SCHOOL OF MATHEMATICAL, PHYSICAL AND COMPUTATIONAL SCIENCES

Advances in Machine Learning Algorithms for Financial Risk Management

by

Emmanuel Osei-Brefo

Thesis submitted for the degree of Doctor of Philosophy

PhD Computer Science

Department of Computer Science

November 2023

UNIVERSITY OF READING

Declaration of original authorship

Declaration: I confirm that this is my own work and the use of all material from other sources have been properly and fully acknowledged.

Emmanuel Osei-Brefo

Acknowledgements

I would like to express my deepest gratitude and appreciation to my main supervisor, Professor Xia Hong for her enthusiasm, guidance, unwavering support, and invaluable expertise throughout my PhD journey. Her insights, constructive feedback, and encouragement have been instrumental in shaping the direction of this work. I would also like to thank my second supervisor, Professor Richard Mitchel for his inputs and proof-reads. I express my deepest gratitude and appreciation to my family members especially my wife Sabina, my mother, Mary Boahin and children; Declan, Perez, Mary-Anne and Emmanuel Jnr whose love and understanding have been the foundation upon which my academic pursuits have been built. Their belief in my abilities and the sacrifices they made have been a constant source of motivation, and I am forever grateful for their support. I also extend my gratitude to my fellow researchers and colleagues for their engaging discussions and collaborative efforts that made the research environment stimulating and inspiring. I thank the faculty members, administrative staff and support personnel at the Department of Computer Science at the University of Reading for fostering a conducive academic environment and providing the resources necessary for the successful completion of this thesis. I also thank my friends who have stood by me during the highs and lows of this journey for their unwavering friendship especially, Thomas Asante Mensah, Baaba Amoah and Dr. Kwaku Aduse-Poku for their help and advice. I acknowledge the collective effort of all those who have been involved in this endeavour, directly or indirectly.

Abstract

In this thesis, three novel machine learning techniques are introduced to address distinct yet interrelated challenges involved in financial risk management tasks. These approaches collectively offer a comprehensive strategy, beginning with the precise classification of credit risks, advancing through the nuanced forecasting of financial asset volatility, and ending with the strategic optimisation of financial asset portfolios.

Firstly, a Hybrid Dual-Resampling and Cost-Sensitive technique has been proposed to combat the prevalent issue of class imbalance in financial datasets, particularly in credit risk assessment. The key process involves the creation of heuristically balanced datasets to effectively address the problem. It uses a resampling technique based on Gaussian mixture modelling to generate a synthetic minority class from the minority class data and concurrently uses k-means clustering on the majority class. Feature selection is then performed using the Extra Tree Ensemble technique. Subsequently, a cost-sensitive logistic regression model is then applied to predict the probability of default using the heuristically balanced datasets. The results underscore the effectiveness of our proposed technique, with superior performance observed in comparison to other imbalanced preprocessing approaches. This advancement in credit risk classification lays a solid foundation for understanding individual financial behaviours, a crucial first step in the broader context of financial risk management.

Building on this foundation, the thesis then explores the forecasting of financial asset volatility, a critical aspect of understanding market dynamics. A novel model that combines a Triple Discriminator Generative Adversarial Network with a continuous wavelet transform is proposed. The proposed model has the ability to decompose volatility time series into signal-like and noise-like frequency components, to allow the separate detection and monitoring of non-stationary volatility data. The network comprises of a wavelet transform component consisting of continuous wavelet transforms and inverse wavelet transform components, an auto-encoder component made up of encoder and decoder networks, and a Generative Adversarial Network consisting of triple Discriminator and Generator networks. The proposed Generative Adversarial Network employs an ensemble of unsupervised loss derived from the Generative Adversarial Network component during training, supervised loss and reconstruction loss as part of its framework. Data from nine financial assets are employed to demonstrate the effectiveness of the proposed model. This approach not only enhances our understanding of market fluctuations but also bridges the gap between individual credit risk assessment and macro-level market analysis.

Finally the thesis ends with a proposal of a novel technique for Portfolio optimisation. This involves the use of a model-free reinforcement learning strategy for portfolio optimisation using historical Low, High, and Close prices of assets as input with weights of assets as output. A deep Capsules Network is employed to simulate the investment strategy, which involves the reallocation of the different assets to maximise the expected return on investment based on deep reinforcement learning. To provide more learning stability in an online training process, a Markov Differential Sharpe Ratio reward function has been proposed as the reinforcement learning objective function. Additionally, a Multi-Memory Weight Reservoir has also been introduced to facilitate the learning process and optimisation of computed asset weights, helping to sequentially re-balance the portfolio throughout a specified trading period. The use of the insights gained from volatility forecasting into this strategy shows the interconnected nature of the financial markets. Comparative experiments with other models demonstrated that our proposed technique is capable of achieving superior results based on risk-adjusted reward performance measures.

In a nut-shell, this thesis not only addresses individual challenges in financial risk management but it also incorporates them into a comprehensive framework; from enhancing the accuracy of credit risk classification, through the improvement and understanding of market volatility, to optimisation of investment strategies. These methodologies collectively show the potential of the use of machine learning to improve financial risk management.

Table of Contents

List of Tables

List of Figures

Nomenclature

Acronyms and Abbreviations

1-D	one dimensional
2-D	Two dimensional
Ad-Attack	Adversarial Attack Models
AUC	Area Under Curve
ADASYN	Adaptive synthetic oversampling approaches
AFT	Accelerated Time Model
ANN	Artificial Neural Network
ARCH	Autoregressive Conditional Heteroscedasticity
ARIMA	Autoregressive integrated moving average
BCBS	Basel Committee on Banking Supervision
CAPM	Capital Asset Pricing Model
CapsNet	Capsules Neural Network
CECL	Current Expected Credit Losses
CVaR	Conditional Value at Risk
CBUT	cluster-based under-sampling technique
cwt-TriGAN	continuous wavelet transform triple discriminator GAN network
cwt-TriGAN	continuous wavelet transform triple discriminator GAN network
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Networks

CWT	Continuous Wavelet Transform
C-RNN-GAN	continuous recurrent neural network GAN
DBN	Deep Belief Networks
ENN	Edited Nearest Neighbour Method
ETFs	Exchange traded funds
EM	Expectation Maximisation
E-step	Estimation Step
FRM	Financial Risk Management
FTSE	Financial Times Stock Exchange
FASB	Financial Accounting Standards Board
FIRB	Foundation Internal Ratings-Based Approach
GARCH	Generalised Autoregressive Conditional Heteroskedasticity
IASB	International Accounting Standards Board
IFRS 9	International Financial Reporting Standard 9
IRB	Internal Ratings-Based Approach
DPG	Deterministic Policy Gradient
D-GAN	Deep Generative Adversarial Network
FN	False Negatives
FP	False Positives
FX	Foreign Exchange
GBP	British Pounds
Hz	Hertz
LLP	Financial Loan Loss Provisioning
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GMMOT	Gaussian Mixture Model Oversampling Technique
GRU	Gated Recurrent Unit
HDRCS	Hybrid Dual Resampling with Cost-sensitive Technique
LSTM	Long Short Term Memory

LCU	Local Credit Union
MAD-GAN	Multivariate Anomaly Detection
MAE	Mean Absolute Error
ML	maximum likelihood
MPV	Mean portfolio value
MCT	Minority Cloning Technique
MDP	Markov Decision Process
MPT	Modern portfolio theory
MMWR	Multi-Memory Weight Reservoir
M-step	Maximisation Step
NLP	Natural language processing
RMSE	Root Mean Square Error
SAE	stacked autoencoders
ST	spatio temporal events
STFT	short-time Fourier transform
TN	True Negatives
TP	True Positives
VAE	Variational autoencoder
WT	wavelet transforms
SV	Stochastic Volatility
SVM	Support Vector Machine
VaR	Value at Risk
RCGAN	Recurrent Conditional GAN
SMOTE	Synthetic Minority Oversampling Technique
TimeGAN	Time series Generative Adversarial Network
US	United States
RNN	Recurrent Neural Network
Primary-Caps	primary capsule layer
RL	Reinforcement Learning

3CNNRL + PVM	3 CNN and Portfolio vector memory RL model
5CNNRL + PVM	5 CNN and Portfolio vector memory RL model
5CNNRL + PVM + mDSR	5 CNN with Portfolio vector memory
5CNNRL +MMWR + mDSR	5 CNN with Multi-Memory Weight Reservoir and mDSR RL model
CapsNetRL + PVM	CapsNet with Portfolio vector memory RL model
CapsNetRL + PVM + mDSR	CapsNet with Portfolio vector memory and mDSR RL model
CapsNetRL +MMWR	CapsNet with Multi-Memory Weight Reservoir RL model
CapsNetRL +MMWR + mDSR	CapsNet with Multi-Memory Weight Reservoir and mDSR RL model
CSLR	Cost-Sensitive Logistic Regression
FPV	Final portfolio value
IoT	Internet of Things
KNN	K-Nearest Neighbours
MDA	Multiple discriminant analysis
mDSR	Markov Cumulative Discounted Sharpe Ratio
PH	Cox proportional hazards model
D_1	Discriminator network 1
D_2	Discriminator network 2
D_3	Discriminator network 3 and mDSR RL model

List of Notations Used

Notations used for Chapters 1 and 2

$\mathbf{x_i}$	vector of covariates
y_i	Dependent Variable
β_0	Intercept
β	Regression coefficient
$\boldsymbol{\beta}$	Regression co-efficient and intercept
n	Number of dimensions
N	Size of dataset
p_i	probability success
$1 - p_i$	probability of failure
$\ell(\beta)$	Log-likelihood function for logistic regression
$\mathbf{x}$	represent an n-dimensional data vector
$\lvert\mathbf{\Sigma}_j\rvert$	the determinant of $\mathbf{\Sigma}_j$
$\phi_{\vert}$	the mixture weights.
k'	represents the number of components of the Gaussian mixture
$\mathbf{z}$	unobserved latent variables
$\boldsymbol{\mu}_j$	n-dimensional mean vector
$\mathbf{\Sigma}_j$	covariance matrix
$\ell(\phi, \mu, \Sigma)$	Log-likelihood function for GMM
$\mathbf{x_t}$	Input vector for LSTM
h_t	Output unit State for LSTM
$\mathbf{c}_t$	LSTM Cell state vector
$\mathbf{W}$	LSTM parameter matrices
b	LSTM parameter vector
f_t	forget gate vector for LSTM

i_t	input gate vectors for LSTM
o_t	output gate vector for LSTM
G	Generator model
$\boldsymbol{\theta}$	parameters of GAN network
p_z	Known probability distribution
p_g	generator distribution
p_{data}	distribution of the data
$\bigtriangledown_{disc}$	discriminator error function using Gradient descent
$\bigtriangledown_{gen}$	generator error function using Gradient descent
X_r	batch of real data randomly drawn
	from the training data for GAN
X_g	batch of generated data GAN
$\boldsymbol{\theta}$	GAN parameters
Z_g	sample of the batch of random noise data
$\mathcal{X}$	vector space of temporary features for timeGAN
$\mathcal{S}$	vector space of static features for timeGAN
H_S, H_X	latent vector spaces corresponding to feature spaces S, X
e_S	embedding network for static features for timeGAN
e_X	embedding network for temporal features for timeGAN
$r_{\mathcal{S}}$	recovery networks for static embeddings
$r_{\mathcal{X}}$	recovery networks for temporal embeddings
$\mathcal{L}_R$	reconstruction loss
$\boldsymbol{\theta}_e$	parameters of the encoder for timeGAN
$\boldsymbol{\theta}_r$	parameters of the decoder network for timeGAN
$\boldsymbol{\theta}_g$	parameters of the generator network for timeGAN
$\boldsymbol{\theta}_d$	parameters of the discriminator network for timeGAN
λ, γ	regularisation parameters for timeGAN

Notations used for Chapter 3

P_d	Probability of default
L_{gd}	Loss Given Default
E_{ad}	Exposure at Default
R_r	Recovery Rate
$\mathcal{U}$	Unexpected Loss
E_l	Expected Loss
ε	Extreme Possible Losses
σ	standard deviation of returns
t	period of time
σ_A	Annualised volatility
μ	Mean
r_t	daily returns
P_t	The closing price of an asset at time t
r_t^r	rate of return
r_t^l	logarithmic rate of return
$\mathbf{w}_t$	portfolio vector weight at period t
K	number of assets in a portfolio
$\mathbf{v}_t$	Opening Price Vector for period t
$\mathbf{v}_t^{hi}$	Highest Price Vector for period t
$\mathbf{v}_t^{lo}$	Lowest Price Vector for period t
$\mathbf{y}_t$	Price relative vector for period t
$\mathbf{w}_0$	initial portfolio weight vector
P_0	Initial amount invested
P_f	final portfolio value
r_k	expected return (per period) of the asset k
ρ_{kj}	correlation between the returns of the assets k and j
σ_k	standard deviation of return for asset k
σ_{kj}	correlation between the returns of assets k and j
R	desired expected return from the chosen portfolio
w_k	The proportion of the total investment associated with the asset k

Notations used for Chapter 4

$\boldsymbol{X}$	Imbalanced Data set
$\mathbf{X}^+$	Independent Minority class dataset
$\mathbf{X}^-$	the majority class data
$\mathbf{x}^+_{os}$	minority class Synthetic data sample generated
$\mathbf{x}^+_R$	Remainder of minority sample from the SMOTE operation
y_i	binary class
K'	The number of clusters used in k-means clustering.
$\mathbf{x}^+_k$	one of k^{th} neighbours of the minority class data
Φ_r	random value generated from the application of the uniform probability distribution in SMOTE
N^+	Number of Minority class samples in the data
N^-	Number of Majority Class Samples in the Data
N	Number of data points for both the majority and minority classes
λ^*	desired imbalance ratio
N^+_{new}	New samples to be drawn
F	desired top features
k	Number of neighbours
w^-	the weight for the majority class, $y_i = 1$
w^+	the weight for the minority class, $y_i = 0$
r	Resampling coefficient

Notations used for Chapter 5

ϕ	scaling factor
C	wavelet coefficients
t	translation factor
$v(x)$	recent Volatility input values
ψ	Morlet mother wavelet
f_ϕ	frequency
f_c	central frequency of the Mother wavelet
$\mathbf{v}$	original 1-D volatility signal
$\mathcal{C}$	vector space of the coefficients obtained
	after applying CWT on volatility data
M	total number of scales
T_s	Forecast time steps
$\hat{p}(\mathbf{C}_T)$	density distribution
Λ	the measure of distance between distributions
$E_\mathcal{C}$	recurrent embedding network for coefficient temporal features
$R_\mathcal{C}$	Decoder networks for coefficient temporal embeddings
$\mathbf{z}_t$	random vector
$D_{\mathcal{C}_i}$	respective classification layers for the
	output layers for Discriminators 1,2 3
$\overrightarrow{\mathbf{q}}_{t_i}$	sequence of forward states for Discriminators
$\overleftarrow{\mathbf{q}}_{t_i}$	sequence of backward states for Discriminators
$\overrightarrow{f}_{\mathcal{C}_i}$	LSTM of the forward states for discriminators
$\overleftarrow{f}_{\mathcal{C}_i}$	LSTM of the backward states for Discriminators
$\boldsymbol{\theta}_{d_1}$	Parameters for the discriminator network 1
$\boldsymbol{\theta}_{d_2}$	Parameters for the discriminator network 2
$\boldsymbol{\theta}_{d_3}$	Parameters for the discriminator network 3

Notations used for Chapter 5

$\mathcal{L}_{D_i}$	Average of individual Discriminator losses
w_i	proportional weights used
$\mathcal{L}_S$	Supervised loss function
$\mathcal{L}_G$	Generator loss
$\overline{\boldsymbol{\theta}_d}$	Average of the 3 Discriminator losses
Λ	Measure of distance between the distributions

Notations used for Chapter 6

$\mathbf{w_j}$	vector output of a capsule j
o_j	total input of a capsule j
$\mathbf{\Phi}$	Weight matrix of Capsules
Φ_{ij}	Weight coefficient of Capsules i and j
$\hat{\mathbf{u}}_{j\|i}$	Prediction vectors of Capsules i and j
c_{ij}	coupling coefficient of Capsules i and j
F	Number of features
$\mathcal{L}_M$	The Marginal Loss
$\mathcal{L}_r$	Reconstruction Loss calculated by the CapsNet
$\mathcal{L}_T$	Total Loss
$\mathbf{w}_{t_{init}}$	initial non-optimised portfolio weights output
θ_{init}	initial random set of parameters used by CapsNet to start gradient update
$\mathbf{w}_t^{opt}$	optimised portfolio vector weights
$\pi_{\theta_{init}}$	Initial policy used at the start of the training process
θ	capsnet parameter
π	policy
π_θ	policy parameterised by capsnet
f	Number of features
t'	Number of input periods before t
D_t	differential Sharpe ratio
$R_{(d)}$	Markov Differential Sharpe Ratio
ϖ	Sharpe ratio used as instantaneous reward for CapsNet-based agent at time t
ϖ_s	Sharpe ratio used as instantaneous reward for secured portfolio at time t
η	decay rate
M_A	Main Reservoir A
M_B	Main Reservoir B
M_C	Cache Reservoir
$\mathbf{w}_t^A$	portfolio weight vectors stored in Main Reservoir A
$\mathbf{w}_t^B$	portfolio weight vectors stored in Main Reservoir B
$\mathbf{w}_t^C$	represent the portfolio weight vectors Cache reservoir
bs	total batch size sampled from Cache reservoir and Main reservoir B
bs_1	mini-batch drawn for training from the Cache reservoir
$\mathcal{L}_M$	Loss function between predicted and actual portfolio allocation
$\mathbf{X}_t$	Input price tensor

Notations used for Chapter 6

$\hat{\mathbf{X}}_t$	reconstructed input data
$\mathbf{w}_{t-1}^{opt}$	optimised portfolio weights for the previous period
bs_2	mini-batch drawn for training from the Main Reservoir B
$\varkappa$	The proportional weight to be used for Cache reservoir
$p_\Omega(t_s)$	exponentially distributed probability
$p_\beta(t_s)$	geometrically distributed probability
n_{bs_1}	number of periods in each mini-batch Cache Reservoir
n_{bs_2}	number of periods in each mini-batch Main Reservoir B

Chapter 1

Introduction

In the dynamic field of financial risk management, the use of advanced machine learning techniques has emerged as a transformative force. This introduction sets the stage for our research into the complex connection between machine learning and financial risk management, where novel methodologies are developed and applied to address some critical challenges in the field. This chapter establishes the context and motivation for our research and outlines the specific contributions made in this thesis in the field of computer science and financial risk management.

1.1 Background

In the current information age, everything around us is data-driven and digitally recorded (Sarker, 2021). This has provided individuals, organisations, and society with the opportunity to collect large amounts of data (Injadat et al., 2021). There exist different kinds of data, such as the Internet of Things (IoT) data, cybersecurity data, financial data, smart city data, business data, smartphone data, social media data, health data plus many more (Sarker, 2021). These data can be structured, semi-structured, or unstructured. The ability to extract insights from these data can be used to build different intelligent applications in various domains such as finance (Sarker, 2021). However, humans do not have the cognitive ability to analyse and understand such large amounts of data due to information overload (Injadat et al., 2021). Thus, data management tools and techniques with the capability to extract useful insights from a large amount of data are needed. As evidenced by their remarkable successes in the fields of computer vision, natural language processing, and robotics, machine learning (ML) techniques can provide a mechanism for humans to process such large amounts of data, gain insight into the behaviour of the data, and make a more informed decision as required.

The 2007–2009 global financial crisis exposed the weak financial risk management (FRM) practises in place at the time. Prior to this, each risk type was managed independently of the others through a process called risk decomposition, which entailed the identification and management of each individual risk separately. The financial performance of a firm can be negatively affected by various adverse events, such as financial market movements, defaults on loans, fraud, cybercrime, and loss of customers. Based on these sources of risk factors, the main types of risk faced by most financial institutions can be categorised as credit risk, market risk, and operational risk. Credit risk refers to the uncertainties involved in one party's inability to perform its contractual financial obligations to another party. An example is a debtor's failure to pay the interest or principal on a loan and a counter-party's failure to perform a settlement. The creditworthiness of households is important because it does not only influence the lending decisions of financial institutions but rather it also impacts the broader economic stability. Financial risk management tasks have been shown to be generally challenging because they use sparse and complex data structures. Subsequently, identifying, quantifying, and managing risk plays a crucial role in any institution.

Financial risk management primarily entails the minimisation of losses and the maximisation of profits by firms, and the processes involved in these tasks are information-driven (Crouhy et al., 2014a). Machine learning technologies and methods have recently been adapted for a variety of financial risk management tasks because the modern investor has become sophisticated and therefore demands complex financial products from financial institutions (Fabozzi et al., 2021). This current trend has led to the difficulties of these firms in accurately evaluating the exposure of their large and complex financial portfolios to the evolving financial markets using classical and statistical approaches. Therefore, the need to manage these risks holistically through a risk aggregation procedure has attracted ubiquitous interest in the use of machine learning and deep learning for financial risk management.

Portfolio optimisation is a formal mathematical approach used to make investment decisions in a collection of financial assets. Portfolios are a combination of a feasible set of assets and are specified in terms of weights. It involves maximising the returns of an investment portfolio while minimising the risks associated with it (Soleymani and Paquet, 2020).

Machine learning is a broad area and covers various classes of algorithms for pattern recognition and decision making (Dixon et al., 2020). In recent times, the popularity of machine learning approaches to learning has been on the rise (Sarker, 2021). This is due to the recent progress attained by machine learning models, which have consistently shown their ability to detect abstract patterns in data with greater accuracy levels exceeding that of human experts (Nichols et al., 2019). Extending classical methods also use deep neural networks that typically consist of more than one hidden layer, organised in deep nested network architectures for their operations (Janiesch et al., 2021). Machine learning algorithms can be grouped into four main categories; these are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (Sarker, 2021).

Supervised Learning: Supervised learning is a type of machine learning algorithm that learns from labelled data to help predict outcomes of unseen data (Bishop, 2006; Sarker, 2021). Under this, a function that maps an input to an output is used to learn a machine-learning task. It uses labelled training data and a collection of training examples to predict the outcomes of the target vectors. Supervised learning tasks can be sub-grouped into classification and regression tasks. The most significant distinction between classification tasks and regression tasks is that classification tasks predict distinct class labels, while regression tasks predict continuous values(Bishop, 2006; Sarker, 2021). On the other hand, a regression task is a type of supervised machine learning algorithm that allows the prediction of continuous values using the value of one or more independent predictor variables. Regression models are used in different fields such as share price forecasting, volatility forecasting, cost estimation, trend analysis, marketing, time series estimation, house price prediction and many more (Bishop, 2006). Classification is a type of supervised learning method in machine learning, in which the aim is to assign each input variable to one of the finite numbers of discrete categories (Bishop, 2006).

In the machine learning domain, classification tasks play a crucial role in analysing data and making predictions. For instance, in classification tasks like fraud detection, the goal is to predict the class of given data points, such as determining whether a transaction is 'fraud' or 'not fraud.' Another example is the prediction of house prices, where supervised learning techniques help estimate property values based on various features. Similarly, financial risk management relies on supervised learning to assess credit risk, which involves estimating the probability of default (P_d) for borrowers. This estimation is pivotal in valuing credit derivatives, gauging a borrower's creditworthiness, and determining economic or regulatory capital for financial institutions. Examples of supervised learning include spam detection, text classification, and prediction of house prices.

The foundation of credit risk lies in the Probability of Default (P_d), Exposure at Default (E_{ad}), and Loss Given Default (L_{gd}) (Thomas, 2009). These interconnected drivers underpin the credit risk assessment and are particularly critical as default occurrences increase, causing a decrease in recovered amounts. The probability of default (P_d) is defined as the likelihood that a borrower will not be able to pay a loan on the agreed time schedule. The accurate estimation of P_d, especially for imbalanced datasets, is vital in financial risk management, given its profound implications for risk valuation and capital allocation. The Exposure At Default, E_{ad}, represents the amount owed by the debtor at the time of default and cannot be greater than the loan amount originally granted. It is used by financial institutions to estimate potential losses and for them to set aside capital reserves accordingly. This helps financial institutions to manage and mitigate risks, price loans appropriately, and maintain adequate capital reserves. The Loss Given Default(L_{gd}) on the other hand is the fraction of the investment a financial institution stands to lose should the debtor default. It is calculated as the final loss a bank or a financial institution suffers when a borrower defaults and is represented as a percentage of total exposure when the account falls into arrears.

Unsupervised learning: Unsupervised learning is a type of machine learning algorithm in which unlabelled datasets are analysed without human interaction or interference. Therefore, it involves the finding of patterns within the data without the presence of any ground truth or labelled data (Sarker, 2021). It is a data-driven process widely used to extract features and identify meaningful trends and structures (Sarker, 2021). It also involves grouping of the data into clusters or arranging them into organised groups for exploratory purposes. Examples of unsupervised learning tasks are clustering, Generative Adversarial Networks (GANs)(Goodfellow et al., 2014), density estimation (Biprodip and Mahit, 2017), feature learning, dimensionality reduction, and anomaly detection (Sarker, 2021).

Semi-supervised: Semi-supervised learning is a type of machine learning algorithm that can be defined as a combination of supervised and unsupervised learning algorithms since it uses both labelled and unlabelled data in its operation (Sarker, 2021). Using this hybridisation concept, machine learning algorithms can learn to label unlabelled data. Therefore, it falls between learning "with supervision" and learning "without supervision". In the real world, where unlabelled data are in abundance and labelled data are scarce, the use of semi-supervised learning can be a useful tool (He et al., 2016). The main aim of a semi-supervised learning algorithm is to provide better results for prediction tasks compared to prediction tasks produced solely from the use of the labelled data. Semi-supervised learning can be used in applications such as machine translation, fraud detection, data labelling, and text classification (Sarker, 2021).

Reinforcement learning: Reinforcement learning is a type of machine learning algorithm that enables agents and machines to find the appropriate actions to take in a given situation to maximise the reward (Bishop, 2006; Silver et al., 2014). Reinforcement learning is different from other machine learning paradigms, such as supervised learning, because it is goal-directed learning through interaction with only a reward signal since there is no supervisor present and the actions of agents affect the subsequent data received. It is also different from the unsupervised learning paradigm because its sole aim is to maximise the reward signal rather than trying to find hidden structures. Under this method, there exists a sequence of states and actions in which the algorithm interacts with its environment (Bishop, 2006).

1.2 Motivation

Previous research has attempted to apply machine learning techniques to financial risk management; however, these efforts remain fragmented. While various methodologies have been explored, there remains a need for an integrated solution that addresses challenges such as data unavailability, data quality concerns such as an imbalanced dataset, noisy data, non-stationarity of data and algorithm-based issues such as model non-convergence. Our work bridges these gaps by proposing innovative techniques that include data balancing, volatility forecasting, and portfolio optimisation.

Imbalanced data sets occur in financial credit data, where the number of defaulters or bad payers (positive class observations), is significantly lower than the number of non-defaulters or good payers (negative class observations). Therefore, it is essential to investigate and implement a robust estimation technique to predict the probability of default (P_d) for consumers, since incorrect P_d results in a false valuation of risk, an incorrect rating, and incorrect pricing of financial instruments.

The prices of financial assets and interest rates continuously change, driving the values of securities and other financial assets up and down. These price movements referred to as volatility, create the potential for loss, as they are deemed to be the engine of market risk. Depending on the underlying assets, these market risks can take the form of interest rate risk, equity risk, and currency risk. Therefore, it is important to investigate other novel techniques to accurately estimate the volatility of financial markets, since financial markets serve as the foundation of every economy.

Although advances have been made in the use of deep reinforcement learning for portfolio optimisation with which excellent results have been obtained (Jiang et al., 2017; Liang et al., 2018), there are still some gaps in the existing literature on how to maximise the profitability of a portfolio using deep reinforcement learning. Nine challenges have been identified with the use of reinforcement learning in real-life scenarios (Dulac-Arnold et al., 2021). There are issues such as sample efficiency, credit assignment, and exploration versus exploitation. The specification of a reward function may not represent the intention of the designer and may cause problems (Li, 2022). Issues such as delusional bias results in behaviours such as overestimation or under-estimation, instability and divergence in learning algorithms (Lu et al., 2018), and Markov reward expressivity (Abel et al., 2022) have been identified by researchers as some of the issues in reinforcement learning. The Deep Deterministic Policy Gradient (DDPG) algorithm has been tested against different trading problems to find optimal or near-optimal control strategies in dynamic portfolio optimisation and reinforcement learning (Chaouki et al., 2020). However, the performance of the model might be limited to the specific trading problems and scenarios tested, which could possibly affect its ability to generalise and might also be susceptible to the problem of delusional bias. Liang et al. (2018), compared DDPG with other algorithms under various settings, which offered good insights into parameter tuning and feature selection, with a focus on the China stock market adding value due to its unique characteristics. However, the findings of their work were limited to the Chinese stock market, and as such may not be directly applicable to other markets due to differences in market dynamics and regulations.

Cui et al. (2024) also proposed a Deep Reinforcement Learning (DRL) hyper-heuristic approach that incorporated domain knowledge and low-level trading strategies and was designed to narrow down the search space and improve portfolio optimisation. However, the model might suffer from more complexities without the ability to generalise well to different market conditions due to the integration of domain knowledge and trading strategies. Lucarelli and Borrotti (2020) used local agents to learn asset behaviours and a global agent for the reward function to provide a balanced approach to understanding both individual asset dynamics and the overall portfolio performance. However, the use of both local and global agents might introduce challenges in the coordination and synthesis of the different learning components. This could also lead to complexities in defining and optimising the reward function that aligns with investment goals. The works of Pigorsch and Schäfer (2021) also proposed the High-Dimensional Stock Portfolio Trading with Deep Reinforcement Learning which has the ability to handle high-dimensional portfolios and accommodate datasets with data gaps and varying history lengths making them highly versatile. However, the model requires an increased computational resources to operate efficiently due to its ability to handle high-dimensional data and complex scenarios.

The determination of a reward signal has been shown to be the most challenging part of the design of a reinforcement learning problem (Littman et al., 2017; Dulac-Arnold et al., 2021). Existing reward functions result in sub-optimal convergence and unstable results. Therefore, it is essential to investigate and design other novel reward functions to address these shortcomings.

Furthermore, existing reinforcement learning approaches of researchers, such as Jiang et al. (2017); Almahdi and Yang (2017); Liang et al. (2018); Soleymani and Paquet (2020), use a model-free RL algorithm made of single buffer memory that does not consider the instability and risk associated with the noisy and non-stationary financial market environment in its operations. Therefore, it is essential to investigate and introduce a robust and efficient replay memory design together with a training scheme capable of overcoming the shortcomings of existing ones.

1.3 Connection between Credit Risk, Volatility Forecasting, and Portfolio Optimisation

Improved credit risk assessments, improved forecasting volatility and the effective optimisation of financial asset portfolios are interrelated and collectively contribute to effective financial risk management strategies. Credit risk assessment is fundamental in financial risk management since it influences investment decisions and risk evaluations. Accurate credit risk predictions can affect the volatility forecasts of financial assets, because it directly affects investor confidence and market stability. Additionally, the volatility of financial assets is an important component in the construction of optimised portfolios. Effective volatility forecasting enables more informed decisions in portfolio construction, balancing potential returns against associated risks.

Finally, portfolio optimisation relies on both credit risk assessments and volatility forecasts. A detailed understanding of these elements enables better allocation of assets which ensures that portfolios are not only diversified but also tailored to withstand market fluctuations and credit risk events. The application of effective machine learning techniques to these 3 areas has the potential to change the current approach to financial risk management. The use of advanced machine learning algorithms can lead to accurate credit risk assessments, more reliable volatility forecasts, and efficient portfolio optimisations. This enhances the overall stability and performance of the financial markets in the long run.

1.4 Aims and Objectives

The overarching aim of this research is to explore and investigate the problems inherent in the application of machine learning to financial risk management tasks. Based on the main aim described above, the main objectives of this research are the following.

- Investigate the use of machine learning techniques to enhance credit risk assessment on imbalanced datasets using a novel Hybrid Dual Resampling and Cost-Sensitive Learning Technique (HDRCS), which results in a better imbalance ratio obtained heuristically.
- Investigate and improve the forecasting of noisy, non-stationary, and unavailable financial datasets such as volatility using Generative Adversarial Network (GAN) and continuous wavelet transform.
- Investigate and optimise portfolio performance and asset allocation using the Capsules Neural Network and the Multi-Memory Weight Reservoir (MMWR) training scheme capable of improving the robustness of reinforcement learning agents.
- Design a stable and effective reward function for reinforcement learning-based portfolio optimisation. This will be monitored by performing a combination of empirical testing, statistical analysis, and qualitative evaluation. The empirical testing will involve the running of the RL model multiple times with the same settings to observe the variability in the performance metrics used. It will also be bench-marked against other reward functions to assess the model's performance using alternative reward structures

These objectives served as a road map through which the aim of this research has been achieved. Consequently, it has led to the development, investigation and evaluation of a series of techniques, with a number of research contributions being made.

The next section highlights the contributions that have been made in this thesis.

1.5 Summary of Contributions

The research and investigations conducted in this thesis on the application of machine learning techniques to financial risk management have led to a number of contributions in this domain. A summary of the contributions made in this thesis is as follows:

- Proposed the Hybrid Dual Resampling with Cost-sensitive Technique (HDRCS), which is a fully featured technique for imbalanced data sets and involves the simultaneous use of Gaussian Mixture Model Oversampling Technique (GMMOT) to over-sample the minority class, the cluster-based under-sampling technique (CBUT) to reduce the excess number of data points and Cost-sensitive logistic regression (CSLR) algorithm on the resampled data. The significance of the HDRCS approach lies in its potential to improve credit risk assessment and help address the challenge involved in the use of imbalanced data. It allows financial institutions to make more accurate credit decisions, minimising the risk of false valuations, incorrect risk ratings, and mispriced financial instruments.

- Proposed continuous wavelet transform triple discriminator GAN network (cwt-TriGAN) which utilises continuous wavelet transform and ensemble loss function obtained from a three discriminator network, which overcomes the problem of mode collapse, leading to improved forecasting of volatility time series. Previous studies have not utilised this architecture. Detailed experiments have been undertaken on nine financial assets using the cwt-TriGAN and other models, such as TimeGAN and RCGAN.

- Empirical investigation to determine the effect of time steps on the accuracy of the volatility prediction has been undertaken. Additionally, reconstruction techniques used to obtain the original volatility for the output of the continuous wavelet transform have been investigated.

- Proposed a Multi-Memory Weight Reservoir (MMWR) training scheme using Capsules Neural Network, which facilitates and improves the optimisation process of the portfolio weights. This helps in the sequential re-balancing of the portfolio throughout the trading period using a continuous action space. This methodology addresses challenges such as sample efficiency and exploration-exploitation trade-offs, providing investors with a tool to navigate complex and dynamic market conditions. The practical impact of MMWR extends to investment practises, where its adoption can lead to more resilient and adaptive portfolios that optimise risk-adjusted returns in volatile markets.

- Proposed discounted cumulative reward function known as a Markov differential Sharpe ratio that provides stability and optimises the training process. The significance of the proposed Markov differential Sharpe ratio lies in its ability to provide stability and optimise the training process in portfolio optimisation using reinforcement learning. This proposed technique ensures that the learning process results in optimal portfolio weights over time.

List of Publications

- Osei-Brefo, E., Mitchell, R., Hong, X. (2023). Hybrid Dual-Resampling and Cost-Sensitive Classification for Credit Risk Prediction. In: Bramer, M., Stahl, F. (eds) Artificial Intelligence XL. SGAI 2023. Lecture Notes in Computer Science(), vol 14381. Springer, Cham.
- Zehao Liu, Emmanuel Osei-Brefo, Siyuan Chen, and Huizhi Liang. 2020. UoR at SemEval-2020 Task 8: Gaussian Mixture Modelling (GMM) Based Sampling Approach for Multi-modal Memotion Analysis. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pages 1201–1207, Barcelona (online). International Committee for Computational Linguistics.

1.6 Outline of Thesis

The rest of this thesis is organised as follows: **Chapter 2** covers the fundamentals of the main machine learning algorithms used in this thesis. This includes the background information on density estimation using the Gaussian Mixture Model (GMM), background information on the use of deep learning models such as the use of Long short-term memory (LSTM) and Generative Adversarial Networks (GAN) for time-series forecasting. The same chapter also introduces the concept of a continuous wavelet transform and the basic concepts of reinforcement learning.

Chapter 3 also introduces the background of the financial risk management techniques used in this thesis. It provides the relevant financial background on credit risk modelling, market risk, volatility forecasting, and portfolio optimisation concepts used in this thesis.

Chapter 4 of this thesis elaborates on the use of our proposed Hybrid Dual Resampling with Cost-Sensitive Technique (HDRCS) to overcome the problem of insufficient and imbalanced credit datasets. It captures and evaluates the novel Hybrid Dual Resampling with a Cost-sensitive Technique using the Gaussian mixture model to model the credit risk of imbalanced datasets. The technique involves the simultaneous use of cluster-based undersampling of the majority class, the use of the Gaussian Mixture Model (GMM) to oversample the minority class, and a cost-sensitive learning algorithm. This chapter also covers the general background in the field of data imbalance and credit risk modelling using Logistic Regression. It reviews some of the resampling techniques for imbalanced datasets such as the Synthetic Minority Oversampling Technique (SMOTE), Gaussian Mixture Model Oversampling, and Cluster Based Under-sampling Techniques (CBUT).

Chapter 5 also covers detailed information on the novel triple-discriminator used within the proposed Continuous Wavelet Transform Triple Discriminator Generative Adversarial Network (cwt-TriGAN) for volatility estimation. It captures the detailed architecture of the proposed cwt-TriGAN, including the in-depth operation of the eight component functions that make up our proposed cwt-TriGAN and evaluates the results of other forecasting techniques.

Chapter 6 covers the use of CapsNet-based reinforcement learning for portfolio optimisation. It covers our proposed Multi-Memory Weight Reservoir (MMWR) training scheme using Capsules Neural Network and also captures the proposed Markov differential Sharpe ratio for portfolio optimisation.

Finally, Chapter 7 covers the conclusion and future work of the proposed novel techniques used in this thesis.

Chapter 2

Machine Learning Preliminaries

2.1 Introduction

This chapter provides the relevant preliminaries on the machine learning algorithms used in this thesis. Hence, the basic machine learning background knowledge of our proposed contributions is introduced here. This includes background information on logistic regression, Gaussian Mixture Model (GMM) density estimation, deep learning for time-series forecasting, and reinforcement learning, respectively.

2.2 Logistic Regression

Logistic regression is a predictive analytic model used in classification problems and can be used to explain the relationship between a dependent binary output and one or more independent variables. Consider data for learning consisting of $(\mathbf{x_i}, y_i)$, $i = 1, 2, ..., N$, where $\mathbf{x}_i = (x_{i_1}, x_{i_2}, ..., x_{i_n}) \in \mathbb{R}^n$ is a vector of covariates and $y_i \in \{0, 1\}$ is a dependent variable. The Logistic Regression model predicts y as a function of $\mathbf{x}$ with the assumption that the independent variables are linearly related to the log odds.

Let $y = p(y = 1|\mathbf{x}) = 1 - p(y = 0|\mathbf{x})$. Logistic Regression model assumes the probability $p_i = p(y = 1|\mathbf{x}_i)$ depends on n covariates $x_{i_1}, x_{i_2}, ..., x_{i_n}$ through:

$$\begin{aligned}\log\left(\frac{p_i}{1-p_i}\right) &= \beta_0 + \beta_1 x_{i1} +, ..., +\beta_n x_{in} \\ &= [1 \;\; \mathbf{x}_i^T]^T \boldsymbol{\beta}\end{aligned}$$

The probability defined in (2.1) below:

$$
\begin{aligned}
p_i &= \left(\frac{\exp\Big([1 \;\; \mathbf{x}_i^T]^T \boldsymbol{\beta}}{1 + \exp\Big([1 \;\; \mathbf{x}_i^T]^T \boldsymbol{\beta}\Big)} \right) && (2.1)\\
&= \left(\frac{\beta_0 + \beta_1 x_{i1} +, ..., \beta_n x_{in}\Big)}{1 - \exp\Big(\beta_0 + \beta_1 x_{i1} +, ..., \beta_n x_{in}\Big)} \right) && (2.2)
\end{aligned}
$$

where, $\boldsymbol{\beta} = [\beta_0, \beta_1, ..., \beta_n]^T$ are unknown parameters referred to as regression coefficients associated with each independent variable in the model. β_0 represents the intercept. Assuming $y_1, ..., y_N$ are independent, the joint distribution is shown in equation (2.3) as:

$$
L = \prod_{i=1}^{N} p_i^{y_i} \Big(1 - p_i\Big)^{1-y_i} \tag{2.3}
$$

The log-likelihood function is given as:

$$
\begin{aligned}
\ell(\boldsymbol{\beta}) = \log L &= \log \left\{ \prod_{i=1}^{N} p_i^{y_i} \left(1 - p_i\right)^{1-y_i} \right\} \\
&= \sum_{i=1}^{N} \Bigg\{ y_i \, \log(p_i) + (1 - y_i) \log(1 - p_i) \Bigg\}, && (2.4)
\end{aligned}
$$

The optimal $\boldsymbol{\beta}$ is found using Newton's method iteratively until it converges to the Maximum Likelihood estimates (Alan, 1990).

2.3 Gaussian Mixture Model Density estimation

Logistic regression is used in imbalanced credit risk modelling in Chapter 4 of this thesis. Similarly, this section describes the Gaussian Mixture Model (GMM) density estimation and the EM Algorithm in detail, which are also used in the proposed Hybrid Dual-Resampling Cost-sensitive (HDRCS) of Chapter 4 for credit risk prediction.

A Gaussian Mixture Model (GMM) represented in Figure 2.1 is a parametric probability density function represented as a weighted sum of the densities of Gaussian components (Biprodip and Mahit, 2017). Density estimation is the construction of an estimate using the observed data of an underlying unobservable probability density function (Bishop, 2006). The unobservable density function is thought of as the density according to which a large population is distributed; the data are usually thought of as a random sample from that population. It involves the modelling of the probability distribution $p(x)$ of a random variable x given a finite set $x_1, x_2, ..., x_N$ of observations.

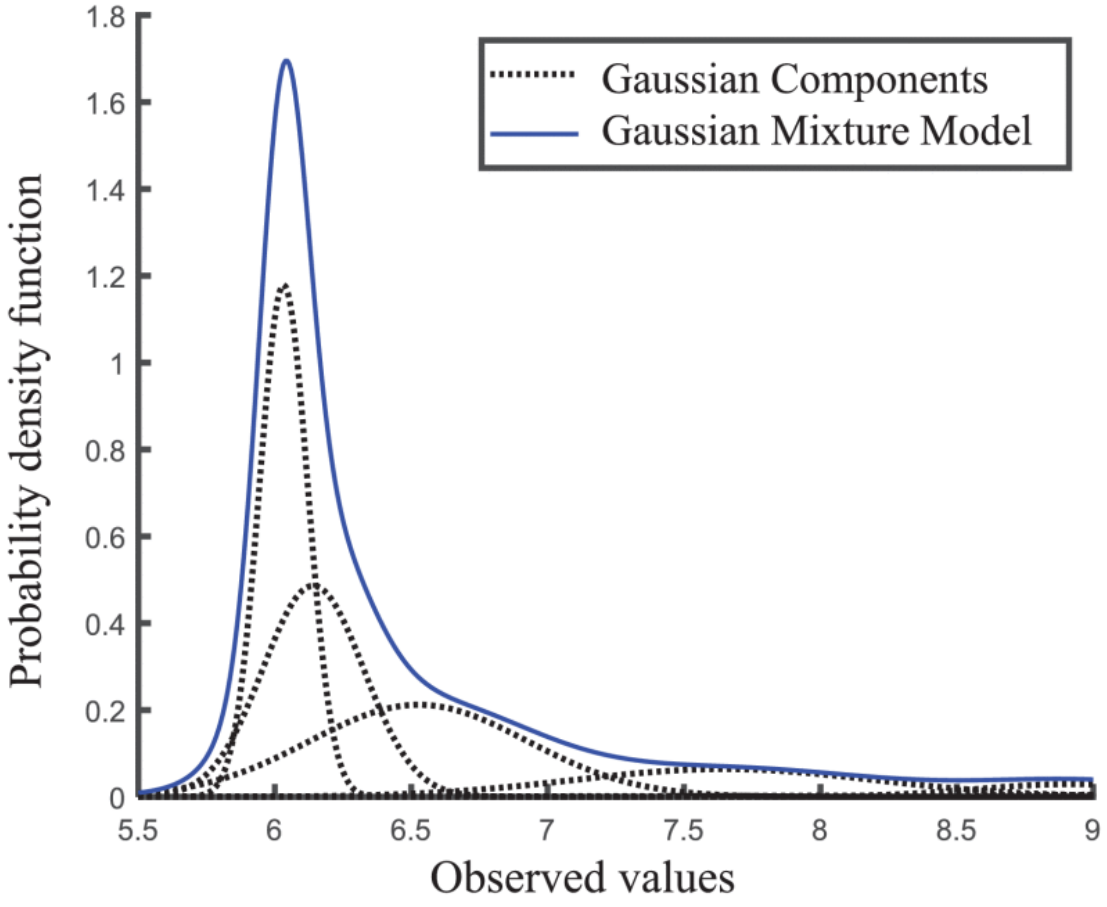


Figure 2.1: Gaussian mixture model representation

GMM is commonly used as a parametric model of the probability distribution of continuous measurements. While the Gaussian distribution possesses some important analytical properties, it suffers from some significant limitations when used on its own for the modelling of real data sets. This is because it does not have the ability to accurately capture the structure of the data set. Hence, a linear combination of two more distributions, such as the Gaussian distribution, is necessary to overcome this shortfall in order to provide a better characterisation of the data set. A linear combination of two or more Gaussians results in very complex densities called the Gaussian mixture model (GMM)(Kwon et al., 2018).

Let:
$\mathbf{x}$ represent an n-dimensional data vector; ϕ_j is the mixture weights.
$j = 1, ..., k'$ represents the number of components of the Gaussian mixture.
$\mathbf{z}$ is a k'-Dimensional binary random variable, with element $\mathcal{Z} \in \{0, 1\}$
$\boldsymbol{\mu}_j$ is a n-dimensional mean vector , $\boldsymbol{\Sigma}_j$ represents a $n \times n$ covariance matrix and

$|\mathbf{\Sigma}_j|$ denotes the determinant of $\mathbf{\Sigma}_j$. (Bishop, 2006)

The weighted sum of j component Gaussian densities that form the Gaussian mixture model is specified by equations (2.5)

$$\sum_{j=1}^{k'} \phi_j = 1 \tag{2.5}$$

The following procedure can be used to estimate ϕ, μ, Σ. Given a set of data, $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$, the maximum likelihood estimation can be applied using equation (2.6)

$$\ell(\phi,\mu,\Sigma) = \sum_{i=1}^{N} \log(p(\mathbf{x}^{(i)};\phi,\mu,\Sigma)) = \sum_{i=1}^{N} \log \sum_{z^{(i)}}^{k'} p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)};\mu,\Sigma)\, p(\mathbf{z}^{(i)};\phi) \tag{2.6}$$

Simplifying the log-likelihood function results in (2.7)

$$\ell(\phi,\mu,\Sigma) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)};\mu,\Sigma) \; + \log p(\mathbf{z}^{(i)};\phi) \tag{2.7}$$

The likelihood function can be maximised through the partial derivative over ϕ, μ, Σ, shown below:

$$\begin{aligned}
\phi_j &= \frac{1}{N}\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\} \\
\mu_j &= \frac{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}} \\
\Sigma_j &= \frac{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}(\mathbf{x}^{(i)} - \mu_j)(\mathbf{x}^{(i)} - \mu_j)^T}{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}}
\end{aligned}$$

Maximum likelihood estimation can be used to estimate the parameters if $\mathbf{z}$ is known. However, it becomes more complicated if $\mathbf{z}$ is unknown. However Maximum likelihood becomes non-beneficial to use if there are latent variables, $\mathbf{z}$. These are variables that interact with those in the data set but are hidden or unobservable. There are many methods used to estimate the parameters of a GMM, but the Expected Maximisation (EM) is the most frequently used method (Kwon et al., 2018). This is because EM is a powerful method to find maximum likelihood solutions for models with latent variables, which involves iteratively finding the maximum likelihood estimates of the parameters of statistical models (Bishop, 2006). The Expectation-Maximisation algorithm is used to estimate $\mathbf{z}$ and other parameters. The EM chooses some random values for the missing data points and then estimates a new set of data. These new values are then used recursively for estimation.

Expected Maximisation(EM) algorithm in GMM

The two steps involved in the EM algorithm are the Estimation Step (E-step) and the Maximisation Step (M-step). Firstly, the model parameters and the latent variables, $z^{(i)}$ can be randomly initialized. In the E-step, the algorithm tries to guess the value of latent variables, $z^{(i)}$ based on the parameters, while in the M-step, the algorithm updates the value of the model parameters based on the guess of $z^{(i)}$ of the E-step. These two steps are repeated until convergence is reached.

- **The Estimation Step (Expectation of the unobserved data)**: In the expectation step, the values of the current parameters are used to find the posterior distribution of the hidden variables given by Equation (2.8)

  For each i, j set:

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) \tag{2.8}$$

- **The Maximisation Step**: The parameters of the unobserved data are updated by maximising the likelihood function generated in the **E-step** above. The revised parameter estimate is determined by maximising the functions. Update the parameters as:

$$\begin{aligned} \phi_j &= \frac{1}{N} \sum_{i=1}^{N} 1\{z^{(i)} = j\} \\ \mu_j &= \frac{\sum_{i=1}^{N} 1\{z^{(i)} = j\} \mathbf{x}^{(i)}}{\sum_{i=1}^{N} 1\{z^{(i)} = j\}} \\ \Sigma_j &= \frac{\sum_{i=1}^{N} 1\{z^{(i)} = j\} (\mathbf{x}^{(i)} - \mu_j)(\mathbf{x}^{(i)} - \mu_j)^T}{\sum_{i=1}^{N} 1\{z^{(i)} = j\}} \end{aligned}$$

  where $\mathbf{z}$ is the unobserved latent variables and $\mathbf{x}$ is the observed variables.

Using the Bayes rule, the E-step leads to equation (2.9):

$$p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi}{\sum_{l=1}^{k'} p(x^{(i)} | z^{(i)} = l; \mu, \Sigma) p(z^{(i)} = l; \phi)} \tag{2.9}$$

Using the GMM setting results in this equation (2.10), where each component density of the GMM is an n-variate Gaussian function.

$$p(\mathbf{x}^{(i)}|z^{(i)} = j; \phi, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma_j|}} \exp\left\{-\frac{1}{2}(x^{(i)} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_j)\right\} \tag{2.10}$$
$$p(z^{(i)} = j; \phi) = \phi_j$$

This allows the possibility of a switch between the E-step and the M-step for any randomly initialised parameters.

2.4 Deep Learning for Time-series Forecasting

In recent years, there has been a renewed interest in utilising deep learning models to predict financial time series data. Notably, models like the Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Deep Belief Networks (DBN) have gained prominence in time-series predictions.

Long Short Term Memory

Long short-term memory (LSTM) was proposed by Hochreiter and Schmidhuber (1997) and has been used to improve the volatility prediction of S & P 500 index and Apple stocks (Liu, 2019). Xiong et al. (2016) also utilised 27 features to predict the S&P 500 volatility in their report which had 25 domestic trends combined with past volatility and return used as inputs of LSTM. LSTMs are a specialised type of Recurrent Neural Network (RNN) with a direct cycle connected between units where temporal patterns are learnt using sequential data. This architecture enables them to store important information from previous inputs, which are then used to update the current output. Through this mechanism, the system obtains memory and, therefore, acts as an intelligent system. Long-Short-Term Memory (LSTM) is a variant of the RNN architecture that is used to train over long sequences and retain memory, allowing long-term dependencies and therefore provides a solution to the problem of vanishing gradient that arises during training by introducing a few more gated units to control access to the cell state (Gers et al., 2000). The main characteristic of hidden layers consisting of memory cells and gates is to store information or forget it if necessary (Fischer and Krauss, 2018). Mathematically, the LSTM can be written as shown in Equation (2.11)

$$
\begin{aligned}
f_t &= \sigma(\mathbf{W}_{fh} h_{t-1} + \mathbf{W}_{fx}\,\mathbf{x}_t + \mathbf{b}_f) \\
i_t &= \sigma(\mathbf{W}_{ih} h_{t-1} + \mathbf{W}_{ix}\,\mathbf{x}_t + \mathbf{b}_i) \\
\hat{\mathbf{c}}_t &= \sigma(\mathbf{W}_{\hat{\mathbf{c}}h} h_{t-1} + \mathbf{W}_{\hat{\mathbf{c}}x}\,\mathbf{x}_t + \mathbf{b}_{\hat{\mathbf{c}}}) \\
\mathbf{c}_t &= f_t \cdot \mathbf{c}_{t-1} + i_t \cdot \hat{\mathbf{c}}_t \\
o_t &= \sigma(\mathbf{W}_{oh} h_{t-1} + \mathbf{W}_{ox}\,\mathbf{x}_t + \mathbf{b}_o) \\
h_t &= o_t \cdot \tanh(\mathbf{c}_t)
\end{aligned}
\tag{2.11}
$$

where $\mathbf{x_t}$ is the input vector, h_t is the output unit State, $\mathbf{c}_t$ is the cell state vector, $\mathbf{W}$ is the parameter matrices, whilst f_t, i_t and o_t represents the forget gate, input gate and output gate vectors respectively. σ is a sigmoid function that bounds the output between 0 and 1 and tanh is a logistic sigmoid function that bounds the output value between -1 and 1. The forget gate f_t decides which information in the cell state should be kept or rejected. The closer the value of f_t is to 1, the higher the probability that the information associated with it will be allowed to pass through it, while values closer to zero are rejected and are not allowed to pass through the gate.

Generative Adversarial Neural Network (GANs)

Generative Adversarial Neural Networks (GANs) (Goodfellow et al., 2014) have attracted a great deal of attention from the research community since their introduction in 2014 and much of their activities are focused on image generation (Reed et al., 2016). They are classes of unsupervised deep learning frameworks with the ability to learn an unknown probability distribution of a given data set and can map the learnt distribution to generate synthetic data sets that follow the same distribution (Takahashi et al., 2019). GANs involve the pitching of two deep neural networks against each other. The first network known as the Generator is trained to generate new samples, while the second network known as the discriminator is fed with either the training data sample or the synthetic data sample produced by the Generator and aims to distinguish the true samples from the generated samples (Goodfellow et al., 2014). The two networks are adversarially trained in a zero-sum game where the Generator is trained to trick the discriminator whilst the discriminator is trained to maximise the classification accuracy. This adversarial training continues until Nash equilibrium is achieved; where the discriminator cannot distinguish the generated samples from the true samples. At this point, the Generator is able to capture the data distributions (Takahashi et al., 2019). Figure 2.2 demonstrates the basic architecture of the GAN and the minimax game that the two networks play.

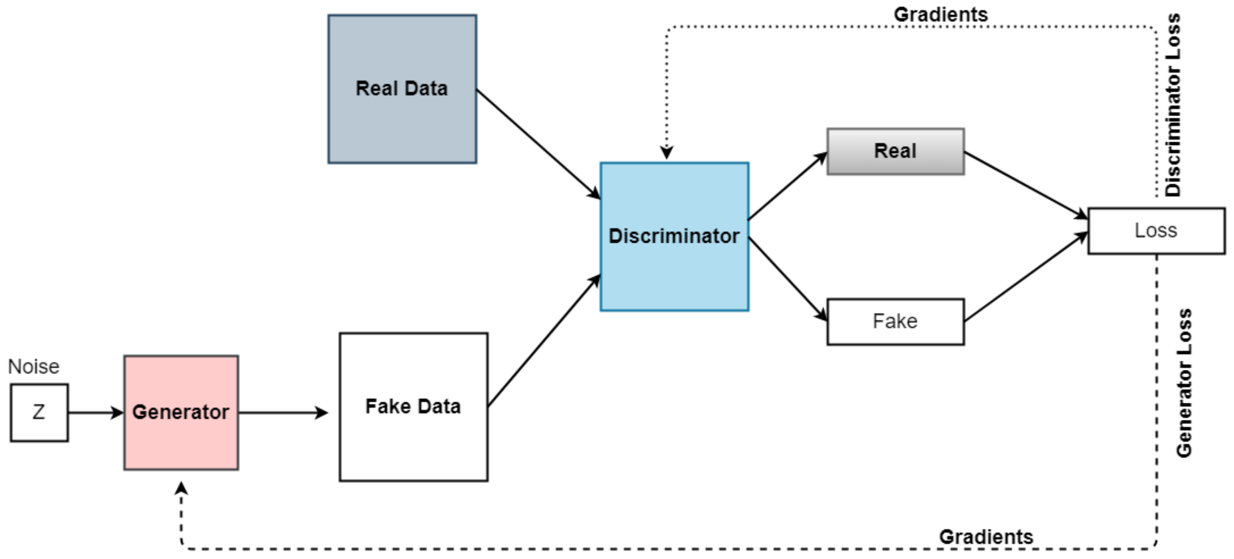


Figure 2.2: Generative Adversarial Network

Denote the training dataset $\mathcal{D} = (\boldsymbol{x}_n)_{n=1}^{N}$, the generative adversarial nets (GAN) aim to learn the Generator's distribution over the data $\boldsymbol{x}$. During the training of the GAN network, the Generator, G learns to map and transform a known probability distribution p_z into the distribution of the Generator p_g to resemble the distribution of the data, p_{data}. Using this adversarial network, both the generator and discriminator models can be trained using back-propagation and dropout algorithms (Hinton et al., 2012; Srivastava et al., 2014). The minimax game played by the GAN network is represented in equation (2.12):

$$\min_{\mathbf{g}} \max_{\boldsymbol{\theta}} \Lambda(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_{\boldsymbol{\theta}}(\boldsymbol{x})] + \mathbb{E}_{z \sim p_z(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\theta}}(G_{\mathbf{g}}(z)))] \qquad (2.12)$$

With GANs, the data are directly trained with a neural network, without making any assumptions about the distribution of the data when building the model. The learning is performed through the discriminator and Generator learning steps described as follows:

Discriminator Learning

This involves learning $\boldsymbol{\theta}$ given a Generator G. The aim is to estimate the parameters, $\boldsymbol{\theta}$ that maximise the classification of real and fake data. Gradient descent algorithms, such as Adam optimiser, are usually used to perform the discriminator learning and yield discriminator error function, $\bigtriangledown_{disc}$ on parameters $\boldsymbol{\theta}$:

$$
\begin{aligned}
&\bigtriangledown_{disc}(X_r, X_g) = \tilde{data}(X_r) + \tilde{Z}(X_g),\\
&\text{with :}\\
&\tilde{data}(X_r) = \frac{1}{b}\sum_{\boldsymbol{x}\in X_r} \log(D_{\boldsymbol{\theta}}(\boldsymbol{x}));\\
&\tilde{Z}(X_g) = \frac{1}{b}\sum_{\boldsymbol{x}\in X_g} \log(1 - D_{\boldsymbol{\theta}}(\boldsymbol{x})),
\end{aligned}
$$

where X_r represents a batch of real data randomly drawn from the training data set and X_g represents a batch of generated data from G. Unlike the discriminator learning step above, the Generator step performs only one per iteration. Through the iteration of the Generator and discriminator steps together for a significant number of times, the GAN ends up with a Generator parameter **g** that resembles that of the real data.

Generator learning

This involves adapting **g** to the new parameters $\boldsymbol{\theta}$. This process is carried out by a gradient descent of the error function $\bigtriangledown_{gen}$ in the Generator parameters **g** :

$$
\begin{aligned}
\bigtriangledown_{gen}(Z_g) &= \tilde{B}(\{G_{\boldsymbol{g}}(\boldsymbol{z}) | \boldsymbol{z} \in Z_g\})\\
&= \frac{1}{b}\sum_{\boldsymbol{x}\in\{G_{\boldsymbol{g}}(\boldsymbol{z})|\boldsymbol{z}\in Z_g\}} \log(1 - D_{\boldsymbol{\theta}}(\boldsymbol{x}))\\
&= \frac{1}{b}\sum_{\boldsymbol{z}\in Z_g} \log(1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{g}}(\boldsymbol{z})))
\end{aligned}
$$

where Z_g represents a sample of batch b of random noise data generated from $p_z(z)$

Time series GAN

Time Series GAN (TimeGAN) is a generative time-series model introduced by Yoon et al. (2019). It offers a unique approach to adversarially train and jointly learn an embedding space using both supervised and unsupervised losses. TimeGAN brings together concepts from auto-regressive models, GANs, and time-series representation learning (Yoon et al., 2019). TimeGAN consists of four network components which are; an embedding function, recovery function, Generator, and discriminator functions.

The embedding and recovery functions: Provide a mapping between the feature and latent space, which allows the Generator and discriminator functions to learn the underlying temporal dynamics of the data through low-dimensional representations. The latent dynamics of both real and synthetic data are synchronised through supervised loss.

Denote the training data set, $\mathcal{D} = (\mathbf{S}_n, \mathbf{X}_{n,1:T_n})_{n=1}^{N}$, $\mathbf{S}_n \in \mathcal{S}$, with $\mathcal{S}$ being a vector space of static features, $\mathbf{X}_n \in \mathcal{X}$, with $\mathcal{X}$ being a vector space of temporary features. Let H_S, H_X denote the latent vector spaces corresponding to the feature spaces S, X. The embedding function, e implemented as a recurrent network, takes static and temporal features to their latent codes $\mathbf{h}_S, \mathbf{h}_{1:T} = e(\mathbf{S}, \mathbf{X}_{1:T})$ as shown in Equation (2.13):

$$\mathbf{h}_{\mathcal{S}} = e_S\left(S\right), \qquad \mathbf{h}_t = e_X\left(\mathbf{h}_{\mathcal{S}}, \mathbf{h}_{t-1}, \mathbf{X}_\mathbf{t}\right) \tag{2.13}$$

where e_S represents an embedding network for static features and e_X is a recurrent embedding network for temporal features. The opposite direction has a recovery function, r, implemented through a feed-forward network at each time step that takes static and temporal codes back to their feature representations $\tilde{\mathbf{S}}, \tilde{\mathbf{X}}_{1:T} = r(\mathbf{h}_s, \mathbf{h}_{1:T})$ as demonstrated in equation (2.14):

$$\tilde{\mathbf{S}} = r_{\mathcal{S}}(\mathbf{h}_s), \qquad \tilde{\mathbf{X}}_t = r_{\mathcal{X}}\left(\mathbf{h}_t\right) \tag{2.14}$$

where $r_{\mathcal{S}}$ and $r_{\mathcal{X}}$ are recovery networks for static and temporal embeddings respectively. The reconstruction loss $\mathcal{L}_R$ involved in this procedure is calculated and mathematically represented as the equation (2.15):

$$\mathcal{L}_R = \mathbb{E}_{S, X_{1:T \sim p}}\left[\parallel \mathbf{S} - \tilde{\mathbf{S}} \parallel_2 + \sum_t \parallel \mathbf{X_t} - \tilde{\mathbf{X}}_\mathbf{t} \parallel_2\right] \tag{2.15}$$

The Sequence Generator and discriminator functions: Rather than producing synthetic output directly in the feature space, the Generator first outputs into the embedding space. Denoting the vector spaces over which known distributions are defined as $\mathbf{z}_s, \mathbf{z}_x$, and from which random vectors are drawn, the Generator function, G takes a tuple of static and temporal random vectors to synthetic latent codes $\hat{\mathbf{h}}_S, \hat{\mathbf{h}}_{1:T} = G(\mathbf{z}_s, \mathbf{z}_{1:T})$

Mathematically, the Generator is implemented as shown in equation (2.16):

$$\hat{\mathbf{h}}_{\mathcal{S}} = G_{\mathcal{S}}\left(\mathbf{z}_{\mathcal{S}}\right), \qquad \hat{\mathbf{h}}_t = G_{\mathcal{X}}\left(\hat{\mathbf{h}}_{\mathcal{S}}, \hat{\mathbf{h}}_{t-1}, \mathbf{z}_t\right) \tag{2.16}$$

where $G_{\mathcal{S}} : \mathcal{Z}_{\mathcal{S}} \to \mathcal{H}_{\mathcal{S}}$ is a Generator network for static features and $\mathcal{G}_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times \mathcal{Z}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{X}}$ represents a recurrent Generator for temporal features. The random vector $\mathbf{z}_{\mathcal{S}}$, can be sampled from any distribution of choice, while the $\mathbf{z}_t$, which follows a stochastic process, can use the Gaussian distribution and the Wiener process.

The discriminator function, $D : \mathcal{H}_{\mathcal{S}1} \times \prod_{t_1} \mathcal{H}_{\mathcal{X}1} \rightarrow [0,1]_1 \times \prod_{t_1} [0,1]_1$ which also operates from the embedding space, receives the static and temporal codes and returns the classifications, $\tilde{y}_{\mathcal{S}}, \tilde{y}_{1:T} = D(\tilde{\boldsymbol{h}}_{\mathcal{S}}, \tilde{\boldsymbol{h}}_{1:T})$ as:

$$\tilde{y}_{\mathcal{S}_1} = D_{\mathcal{S}}\left(\tilde{\mathbf{h}}_{\mathcal{S}}\right), \qquad \tilde{y}_t = D_{\mathcal{X}}\left(\overleftarrow{\mathbf{u}}_t, \overrightarrow{\mathbf{u}}_t\right) \tag{2.17}$$

where $\overrightarrow{\boldsymbol{u}}_t = \overrightarrow{c}_{\mathcal{X}}(\tilde{\boldsymbol{h}}_{\mathcal{S}}, \tilde{\boldsymbol{h}}_t, \overrightarrow{\boldsymbol{u}}_{t-1})$ and $\overleftarrow{\mathbf{u}}_t = \overleftarrow{c}_{\mathcal{X}}(\tilde{\boldsymbol{h}}_{\mathcal{S}}, \tilde{\boldsymbol{h}}_t, \overleftarrow{\mathbf{u}}_{t+1})$ respectively represents the sequences of forward and backwards hidden states using the recurrent functions and $D_{\mathcal{S}}, D_{\mathcal{X}}$ denotes the output layer classification functions.

Gradients are calculated on the unsupervised loss $\mathcal{L}_U$ to allow maximisation of the likelihood of providing a correct classification for the discriminator or minimisation of the likelihood of providing a correct classification for the Generator. The unsupervised loss $\mathcal{L}_U$, is calculated as :

$$\mathcal{L}_U = \mathbb{E}_{S,X_{1:T\sim p}}\left[\log y_s + \sum_t \log y_t\right] + \mathbb{E}_{S,X_{1:T\sim\hat{p}}}\left[\log(1-\hat{y}_s) + \sum_t \log(1-\hat{y}_t)\right] \tag{2.18}$$

The Optimisation process: Let $\theta_e, \theta_r, \theta_g, \theta_d$ denote the parameters of the Encoder, Decoder, Generator, and discriminator networks, respectively. The θ_e and θ_r are jointly used to train on the reconstruction loss and the supervised loss as shown in:

$$\min_{\theta_e,\theta_r}(\lambda\mathcal{L}_S + \mathcal{L}_R)$$

The Generator and the discriminator networks uses equation (2.19) to adversarially train each other

$$\min_{\theta_g}\left(\gamma\,\mathcal{L}_S + \max_{\theta_d}(\mathcal{L}_U)\right) \tag{2.19}$$

where $\lambda, \gamma \geq 0$ are regularisation parameters that balance these losses.

In conclusion, TimeGAN's unique approach leverages the power of LSTM-based generative modelling and GANs to capture and replicate the underlying temporal dynamics of time-series data, making it a promising tool for time-series forecasting.

2.5 Continuous Wavelet Transform (CWT)

Wavelet transforms are mathematical techniques used to analyse data with features that vary across different scales. They can be used to de-noise data since it has the ability to handle non-stationary financial time series data (Ramsey, 1999). The main feature of the wavelet transform is its ability to analyse the frequency and time components of financial time series simultaneously, unlike the Fourier transform which lacks that ability (Lee et al., 2019). Fourier analysis involves breaking down a signal into sine waves of particular frequencies, whereas wavelet analysis involves the division of signals into shifted and scaled versions of a wavelet. A wavelet is a wavelike oscillation that quickly fades away, unlike a sine wave from the Fourier transform (Misiti, 2000). Compared to the Fourier transform, wavelet analysis is a relatively new technique in signal processing (Cohen et al., 1993), and is made up of mathematical functions that decompose data into different frequency components, after which each component is studied with a resolution that corresponds to its scale, where a scale denotes a time steps (Ramsey and Zhang, 1997). One main advantage of wavelets is their ability to analyse a localised area of a larger signal (Misiti, 2000). The different types of mother wavelets for both discrete and continuous wavelets are represented in Figure 2.3. In the first row are the mother wavelets for discrete wavelets, and in the second row are several mother wavelets for continuous wavelets, including the Morlet wavelet shown in red.

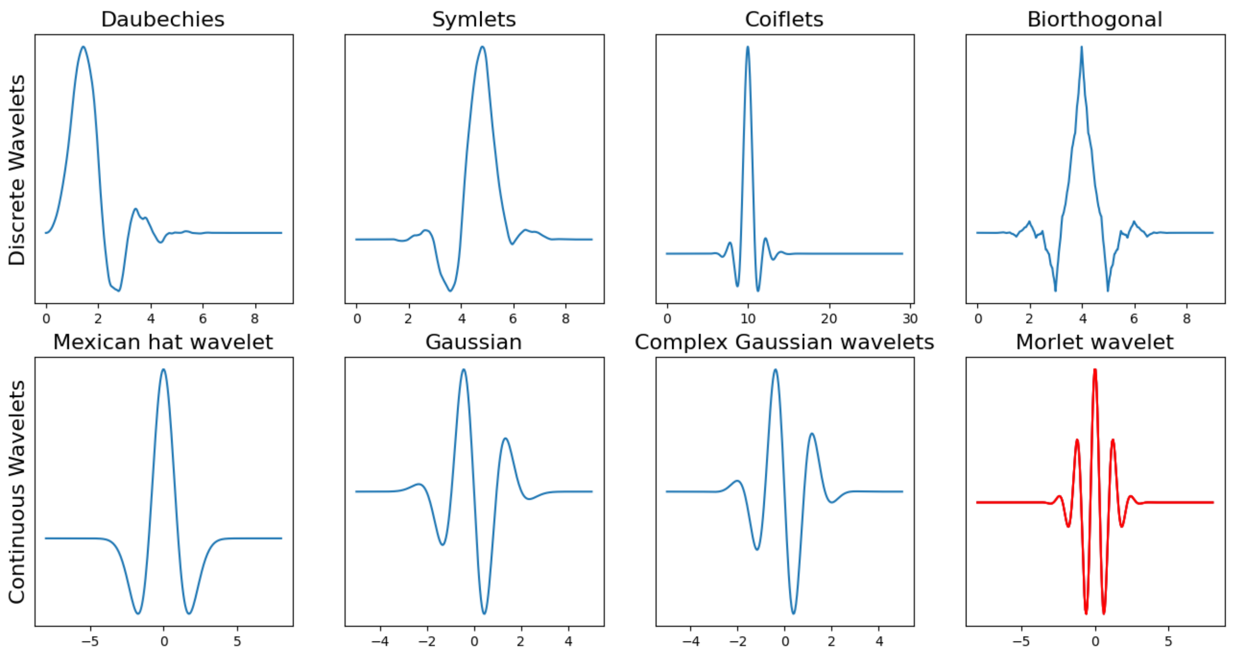


Figure 2.3: Various families of mother Wavelets available. In the first row are the mother wavelets for discrete wavelets and in the second row are several mother wavelets for continuous wavelets including the Morlet wavelet shown in red

The continuous wavelet transform (CWT) (Luo et al., 2018), which is a form of the wavelet transform, is a time-frequency transformation that is suitable for analysing non-stationary signals. The CWT is comparable to the short-time Fourier transform (STFT), but the

STFT uses a fixed window to generate a local frequency analysis, while the CWT uses variable windows to cover the time-frequency plane (Gencay et al., 2001). The continuous wavelet transform is the sum over all time of the signal multiplied by scaled and shifted versions of the wavelet (Misiti, 2000). This results in wavelet coefficients, C, which are a function of scale and position. The wavelet function of a continuous wavelet transform (CWT) can be mathematically defined as shown in Equation (2.20) :

$$C(\phi, t) = \phi^{-\frac{1}{2}} \int_{-\infty}^{\infty} v(x)\psi\left(\frac{x-t}{\phi}\right) dx \tag{2.20}$$

$$\text{with,} \qquad \psi(t) = \exp^{(-\frac{t^2}{2})} \cos(5t)$$

where ϕ represents the scaling factor, C represents the wavelet coefficients, t is the translation factor, $v(x)$ represents the recent Volatility input values and ψ represents the Morlet mother wavelet.

The wavelets undergo two operations, which are shifting and scaling. Shifting refers to the delay of the onset of the wavelet, while scaling refers to the stretching or compression of the wavelet(Misiti, 2000). Smaller scales show a more compressed wavelet (Subbey et al., 2007). Thus, wavelet transforms convert the original time series data into frequencies, which are linked to the specific times when these frequencies occur, using wavelet operations. The wavelet transform decomposes a time series signal into different scales, helping to distinguish seasonality, reveal structural breaks and volatility clusters, and identify local and global dynamic properties of a time series signal at specific time scales (Gencay et al., 2001). Consequently, the wavelet is classified as a useful tool for handling highly irregular financial time series (Popoola and Ahmad, 2006).

The relation between the scale factor and the frequencies is shown as:

$$f_\phi = \frac{f_c}{\phi}$$

where f_ϕ is the frequency, f_c is the central frequency of the Mother wavelet. A higher scale factor (longer wavelet) corresponds to a smaller frequency, so by scaling the wavelet in the time domain, smaller frequencies can be analysed (resulting in a higher resolution) in the frequency domain. By using a smaller scale, there is more detail in the time domain.

Reconstruction loss on the continuous Wavelet transform

The inverse wavelet transform can be used to recover the original volatility signal, v using equation (2.21) . The reconstructed volatility time series was the sum of the real part of the wavelet transform over all the scales and is given in equation (2.21):

$$v(t) = \int_{-\infty}^{\infty} \int_{0}^{\infty} \mathcal{C}(v)(\phi, x)\phi^{\frac{-5}{2}} \psi \left(\frac{t-x}{\phi} \right) dx d\phi \tag{2.21}$$

(Luo et al., 2018)

2.6 Problem Formulation of cwt-TriGAN

For a given volatility data, let $\mathcal{C}$ be the vector space of the coefficients obtained after applying CWT on volatility data, with $\mathbf{C} \in \mathcal{C}$ representing random vectors that can be initiated with specific $\mathbf{c}$ values. Let p represent the distribution of $\mathbf{C}_T$. During data training, individual samples can be indexed by $n \in \{1, ..., N\}$. Re-shape the CWT features into 2D features of $T_s \times M$ size, where M is the total number of scales and T_s is the number of time steps. Therefore, the training data can be denoted by $\mathcal{V} = \{\mathbf{c}_{n,T_n}\}_{n=1}^{N}$. Since the use of the standard GAN framework results in optimisation difficulties, cwt-TriGAN can be used to learn the density distribution, $\hat{p}(\mathbf{C}_T)$ of training data $\mathcal{V}$ that best approximates $p(\mathbf{C}_T)$. This can then be used together with an auto-regressive decomposition of $p(\mathbf{C}_T) = p(\mathbf{C}_t|\mathbf{C}_{t-1})$ to specifically focus on the conditional objectives of learning a distribution density $\hat{p}(\mathbf{C}_t|\mathbf{C}_{t-1})$ that best approximates $p(\mathbf{C}_t|\mathbf{C}_{t-1})$ at any time t.

cwt-TriGAN Objective functions

The objective function refers to measuring the difference between the coefficients of the volatility data distribution and the generated coefficients of the volatility distribution. The two training objectives are global and local objectives. These separate the sequence-level objective into a series of step-wise objectives. The first objective representing the global objective is represented by Equation (2.22):

$$\min_{\hat{p}} \Lambda \left(p(\mathbf{C}_T) \parallel \hat{p}(\mathbf{C}_T) \right) \tag{2.22}$$

The second objective representing the local objective is represented by equation (2.23):

$$\min_{\hat{p}} \Lambda(p(\mathbf{C}_t|\mathbf{C}_{t-1}) \parallel \hat{p}((\mathbf{C}_t|\mathbf{C}_{t-1})) \tag{2.23}$$

where Λ is the measure of distance between the distributions.

Under an ideal cwt-TriGAN, equation (2.22) uses Jensen-Shannon divergence (Menendez et al., 1997) to minimise the loss function, while equation (2.23) which is the local objective uses Kullback-Leibler divergence (Van Erven and Harremos, 2014) through maximum likelihood (ML) to minimise the loss function. Here, the original coefficients of the volatility data serve as the supervisor in the maximum likelihood training. Therefore, the main objective is the combination of the GAN objective in equation (2.22) and the ML objective in equation (2.23). This procedure leads to a training process that involves the addition of a supervised loss to guide adversarial learning of the cwt-TriGAN.

The next section covers the fundamentals of reinforcement learning.

2.7 Fundamentals of Reinforcement Learning

This section covers the fundamentals of reinforcement learning. Reinforcement learning is a machine learning paradigm that models a sequence of decisions over state space. As stated by Dixon et al. (2020), the agent in reinforcement learning learns more about the environment in order to perform better on the tasks assigned to it. These tasks can be mapped as the problem of conducting an optimal action space. A reinforcement learning programme can be modelled as a Markov Decision Process (MDP), which is an optimal control problem and involves choosing action space over some period of time with the aim of maximising some objective function, which invariably depends on both the future states and actions taken.

Policy Gradients which are a special class of reinforcement learning are used to estimate the gradient of the function approximator. All reinforcement learning agents possess explicit goals; have a high awareness of their environment, and have the ability to select actions that influence their environments. Among all the available machine learning paradigms, reinforcement learning is the closest to the type of learning that humans and other animals do, and the fundamental algorithms used in reinforcement learning were originally inspired by biological learning systems.

Figure 2.4 shows a typical framework for RL operations. It is made up of an agent, which is the learner and the decision-maker, and the environment with which it interacts. The interaction of the agent with the environment is a continuous process, where the agent selects actions and the environment responds to these actions and presents a new state to the agent. As highlighted above, the environment also produces rewards that the agent tries to maximise over time through its observations and choice of actions. In Figure 2.4, it can be seen that the interaction between the agent and the environment occurs at each sequence of time steps, $t = 0, 1, 2, 3...T$. At each time step, the Environment state $S_t \in \mathcal{S}$ sends some of its representation to the agent, where the agent selects an action, $A_t \in \mathcal{A}(s)$. At time step $t + 1$, the Agent obtains a **reward** , $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ as a result of its chosen action and transitions to a new state S_{t+1}. The Markov Decision Process and the agent give rise to a sequence such as $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ..., S_T, A_T, R_{T+1}$ (Silver, 2015)

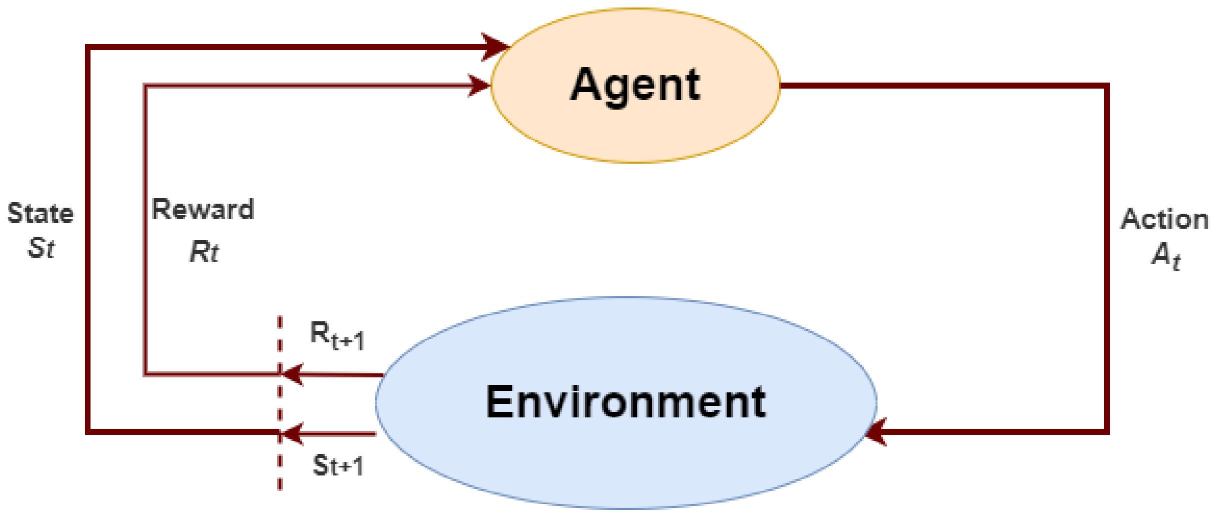


Figure 2.4: Basic design of Reinforcement learning showing the Agent-Environment interaction (Sutton and Barto, 2018)

In addition to an agent and the environment, an RL agent may include one or more of these components:

- **Policy function:** A policy π defines the behaviour of an agent and it is defined as the distribution over actions given states (Silver, 2015). This contains an agent's learning behaviour function at a given time t. It serves as a mapping from perceived states of the environment to actions taken when in those states. The policy can be a simple function or look-up table in some cases, while in others too, it can involve complex computations. Generally, policies may be stochastic or deterministic and serve as the core of reinforcement learning agents, as only they can determine behaviour (Sutton and Barto, 2018). It depends on the current state and not the historical states, since they are time-independent and stationary. Mathematically, it is represented in equation 2.24:

$$\pi(a|s) = P = [A_t = a]|S_t = s] \tag{2.24}$$

- **Reward:** The goal of a reinforcement learning problem is defined by the reward. At each time step, the environment sends a scalar in the form of a number called *reward* is sent to the reinforcement learning agent by the environment. The main objective of the agent is to maximise the total reward received in the long term. Therefore, the reward signal defines the positive and negative events of the agent. In a biological system, this can be loosely equated with experiences of pleasure or pain. The reward signal serves as the primary basis on which the policy is altered. If an action selected by the policy results in a low reward, then there could be changes in the policy to select other actions for the same situation in the future (Sutton and Barto, 2018).

- **Value function :** This demonstrates how good each state and action are in the long term compared to a reward signal that indicates what is good in the short or immediate term. The value of a state can be said to be analogous to the total amount of reward an agent can expect to have in the future. There could be no values without rewards, and the only reason behind the estimation of values is to achieve more rewards. For value-based RL agents, values are the most important element for evaluation and decision-making. The choice of actions taken at each time step is based on value judgements, since actions that result in states of the highest value are sought rather than the highest reward. Unfortunately, it is easier to determine rewards as compared to values because rewards can be directly obtained from the environment, but values must be estimated and re-estimated from the series of observations made by an agent over its whole lifetime.

- **Model:** This only occurs in model-based methods to solve reinforcement learning. This shows the agent's representation of the environment and mimics the behaviour of the environment. As an example, given a state and action, a model could predict the next state and the next level. Models are used for planning, whereby possible future situations are considered before they actually occur. There is a model-free method that explicitly uses try-and-error learners without planning, unlike model-based methods (Sutton and Barto, 2018).

Markov decision process(MDPs)

Markov decision process describes the environment used for reinforcement learning(Sutton and Barto, 2018). There could be a fully-observable environment or a partially-observable environment. Most RL problems can be formalised as MDPs. Bandits are a special case of MDPs with only one state. The central idea behind an MDP is the Markov property, which states that the future is independent of the past given the present (Silver, 2015). All the relevant information about history is captured by the state, and once the state is known, the history might be thrown away. A state S_t is said to have the Markov property only if equation (2.25) is observed:

$$P\left[S_{t+1}|S_t\right] = P\left[S_{t+1}|S_1, ..., S_t\right] \tag{2.25}$$

Bellman Equation for Markov Reward Process

The value function, $v(s) = E[G_t|S_t = s]$, shows the Expected returns ($E[G_t)$, given that it starts at state s. It can be divided into two parts, which are:

- Immediate reward, R_{t+1}
- Discounted value of the next state $\gamma v\left(S_{t+1}\right)$

This is mathematically represented in equation 2.26:

$$\begin{aligned} v(s) &= E[G_t|S_t = s] \\ &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ...|S_t = s] \\ &= E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + ...)|S_t = s] \\ &= E[R_{t+1} + \gamma G_{t+1} + ...|S_t = s] \\ &= E[R_{t+1} + \gamma v(S_{t+1})|S_t = s] \end{aligned} \tag{2.26}$$

A Markov decision process has been described as a Markov reward process with decisions and is a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\ )$, where:

- $\mathcal{S}$ represents a finite set of states
- $\mathcal{A}$ represents the final set of actions undertaken by the Agent
- $\mathcal{P}$ represents a state transition probability matrix, $\mathcal{P}^a_{ss'} = P[S_{t+1} = s'|S_t = s, A_t = a$
- $\mathcal{R}$ represents the reward function, $\mathcal{R}^a_s = E[R_{t+1}|S_t = s, A_t = a$ γ represents the discount factor where $\gamma \in [0, 1]$

Value functions

The expected return that starts from state s which follows policy, π is known as the *state-value function* $v_\pi(s)$ (Silver, 2015). it is mathematically expressed as shown in equation 2.27:

$$v_\pi(s) = E_\pi[G_t|S_t = s] \tag{2.27}$$

The *action-value function* $q_\pi(s,a)$ is defined as the expected return that starts from the state s, takes action a and follows a policy π. It can be mathematically defined as equation 2.28:

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a] \tag{2.28}$$

Optimal Value Function

The optimal value function sets out the best possible performance in a Markov decision process, and an MDP is deemed to be solved when the optimal value function is known (Silver, 2015). The maximum value function overall policies are defined as the optimal state-value function $v_*(s)$ and it is mathematically expressed as equation 2.29:

$$v_*(s) = \max_\pi v_\pi(S) \tag{2.29}$$

The optimal action-value function $q_*(s,a)$ on the other hand is defined as the maximum action-value function over all policies. It can be mathematically defined in equation 2.30 as:

$$q_*(s,a) = \max_\pi q_\pi(S,a) \tag{2.30}$$

Optimal Policy

As proposed by Silver (2015), defining a partial order over policies is shown in equation (2.31) as:

$$\pi \geq \pi' \rightarrow v_\pi(S), \forall S \tag{2.31}$$

There exists an optimal policy π_* which is better than or equal to all other policies, for any Markov Decision process, $\pi_* \geq \pi, \pi\forall$. Also, all optimal policies achieve the optimal value function, $v_{\pi*}(S) = v_*(S)$. Finally, all optimal policies achieve the optimal action-value function $q_{\pi*}(S,a) = q_*(S,a)$.

2.8 summary

In summary, this chapter has covered some machine learning concepts and fundamentals as preliminaries to the proposed solutions for the problems of credit risk modelling, volatility forecasting, and portfolio optimisation to be detailed in this thesis. It initially introduced the relevant concepts of logistic regression and density estimation using the Gaussian Mixture Model (GMM) to be used in credit risk modelling. Deep Learning for Time-series forecasting and reinforcement learning has been introduced as technical background for the problems of volatility forecasting and portfolio optimization. A limitation of TimeGAN that has been identified is its sensitivity to the quality of input data, which makes its performance to be affected by noisy and poor data quality.

In the chapter that follows, the basic financial background knowledge underpinning credit risk modelling, volatility forecasting and portfolio optimisation is presented.

Chapter 3

Background of Financial Risk Management

3.1 Introduction

This chapter serves as a comprehensive review of the foundational concepts of financial risk management, with a specific focus on key aspects of finance that are later explored through machine learning algorithms in the subsequent chapters of this thesis. The central theme of this chapter revolves around credit risk modelling, volatility forecasting, and portfolio optimisation, forming the core pillars on which this thesis is built. The remainder of this chapter is organised as follows. Section 3.3 introduces the fundamental principles of credit risk modelling, offering insights into its practical applications. Section 3.4 introduces the fundamentals of Market Risk and Volatility forecasting. Finally, Section 3.5 delves into essential financial concepts used in portfolio optimisation.

3.2 Financial risk management

Financial risk management involves the identification, assessment and mitigation of potential risks that may impact an organisation's financial performance (Crouhy et al., 2014b). These risks include credit risk, market risk, liquidity risk, operational risk, and many others (Apostolik et al., 2009). During a severe crisis, such as the 2008-2009 financial crisis, these risks can flow from credit risk to liquidity risk and to market risk. Various risks are faced by financial institutions (Ghenimi et al., 2017). Effective management of these risks is the key to bank performance and survival. Financial institutions actively utilise financial risk management techniques to detect, manage and measure these risks (Apostolik et al., 2009).

Credit risk is defined as the risk that a borrower defaults on a loan or other related financial obligation and is the largest risk faced by financial institutions (Siddiqi, 2017; Borsuk, 2023). However, market risk can be defined as the risk of losses due to movements in market prices such as stock prices, interest rates, exchange rates, and commodity prices(Jorion, 2007). It includes interest rate risk, equity risk, foreign exchange risk, and commodity risk. Therefore, the movement in prices known as volatility is an important concept for research.

3.3 Credit Risk Modelling

This section briefly introduces the fundamentals of credit risk modelling and how it is applied in practice. The probability that a customer will default can be estimated using application scoring at the time of the application for the credit facility. A scorecard is made of a group of characteristics statistically determined to be predictive of the separation of good loans from bad ones (Siddiqi, 2017). Alternatively, behavioural scoring is done by observing recent payment and purchase behaviour of customers who have been granted credit facilities to predict the probability of default in 12 months or other fixed time window (Kennedy et al., 2013). This is used to make lending decisions on current customers, such as increasing or decreasing credit limits, offering new financial products, or new interest rates according to the customer's behaviour. Credit risk has also been known to affect the portfolio of banks, posing a liquidity risk to their operations (Ghenimi et al., 2017). It is argued that this can eventually have a negative effect on both financial industries and economies (Yu et al., 2018). Credit risk models can be grouped into corporate models and consumer models (Zamore et al., 2018). Although both models share the same fundamental assumptions, the majority of available literature tends to focus on corporate models.

For corporate and institutional credit risk modelling, two major models that are frequently used are structural firm value models and reduced-form intensity models (Yu et al., 2018). Structural models based on capital structure theory (Modigliani and Miller, 1958), can be traced back to the works of Black and Scholes (1973) and Merton (1974). The main idea involves modelling the default of a company by using its asset value and adopting a geometric Brownian motion to describe the value of the asset (Yu et al., 2018). It assumes the occurrence of default when the asset value of a firm falls below its total debt value.

The reduced-form intensity-based model was pioneered by Jarrow and Turnbull (1995); Madan and Unal (1998). Unlike structural models, reduced-form models have the ability to determine the probability of default without making any assumptions about the source of the credit risk premium (Benzschawel, 2012). They are based on the risk-neutral pricing theory, which states that the market value of a risky asset equals the discounted present value of future cash flows using the risk-free rate.

For retail and consumer credit decision, various models have been implemented to estimate the Probability of Default of customers.

Credit Risks

Failure to manage credit risks adequately could exceed direct accounting losses (Yu et al., 2018). This is because it entails transaction costs, opportunity costs, and expenses associated with non-performing assets over and above the accounting loss. The main objective of credit scoring (Thomas, 2009) is the development of a credit system that can rank customers in terms of their relative default risks in such a way that customers above some cutoff score are classified as less risky than those below. These models can be classified into application scoring and behavioural scoring, with the main objective of classifying whether a customer will default or not (Thomas, 2009; Siddiqi, 2017). In today's banking and financial services activities, credit risk management plays a pivotal and significant role, since it involves the practice of reducing losses through the understanding of the financial institution's bank reserve and loan loss reserves.

The reduction of losses suffered by financial institutions plays a crucial role in ensuring their business continuity. The profitability of a loan is determined by several factors, such as the creditworthiness of a borrower, the quality of the collateral, and the regulatory regime in which the financial institution finds itself in (Engelmann and Pham, 2020). Regulators and accounting boards around the world have undertaken several initiatives aimed at stabilising the financial system (Engelmann and Pham, 2020). In 1998, the Basel Committee on Banking Supervision (BCBS) introduced the Basel I Accord to implement the minimum regulatory capital requirements for financial institutions (BCBS, 1988). The main purpose of this was to ensure that banks were solvent without any threat of bankruptcy. This regulation allowed banks to swap low-risk assets with high-risk assets, as loans of different risk levels were deemed to have the same value (BCBS, 1988). However, this arrangement under the Basel I Accord led to the occurrence of regulatory arbitrage (Zhu, 2008). To overcome this loophole, Basel II was introduced in 2006. This allowed banks to calculate capital requirements to manage credit risks using a standardised approach, the Foundation

Internal Ratings-Based Approach (FIRB) and the Internal Ratings-Based Approach (IRB) (Yu et al., 2018).

In response to the global financial crisis in 2008, and after a series of consultations, BCBS proposed Basel III in December 2009 (BCBS, 2020). Basel III, which is an extension of the Basel II framework, introduced new capital and liquidity standards required to strengthen banking and financial sector regulation and supervision. This led to an improvement in the quality of banks' capital and provided banking regulators with more flexibility in setting capital levels for individual financial institutions (BCBS, 2020). The Basel III framework is based on three pillars which are:

- Pillar 1: This represents the minimum regulatory capital requirements that financial institutions should have. This requirement results in the reduction of the risk appetite of financial institutions since the holding of riskier assets would increase the Pillar 1 capital requirement for that institution.
- Pillar 2: This is the supervisory review and evaluation process that allows banking regulators to evaluate a bank's risk profile through the analysis of the institution's business model, governance structure, risks to capital, and liquidity. From this perspective, this pillar allows regulators to focus on the assessment of risks not covered or not fully covered by Pillar 1. The additional capital requirements will be translated into additional Pillar 2 capital, which the institution is expected to hold in addition to its Pillar 1 capital. If the institution's provision coverage is considered inadequate by the supervisors, an additional capital addition could be placed to ensure the coverage is within the supervisory risk tolerance limits.
- Pillar 3: This represents the requirement of market discipline and allows financial institutions to ensure transparent reporting that enables capital markets to serve as a complementary force to regulate bank behaviour.

As an additional measure in response to the 2008 financial crisis, the accounting boards; International Accounting Standards Board (IASB) and Financial Accounting Standards Board (FASB), respectively, introduced new standards on loan loss provisions for impaired loans, which are now known as International Financial Reporting Standard, IFRS 9 (IASB, 2015) and Current Expected Credit Losses, CECL (FASB, 2016). This was to improve the accounting and reporting of financial assets and liabilities after the financial crisis with the expectation that the earlier recognition of loan losses would enhance financial stability. The profitability and volume of loans that a bank can originate were directly impacted by these reforms. Credit risk management is understood to be a process that starts from the regulatory level to the strategic level of the bank, then continues to the operational level (Win, 2018). The above statement shows that each level involves several decision-making processes, such as risk-return trade-off evaluation and optimisation of stakeholders' targets.

Regulatory Level

The regulatory level consists of the decision-making process of the financial regulatory bodies that set a regulatory framework and guidelines that each financial institution must follow. The main objectives of managing credit risk at this level by the Financial Regulators are: (i) to protect depositors who lend to banks in the form of deposits; (ii) to reduce systematic risks that could cause financial institutions' failures; and (iii) to protect banking confidentiality. As part of the activities at the regulatory level, financial institutions are obliged to reserve a certain level of capital to absorb credit losses. For financial institutions to use the internal-based approach to calculate their risk parameters, approval from regulatory authorities is required. Banking is one of the most regulated industries in the world (Zhu, 2008). Hence, at this level, the regulation of a bank's capital is crucial due to the important role it plays in banks' soundness and risk-taking behaviour and its influence on the competitiveness of banks. Financial institutions must comply with these regulations, as they are legally required to provide an estimate of the components of their credit losses by quantifying these risk parameters. The credit risk of all financial assets is influenced by these three parameters, which are Probability of Default P_d, Loss Given Default L_{gd} and Exposure at default E_{ad} (Caouette et al., 2008).

The Loss Given Default(L_{gd}) is represented mathematically by equation (3.1):

$$L_{gd} = 1 - R_r. \tag{3.1}$$

It is represented as a percentage of total exposure when the account falls into arrears. The R_r is the Recovery Rate which is conditional on the default occurrence and refers to how much is expected to be recovered by the lender from the debtor through reconstruction, renegotiation, or forced sales of assets. The Exposure At Default, E_{ad}, is a critical component in credit risk assessment and used for the calculation of the potential loss a financial institution might incur from a loan or other line of credit. The Expected Loss, E_l, is the expectation of the loss variable represented mathematically in Equation (3.2)

$$E_l = P_d \times E_{ad} \times L_{gd} = P_d \times E_{ad} \times (1 - R_r) \tag{3.2}$$

The expected losses must be covered by the profits from the financial institution and are taken into account when allocating the interest rate to each customer.A simple measure of unexpected losses is the standard deviation of the loss variable shown in equation (3.3).

$$\mathcal{U} = \varepsilon - E_l \tag{3.3}$$

Where $\mathcal{U}$ is the unexpected losses and ε is the Extreme Possible Losses. The unexpected losses that might arise are what is of interest to banking regulators; however, this does not seem to be the best approach for measuring unexpected losses, especially in times of economic crisis. Therefore, financial institutions tend to use economic capital as an unexpected loss measure, since it covers the entire distribution of portfolio loss. It is the amount of capital that a business requires to ensure that its balance sheet remains solvent over a specified period. It is used in the estimation of the riskiness of a venture and in the comparison of the opportunity cost of loan opportunities.

Strategic Level

Credit risk management at the strategic level involves monitoring and controlling the financial institution's overall strategic decision making such as; which type of new credit product needs to be introduced in the market, the total amount to be made available for lending, and other pre-defined loan-qualifying criteria. These decisions are made in such a way that they are in accordance with the regulatory-level framework and guidelines. These decisions will impact the total amount of risk faced by the financial institution. Basel II brought about the division of risks faced by financial institutions into credit risks, market risks, and operational risks (Thomas, 2009).

Operational Level

Credit Risk management at the operational level involves controlling the day-to-day credit decisions and operations of the financial institution. Operations at this level have little or no influence on the main decisions that affect the risk appetite of the financial institution. However, at this level, various risks, including credit risks, are identified and reported back to the strategic level for decisions to be made. Numerous empirical models have also been developed to assess credit risks posed by consumers. These consumer risk models typically involve a type of regression analysis or some form of probability model to evaluate the risk factors that contribute to default (Hassan et al., 2018).

Developments in Banking Supervision and Financial Reporting standards

The IFRS9 and CECL primarily introduced Loan Loss Provisioning(LLP), while the Basel Accords introduced minimum capital requirements with each of them being used as a tool to ensure improved loan performance.

Loan Loss Provisioning under IFRS9 and CECL

The IFRS9 recommends a three-stage process for loan provisioning. These are:

- Stage 1: represents Normal performing loans, where banks have to reserve one year of expected loss.
- Stage 2: represents loans with deteriorated credit quality; where banks have to reserve lifetime expected loss.
- Stage 3: Consists of Defaulted loans, where banks have to build a specific loan provisioning.

For stage 1, the financial institution should hold an equivalent of one year of expected credit losses, while for stages 2 and 3 the institution should hold the equivalent of lifetime expected credit losses. Therefore, the expected credit losses under IFRS9 are mathematically defined as shown in equation (3.4) as compared to that of the Basel Accord represented mathematically earlier in equation (3.2).

$$E_l = \sum_{i=1}^{m} mP_d \times E_{ad} \times L_{gd} = P_d \times E_{ad} \times (1 - R_r) \times d \tag{3.4}$$

Where m represents the time horizon for the expected loss calculation under IFRS9 and is given as 12 Months for stage 1 and a Lifetime for stage 2. At stage 3, the P_d will always be set to 100% since the exposures are already in default at this stage. d represents the discount factor required to discount the losses from the point of default in period i back to the reporting date. Detailed information and experiments on credit risk undertaken in this thesis have been provided in Chapter 4. The next section covers the fundamentals of Market Risk and Volatility forecasting.

3.4 Market Risk and Volatility

This section serves as background information for Chapter 5 which covers our proposed Continuous Wavelet Transform Triple Discriminator GAN (cwt-TriGAN) for Volatility Forecasting. Market risk and volatility are closely related concepts in financial risk management, with volatility often used as a proxy for market risk (Trenca et al., 2015). Market risk refers to the risk that an investment will lose value due to factors affecting the overall performance of the market. This type of risk is also known as systematic risk because it cannot be eliminated through diversification. Examples of market risks include changes in interest rates, inflation, geopolitical events, and economic downturns.

Volatility, on the other hand, is a measure of how much the price of an asset or security changes over time and shows the level of risk associated with the price changes of an asset (Magner et al., 2021). Volatility is an indicator of uncertainty and plays an important role in financial markets (Chen, 2022). High volatility is associated with market turbulence and high price fluctuation, while low volatility signifies calm markets (Anderson et al., 2009). In general, higher volatility indicates higher market risk. This is because assets that are more volatile are generally considered to be riskier, as they are subject to larger price swings and may be more likely to experience sudden and significant losses. Conversely, assets with lower volatility are generally considered to be less risky, as they exhibit less price volatility and are less likely to experience sudden and significant losses. Changes in market conditions can also lead to changes in volatility and market risk. For example, if there is a sudden increase in uncertainty or risk in the market, this may cause investors to become more risk-averse and sell off their investments, leading to increased volatility and market risk. Conversely, if market conditions are stable and there is low uncertainty, volatility and market risk may be lower. Investors and traders must take both market risk and volatility into account when making investment decisions. They must carefully analyse market conditions and assess the potential impact of different factors on the market as a whole and on individual securities. They must also use risk management strategies such as diversification and hedging to protect their portfolios from the effects of market risk and volatility. The modern portfolio theory is used by financial institutions for asset allocation. Therefore, volatility can be seen as a useful parameter used in finance applications such as investment decisions, financial risk management, regulatory policies, and capital budgeting.

The study of volatility is therefore of significant value in financial markets since it is primarily used to estimate market risk and also serves as a key parameter in the pricing of financial derivatives such as options. Volatility has been used in option pricing formulas derived from asset pricing models, such as the Black-Scholes model and other extensions derived from it. Reliable volatility estimates and forecasting play a crucial role in portfolio management and hedging against risk because it has been shown to be associated with risk and uncertainty (Poon and Granger, 2005). A simple risk measure has been used to estimate volatility in many asset pricing models (Xiao and Aydemir, 2007). Volatility fluctuations affect the price of almost every derivative security in the financial markets (Brownlees et al., 2011). Furthermore, the accuracy of volatility forecasts and estimates is vital in areas such as risk management for the calculation of metrics such as Value at Risk (VaR). This is because underestimating expected risk can expose investors to excessive market volatility (Brownlees et al., 2011).

The majority classes of volatility forecasting models that are widely used in modern practice to account for stylised facts of volatility are the following:

- Classical models including moving averages, auto-regressive models, conditional heteroskedastic models (Engle, 1982), and the implied volatility concept (Schlag et al., 2021).
- Machine learning models such as Support Vector Machines (SVMs) (Yang et al., 2020), Artificial Neural Networks (ANNs) (Ge et al., 2022), Recurrent Neural Networks (RNNs), Long Short-term Memories (LSTMs) and Generative Adversarial Networks (GANs). These have an important feature with the ability to capture non-linear behaviour.

Economic indicators have been used to forecast volatility (Bansal and Yaron, 2004; Corradi et al., 2013; Engle et al., 2013); however, the use of purely macroeconomic variables as opposed to financial variables does not act as accurate predictors of volatility (CHRISTIANSEN et al., 2012). Traditional econometric models, such as regression and ARIMA, generally assume that the variance in financial asset returns remains constant (Lin, 2018). The economic models were combined to achieve a marginal but significant predictability of stock volatility (Paye, 2012). Min et al. (2015) argued that econometric models require strict assumptions about the distributions of time series data and may not capture the complex and non-linear underlying patterns of financial data. The use of econometric models to predict the volatility of an asset is inconsistent and unreliable (Liu and Morley, 2009). This is because these models are unable to capture the asymmetries, cyclicality, time irreversibility, sudden bursts at irregular time intervals, and periods of low and high volatility that are observed properties of typical time series data such as volatility (Xiao and Aydemir, 2007).

Market Risk Estimation

Market risk defines the risk that an asset such as equity, fixed income, or a commodity will decline in value as a result of changes in market factors. Value-at-risk (VaR) is a key characteristic of market risk, and its use forms part of the regulatory requirements for a financial institution's capital under market risk. Regulators typically use a daily VaR with a higher confidence level.

Value-at-Risk (VaR) is the most widely used statistical method for quantifying market risk associated with bank portfolios. It is a probabilistic indicator that expresses the potential maximum loss of the market value of the portfolio, which may appear in a certain period of time, for a given confidence level (Trenca et al., 2015). The three different methods used in the calculation of VaR are the historical method, the parametric method, and the Monte Carlo method. The historical method quantifies the hypothetical value of the change of the current portfolio according to the historical fluctuations of the risk factors, using the empirical distribution of the past data and without any assumption related to the returns distributions. The parametric method implies that the daily returns follow a normal distribution. The main disadvantage of this method is the fact that it cannot estimate the important losses because many times the distributions have fat tails are characterised by a large number of unexpected events and do not follow a normal distribution. The Monte Carlo method implies generating scenarios for future prices based on the volatility and the correlations of the assets from the portfolio. After that, for each individual scenario, the portfolio value will be calculated and the final results of the simulation will be reported, whether as a portfolio distribution or as a particular risk measure. The Basel Committee recommends the VaR estimation using a confidence level of 99% and the use of an instantaneous shock of the price is equivalent to a 10-day fluctuation in the prices. VaR is calculated using the daily volatility of an asset under the model building approach (Hull, 2002)

Concept of Volatility Usage

Various stylised facts about the volatility of financial assets' returns have been documented in literature (Masset, 2011), making the study of volatility advantageous since it indicates the possibility of modelling its dynamics. Using modelling for quantitative forecasts can provide a valuable estimate of a future market value, even though some practitioners are of the view that financial instruments are unpredictable due to their stochastic nature (Kolte et al., 2023). Mathematical modelling can be used to detect the dependencies between the current values of financial indicators and their expected future values. The tendency of volatility to cluster and exhibit auto-correlation is an example of how volatility can be predicted as part of their documented stylised facts (Masset, 2011). These features provided the justification and foundation for formalising the concepts of volatility and the use of mathematical models for volatility forecasting.

Stylised facts about volatility

Various stylised facts about the volatility of financial asset returns have been documented in previous research papers (Masset, 2011). This makes the study of volatility advantageous since it confirms the possibility of modelling its dynamics. The stylised facts about volatility are captured as:

- **Volatility clustering**: Volatility is not constant and tends to cluster over time. This means that large movements (returns) are followed by further large movements, while small movements (returns) are followed by the corresponding small movements (Masset, 2011). This is an indication of volatility that shows persistence.
- **Fat tails:** Financial time series possess a fatter tail distribution than those of a normal distribution; this signifies their exhibition of excess kurtosis. Many financial time series have a standardised fourth movement above 3 compared to that of a normal distribution, which is 3 (Fama, 1965; Mandelbrot, 1963).
- **Leverage effects**: Price movements are negatively correlated with volatility (Black, 1976). For stock returns, the measured effect of stock price changes on volatility is too large to be explained solely by leverage effects. This means that the volatility of assets tends to increase when there is a drop in their price. Empirical evidence exists on leverage effects (Tauchen et al., 1996; Nelson, 1991; Sanusi, 2017)
- **Co-movements in Volatility**: The volatility of financial time series across different markets tends to move in a pattern; for example, exchange rate returns for two different currencies can be observed to have large movements in one currency matched with large movements in another (Masset, 2011). This emphasises the importance of multivariate models used in cross-correlations in different markets.

Volatility calculations

Volatility refers to the degree of spread of all the outcomes of a stochastic variable, such as the returns of a financial asset. It is associated with the sample standard deviation of returns, σ over a period of time, t calculated using the formula in equation (3.5).

$$\sigma = \sqrt{\frac{1}{t-1}\sum_{i=1}^{t}(r_t - \mu)^2} \tag{3.5}$$

The annualised volatility is obtained by multiplying the daily volatility by the square root of the number of trading days in a year of 252. Hence the Annualised volatility can be written as:

$$\sigma_A = \sigma \times \sqrt{252}$$

Where μ is the mean r_t is the daily returns of the asset price at a period of time t given in equation (3.6):

$$r_t = \frac{\ln(P_t)}{\ln(P_{t-1})} \tag{3.6}$$

where:
ln= natural log, P_t = The closing price of an asset at time t, P_{t-1} = The previous trading day closing price of an asset at time $t-1$

Forecasting volatility

In recent years, the field of volatility forecasting has received significant attention due to its pivotal role within financial markets (Xiao and Aydemir, 2007). This is because many asset-pricing models use volatility estimates as a simple risk measure. Volatility is also used in option pricing formulas derived from models such as the Black–Scholes model and its extensions. Reliable volatility estimates and forecasts are crucial to hedging against risk and for portfolio management (Xiao and Aydemir, 2007). Some researchers have recently observed some significant differences in terms of information content among volatility estimates calculated at various frequencies. There are the Autoregressive Moving Average (ARMA) models, Autoregressive Conditional Heteroskedasticity (ARCH) models, Stochastic Volatility (SV) models, regime-switching models, and threshold models (Xiao and Aydemir, 2007). The Auto-regressive Conditional Heteroscedastic model (ARCH) is used to model the variance of financial asset returns (Engle, 1982). The ARCH model has been found to undermine the universally accepted hypothesis of a linear relationship between risk and returns (Liu, 2019) and it also has the ability to capture volatility clustering (Xiao and Aydemir, 2007). A conditional variance was added to the lag phase of the ARCH model to form the Generalised Autoregressive Conditional Heteroskedasticity (GARCH) model (Bollerslev, 1986) and there have since been a number of variations to this, which are: Integrated GARCH (iGARCH), Exponential GARCH (EGARCH), GARCH-M and VGARCH (Lin, 2018).

Forecasting time steps

Changes in low-frequency volatility have been reported to have a greater impact on subsequent high-frequency volatility than the opposite. This is due to the heterogeneous nature of market participants, some of whom have short, medium, or long-term investment steps, but are all influenced by long-term moves on the markets (Dacorogna et al., 2001). There has been evidence that the intensity of the relationship between long and short time steps depends on the level of volatility at long steps (Gencay et al., 2010). When volatility at a long-term horizon is low, it usually leads to low volatility at short steps. However, the reverse is not always true and is referred to as an asymmetric vertical dependence property (Gencay et al., 2010). Volatility can be forecasted over different time steps, which could be 1-day volatility, 10-day volatility, 1-month volatility or even a bigger time horizon. In practice, daily volatility is usually used in the calculation of risk metrics; whilst 10-day volatility is frequently used for financial risk management purposes. As an example, under Basel II regulatory requirements, a Financial Institution should track its 10-day VaR. Finally, a 21-day and above time frame could be used for option pricing and portfolio management. The volatility of a certain time frame can be estimated using data from a shorter time frame to calculate the standard deviation. As an illustration, the standard deviation of an asset's daily returns can be calculated to estimate the monthly volatility.

3.5 Fundamentals of Portfolio Optimisation

This section covers some of the key fundamental concepts of Portfolio Optimisation used in this thesis.

Portfolio optimisation refers to strategies used in financial risk management that seek to maximise the returns of an investment portfolio while minimising its risks (Soleymani and Paquet, 2020). It involves the diversification of investments across different asset classes to minimise risk and generate returns that are consistent with the investor's risk tolerance (Soleymani and Paquet, 2020). The portfolio optimisation process can also involve the consideration of factors such as expected returns, volatility, and correlation between assets (Zhang et al., 2020). Before the 2008-2009 financial crisis, most of the existing models at the time assumed market risk to be the only risk assets financial assets were exposed to, with only a few papers discussing portfolio optimisation problems with credit risk. Since then, there has been increased prominence in financial risk management with an increased focus on how risks are detected, measured, reported, and managed (Apostolik et al., 2009). An increasing number of studies have focused on developments in financial risk management and the emerging challenges associated with it (Van Liebergen et al., 2017)

Financial Asset

A financial asset represents anything from which a current or future economic value can be expected to be derived. Some examples of these are stocks, bonds, gold, treasury bills, cash and cash equivalents like closed-end funds and exchange-traded funds (ETFs). In this thesis, financial assets are assumed to be liquid in nature. This means that financial assets can be bought and sold without loss in value during the transaction.

Portfolio

A portfolio is a collection of multiple financial assets such as stocks, bonds, commodities, cash, and cash equivalents. A portfolio may also contain a different number of assets, including real estate, art, and private investments. The combination of assets held in a portfolio at any time period, t is defined as a vector in equation (3.7):

$$\mathbf{w}_t = [w_{1,t}, w_{2,t}, ..., w_{K-1,t}, w_{K,t}]^T \in \mathbb{R}^K \tag{3.7}$$

Where K is the number of assets in the portfolio and $\mathbf{w}_t$ is the portfolio vector weight at period t and the ratio of the total amount invested in each asset is represented as k. The total sum of the elements of $\mathbf{w}_t$ is 1 as represented in Equation (3.8).

$$\sum_{k=1}^{K} w_{k,t} = 1 \tag{3.8}$$

Diversified portfolios reduce exposure to risks.

The closing prices of all the assets in a portfolio with K number of assets are made up of *Price Vector*, $\mathbf{v}_t$ for the period t. The k^{th} element of $\mathbf{v}_t$ given as $v_{k,t}$ serves as the closing price of the k^{th} asset in the t^{th} period. For continuous markets, the elements of $\mathbf{V}_t^o$ represent the opening price of the asset for the period t, the elements of $\mathbf{v}_t$ act as opening prices for the trading period $t+1$ and closing prices for the trading period t. Similarly, $\mathbf{v}_t^{hi}$ and $\mathbf{v}_t^{lo}$ represent the highest and lowest prices of the period, respectively.

$\mathbf{y}_t$, which is the *Price relative vector* of the t^{th} trading period is defined as the element-wise division of the price vector at time t, $\mathbf{v}_t$ and price vector at time $t-1$, $\mathbf{v}_{t-1}$

and represented in equation (3.9):

$$\mathbf{y_t} := \mathbf{v_t} \oslash \mathbf{v_{t-1}} = \left(1, \frac{v_{1,t}}{v_{1,t-1}}, \frac{v_{2,t}}{v_{2,t-1}}, ..., \frac{v_{k,t}}{v_{k,t-1}}\right)^{\mathbf{T}} \tag{3.9}$$

The elements of $\mathbf{y}_t$ are the quotients of the Close prices and Open prices for the various assets in the trading period. The change in total portfolio value can be calculated by using the *Price relative vector*, $\mathbf{y}_t$. If the Portfolio Value at the beginning of the trading period, t is P_{t-1} and ignoring transaction cost, then the total portfolio value at the end of the trading period, t is given by equation (3.10):

$$P_t = P_{t-1}\mathbf{y}_t \cdot \mathbf{w}_{t-1} \tag{3.10}$$

where: $\mathbf{w}_{t-1}$ represents the portfolio weight vector at the start of the period t with k^{th} element, $w_{t-1,k}$ is the proportion of the asset k in the portfolio after the reallocation of the capital.

The initial portfolio weight vector $\mathbf{w}_0$ for a typical portfolio allocation problem is given as:

$$\mathbf{w_0} = [1, 0, 0, ..., 0]^T$$

This demonstrates that all the trading capital is held in cash before trading begins. Assuming there is no transaction cost, the final portfolio value P_f is represented in equation (3.11) as:

$$\begin{aligned} P_f &= P_0 \cdot \exp\left(\sum_{t=1}^{t_{f+1}} r_t^l\right) \\ P_f &= P_0 \prod_{t=1}^{t_{f+1}} \mathbf{y}_t \cdot \mathbf{w}_{t-1} \end{aligned} \tag{3.11}$$

where: P_0 is the initial amount invested, while P_f represents the final portfolio value. The overall objective of the portfolio manager is to maximise the final portfolio value P_f for a given period of time.

Returns

The prices of financial assets are generally not directly useful in investors' decision-making to buy or sell an asset. The returns of these assets are usually the most useful for making such decisions. The returns represent the changes in the prices of assets over time. The *rate of return* for the period t is defined as the net gain or loss of an investment within a specific time frame and is represented as a percentage of the initial cost. Mathematically, it is denoted as in equation (3.12):

$$r_t^r = \frac{P_t}{P_{t-1}} - 1 = \mathbf{y}_t \cdot \mathbf{w}_{t-1} - 1 \tag{3.12}$$

The equivalent **logarithmic rate of return**, is represented mathematically by equation 3.13:

$$r_t^l = \ln \frac{P_t}{P_{t-1}} = \ln \mathbf{y}_t \cdot \mathbf{w}_{t-1} \tag{3.13}$$

Portfolio Value with transaction cost

As indicated by Jiang et al. (2017), there is always a cost associated with either the buying or selling of an asset in the financial markets. Based on this, a recursive formula proposed by Ormos and Urban (2013) and extended by Jiang et al. (2017) can be used to recalculate the final portfolio value presented in equation 3.11. Figure 3.1 illustrates the process of reallocating the weights and also shows the effect of the transaction remainder factor, μ_t on the portfolio at each time period.

From Figure 3.1, the market price movement during Period t, represented by *Price relative vector* $\mathbf{y}_t$, moves the portfolio value and portfolio weights from P_{t-1} and $\mathbf{w}_{t-1}$ to $P_t^{'}$ and $\mathbf{w}^{'}{}_t$ respectively. The actions of buying and selling of assets at time t result in the redistribution of the fund into $\mathbf{w}_t$. As a result of these transactions, the portfolio value is shrunk by a factor of μ_t.

The portfolio weight vector at the beginning of the trading period t is $\mathbf{w_{t-1}}$ and as a result of the movement in the prices of assets in the market, at the end of the same trading period, the weights changes to:

$$\mathbf{w}^{'}{}_t = \frac{\mathbf{y}_t \odot \mathbf{w}_{t-1}}{\mathbf{y}_t \cdot \mathbf{w}_{t-1}}$$

Where $\odot$ is an element-wise multiplication operation. To fulfil the aim of the portfolio manager, portfolio vector weight reallocation from $\mathbf{w_t^{'}}$ to $\mathbf{w_t}$ is carried out through the selling and buying of financial assets at the end of the trading period t. The reallocation process reduces the portfolio value by a factor μ_t, referred to by Jiang et al. (2017) as the transaction remainder factor, where $\mu_t \in [0, 1]$.

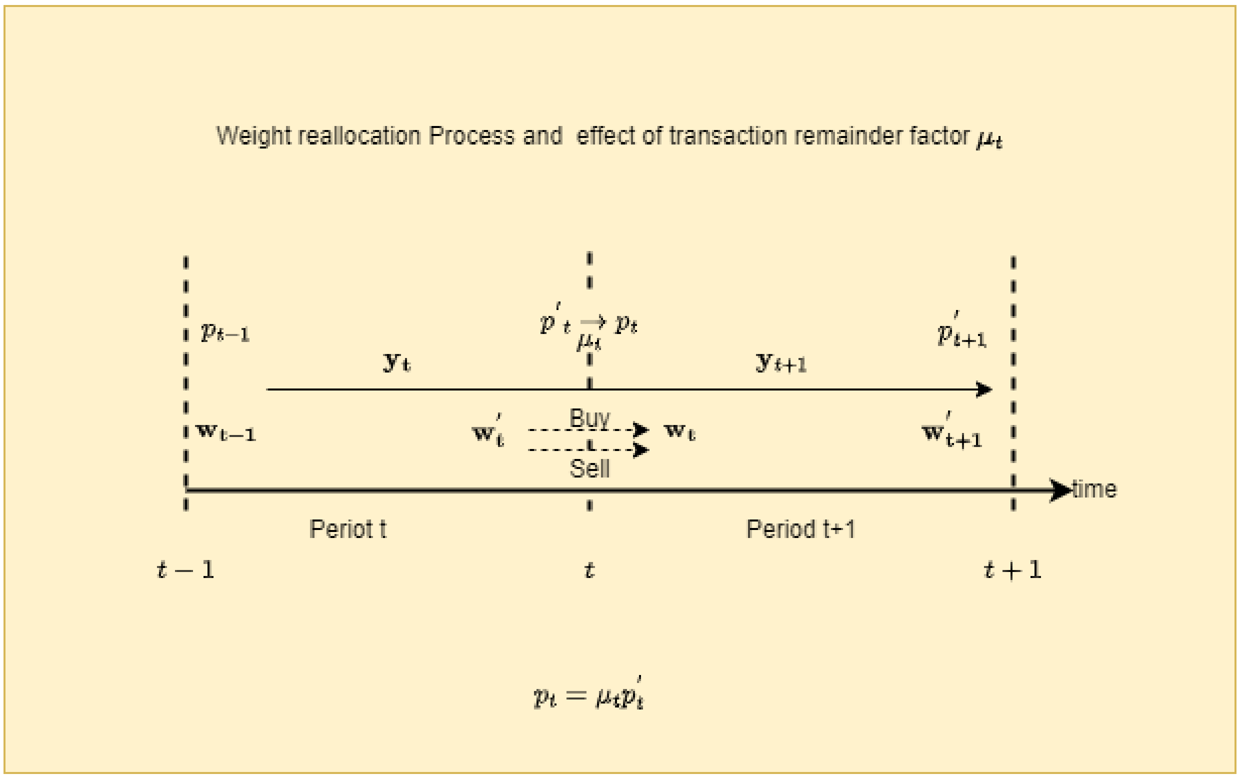


Figure 3.1: illustration of the weight re-allocation process and effect of transaction remainder, μ_t on the portfolio value

Given that P_{t-1} represents the portfolio value at the beginning of the trading period and P'_t represents the portfolio value at the end of the trading period.

$$P_{t-1} = \mu_t P'_t$$

The rate of return and logarithmic rate of return given in equations 3.12 and 3.13 respectively now becomes equation 3.14 and 3.15:

$$r^r_t = \frac{P_t}{P_{t-1}} - 1 = \frac{\mu_t P'_t}{P_{t-1}} - 1 = \mu_t \mathbf{y_t} \cdot \mathbf{w}_{t-1} - 1 \tag{3.14}$$

$$r^l_t = \ln \frac{P_t}{P_{t-1}} = \ln \left(\mu_t \mathbf{y}_t \cdot \mathbf{w}_{t-1}\right) \tag{3.15}$$

When transaction cost is introduced, the final portfolio value shown in equation 3.11 becomes equation shown in 3.16.

$$
\begin{aligned}
P_f &= P_0 \cdot \exp\left(\sum_{t=1}^{t_{f+1}} r_t^l\right) \\
&= P_0 \prod_{t=1}^{t_{f+1}} \mu_t \mathbf{y_t} \cdot \mathbf{w_{t-1}}
\end{aligned}
\tag{3.16}
$$

Portfolio Optimisation

As indicated previously, portfolio optimisation is generally the process of allocating the assets in a portfolio such that it maximises the expected return on investment while at the same time minimising financial risk. In the financial market, one of the most important problems for well-endowed and resourceful investors is how to determine the optimal trading strategy over a specified time frame. Portfolio optimisation can help investors to balance their portfolio's risk-return profile, aligning the portfolio's risk level with the investor's risk tolerance. It can also help investors identify and mitigate potential risks, such as market risk, credit risk, or liquidity risk, by adjusting their portfolio's asset allocation and risk management strategies.

The earliest portfolio optimisation theory, popularly known as modern portfolio theory (MPT), proposed by Markowitz (1952) is a portfolio optimisation problem that highlights how a risk-averse investor can construct a portfolio to minimise the risk of the portfolio given an expected return or, conversely, to maximise the expected return while constraining the risk. Prior to the work of Markowitz (1952), assets were individually analysed to construct a portfolio. Markowitz (1952) proposed that portfolios should be selected based on the overall risk-return assessment. The main assumption behind the MPT is that investors are risk averse and given two portfolios that provide the same expected return, they would always choose the less risky assets. Therefore, there is a trade-off between risk and return when investing, and investors will take an increased risk only if they are compensated with higher expected returns. This assumption led to the formulation of the portfolio problem, which involves finding the weights of assets that minimise the overall risk given an expected return. The variance of the returns of an asset is used to measure the risk of each trading asset.

To represent the basic idea behind the Markowitz mean-variance portfolio model, some notions are introduced.

Let:

K	Be the number of available assets

r_k Be the expected (average, mean) return (per period) of the asset k

ρ_{kj} The correlation between the returns of the assets k and j $(-1 \leq \rho_{kj} \leq +1)$

σ_k Be the standard deviation in return for the asset k

σ_{kj} Be the covariance between the returns for assets k and j $(\sigma_{kj} = \rho_{kj}\sigma_k\sigma_j)$

R Be the desired expected return from the chosen portfolio.

The decision variables are: w_k The proportion of the total investment associated with the asset k ($0 \leq w_k \leq 1$). Using the Markowitz mean-variance method, the portfolio optimisation problem can be mathematically expressed as equation (3.17).

$$\text{minimise} \sum_{k=1}^{K}\sum_{j=1}^{K} w_k w_j \sigma_{kj} \tag{3.17}$$

Subject to:

$$\sum_{k=1}^{K} w_k r_k = R, \tag{3.18}$$

$$\sum_{k=1}^{K} w_k = 1, \tag{3.19}$$

In equation (3.17), the total variance (risk) associated with the portfolio is minimised, subject to equation (3.18) which ensures that the expected return of the portfolio is R. Finally, Equation (3.19) ensures that the weighting of the assets adds up to one, which ensures that all the available cash is invested in the assets. This formulation is a nonlinear programming problem. The standard deviation of an asset is synonymous with risk. The above optimisation problem results in the construction of a smooth non-decreasing curve known as an efficient frontier that demonstrates the best possible trade-off between risk and returns, as shown in Figure 3.2. The efficient frontier is a graphical representation of a set of optimal portfolios that offer the maximum expected return for a predetermined level of risk or the lowest risk for a given level of expected return. Portfolios that lie below the efficient frontier are sub-optimal because they do not provide enough return for the level of risk. Portfolios located on the right of the efficient frontier are sub-optimal because they have a higher level of risk for the defined rate of return. Figure 3.2 shows the running of a smooth continuous curve from the minimum variance portfolio made of 20 assets to the maximum return-to-maximum risk portfolio made up of 1 asset. Here, we can choose to hold any of the portfolios on this efficient frontier.

Based on Markowitz's mean-variance model, the Capital Asset Pricing Model (CAPM)

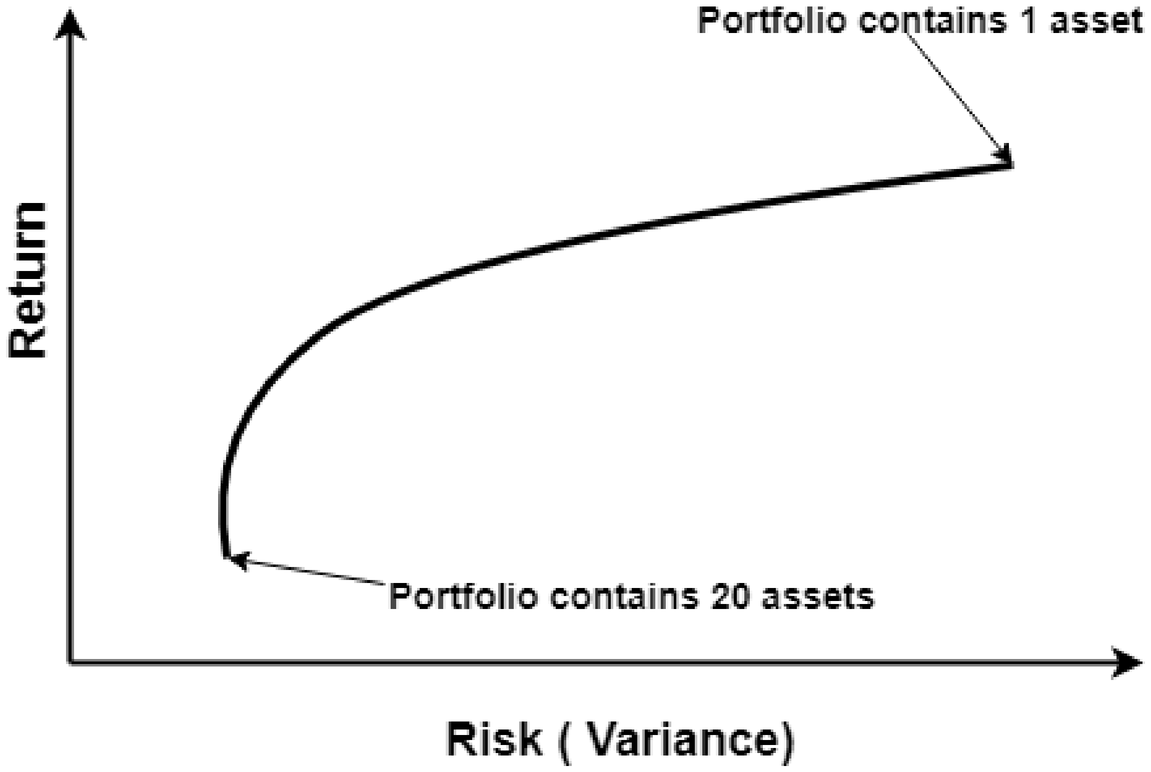


Figure 3.2: Demonstration of Efficient frontier

(Fama and French, 2004) was independently proposed by Sharpe (1964) and Lintner (1965). It describes the relationship between the risk of an asset and the market as a whole. The CAPM states that the expected return of an asset or portfolio equals the return on a risk-free asset plus a risk premium. If an asset is to be added to an already diversified portfolio, CAPM determines a theoretically appropriate rate of return that compensates the investor for taking the risk premium associated with that asset. CAPM assumes that each asset in a portfolio contains a specific risk, but, through diversification, investors can reduce their net exposure to the systematic risk of the market portfolio. The general idea behind CAPM is that investors need to be compensated in two ways: time value of money and risk. The time value of money is represented by the risk-free rate, which means how much return an investor would expect from a risk-free investment over a given period. A rational investor who decides to make a risky investment expects at least to exceed the risk-free rate. The other input to CAPM is the amount of compensation an investor needs to take additional risk. This is calculated by taking the asset's sensitivity to non-diversifiable (specific) risk, often represented by the quantity β in the financial industry, and comparing the returns of the assets with the market premium (return over a risk-free investment). Different approaches measure risk differently; Examples of different risk measures are the variance of returns, Conditional Value at Risk (CVaR) (Rockafellar and Uryasev, 2000), and Sortino ratio (Sortino and van der Meer, 1991). Refer to Chapter 6 for detailed information and experiments carried out on the fundamentals covered in this section.

3.6 Summary

In summary, this chapter has reviewed the background information on financial risk management and covered the key concepts of credit risk modelling, market risk, volatility forecasting, and portfolio optimisation. Additionally, this chapter has provided the literature and background on market risk and volatility. It has also provided the solid groundwork for subsequent chapters that explore machine learning techniques applied to these domains.

In the chapter that follows, the proposed Hybrid Dual Resampling with Cost-sensitive Technique (HDRCS) for credit risk modelling on imbalanced data sets is presented.

Chapter 4

Hybrid Dual Resampling with Cost-sensitive Technique (HDRCS) for Credit Risk Modelling on Imbalanced Data

4.1 Introduction

Credit risk refers to the perspective that a borrower or counter-party will fail to meet its obligations in accordance with agreed terms and conditions. This chapter describes in greater detail the use of the Hybrid Dual Resampling with Cost-sensitive Technique (HDRCS) for the modelling of credit risk on Imbalanced Data sets. Failure to adequately manage credit risks could result in direct accounting losses as it involves transaction costs, opportunity costs, and expenses associated with non-performing assets above the accounting loss (Yu et al., 2018).

Logistic Regression is one of the most widely used statistical models for classification problems which makes use of probability distribution function for estimation and makes it convenient for consumer credit risk modelling (Ershadi and Omidzadeh, 2018). The popularity of using logistic regression is due to its sigmoid shape (Ershadi and Omidzadeh, 2018). It is also due to its ability to be mathematically bound between a range of $0, 1$.

There is the assumption that data are evenly distributed between classes without any bias present in Machine Learning algorithms (D'Arco et al., 2023). Consequently, classifiers such as logistic regression, decision trees, and neural networks perform well when the class distribution of the categorical target or response variable in the data set is balanced (Sun et al., 2007). However, for binary imbalanced class data, the event of interest known as the positive or minority class, for example, probability of default, is under-represented, and the number of cases for the negative or majority class is much higher than the minority class cases (Yap et al., 2014). These are relevant in many real-world problems which involve a binary response variable where the datasets are often imbalanced in nature (Wong et al., 2018) such as document classification (Laza et al., 2011), loan default prediction (Brown and Mues, 2012), fraud detection (Wei et al., 2013) or medical diagnostic classification (Rahman and Davis, 2013). In some cases, the minority class is viewed as the scenario of rare events, for example, the percentage of the minority class is less than 5% (Au et al., 2010). The recent interest in imbalanced datasets has been captured (Yap et al., 2014; Batista et al., 2004). The specific numbers for the imbalance ratio in credit risk datasets vary widely across different datasets and institutions. The class imbalance problem is known to be a major obstacle to the use of a good classifier in machine learning algorithms (Chawla et al., 2004; Sun et al., 2007). Different datasets can show significant variations in their imbalance ratios, depending on their source and nature of the credit data.

The detection of a minority class, especially through the use of logistic regression in imbalanced data sets is challenging and not trivial (Chen et al., 2018), since the performance of classifiers such as Logistic Regression can be poor if it is applied to imbalanced data sets (Wong et al., 2018). Taking credit risk modelling as an example, a dataset with 99% of data from the class of good payers (majority class) and only 1% from the class of bad payers or defaulters (minority class) can still have an accuracy of about 99% if a classifier ignores the data from the bad payers or defaulters class and labels the entire dataset as belonging to the class of the good payers, but it will be unsuitable for credit risk prediction.

Three main approaches can be used to mitigate the problem of imbalanced datasets; these are (i) the data-level approaches, (ii) the algorithm-level approaches, and the hybrid approaches (Chawla et al., 2004; Sun et al., 2007; Johnson and Khoshgoftaar, 2019).

The data-level techniques involve the re-sampling of the imbalanced training dataset before the model is trained (Guo and Viktor, 2004; Batista et al., 2004; Johnson and Khoshgoftaar, 2019). Balanced data can be achieved by resampling the original imbalanced data set to modify the training distributions in order to decrease the level of imbalance or reduce noise through (i) the oversampling of the minority class (Chawla et al., 2002; Liu et al., 2020); (ii) under-sampling the majority class (Japkowicz, 2000); or iii) combining under-sampling and oversampling together as in i) and ii) above (Han et al., 2019). Examples of the oversampling approach are the synthetic minority oversampling technique (SMOTE)(Chawla et al., 2002). There are also other improved SMOTE alternatives such as adaptive synthetic oversampling approaches (ADASYN) (Haibo He et al., 2008) and the Ranked Minority Oversampling in Boosting (RAMOBoost) which uses a probabilistic directed approach (Chen et al., 2010). Other researchers have used under-sampling techniques such as the Minority Cloning Technique (MCT) (Jiang et al., 2015) and the Edited Nearest Neighbour Method (ENN), in which the majority class is treated as unreliable and is subsequently ignored if there are no more minority class labels among its nearest three neighbours (Davies, 1988).

The algorithm-level approaches, on the other hand, do not alter the distribution of the training data (Johnson and Khoshgoftaar, 2019). They improve the existing machine learning methods by adjusting the probabilistic estimate, modifying the cost per class (Provost and Fawcett, 2001; Chawla et al., 2004; Sun et al., 2007), learning from one class (Li et al., 2008), or adding some penalty constants (Lin et al., 2002). Unlike data-level techniques that involve the creation of balanced class distributions through sampling techniques, cost-sensitive learning uses cost matrices that outline the costs associated with the misclassification of the various classes to solve the problem of imbalanced data (Mienye and Sun, 2021). In cost-sensitive learning, penalties are assigned to each class using a cost matrix. The increase in the cost of the minority class is synonymous with an increase in its importance. This decreases the likelihood that the learner will incorrectly classify instances from this minority class (Krawczyk, 2016; Johnson and Khoshgoftaar, 2019).

The Hybrid methods utilise a combination of the data-level and algorithm-level methods in various ways to class imbalance problems (Krawczyk, 2016; Johnson and Khoshgoftaar, 2019). Oversampling techniques have been shown to increase the likelihood of over-fitting in the model construction process (Błaszczyński and Stefanowski, 2015). The use of under-sampling strategies has also been shown to result in the likely elimination of important and useful data present in the majority class (Sun et al., 2015). As a result of these limitations, many researchers have studied the combination of these two techniques. Tomek's modification of a reduced nearest neighbour(Tomek, 1976) has been used in combination with SMOTE as a sampling strategy for imbalanced datasets by researchers. A combination of Gaussian mixture under-sampling and SMOTE has been employed on imbalanced credit data (Han et al., 2019). Examples of this approach involve the combination of oversampling and under-sampling techniques with classifier ensembles through boosting (Schapire, 1990). Some examples of these are SMOTEBoost (Chawla et al., 2003) and RUSBoost (Seiffert et al., 2010).

Financial institutions rely on balanced data sets to help them make informed credit decisions. The main aim of this chapter is to investigate the concept of class imbalance prevalent in financial data sets and to propose a robust solution. This chapter proposes a Hybrid Dual-Resampling Cost-Sensitive (HDRCS) algorithm which simultaneously applies Gaussian mixture modelling (GMM) on the minority class to generate a synthetic minority class, as well as applies k-means clustering to create a new majority class data set. This results in a recommended heuristic imbalance ratio to strike an appropriate balance between the number of the majority class under-sampled and the number of minority class oversampled. The Extra Tree Ensemble technique (Arya et al., 2022) is then applied and a cost-sensitive weighting technique, which further addresses the remaining minor imbalance, is applied to a logistic regression classifier to the final data set based on the most important features.

The rest of the chapter is organised as follows; Section 4.2 discusses the related work on Resampling Techniques for imbalanced datasets. This includes class imbalance in general, Synthetic Minority Over-Sampling Technique (SMOTE), Cluster-based Under-sampling Technique (CBUT), cost-sensitive learning, and our proposed Hybrid Dual Resampling with Cost-sensitive Technique (HDRCS) to balance the datasets. Section 4.3 covers the Gaussian Mixture Model Oversampling Technique (GMMOT) proposed by us as part of our main proposed HDRCS. Section 4.4 introduces the proposed HDRCS algorithm. Section 4.5 presents comparative experiments that demonstrate the effectiveness of the proposed HDRCS algorithm using several real-world credit risk data sets. The conclusions are presented in Section 4.6

4.2 Related work on Re-sampling Techniques for imbalanced datasets

This section's main purpose is to explore the different re-sampling techniques used to resolve the problem of class imbalance in datasets.

Class Imbalance

In a classification task, the problem of class imbalance is an important area that has attracted numerous research studies (Krawczyk, 2016; Zhu and Wang, 2017). In most applications where logistic regression is used to estimate the maximum likelihood of an event, the empirical data often exhibit class imbalance, where one class is represented by many events, whilst the other is much less represented (Oommen et al., 2010). Data sets have also been known to exhibit sampling bias when there is a difference between the class distribution in the sample compared to the actual class distribution in the population. The use of an appropriate density estimation technique to balance datasets is crucial in the maximum likelihood parameter estimation (Oommen et al., 2010). Problems that are intrinsically based on imbalanced data frequently cause imbalances in datasets. Imbalanced datasets provide a disproportionate degree of predictive accuracy, with the majority class having a better percentage of accuracy, while the minority class has poor accuracy measurements (He and Garcia, 2009). Three main approaches can be used to solve the problem of class imbalance (Wang and Yang, 2019); the first approach involves the manipulation of the data sets in order to be balanced (Akbani et al., 2004) and the other is based on the use of improving existing algorithms with the main aim of shifting the decision hyper-plane from the minority class such as the cost-sensitive learning method (Yan et al., 2017). The third approach is the combination of the first and second approaches (Johnson and Khoshgoftaar, 2019). The third approach highlighted above is the relevant technique used in this chapter to solve the problem of imbalanced data.

Synthetic Minority Oversampling Technique (SMOTE)

The SMOTE technique can be used to generate synthetic data using the k-nearest neighbour algorithm (KNN) and Uniform Probability Distribution (Chawla et al., 2002). It does this by introducing synthetic samples along the line segments that join any of the K' minority class nearest neighbours (Ramentol et al., 2011). The principles and operations underpinning these are as follows: At the initialisation stage, the algorithm separates the data into the majority and minority classes. After that, the data for each minority class will have the k^{th} number of its neighbours produced by the k-nearest neighbours (KNN) algorithm. In the process of creating synthetic samples, each minority sample has its own randomly selected nearest neighbour among the k-nearest neighbours. The remainder of the minority sample and its selected nearest neighbour is given in equation (4.1) as:

$$\mathbf{x}_R^+ = |\mathbf{x}^+ - \mathbf{x}_k^+| \tag{4.1}$$

where $\mathbf{x}^+$ is the minority class data and $\mathbf{x}_k^+$ is one of k^{th} neighbours of the minority class data obtained through the use of the uniform probability distribution for random selection. The difference obtained is then multiplied by a random value from the uniform probability distribution to add randomness. The synthetic sample is then obtained by equation (4.2) below:

$$\mathbf{x}_{os}^{+} = \{\mathbf{x}^{+} + (|\mathbf{x}^{+} - \mathbf{x}_{k}^{+}|)\} \times \Phi_r \tag{4.2}$$

where: Φ_r represents the random value generated from the application of the uniform probability distribution. This procedure iterates and repeats itself until the stopping criterion is met, including the target number of samples generated.

Cluster-Based Under-sampling Technique (CBUT)

Cluster centres and the k nearest neighbours of the majority class were used as an under-sampling technique to balance the imbalanced data in which a superior performance was achieved Lin et al. (2017). The clustering-based under-sampling approach was also used to balance imbalanced datasets where improved predictive results were obtained Onan (2019). The processes involved in this technique are as follows. Given a binary imbalanced data set $\mathbf{X}$; made up of the majority class with $\mathbf{X}^{-}$ data sets and the minority class with data sets $\mathbf{X}^{+}$, the first step involves the division of the imbalanced data set into training sets 85% and testing sets 15% or any appropriate percentage division of choice. The second step involves the division of the training set into subsets of a majority class and a minority class. The number of clusters, K' to be used in the k-means clustering is chosen based on the number of data samples in the minority class; thus $K' = N^{+}$. Then, the K' cluster centres, also known as centroids, are produced by fitting the k-means algorithm to the N^{-} data points to produce the N^{+} number of clusters. Where N^{-} and N^{+} represent the number of the majority data class and the number of minority data class, respectively. These centroids replace the entire majority-class dataset, resulting in a reduced number of data samples in the majority class. The reduced majority class is then combined with the minority class to form a balanced training dataset. The classifier is then trained on the balanced training data set and then tested on the test data sets.

Cost-Sensitive Logistic Regression (CSLR)

The Logistic Regression introduced in section 2.2 of Chapter 2 in its standard form assumes an even class distribution of the data and does not take into account their imbalanced nature. To take into account the imbalanced nature of the data, Cost-Sensitive Logistic Regression (CSLR) is used. Cost-sensitive learning is a special type of learning in which the cost of misclassification is taken into account during the training process with the overall goal of minimising the total cost (D'Arco et al., 2023). It involves the modification of the objective function of the classifier to ensure that it focuses more on accurately predicting the minority class (Mienye and Sun, 2021). Hence, instead of optimising the accuracy, the algorithm tries to minimise the total misclassification cost. The given classifier is modified to incorporate varying penalties for each of the classes considered (Krawczyk, 2016). Through the assignment of a higher penalty to the less represented class (minority class), its importance is boosted during the training process. In Cost-Sensitive Logistic Regression (CSLR), the algorithm is modified to take into account the skewed distribution of the data sets (Mienye and Sun, 2021).

4.3 Applied Gaussian Mixture Model Oversampling Technique (GMMOT)

Gaussian Mixture Model Oversampling Technique (GMMOT) involves the use of GMM to over-sample the minority class instances in an imbalanced data set. It uses the fitted normal distributions to generate new samples. The Expectation Maximisation (EM) algorithm can be used to estimate the parameters of the GMM, such as the means and covariances of each component.

The fitted EM algorithm is used to generate the extra data under the proposed GMMOT and the full procedure involved in the technique can be seen in Algorithm 2. For a given imbalanced data set, $\boldsymbol{X} = \{\boldsymbol{x}_i\}$ with binary class $y_i \in \{0, 1\}$, $\boldsymbol{x}$ is m-dimensional feature vector. $\boldsymbol{X}$ can be divided as positive/minority class $\mathbf{X}^+$ and negative / minority class $\mathbf{X}^-$. Under GMMOT, the desired imbalance ratio $\lambda = 1$ is calculated as $\lambda = \frac{N^-}{N^+}$, where, N^- is the number of data points for the majority class and N^+ is the number of data points for the minority class, $N = N^- + N^+$. The number of components used in the GMMOT was based on the silhouette analysis score (Rousseeuw, 1987), where the Silhouette score was used to determine the separation distance between the number of components, K. The optimal number of components corresponded with the highest point on the Silhouette plot, which displays a measure of how close each of the points in one component is to points in the neighbouring components (Pedregosa et al., 2011).

It is proposed to fit the GMM model using $\mathbf{X}^+$, then more minority class samples are generated by sampling from the estimated distribution. To achieve the desired imbalance ratio, $\lambda^* = 1$, N^+_{new} samples are drawn, as

$$N^+_{new} = N^- - N^+$$

which is added to the minority class, to increase its number as $N^+ \leftarrow N^+ + N^+_{new}$, resulting in balanced data.

A detailed summary of the Expectation Maximisation (EM) algorithm used to fit the data points from the data sets is provided in the algorithm 1.

Algorithm 1 The pseudo-code for the Expectation Maximisation used for GMM

```
1: Input: $N$ training samples such that $x_1, x_2 ... x_N \in \mathbf{X}$
2: Initialize the means $\boldsymbol{\mu_j}$, co-variances $\boldsymbol{\Sigma}_j$ and mixture weights $\phi_j$ and evaluate the initial
   log-likelihood.
3: E STEP: Determine the conditional probability for each mixture component $i$ to be
   responsible for observation $x_N$ using current parameter values:
       $$\phi(Z_{ni}) = \frac{\pi_i \mathcal{N}\mathbf{x}_N|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i}{\sum_{j=1}^{j} \pi_j \mathcal{N}(\mathbf{x}_N|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$
4: M STEP: Re-estimate the parameters using the current conditional probabilities
       $$\phi_j = \frac{1}{N}\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}$$
       $$\mu_j = \frac{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}\mathbf{x}^{(i)}}{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}}$$
       $$\Sigma_j = \frac{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}(\mathbf{x}^{(i)} - \mu_j)(\mathbf{x}^{(i)} - \mu_j)^T}{\sum_{i=1}^{N} 1\{\mathbf{z}^{(i)} = j\}}$$
5: The old parameters are replaced by the new ones, and the log-likelihood is calculated
   as follows:
       $$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}; \mu, \Sigma) + \log p(\mathbf{z}^{(i)}; \phi)$$
6: Check that the stopping criteria such as a maximal number of iterations are reached or
   the relative change of the last two log-likelihoods. If the stopping criterion is not met
   return to step 3 to repeat.
7: Output: $\phi = \{\phi_1, ..., \phi_j\}$, $\mu = \{\mu_1, ..., \mu_j\}$, $\Sigma = \{\Sigma_1, ..., \Sigma_j\}$
```

Algorithm 2 shows the full GMMOT procedure used to estimate the distribution of the minority class using Gaussian Mixture Models (GMM).

Algorithm 2 GMMOT Algorithm

```
1: Require: For a given dataset, $\mathbf{X}$ with binary class $y_i \in \{0, 1\}$, desired top $F$ features.
2: Ensure: Logistic model is optimized.
3: Split the training dataset ($\mathbf{X}_{tr}$) into majority class ($\mathbf{X}^-$) and minority class ($\mathbf{X}^+$).
4: Set the number of components used in GMM using Silhouette analysis (Rousseeuw, 1987)
5: Fit the GMM model using the EM algorithm (Bishop, 2006) from $\mathbf{X}^+$, using randomly drawn $N^+_{new} = N^- - N^+$ data samples from the resultant GMM model.
6: Merge $N^+_{new}$ data samples into the minority class, so that $N^+ \leftarrow N^+ + N^+_{new}$, the preprocessed minority data set is still referred to as $\mathbf{X}^+$.
7: Apply Extra Tree Classifier algorithm (Arya et al., 2022) to select the best $F$ features based on the modified two-class datasets $\{\mathbf{X}^-, \mathbf{X}^+\}$
8: Return Predicted Classes for the test data set.
```

4.4 Proposed Hybrid Dual Resampling and Cost-Sensitive Technique (HDRCS)

Cost-Sensitive logistic classifier

For a given imbalanced data set, $\boldsymbol{X} = \{\boldsymbol{x}_i\}$ with binary class $y_i \in \{0, 1\}$, $\boldsymbol{x}$ is m-dimensional feature vector. $\boldsymbol{X}$ can be divided as positive/minority class $\mathbf{X}^+$ and negative / majority class $\mathbf{X}^-$. The imbalance ratio $\lambda \gg 1$ is calculated as $\lambda = \frac{N^-}{N^+}$, where, N^- is the number of data points for the majority class and N^+ is the number of data points for the minority class, $N = N^- + N^+$. Cost-sensitive logistic regression (CSLR) involves modifying the objective function of a logistic classifier to ensure that it is focused more on accurately predicting the minority class (Mienye and Sun, 2021). The coefficients $\boldsymbol{\beta}$ are updated by maximising the log-likelihood function of logistic regression, given by Equation (4.3).

$$\boldsymbol{\beta}^* = \max_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left\{ w^- * \Big(y_i \log(p_i) \Big) + w^+ * \Big((1 - y_i) \log(1 - p_i) \Big) \right\} \tag{4.3}$$

with $p_i = \frac{1}{1+\exp\left(-[1 \quad \boldsymbol{x}_i^T]\boldsymbol{\beta}\right)}$ is the probability of $y(\boldsymbol{x}_i) = 1$, w^- is the weight for the majority class ($y_i = 1$) and w^+ is the weight for the minority class, $y_i = 0$. $\boldsymbol{\beta} = [\beta_0, ..., \beta_m]^T$. The class weights are set such that the total number of effective samples is equal to the total number of samples (N) as (D'Arco et al., 2023):

$$w^- \times N^- + w^+ \times N^+ = N, \quad \text{subject to} \;\; w^- \times N^- = w^+ \times N^+$$

which yields $w^- = \frac{N}{2 \times N^-}$ and $w^+ = \frac{N}{2 \times N^+}$.

By introducing penalties for each of the classes being considered, the algorithm aims to minimise the total misclassification cost. Through the assignment of a higher penalty to the least represented class (minority class), its importance is boosted during the training process. However, in the case of a highly imbalanced data set, it is desirable to use a combination of data-level and algorithm-level approaches (Johnson and Khoshgoftaar, 2019) to further improve the performance and robustness of the classifier.

Hybrid Dual-Resampling and Cost-Sensitive (HDRCS)

In this work, the Hybrid Dual-Resampling and Cost-Sensitive (HDRCS) is introduced which aims to resample the data sets to a desired imbalanced ratio level, followed by the use of CSLR. A summary of the proposed procedure is shown in **Algorithm 3**.

The Algorithm **Algorithm 3** is used for training purposes. The k-means clustering is applied to the majority class $\mathbf{X}^-$, the number of centroids is used to be the majority data set, with the number of centres K' set by using the heuristics below:

$$K' = N^- - N^+ \times r \tag{4.4}$$

Where the resampling coefficient r can take values $\{0, 0.1, 0.2, 0.3\}$ depending on the desired imbalance ratio the user prefers. For any given data, the r value can be calculated using equation (4.5)

$$r = \left(\left(\frac{N^-}{N} \right) - \left(\frac{N^-}{N} - i \right) \right) \geq 0, \quad \text{given} \quad i = \{0, 0.1, 0.2, 0.3\} \tag{4.5}$$

It is assumed that $\frac{N^-}{N} > 0.6$ for any imbalanced data considered, hence $0 \leq r < 0.4$, providing a range of reduction option when using equation(4.4) for the down-sampling dependent on the level of imbalance in the data set. The heuristics to determine how much data samples are reduced by down-sampling is to increase according to either the number of minority data samples (N^+) or the degree of imbalance in the data set. From equations (4.4), it can be seen that at the down-sampling stage using k-means clustering,

Algorithm 3 Proposed Hybrid Dual-Resampling and Cost-Sensitive Learning (HDRCS) logistic classifier.

```
1: Require: For a given dataset, $\mathbf{X}$ with binary class $y_i \in \{0, 1\}$, desired final imbalance ratio $\lambda^*$, desired top $F$ features.
2: Ensure: Logistic model is optimized.
3: Split the training dataset ($\mathbf{X}_{tr}$) into majority class ($\mathbf{X}^-$) and minority class ($\mathbf{X}^+$).
4: for The Majority Class, $\mathbf{X}^-$ : do
5:     Determine the optimal number of centroids to be used to produce the number of clusters, $K'$ using the heuristics below:
           $$K' = N^- - N^+ \times r$$
       where $r = \left(\left(\frac{N^-}{N}\right) - \left(\frac{N^-}{N} - i\right)\right) \geq 0, \quad \text{given} \quad i = \{0, 0.1, 0.2, 0.3\}$
6:     Use k-means clustering algorithm to form $K'$ clusters.
7: end for
8: Synthetic majority data set is composed with $K'$ centres, still referred to as $\mathbf{X}^-$, with $N^- = K'$.
9: Set the number of components used in GMM using Silhouette analysis (Rousseeuw, 1987)
10: Fit the GMM model using the EM algorithm (Bishop, 2006) from $\mathbf{X}^+$,
11: Randomly draw $N^+_{new} = \frac{K'}{\lambda^*} - N^+$ data samples from the resultant GMM model.
12: Merge $N^+_{new}$ data samples into the minority class, so that $N^+ \leftarrow N^+ + N^+_{new}$, the preprocessed minority data set is still referred to as $\mathbf{X}^+$.
13: Apply Extra Tree Classifier algorithm (Arya et al., 2022) to select the best $F$ features based on the modified two-class datasets $\{\mathbf{X}^-, \mathbf{X}^+\}$ which has an imbalance ratio $\lambda^*$
14: Apply the CSLR algorithm to the modified two-class datasets, $\{\mathbf{X}^-, \mathbf{X}^+\}$ which has an imbalance ratio $\lambda^*$.
15: Return Predicted Classes for the test data set.
```

the imbalanced ratio is reduced by r. Since the higher r is, the more imbalanced the data set is, the heuristic for the down-sampling stage is designed so that the higher the imbalance ratio in a data set, the more it is reduced.

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of the densities of Gaussian components, (Biprodip and Mahit, 2017) with its parameters estimated using Expectation Maximisation (EM) (Bishop, 2006). It is proposed to fit the GMM model using $\mathbf{X}^+$, then more minority class samples are generated by sampling from the estimated distribution. For a desired imbalanced ratio, λ^*, N^+_{new} samples are drawn, as

$$N^+_{new} = \frac{K'}{\lambda^*} - N^+,$$

which is added to the minority class, which increases its number as $N^+ \leftarrow N^+ + N^+_{new}$. A series of experiments using trial and error determined that $\lambda^* = 1.67$ can produce the best result.

The motivation of the proposed approach is to introduce a heuristic approach that can realise the dual resampling (down-sample majority class and over-sample minority class) automatically. From (4.5) it can be seen that the more imbalanced the data, there is, the greater the reduction in the down samples using the k-means cluster algorithm. This helps to reduce the demand for generating too many synthetic minority-class samples. Furthermore, the CSLR algorithm addresses the remaining imbalances by setting higher costs for minority data. In doing so, each data-level or algorithm-level component contributes with the aim of obtaining the best performance.

4.5 Experiments

The proposed HDRCS algorithm is compared with several resampling algorithms, including SMOTE (Chawla et al., 2004), CBUT (Lin et al., 2017), GMMOT (Liu et al., 2020) on four real data sets, as shown in Table 4.1 which lists a summary of dimensions and sizes, with more details in Section 4.5. The data sets were subjected to the appropriate data preprocessing techniques, where they were divided into training and test data sets. The split had 85% as a training data set and 15% as a test data set. For fair comparison and completeness, all three imbalanced data resampling techniques SMOTE (Chawla et al., 2004), CBUT (Lin et al., 2017) and GMMOT (Liu et al., 2020) were applied to bring the imbalanced ratio to 1:1, then the same Extra Tree Classifier algorithm was applied to select a set of features (Arya et al., 2022), followed by a logistic classifier. The same algorithm for feature selection, then logistical classifier, was also applied to the original imbalanced data sets, referred to as the baseline model. A series of experiments were conducted using the desired imbalance ratios, λ^* of 1, 1.22, 1.5, 1.67 and 1.86 for each r used to determine the one with the best performance.

Data Description

Data Description Before Resampling

For transparency and reproducible purposes, the description and overview of the datasets used in this study are presented. Three out of the four datasets used are in the public domain and provided by reputable financial institutions. The sources of the dataset, their quality, and their characteristics are described in this section. Table 4.1 shows a summary of the size and imbalance ratio of the four datasets before any resampling technique has been applied to them.

Data set	m, no. of features	N^+	N^-	N	$\%(\frac{N^+}{N})$	Imbalance Ratio λ	r
LCU	24	38	173	211	18%	4.55:1	0.2
UCI German	21	300	700	1000	30%	2.33:1	0.1
GiveMe Some Credit	11	67	933	1000	6.7%	13.98:1	0.3
Lending Club	94	122	1078	1200	10%	8.84:1	0.3

Table 4.1: Size and dimension of original data sets, showing their degrees of imbalance with their respective r values calculated using $r = \frac{N^-}{N} - 0.6$ in the heuristic dual-resampling.

Local Credit Union (LCU) data set

The first data set used is a non-public data set from a local credit union, called LCU data (Osei-Brefo, 2015) in this work. It has 211 observations, of which 173 belonged to the non-defaulters and classified as the good payers. The remaining 38 are bad-payers who defaulted on loan payments. This translates to 82% non-defaulters and 18% defaulters, respectively. This gives an imbalance ratio of 4.55:1, as can be seen in Table 4.1. 63.2% of the bad payers were females, while 36. 8% were males. It has a total of 20 independent features and a target variable. There are 10 categorical features and 11 numerical features in the datasets. The features in the data are membership length, Gender, Dependents, Permanent UK Residence, Age, Employment status, Time with employer, Current Balance, Regular Saver, Previous Loans, past performance, Town, residential status, Time at Address, Loan Applied, Period, Loan Purpose, Top Up, Adverse credit History, Docs, Loan Given, Decision evaluator and Status. The features with some missing values which were treated are Dependents, Time with employer, past performance, residential status, Time at Address, Period, Loan Purpose, adverse credit History, Docs and Decision evaluator variables.

For this data set, the rate of default varies significantly between year groups, demonstrating that 60% of customers aged 20-25 years defaulted on their loans; while about 5% of those aged 65-70 years defaulted on their loans. For the purpose of these experiments, the LCU datasets were subdivided into LCU (a) and LCU (b). The LCU (a) had an extra feature known as GR Decision Eval. It is the pre-screening results offered by a credit risk expert at the financial institution that scored applicants according to their risk profile. The LCU (b) did not have the GR Decision Eval feature. This means that there was no credit risk expert who scored the applicants according to their risk profile.

German Data data

The second set of data used was from the UCI depository called German data (Hofmann, 1994). It has 1000 observations with 20 independent features and a target variable. The majority class had 700 observations, whilst the minority class had 300 observations. This represents a class imbalance ratio of 2.33:1, where 70% belonged to the majority class and 30% belonged to the minority class, as can be seen in Table 4.1.

Give me Some Credit data

The third set of data used was offered by a US-based company called Give Me Some Credit (Credit, 2020). It has an original data size of 150,000 customers and 10 independent features together with 1 target or dependent variable, making it a total of 11 features. Stratified sampling was used to extract a small sample size of 1000 datasets, which was very representative of the original datasets based on the inherent class imbalances. 933 of this belonged to the majority class and 67 belonged to the minority class that defaulted; representing 93. 3% and 6. 7% of the datasets, respectively. These are considered to be of an imbalanced nature with an imbalance ratio of 13.93:1, as can be seen in Table 4.1. These data had 16. 4% of customers aged 30-35 years who defaulted on their loans; while 3% of those aged 60 to 65 years who defaulted on their loans.

Lending Club Data data

This publicly available data used in this work were provided by the Lending Club (Club) Due to its size, only 1200 observations were used for this study using a stratified sampling approach. The majority class had 1078 observations, which represents 89.83% of the data, whilst the minority class had 122 observations, which represented 10.17% of the observations. This gives a class imbalance ratio of 8.84:1, as can be seen in Table 4.1. These data had 94 features, with their full description provided in the link to the source.

Data Description After Resampling

Tables 4.2, 4.3, 4.4 and 4.5 respectively show a summary of the data statistics after the application of our proposed resampling technique on the LCU data, UCI German data, GiveMe Credit Data and Lending Club Data. It respectively shows the number of data points resampled for each r value with their respective desired imbalance ratios λ^* used for the LCU data, UCI German data, GiveMe Credit Data and Lending Club Data. They also show the ratio of the split between the % Majority class and the % minority class that corresponds to each desired imbalance ratio.

The tables 4.2, 4.3, 4.4 and 4.5 show that, as the imbalance ratio (λ^*) increases, the proportion of the minority class (N^+) decreases relative to the majority class (N^-). For each of the datasets used, this trend is observed across different datasets with varying values of r. As λ^* rises, the percentage of the majority class in the dataset increases, moving from a balanced 50%/50% distribution with an imbalance ratio of 1 to a more imbalanced state of 65%/35% with an imbalance ratio of 1.86. This pattern demonstrates the impact of the HDRCS technique in adjusting the class distribution and reflects on how the minority class is under-sampled or the majority class is oversampled to achieve the desired imbalance. This resampling technique is critical in the evaluation of the performance of the HDRCS under different degree of class imbalance.

Tables 4.2, 4.3, 4.4 and 4.5 respectively show the number of oversampled minority data points, under-sampled majority data points, the total number of data points obtained for each of the possible desired imbalance ratio used for each r values and the top 5 selected. It can be seen that the total number of data points gradually reduces as r increases from 0.0 to 0.3. Likewise, given any r-value, the total data points obtained decrease as the desired imbalance ratio increases.

Data set	r	Imbalance Ratio λ^*	F, no. selected features	N^+	N^-	N	% Maj / Min $(\frac{N^-}{N})/(\frac{N^+}{N})$
LCU	0.0	1	5	173	173	346	50%/50%
		1.22	5	141	173	314	55%/45%
		1.5	5	110	173	278	60%/40%
		1.67	5	103	173	276	63%/37%
		1.86	5	93	173	266	65%/35%
		2.33	5	74	173	247	70%/30%
LCU	0.1	1	5	169	169	338	50%/50%
		1.22	5	138	169	307	55%/45%
		1.5	5	109	169	278	60%/40%
		1.67	5	101	169	270	63%/37%
		1.86	5	90	169	259	65%/35%
		2.33	5	72	169	241	70%/30%
LCU	0.2	1	5	165	165	330	50%/50%
		1.22	5	135	165	300	55%/45%
		1.5	5	110	165	275	60%/40%
		1.67	5	98	165	263	63%/37%
		1.86	5	88	165	253	65%/35%
		2.33	5	70	165	231	70%/30%
LCU	0.3	1	5	161	161	322	50%/50%
		1.22	5	131	161	292	55%/45%
		1.5	5	107	161	268	60%/40%
		1.67	5	96	161	257	63%/37%
		1.86	5	86	161	247	65%/35%
		2.33	5	69	161	230	70%/30%

Table 4.2: Sizes and dimensions of the datasets produced, showing the desired imbalance ratio λ^* and the number of data samples oversampled for each r value and imbalance ratio for the LCU dataset after the heuristic dual-resampling approach and feature selection were applied.

Data set	r	Imbalance Ratio λ^*	F, no. selected features	N^+	N^-	N	% Maj / Min $(\frac{N^-}{N})/(\frac{N^+}{N})$
UCI German	0.0	1	5	700	700	1400	50%/50%
		1.22	5	573	700	1273	55%/45%
		1.5	5	466	700	1166	60%/40%
		1.67	5	419	700	1119	63%/37%
		1.86	5	376	700	1076	65%/35%
UCI German	0.1	1	5	670	670	1340	50%/50%
		1.22	5	541	670	1211	55%/45%
		1.5	5	446	670	1116	60%/40%
		1.67	5	401	670	1071	63%/37%
		1.86	5	360	670	1030	65%/35%
UCI German	0.2	1	5	640	640	1280	50%/50%
		1.22	5	524	640	1164	55%/45%
		1.5	5	426	640	1066	60%/40%
		1.67	5	383	640	1023	63%/37%
		1.86	5	344	640	984	65%/35%
UCI German	0.3	1	5	610	610	1220	50%/50%
		1.22	5	500	610	1110	55%/45%
		1.5	5	406	610	1016	60%/40%
		1.67	5	365	610	975	63%/37%
		1.86	5	327	610	937	65%/35%

Table 4.3: Sizes and dimensions of the data points generated, showing the desired imbalance ratio λ^* and the number of data samples oversampled for each r value and imbalance ratio for the UCI German dataset after the heuristic dual-resampling approach and feature selection were applied.

Data set	r	Imbalance Ratio λ^*	F, no. selected features	N^+	N^-	N	% Maj / Min $(\frac{N^-}{N})/(\frac{N^+}{N})$
		1	5	933	933	1866	50%/50%
Give		1.22	5	764	933	1697	55%/45%
Me		1.5	5	622	933	1555	60%/40%
Credit	0.0	1.67	5	558	933	1491	63%/37%
		1.86	5	501	933	1434	65%/35%
		1	5	926	926	1852	50%/50%
Give		1.22	5	759	926	1685	55%/45%
Me		1.5	5	617	926	1543	60%/40%
Credit	0.1	1.67	5	554	926	1480	63%/37%
		1.86	5	497	926	1423	65%/35%
		1	5	640	640	1280	50%/50%
Give		1.22	5	524	640	1164	55%/45%
Me		1.5	5	426	640	1066	60%/40%
Credit	0.2	1.67	5	383	640	1023	63%/37%
		1.86	5	344	640	984	65%/35%
		1	5	610	610	1220	50%/50%
Give		1.22	5	500	610	1110	55%/45%
Me		1.5	5	406	610	1016	60%/40%
Credit	0.3	1.67	5	365	610	975	63%/37%
		1.86	5	327	610	937	65%/35%

Table 4.4: Sizes and dimensions of the datasets produced, showing the desired imbalance ratio λ^* and the number of data samples oversampled for each r value and imbalance ratio for the Give credit dataset after the heuristic dual-resampling approach and feature selection were applied.

Data set	r	Imbalance Ratio λ^*	F, no. selected features	N^+	N^-	N	% Maj / Min $(\frac{N^-}{N})/(\frac{N^+}{N})$
Lending Club	0.0	1	5	1078	1078	2156	50%/50%
		1.22	5	883	1078	1961	55%/45%
		1.5	5	718	1078	1796	60%/40%
		1.67	5	645	1078	1723	63%/37%
		1.86	5	579	1078	1657	65%/35%
Lending Club	0.1	1	5	1065	1065	2130	50%/50%
		1.22	5	872	1065	1937	55%/45%
		1.5	5	710	1065	1775	60%/40%
		1.67	5	637	1065	1702	63%/37%
		1.86	5	572	1065	1637	65%/35%
Lending Club	0.2	1	5	1053	1053	2106	50%/50%
		1.22	5	863	1053	1916	55%/45%
		1.5	5	702	1053	1755	60%/40%
		1.67	5	630	1053	1683	63%/37%
		1.86	5	566	1053	1619	65%/35%
Lending Club	0.3	1	5	1041	1041	2082	50%/50%
		1.22	5	853	1041	1894	55%/45%
		1.5	5	694	1041	1735	60%/40%
		1.67	5	623	1041	1664	63%/37%
		1.86	5	559	1041	1600	65%/35%

Table 4.5: Sizes and dimensions of the datasets produced, showing the desired imbalance ratio λ^* and the number of data samples oversampled for each r value and imbalance ratio for the Lending Club dataset after the heuristic dual-resampling approach and feature selection were applied.

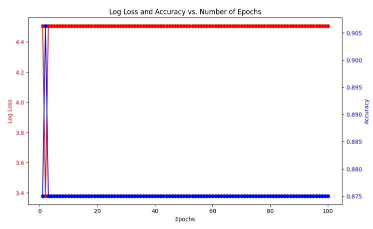
Results and Discussion

Tables 4.6, 4.7, 4.8, 4.9 and 4.10 respectively show the results of the proposed HDRCS algorithm with desired imbalance ratio of 1.67 compared to its variant with a desired imbalanced ratio of 1 using logistic regression model and the other balanced algorithms such as SMOTE(Chawla et al., 2004), CBUT (Lin et al., 2017) and GMMOT (Liu et al., 2020) when applied to LCU (a) data, LCU (b) data, UCI German Data, GiveMe Some Credit data, and Lending Club data, respectively. These tables of results capture the respective evaluation metrics used in this study, which are Precision, Recall, Accuracy, f1-score and Area Under Curve (AUC). The rest are True Negatives (TN), False Positives (FP), False Negatives (FN) and True Positives (TP).

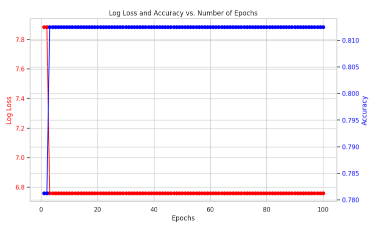
Data Set	Metric	Models					
						HDRCS(r=0.3)	
		Baseline	SMOTE	CBUT	GMMOT	$\lambda^* = 1.67$	$\lambda^* = 1$
Local Credit Union (a)	TN	23	23	21	24	23	23
	FP	3	3	5	2	3	3
	FN	**1**	3	2	2	0	2
	TP	5	3	4	4	6	4
	Precision	0.63	0.50	0.44	**0.67**	**0.67**	0.57
	Recall	0.83	0.5	0.67	0.67	**1**	0.67
	Accuracy	0.88	0.81	0.78	0.88	**0.91**	0.84
	f1-score	0.71	0.50	0.53	0.67	**0.80**	0.62
	AUC	0.86	0.69	0.74	0.79	**0.94**	0.78

Table 4.6: Modeling performance for LCU (a) datasets

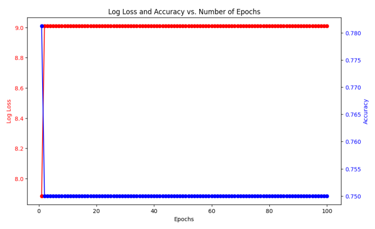
A plot of the loss function and accuracy obtained for the LCU(a) during the training process against the number of epochs is shown in figure 4.1



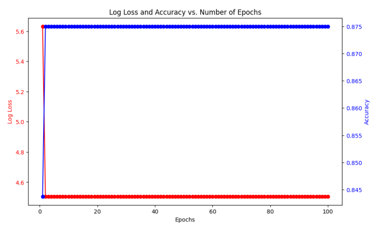
(a) Plots of the Loss function and accuracy against the number of epochs for the baseline model



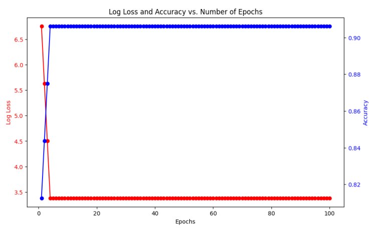
(b) Plots of the Loss function and accuracy against the number of epochs for the SMOTE model



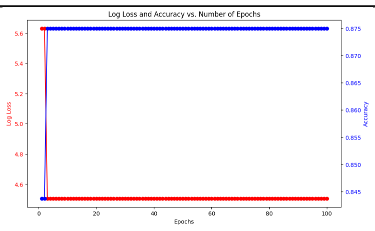
(c) Plots of the Loss function and accuracy against the number of epochs for the CBUT model



(d) Plots of the Loss function and accuracy against the number of epochs for the GMMOT model



(e) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with desired imbalance ratio of 1.67



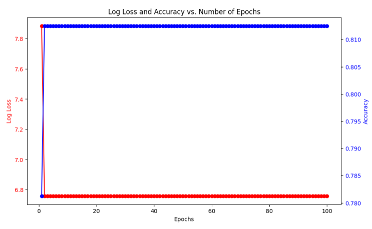
(f) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with logistic regression on balanced data with a desired imbalance ratio of 1

Figure 4.1: Log loss and accuracy plots of all the models used for the LCU (a) data on the LCU(a) data obtained during the training process. The top row models are the baseline model, SMOTE model and CBUT model and the bottom rows are the GMMOT model, HDRCS model with a desired imbalance ratio of 1.67 and HDRCS model with logistic regression on balanced data with an imbalance ratio of 1

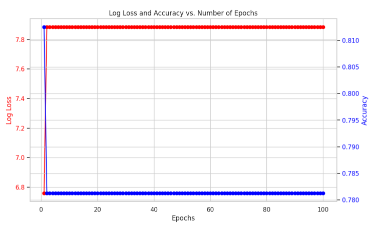
Data Set	Metric	Models					
						HDRCS(r=0.3)	
		Baseline	SMOTE	CBUT	GMMOT	$\lambda^* = 1.67$	$\lambda^* = 1$
Local Credit Union (b)	TN	26	24	20	21	21	21
	FP	0	2	6	5	4	5
	FN	6	5	2	2	2	2
	TP	0	1	4	4	4	4
	Precision	0	0.33	0.40	0.44	**0.50**	0.4
	Recall	0	0.17	**0.67**	**0.67**	**0.67**	**0.67**
	Accuracy	**0.81**	0.78	0.75	0.78	**0.81**	0.78
	f1-score	0	0.22	0.50	0.53	**0.57**	0.53
	AUC	0.5	0.54	0.72	0.74	**0.76**	0.74

Table 4.7: Modelling performance for LCU(b) dataset.

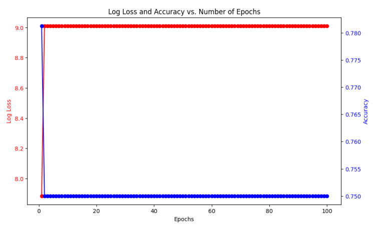
A plot of the loss function and accuracy obtained for the LCU(b) during the training process against the number of epochs is shown in figure 4.2



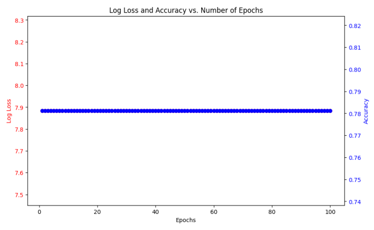
(a) Plots of the Loss function and accuracy against the number of epochs for the baseline model



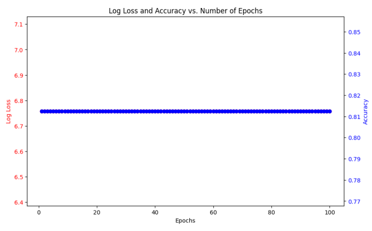
(b) Plots of the Loss function and accuracy against the number of epochs for the SMOTE model



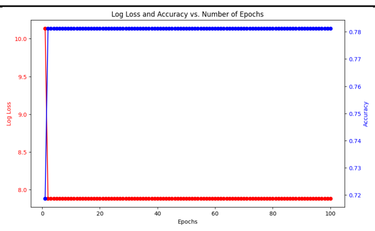
(c) Plots of the Loss function and accuracy against the number of epochs for the CBUT model



(d) Plots of the Loss function and accuracy against the number of epochs for the GMMOT model



(e) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with the desired imbalance ratio of 1.67



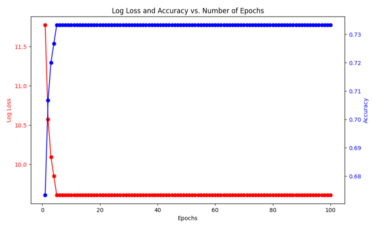
(f) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with logistic regression on balanced data with a desired imbalance ratio of 1

Figure 4.2: Log loss and accuracy plots of all the models used for the LCU(b) Data obtained during the training process. The top row models are the baseline model, SMOTE model and CBUT model and the bottom rows are the GMMOT model, HDRCS model with a desired imbalance ratio of 1.67 and HDRCS model with logistic regression on balanced data with an imbalance ratio of 1

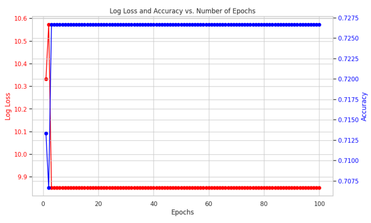
Data Set	Metric	Models					
						HDRCS(r=0.1)	
		Baseline	SMOTE	CBUT	GMMOT	$\lambda^* = 1.67$	$\lambda^* = 1$
UCI German	TN	**91**	99	67	71	69	65
	FP	**20**	35	38	34	36	40
	FN	20	**6**	13	10	10	10
	TP	19	10	32	**35**	35	35
	Precision	0.49	0.22	0.46	**0.51**	0.49	0.47
	Recall	0.49	0.63	0.71	**0.78**	**0.78**	**0.78**
	Accuracy	**0.73**	0.73	0.66	0.71	0.69	0.67
	f1-score	0.49	0.33	0.56	**0.61**	0.60	0.58
	AUC	0.65	0.68	0.67	**0.73**	0.72	0.70

Table 4.8: Modelling performance for UCI German dataset.

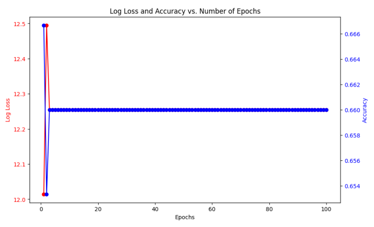
A plot of the loss function and accuracy obtained for the UCI German Data during the training process against the number of epochs is shown in figure 4.3



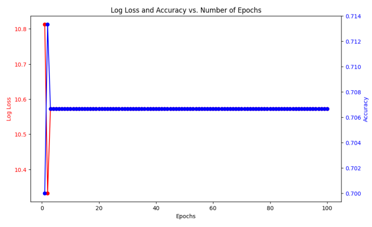
(a) Plots of the Loss function and accuracy against the number of epochs for the baseline model



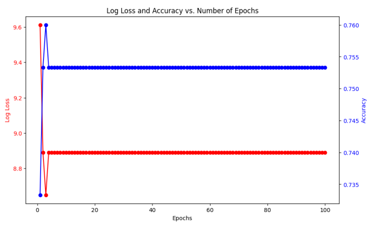
(b) Plots of the Loss function and accuracy against the number of epochs for the SMOTE model



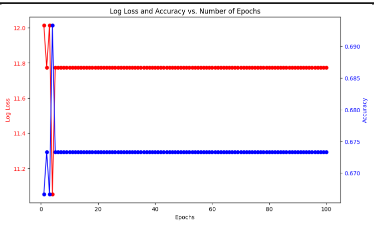
(c) Plots of the Loss function and accuracy against the number of epochs for the CBUT model



(d) Plots of the Loss function and accuracy against the number of epochs for the GMMOT model



(e) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with the desired imbalance ratio of 1.67



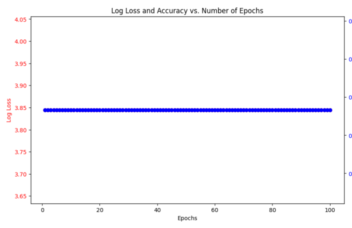
(f) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with logistic regression on balanced data with a desired imbalance ratio of 1

Figure 4.3: Log loss and accuracy plots of all the models used for the UCI German Data obtained during the training process. The top row models are the baseline model, SMOTE model and CBUT model and the bottom rows are the GMMOT model, HDRCS model with a desired imbalance ratio of 1.67 and HDRCS model with logistic regression on balanced data with an imbalance ratio of 1

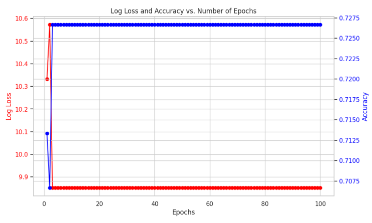
Data Set	Metric	Models					
						HDRCS(r=0.3)	
		Baseline	SMOTE	CBUT	GMMOT	$\lambda^* = 1.67$	$\lambda^* = 1$
GiveMe Some Credits	TN	134	99	97	137	138	138
	FP	**0**	35	43	3	2	2
	FN	16	6	2	6	5	6
	TP	0	**10**	8	5	6	5
	Precision	0	0.22	0.17	0.63	**0.75**	0.71
	Recall	0.0	0.63	**0.82**	0.45	0.55	0.45
	Accuracy	0.89	0.73	0.70	0.94	**0.95**	**0.95**
	f1-score	0	0.33	0.29	0.53	**0.63**	0.56
	AUC	0.5	0.68	0.76	0.73	**0.77**	0.72

Table 4.9: Modelling performance for Give Me Some Credit dataset.

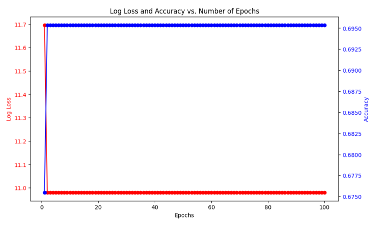
A plot of the loss function and accuracy obtained for the Give Me Credit Data during the training process against the number of epochs is shown in figure 4.4



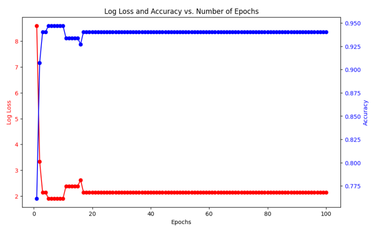
(a) Plots of the Loss function and accuracy against the number of epochs for the baseline model



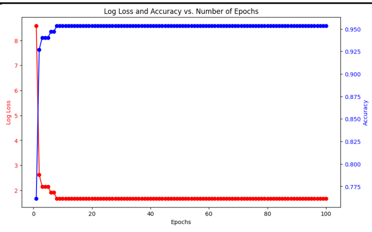
(b) Plots of the Loss function and accuracy against the number of epochs for the SMOTE model



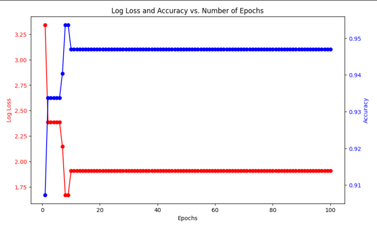
(c) Plots of the Loss function and accuracy against the number of epochs for the CBUT model



(d) Plots of the Loss function and accuracy against the number of epochs for the GMMOT model



(e) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with desired imbalance ratio of 1.67



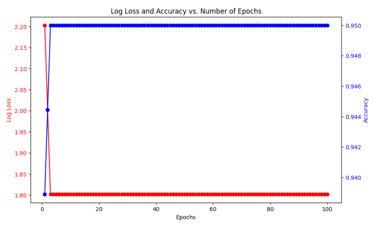
(f) Plots of the Loss function and accuracy against the number of epochs of logistic regression model on balanced data with a desired imbalance ratio of 1

Figure 4.4: Log loss and accuracy plots of all the models used for the Give Me Credit Data obtained during the training process. The top row models are the baseline model, SMOTE model and CBUT model and the bottom rows are the GMMOT model, HDRCS model with a desired imbalance ratio of 1.67 and logistic regression on data with an imbalance ratio of 1

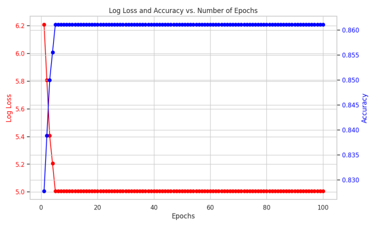
Data Set	Metric	Models					
						HDRCS(r=0.3)	
		Baseline	SMOTE	CBUT	GMMOT	$\lambda^* = 1.67$	$\lambda^* = 1$
Lending Club	TN	160	137	96	157	162	154
	FP	**0**	23	66	5	0	8
	FN	9	2	1	6	8	4
	TP	11	18	18	13	11	15
	Precision	**1**	0.44	0.21	0.72	**1**	0.65
	Recall	0.55	0.90	**0.95**	0.68	0.58	0.79
	Accuracy	0.95	0.86	0.63	0.94	**0.96**	0.93
	f1-score	0.71	0.59	0.35	0.70	**0.73**	0.71
	AUC	0.78	**0.88**	0.77	0.83	0.79	0.87

Table 4.10: Modelling performance for Lending Club dataset.

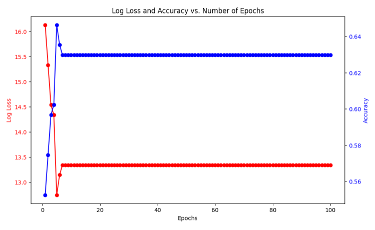
A plot of the loss function and accuracy obtained for the Lending Club Data during the training process against the number of epochs is shown in figure 4.5



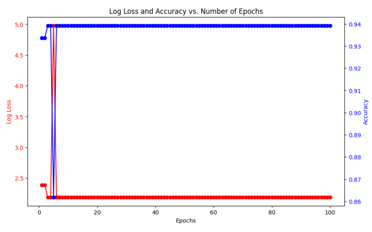
(a) Plots of the Loss function and accuracy against the number of epochs for the baseline model



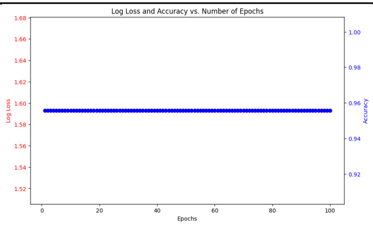
(b) Plots of the Loss function and accuracy against the number of epochs for the SMOTE model



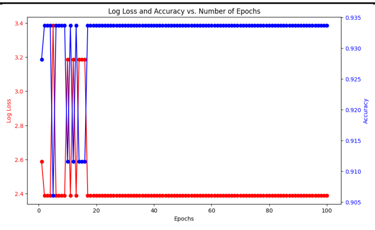
(c) Plots of the Loss function and accuracy against the number of epochs for the CBUT model



(d) Plots of the Loss function and accuracy against the number of epochs for the GMMOT model



(e) Plots of the Loss function and accuracy against the number of epochs for the proposed HDRCS model with the desired imbalance ratio of 1.67



(f) Plots of the Loss function and accuracy against the number of epochs with logistic regression model a data with a desired imbalance ratio of 1

Figure 4.5: Log loss and accuracy plots of all the models used for the Lending Club Data obtained during the training process. The top row models are the baseline model, SMOTE model and CBUT model and the bottom rows are the GMMOT model, HDRCS model with a desired imbalance ratio of 1.67 and logistic regression model with an imbalance ratio of 1

Analysis of the results of the models on the imbalanced LCU (a) datasets from Table 4.6 shows that having a credit risk expert in the assessment and risk profiling of loan applicants leads to an increase in performance metrics compared to the LCU (b) datasets which did not have a credit risk expert evaluation of loan applicants, as shown in Table 4.7.

It can therefore be seen in Table 4.6 that the application of the baseline model on the LCU (a) data set resulted in Precision of 0.63, Recall of 0.83, Accuracy of 0.88, f1-score of 0.71 and AUC of 0.86 as compared to the application of the same model on the LCU(b) dataset in Table 4.7 which had relatively lower precision of 0, recall of 0, Accuracy of 0.81, f1-Score of 0 and AUC of 0.5. This was equally true when the SMOTE technique was applied to both LCU (a) and LCU (b), where the f1 score improved to 0.50 in LCU (a) from 0.22 in LCU (b) resulting in a 127.27% increase in the f1-score with the introduction of a credit risk expert. This trend was consistent for all the other eight metrics used in this model. This observation was also true when the CBUT technique was applied to the LCU(a) and LCU(b) data where the f1-score increased from 0.50 to 0.53, representing an increase of 6.00% as a result of the introduction of a credit risk expert into the process. The GMMOT had similar trends, in which the introduction of a credit risk expert into the decision-making process resulted in an increase of f1-score from 0.53 to 0.67 representing an increase of 26.42%. Overall it can be seen that the proposed HDRCS model with a desired imbalance ratio of 1.67 achieved the highest Precision, highest Recall, highest Accuracy, highest f1-score and highest AUC values when there was the involvement of a credit risk expert for the LCU(A) with figures of 0.67, 1, 0.91, 0.80 and 0.94 respectively. This superior performance of our proposed HDRCS model with a desired imbalance ratio of 1.67 was equally true when it was applied to the LCU(b), where there was not any involvement of a credit risk expert in the scoring process, with the proposed HDRCS model achieving the highest Precision, highest Recall, highest Accuracy, highest f1-score and highest AUC values 0.50, 0.67.0.81, 0.57 and 0.76 respectively. For both LCU(a) and LCU(b) the HDRCS model with a desired imbalance ratio of 1.67 outperformed the HDRCS model with a desired imbalance ratio of 1.

From Table 4.8, it can be observed that the GMMOT model achieved the highest performance metric in 5 of the total available metrics when applied to the UCI German dataset. It had the highest precision of 0.51, the highest recall of 0.78, the highest f1 score of 0.61, and the highest AUC of 0.73. The baseline model was able to achieve the highest accuracy figure of 0.73 when applied to the UCI German data set. The proposed HDRCS could only tie in with the GMMOT in terms of the highest recall achieved. Nonetheless, it compared well with the highest-performing GMMOT model, with a very small difference between their performance measures. One hypothesis that could be attributed to why the model failed to dominate as the highest performing model on this dataset could be because this dataset happened to be the least imbalanced dataset out of all the four datasets used, with an imbalance ratio of 2.33:1. However, the model was designed for datasets with relatively high imbalanced ratios.

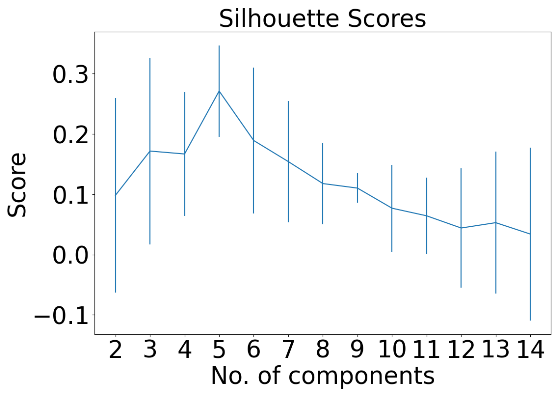
For the Give Me Credit data set shown in Table 4.9, the proposed HDRCS model with a desired imbalance ratio of 1.67 achieved superior performance in terms of Precision, Accuracy, f1-score and AUC metrics with values of 0.75, 0.95, 0.63, and 0.77 respectively. The f1-score and AUC of our proposed model are 100% and 54% above the f1-score and AUC figures obtained by the baseline model, which had an f1-score and AUC figures of 0 and 0.5, respectively. The baseline model performed poorly on this data set in terms of f1-score and AUC due to the highly imbalanced nature of this dataset; with the highest imbalance ratio of 13.98:1, which is the highest among the four data sets used.

Finally, from figure 4.10, it can be observed that the proposed model achieved the highest performance in the precision metric with an impressive value of 1, the highest accuracy value of 0.96, and the highest f1-score of 0.73. The CBUT model had the lowest f1-score of 0.35 but achieved the highest recall value of 0.95, while the SMOTE achieved the highest AUC value of 0.88.

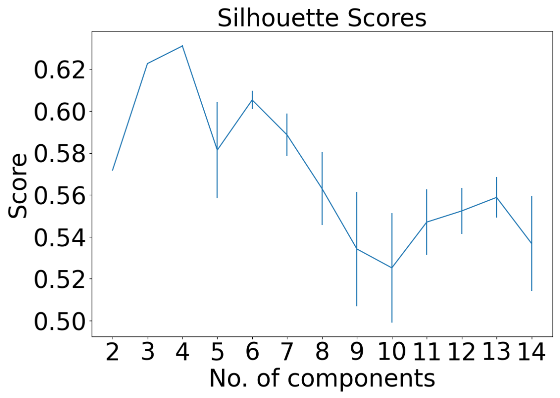
From Tables 4.6, 4.7, 4.8, 4.9 and 4.10 it can be observed that of the 5 metrics used in all data sets used, the HDRCS achieved superior performance in 18 of the total 25 available performance metrics. This represented 72% of the total, which is 48% higher than the metric achieved by the next best performing model, GMMOT in all data sets and models used with the superior performance of the total of 8 performance metrics. Again, the proposed HDRCS obtained the highest proportion of the 5 f1-scores available across the 5 datasets used. It achieved a superior f1-score in 4 of the 5 data sets used in this work, representing 80% of the total available.

The equation $r = \frac{N^-}{N} - 0.6$ obtained from equation (4.5) was used to estimate the optimal r value of each dataset used. This is because while $\lambda^* = 1$ may help to create a perfectly balanced data set, it equally means that more synthetic data samples than true data samples are needed from the GMM model, which may introduce inaccuracies since there is a lack of original minority data samples. For example, the choice $\lambda^* = 1.67$ that was used in the experiments would still have a more balanced data set than the original data set, but with higher fidelity to the original minority data set. As was observed from tables 4.2, 4.3, 4.4 and 4.5, an increase in r correlates with a shift in the imbalance ratios. This shift impacts the representation of majority and minority classes. This trend underscores the important role played by the resampling coefficient, r in the dual-resampling process of the HDRCS technique, and hints at its potential to achieve an optimal class distribution for model training in credit risk assessment. This is important in credit risk prediction, where the balance between accurate detection of defaulters (minority class) and non-defaulters (majority class) is key to effective and fair risk assessment. The relationship between r and class distribution offers a promising avenue for refining predictive models, ensuring they are not only accurate but also fair in their predictions.

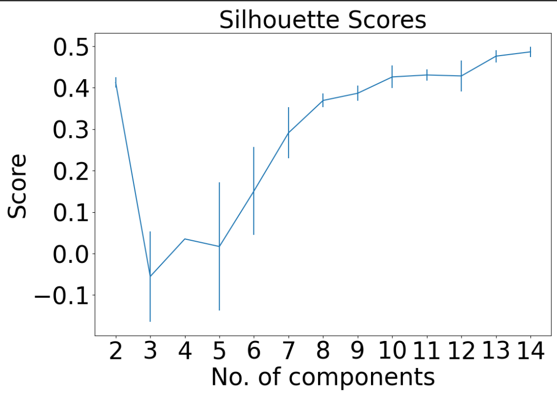
Choosing the number of components



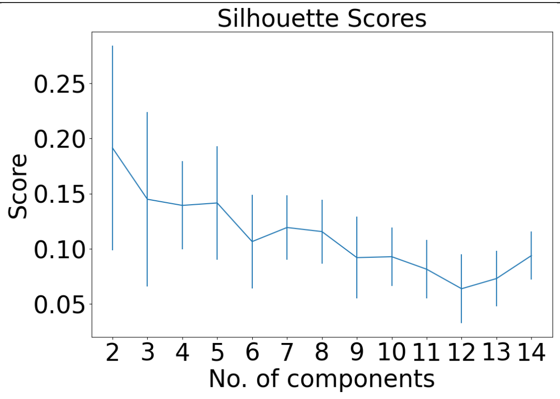
(a) Plots of the Silhoette score for LCU data



(b) Plots of the Silhoette score for German data



(c) Plots of the Silhoette score for GiveMe-Credit data



(d) Plots of the Silhoette score for the Lending Club datasets

Figure 4.6: Comparision of the plots of the Silhoette score for all the data sets used

The results from the above indicate that the use of GMM oversampling and K-means undersampling together with the cost-sensitive logistic regression achieved the desired outcome. It can be seen in figure 4.6 that for the LCU Data, when the highest silhouette score of about 0.26 was obtained when the number of components in the minority class was 5. Hence the number of components used for the LCU for the GMMOT was 5. Likewise, the UCI German data had the highest silhouette score of 0.625 when 4 components were used. For the Give Me credit data, the use of 13 components resulted in the highest Silhoette score of 0.45 and finally the Lending Club data obtained the highest silhouette score of 0.2 when the number of components were 2. There was not any significant number of outliers detected, during the data preprocessing stage which could have skewed the mean and variance of the Gaussian distributions and subsequently affect the performance of the model. To mitigate the potential effects of these outliers, cross-validation testing was used during the training phase of the model.

Sensitivity Analysis of HDRCS

In this section, a comprehensive sensitivity analysis of the HDRCS (Hybrid Dual Resampling Cost-Sensitive) model on all 4 imbalanced datasets used is conducted. The aim is to investigate the impact of varying the r coefficient and the corresponding desired imbalance ratios (λ^*) on various performance metrics used, to provide greater insight into the model's behaviour under different conditions. This analysis offers insights that are invaluable in addressing class imbalance effectively.

Our analysis begins by considering different datasets, each characterised by a distinct r-value parameter. Concurrently, variations of the imbalance ratios (λ^*) are explored to examine their influence on the HDRCS model performance. The sensitivity to these parameters highlights the inherent complexity of the model's response to dataset characteristics. Our investigation reveals compelling variations in the HDRCS model's performance metrics, which encompass Precision, Recall, Accuracy, F1-score, and AUC. As the r-value and λ^* change, these metrics reflect the model's adaptability and response to the evolving dataset conditions. Such sensitivity underscores the importance of parameter selection. It can be observed that the choice of desired imbalance ratio, λ^* plays an important role in determining the model's performance. It can also be observed from the table 4.11 that higher imbalance ratios, with $\lambda^* = 1.67$ and $\lambda^* = 1.86$, are associated with an increase in False Positives (FP), adversely the model's impacting Precision and F1-score. This observation underscores the need to carefully consider the trade-offs between different performance metrics.

The analysis also identifies parameter combinations that result in superior model performance for specific scenarios. For instance, when $\lambda^* = 1.22$ and r-values of 0.1 are used, the HDRCS model consistently demonstrates high Precision, Recall, Accuracy, F1-score, and AUC. This finding suggests that the careful selection of r-value and λ^* can lead to improved performance of the model.

The HDRCS model's performance on the LCU (a) dataset at different r-values and imbalance ratios reveals notable trends. Specifically, at r-value of 0.0, the model maintains high Accuracy and AUC across most λ^* values, indicating robustness to class imbalances. However, the model's performance varies at higher r-values; for example, at r=0.1 and $\lambda^* = 1$, there's a notable improvement in both Accuracy and AUC with figures of 0.91 and 0.94 respectively. Interestingly, the f1-score is highest at 0.8 for r=0.1 and $\lambda^* = 1$. This pattern suggests the HDRCS model's response to imbalances varies with r, impacting its predictive reliability and the trade-off between Precision and Recall for this data set. These observations imply that the model's ability to correctly classify cases improves with a slight

Data Set	r-value	Metric	HDRCS Model				
			$\lambda^* = 1$	$\lambda^* = 1.22$	$\lambda^* = 1.5$	$\lambda^* = 1.67$	$\lambda^* = 1.86$
		TN	24	24	24	22	**24**
		FP	2	2	2	**4**	**2**
		FN	2	2	2	2	2
		TP	4	4	4	4	4
Local Credit Union (a)	r=0.0	Precision	0.67	0.67	0.67	0.5	0.67
		Recall	0.67	0.67	0.67	0.67	0.67
		Accuracy	**0.88**	0.88	0.88	0.81	**0.88**
		f1-score	0.67	0.67	0.67	0.57	0.67
		AUC	0.79	0.79	0.79	0.76	0.79
		TN	23	24	22	24	**24**
		FP	3	2	4	2	2
		FN	**0**	2	2	2	2
		TP	6	4	4	4	4
Local Credit Union (a)	r=0.1	Precision	0.67	0.67	0.50	0.67	**0.67**
		Recall	**1**	0.67	0.67	0.67	0.67
		Accuracy	0.91	0.88	0.81	**0.88**	**0.88**
		f1-score	**0.8**	0.67	0.57	0.67	0.67
		AUC	**0.94**	0.79	0.76	0.79	0.79
		TN	24	24	23	24	23
		FP	2	2	3	**2**	3
		FN	2		2	2	2
		TP	4	4	4	4	4
Local Credit Union (a)	r=0.2	Precision	0.67	0.67	0.57	**0.67**	**0.57**
		Recall	**0.67**	0.67	0.67	0.67	0.67
		Accuracy	**0.88**	0.88	0.84	**0.88**	0.84
		f1-score	**0.67**	0.67	0.62	0.67	0.62
		AUC	0.79	0.79	0.78	0.79	0.78
		TN	23	24	23	23	**23**
		FP	3	2	3	3	3
		FN	2	2	2	0	2
		TP	4	4	4	6	4
Local Credit Union (a)	r=0.3	Precision	0.57	0.67	0.57	**0.67**	0.57
		Recall	0.67	0.67	0.67	1	0.67
		Accuracy	0.84	0.88	0.84	**0.91**	**0.84**
		f1-score	0.62	0.67	0.62	0.80	0.62
		AUC	0.78	0.79	0.78	**0.94**	0.78

Table 4.11: sensitivity analysis for HDRCS for LCU (a) datasets

increase in the r-value, but this improvement is not consistent across all imbalance ratios for this dataset. This trend highlights the importance of tuning the r-value to optimise the model's performance, especially in scenarios with varying degrees of class imbalance.

Data Set	r-value	Metric	HDRCS Model				
			$\lambda^* = 1$	$\lambda^* = 1.22$	$\lambda^* = 1.5$	$\lambda^* = 1.67$	$\lambda^* = 1.86$
LCU(b) Data	r=0.0	TN	21	21	22	19	23
		FP	5	5	4	7	3
		FN	2	2	3	2	3
		TP	4	4	3	4	3
		Precision	0.44	0.44	0.43	0.36	0.50
		Recall	0.67	0.67	0.50	0.67	0.50
		Accuracy	0.78	0.78	0.78	0.72	0.81
		f1-score	0.53	0.53	0.46	0.47	0.50
		AUC	0.74	0.74	0.67	0.70	0.69
LCU(b) Data	r=0.1	TN	21	19	20	21	21
		FP	5	7	6	5	5
		FN	2	2	3	2	3
		TP	4	4	3	4	3
		Precision	0.44	0.36	0.33	0.44	0.38
		Recall	0.67	0.67	0.5	0.67	0.5
		Accuracy	0.78	0.72	0.72	0.78	0.75
		f1-score	0.53	0.47	0.40	0.53	0.43
		AUC	0.74	0.70	0.63	0.74	0.65
LCU(b) Data	r=0.2	TN	23	21	23	18	19
		FP	3	5	3	8	7
		FN	3	3	3	3	2
		TP	3	3	3	3	4
		Precision	0.50	0.38	0.50	0.27	0 .36
		Recall	0.50	0.50	0.50	0.50	0.67
		Accuracy	0.81	0.75	0.81	0.66	0.72
		f1-score	0.50	0.43	0.50	0.35	0.47
		AUC	0.69	0.65	0.69	0 .60	0 .70
LCU(b) Data	r=0.3	TN	21	21	22	21	**23**
		FP	5	5	5	4	5
		FN	2	3	3	2	3
		TP	4	3	3	4	3
		Precision	0.44	0.38	0.38	0.50	0.38
		Recall	0.67	0.50	0.50	0.67	0.50
		Accuracy	0.78	0.75	0.75	0.81	0.75
		f1-score	0.53	0.43	0.43	0.57	0.43
		AUC	0.74	0.65	0.65	0.76	0.65

Table 4.12: sensitivity analysis for HDRCS for LCU (b) datasets

As can be observed in Table 4.12, the sensitivity analysis of the HDRCS model on the LCU (b) datasets shows various trends. There exists fluctuation in metrics such Precision, Recall, Accuracy, f1-score, and AUC as the r-value increases. Specifically, Accuracy generally remains above 0.75 for all r-values, indicating a stable model performance across different imbalance scenarios. However, the Precision and Recall values show more variability, which impacts the stability of the f1-score. As an example, at r=0.0 and $\lambda^* = 1$, Precision is 0.44, which increases to 0.50 at λ=1.86. This indicates the model's varying ability to correctly identify positive cases as the class imbalance changes. Such insights are essential for understanding the model's behaviour under different conditions, guiding adjustments for optimal performance in credit risk prediction. As can be seen in Table 4.13, the sensitivity analysis of the HDRCS model on the UCI German dataset indicates that the f1-score exhibits a fluctuating pattern across different r-values and imbalance ratios. Notably, the f1-score remains relatively consistent around 0.60 to 0.63 at r=0.0, but as r-value increases to 0.1 and beyond, there is a slight decrease in the f1-score, particularly noticeable at higher imbalance ratios. This trend suggests that while the model's ability to balance Precision and Recall is relatively stable at lower r-values, it becomes slightly difficult under conditions of increased class imbalance as the r-value rises. This could be seen as a limitation of the HDRCS model. As can be seen in the Table 4.14 for the sensitivity analysis of the HDRCS model on the Give Me Credit dataset, there exists a noticeable trend in the f1-score across different r-values and imbalance ratios. The model generally maintains a high Accuracy of about 0.95 across all r-values. However, the f1-score shows a varied outcome. For instance, at r=0.0 with $\lambda^* = 1$, the f1-score is 0.56, which increases to 0.63 at $\lambda^* = 1.86$. This pattern indicates that the HDCRS model's ability to balance Precision and Recall improves slightly with an increase in class imbalance, particularly at higher r-values. This insight is critical for understanding the model's effectiveness in handling class imbalance in credit risk prediction.

As can be seen in Table 4.15, The sensitivity analysis of the HDRCS model on the Lending Club dataset shows an interesting trend in the performance metrics across different r-values and imbalance ratios. A notable observation is the model's Precision, which reaches 1.0 at higher imbalance ratios for r-values of 0.2 and 0.3, suggesting an increased ability to predict positive classes correctly under these conditions. However, this high Precision coincides with a decrease in Recall, particularly at r=0.3 and $\lambda^* = 1.86$. The f1-score tends to be higher at r=0.3 across different imbalance ratios, indicating an overall effective balance between the sensitivity and specificity of the model under these settings.

4.6 Summary

This paper has highlighted the potential problems caused by the use of insufficient and imbalanced data sets in classification tasks. The Hybrid Dual-Resampling and Cost Sensitive Algorithm (HDRCS) has been proposed as a solution to address the class imbalance prob-

Data Set	r-value	Metric	HDRCS Model				
			$\lambda^* = 1$	$\lambda^* = 1.22$	$\lambda^* = 1.5$	$\lambda^* = 1.67$	$\lambda^* = 1.86$
		TN	74	71	70	70	74
		FP	31	34	35	35	31
		FN	10	10	11	10	13
		TP	35	35	34	35	32
UCI	r=0.0	Precision	0.53	0.51	0.49	0.5	0.5
German		Recall	0.78	0.78	0.75	0.78	0.71
Data		Accuracy	0.73	0.71	0.69	0.7	0.71
		f1-score	0.63	0.61	0.60	0.61	0.59
		AUC	0.74	0.73	0.71	0.72	0.71
		TN	65	67	73	69	70
		FP	40	38	32	36	35
		FN	10	12	12	10	12
		TP	25	33	33	35	33
UCI	r=0.1	Precision	0.47	0.46	0.51	0.49	0.49
German		Recall	0.78	0.73	0.73	0.78	0.73
data		Accuracy	0.67	0.67	0.71	0.69	0.69
		f1-score	0.58	0.57	0.60	0.60	0.58
		AUC	0.70	0.69	0.71	0.72	0.70
		TN	63	73	73	69	70
		FP	42	32	32	36	35
		FN	7		11	13	11
		TP	38	34	32	34	34
UCI	r=0.2	Precision	0.48	0.52	0.50	0.49	0.49
German		Recall	0.84	0.76	0.71	0.76	0.76
data		Accuracy	0.67	0.71	0.70	0.69	0.69
		f1-score	0.61	0.61	0.59	0.59	0.60
		AUC	0.72	0.73	0.70	0.71	0.71
		TN	64	71	74	70	70
		FP	41	34	31	35	35
		FN	8	13	13	11	12
		TP	37	32	32	34	33
UCI	r=0.3	Precision	0.47	0.48	0.51	0.49	0.49
German		Recall	0.82	0.71	0.71	0.76	0.73
Data		Accuracy	0.67	0.69	0.71	0.69	0.69
		f1-score	0.60	0.58	0.59	0.59	0.58
		AUC	0.70	0.69	0.71	0.71	0.70

Table 4.13: sensitivity analysis for HDRCS for UCI German datasets

lem in credit risk prediction. The HDRCS algorithm combines Gaussian mixture modelling (GMM) for generating synthetic minority classes and k-means clustering for creating a new data set for the majority class, resulting in a dataset with a desired imbalance ratio suitable for cost-sensitive logistic regression prediction operations. The HDRCS algorithm offers a

Data Set	r-value	Metric	HDRCS Model				
			$\lambda^* = 1$	$\lambda^* = 1.22$	$\lambda^* = 1.5$	$\lambda^* = 1.67$	$\lambda^* = 1.86$
		TN	138	138	135	136	138
		FP	2	2	5	4	2
		FN	6	6	5	6	5
		TP	5	5	6	5	6
GiveMe	r=0.0	Precision	0.71	0.71	0.55	0.56	0.75
Credit		Recall	0.45	0.45	0.55	0.45	0.55
Data		Accuracy	0.95	0.95	0.93	0.93	0.95
		f1-score	0.56	0.56	0.55	0.50	0.63
		AUC	0.72	0.72	0.75	0.71	0.77
		TN	137	138	138	138	138
		FP	3	2	2		2
		FN	6	6	5	6	5
		TP	5	5	5	6	5
GiveMe	r=0.1	Precision	0..63	0.71	0.75	0.75	0.75
Credit		Recall	0.45	0.45	0.55	0.45	0.55
Data		Accuracy	0.94	0.95	0.95	0.95	0.95
		f1-score	0.53	0.56	0.63	0.56	0.63
		AUC	0.72	0.72	0.77	0.72	0.77
		TN	137	135	125	138	138
		FP	3	5	15	2	2
		FN	6	6	1	5	6
		TP	5	5	10	6	6
GiveMe	r=0.2	Precision	0.63	0.50	0.40	0.75	0.55
Credit		Recall	0.45	0.45	0.91	0.55	0.55
data		Accuracy	0.94	0.93	0.89	0.95	0.95
		f1-score	0.53	0.48	0.56	0.63	0.63
		AUC	0.72	0.71	0.90	0.77	0.77
		TN	138	138	138	138	138
		FP	2	2	2	2	2
		FN	6	6	5	5	5
		TP	5	5	6	6	6
GiveMe	r=0.3	Precision	0.71	0.71	0.75	0.75	0.75
Credit		Recall	0.45	0.45	0.55	0.55	0.55
Data		Accuracy	0.95	0.95	0.95	0.95	0.95
		f1-score	0.56	0.56	0.63	0.63	0.63
		AUC	0.72	0.72	0.77	0.77	0.77

Table 4.14: sensitivity analysis for HDRCS for Give Me Credit datasets

comprehensive hybrid approach that leverages both data-level and algorithm-level techniques to address the class imbalance problem in the credit risk domain. Its effectiveness in achieving balanced datasets and accurate predictions makes it a valuable tool for financial risk management. The performance of the HDRCS algorithm on four real-world credit risk

Data Set	r-value	Metric	HDRCS Model				
			$\lambda^* = 1$	$\lambda^* = 1.22$	$\lambda^* = 1.5$	$\lambda^* = 1.67$	$\lambda^* = 1.86$
Lending Club Data	r=0.0	TN	155	162	144	162	141
		FP	7	0	18	0	21
		FN	6	8	4	6	4
		TP	13	11	15	13	15
		Precision	0.65	1	0.45	1	0.42
		Recall	0.68	0.58	0.79	0.68	0.79
		Accuracy	0.93	0.96	0.88	0.97	0.86
		f1-score	0.67	0.73	0.58	0.81	0.55
		AUC	0.82	0.79	0.84	0.84	0.83
Lending Club Data	r=0.1	TN	157	158	155	155	155
		FP	5	4	7	7	7
		FN	6	8	5	4	4
		TP	13	11	14	15	15
		Precision	0.72	0.73	0.67	0.68	0.68
		Recall	0.68	0.58	0.74	0.79	0.79
		Accuracy	0.94	0.93	0.93	0.94	0.96
		f1-score	0.70	0.65	0.70	0.73	0.73
		AUC	0.83	0.78	0.85	0.87	0.87
Lending Club data	r=0.2	TN	143	160	162	162	162
		FP	19	2	0	0	0
		FN	4	6	7	7	8
		TP	15	13	12	12	11
		Precision	0.44	0.87	1	1	1
		Recall	0.79	0.68	0.63	0.63	0.58
		Accuracy	0.87	0.96	0.96	0.96	0.96
		f1-score	0.57	0.76	0.77	0.73	0.73
		AUC	0.84	0.84	0.82	0.82	0.79
Lending Club Data	r=0.3	TN	154	162	144	162	141
		FP	8	0	18	0	21
		FN	4	7	4	8	4
		TP	15	12	15	11	15
		Precision	0.65	1	0.45	1	0.42
		Recall	0.79	0.63	0.79	0.58	0.79
		Accuracy	0.93	0.96	0.88	0.96	0.88
		f1-score	0.71	0.77	0.58	0.73	0.55
		AUC	0.87	0.82	0.84	0.79	0.83

Table 4.15: sensitivity analysis for HDRCS for Lending Club datasets

datasets which are; LCU (a), LCU (b), UCI German, Give Me Some Credit, and Lending Club, has been evaluated. The experimental results have demonstrated that the proposed HDRCS algorithm outperformed other algorithms in most of the metrics used. The experimental results have demonstrated the effectiveness of the HDRCS algorithm compared to other resampling approaches, including SMOTE, CBUT, and GMMOT. The recommended heuristic Imbalance Ratio provides a practical means to achieve better balancing. It is worth noting that while our proposed method achieved some superior results, it has its own limitations which are:

- Data Dependency: The HDRCS algorithm's performance depends on the quality and the ability of the GMM to accurately capture the distributions of the minority class data used.
- Complexity in Implementation: The hybrid nature of the model, combining resampling and cost-sensitive approaches, can lead to increased complexity in implementation and parameter tuning.
- Class Imbalance Sensitivity: While designed to address class imbalance, the technique might still be sensitive to extreme imbalances.
- Computational Cost: Given its hybrid approach, the HDRCS model might incur higher computational costs compared to simpler models.

While addressing data imbalances improved the accuracy of credit risk predictions, a comprehensive risk management strategy also requires precise volatility forecasting. Therefore, in the following chapter, we will focus on the use of Continuous Wavelet Transform Triple Discriminator GAN (cwt-TriGAN) to enhance the forecasting accuracy of Volatility.

Chapter 5

Continuous wavelet Transform Triple Discriminator GAN (cwt-TriGAN) for Volatility Forecasting

5.1 Introduction

Building on the enhanced predictive models for credit risk developed in the previous chapter, this chapter explores how the forecasting of volatility, which is an essential component in the understanding of market risks can be improved. In recent years, volatility forecasting has attracted renewed attention after several major financial market crises. Accurate volatility forecasting plays an important role in the stock market and is required by both market participants and policymakers (Liu and Pan, 2020). Modern portfolio theory (Markowitz, 1952) has a positive trade-off between risk and return as its cornerstone, which therefore motivates the need for accurate forecasting and estimation of risk. The accurate description of how stock prices and stock indices fluctuate and the determination of the future rate of return of the stock market have attracted attention from both the research and investment communities (Lin, 2018).

Machine learning techniques such as Artificial Neural Networks (ANN), fuzzy logic, Support Vector Machines (SVM), and particle swarm optimisation have been used for modelling and forecasting of financial data (Liu, 2019). These machine learning models have the ability to fit financial time series data better than the classical models due to their ability to learn patterns of complex data without any prior assumptions about their distribution (Lotfi et al., 2016; Kim et al., 2009). Several research studies have used ANN to forecast the volatility of financial data; the most notable is the use of ANN to forecast the directional movement of the implied volatility of the derivatives market (Ahn et al., 2012). Over the past few years, there has been renewed interest in the use of deep learning models for the prediction of financial time series data (Bao et al., 2017). Models such as Gated Recurrent Unit (GRU), Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Deep Belief Networks (DBN) (Hinton et al., 2006), and other similar architectures have been used for time series predictions.

Generative adversarial networks (GAN) (Goodfellow et al., 2014), are a type of machine learning algorithm used in unsupervised learning. It is composed of two parts: a Generator network and a Discriminator network. The Generator produces samples, while the Discriminator attempts to determine whether a given sample is real or fake. GANs (Goodfellow et al., 2014) as an emerging research field in finance have been applied to financial datasets. They have also been used for stock price prediction through the optimisation of the prediction model to minimise the loss of the Discriminator to help predict the price movement (Zhou et al., 2018). Additionally, GANs have also been used to develop systematic trading strategies and discover combinations of trading strategies (Koshiyama et al., 2019) and have been used for fraud detection (Fiore et al., 2017). TimeGAN (Yoon et al., 2019) was proposed to generate realistic time series data by adding an auto-regressive model to unsupervised models in their GAN network.

The use of machine learning models for financial risk management is limited by noisy data and a lack of data availability, as observed by Caiafa et al. (2021). Market noise, statistical noise and information noise are types of noise found in financial data. Therefore, it is essential that these limitations are adequately investigated and a solution proposed. Therefore, the wavelet transform has been widely used to filter and extract single-dimensional signals due to its ability to fix the noisy features of financial time series (Hsieh et al., 2011). Therefore, it is an effective method to map the dynamic nature of non-stationary financial data (Ramsey, 1999). The success of a deep learning algorithm is dependent on the loss function used (Zabihzadeh, 2021). There are different loss functions, each with its own strengths and weaknesses since no single loss function is perfect. The strength of each of them covers only those with some sort of optimal similarity embedding when training. Ensemble loss functions (Fort et al., 2019; Hajiabadi et al., 2020) involve the combination of losses to achieve superior results. Through this, the coefficients of loss functions and weight parameters of an artificial neural network are jointly learned through back-propagation.

Due to the crucial role volatility plays in portfolio management and risk hedging, it is imperative to have reliable methods for its estimation and forecasting. Therefore, it is essential to have a robust preprocessing mechanism to remove the noise associated with it and also transform its 1-dimensional (1-D) feature space into a high-dimensional feature space. This makes it suitable as an input for the GAN forecasting model. This is because volatility, which is one-dimensional (1-D) time series data and noisy in nature, is not suitable to use as an input for GAN; therefore, this chapter proposes a method that combines GANs with Continuous Wavelet Transform (CWT) (Torrence and Compo, 1998) to achieve this. It uses CWT to decompose noisy and non-stationary data into de-noised wavelets as input for the GAN model. Our proposed architecture employs supervised loss, reconstruction loss, and discriminators ensemble loss derived from the use of a triplicate Discriminator to overcome the problems associated with the use of GAN as a forecasting tool, such as mode collapse and model in-convergence, which makes it difficult to train. Existing methods often struggle to capture the complex and dynamic nature of financial time series data. In response, a novel technique that provides a comprehensive solution is presented. Our proposed approaches, which serve as our main contributions, are as follows:

- Continuous Wavelet Transform Triple Generative Adversarial Network (cwt-TriGAN): This uses a Triple Discriminator network as part of a GAN model to overcome the problem of mode collapse normally encountered when using GANs for prediction. It uses a supervised loss, a reconstruction loss, and discriminator losses to train a GAN model for volatility forecasting. Discriminator Losses leverage a novel ensemble loss function made up of hinge loss, least square loss function, and cross-entropy loss to train the model.
- Investigation into the use of continuous wavelet transform on nine financial datasets to improve the forecasting and data-generation ability of volatility time-series using cwt-TriGAN architecture. A detailed volatility analysis using a scaleogram has also been contributed.

The rest of the chapter is organised as follows: The related work is provided in Section 5.2. Our proposed cwt-TriGAN and its architecture are introduced in Section 5.3. Our experimental setup and results are presented in Section 5.4. The summary, containing the limitations of this work is captured in Section 5.5.

5.2 Related Works

Most of the successful deep learning applications used in different time series domains have been reviewed (Hassan et al., 2019). Recurrent Neural Network (RNN) has been highlighted as a type of deep neural network that can be used for sequential and time-series modelling (Xing et al., 2019). Variational RNN, which is a hybrid, has been used to jointly model sequential data using its' mean and variance (Chung et al., 2015). LSTM has also been used in high-frequency data from four future markets, namely petroleum, oil, flammable gas, and fuel, to demonstrate its effective use in predicting volatility (Prokopczuk and Simen, 2014). GANs were used for the generation of time series data in which the data generated from the previous time step and the noise vector were used as input to generate new data recurrently (Mogren, 2016). (Esteban et al., 2017). However, many research studies have directly applied the GAN framework in a temporal setting. The first continuous recurrent neural network GAN (C-RNN-GAN) (Mogren, 2016) applied the GAN architecture to sequential data using LSTM networks as both a Generator and a Discriminator. It involved the use of data generated from the previous time step and a noise vector as input to generate new data recurrently (Mogren, 2016). It was then followed by the work of Esteban et al. (2017), where small architectural differences such as eliminating the dependence on the previous output were introduced while conditioning on additional input to generate real-valued multidimensional time series medical records data obtained from the intensive care unit. Since then, many studies have used these frameworks to generate synthetic sequences in various domains such as finance (Zhang et al., 2019), text (Zhang et al., 2016) and bio-signals (Haradal et al., 2018). GANs have also been used for anomaly detection, such as MAD-GAN (Li et al., 2019), where multivariate anomaly detection was used to detect anomalies in time series data. They have also been used for stock price prediction (Zhang et al., 2019) such as ForGAN (Koochali et al., 2019), which is a one-step-ahead probabilistic forecasting with generative adversarial networks that uses a conditional generative adversarial network (CGAN) (Mirza and Osindero, 2014) and Fin-GAN (Vuletić et al., 2023). A Conditional Generative Adversarial Network (CGAN)(Mirza and Osindero, 2014) can learn and simulate categorical and continuous financial time series variables (Fu et al., 2020) because CGAN has the ability to learn normal and heavy tail distributions and, as such, can be used to generate conditional predictive distributions consistent with training data distributions. A deep generative adversarial network (D-GAN)(Saxena and Cao, 2019) was proposed for accurate prediction of spatio-temporal events (ST) and involved the combination of GAN and VAE to jointly learn the generation and variational inference of data in an unsupervised manner.

TimeGAN(Yoon et al., 2019) was proposed to generate realistic time series data by combining deep auto-regressive models with unsupervised models for supervised loss and the GAN network. An Adversarial Attack on Probabilistic Autoregressive Forecasting Models (Ad-Attack) (Dang-Nhu et al., 2020) has also been proposed. This used a GAN framework to generate attacks with small input perturbations in stock market trading and prediction of electricity prices. Further research in the field involved the proposal of SimGAN (Golany et al., 2020). This used a system of ordinary differential equations representing heart dynamics and incorporated them into the optimisation process of a generative adversarial network to synthesise biologically plausible heart signal electrocardiogram (ECG) training data samples. DeepTrader (Wang et al., 2021) has also been proposed, where GAN was used to predict the volatility of stock prices using the daily closing price of the S & P 500 index. A combination of wavelet transform and convolutional Neural Networks (CNN) has been used for time series prediction (Mittelman, 2015). Wavelet transforms (WT) were also combined with stacked Auto-encoders (SAEs) and long-short-term memory (LSTM) to eliminate noise and were used for stock price forecasting (Bao et al., 2017).

Our proposed cwt-TriGAN is described in the next section.

5.3 Proposed Triple Discriminator GAN with Continuous wavelet Transform (cwt-TriGAN) for Volatility Forecasting

In this section, the method used to implement our Triple Discriminator GAN with cwt is presented. An overview of the cwt-TriGAN, which includes coverage of the Auto-encoder and GAN components, is first introduced. It is then followed by the description of the architecture of the cwt-TriGAN. The final part covers all three loss functions and training schemes used.

Overview of cwt-TriGAN

To generate a de-noised signal for a one-step-ahead volatility prediction, a deep learning framework for financial time series that utilises a deep learning-based forecasting scheme with integrated stacked auto-encoders and LSTM-based GAN were used. Figure 5.1 represents the network architecture and network flow of the proposed cwt-TriGAN.

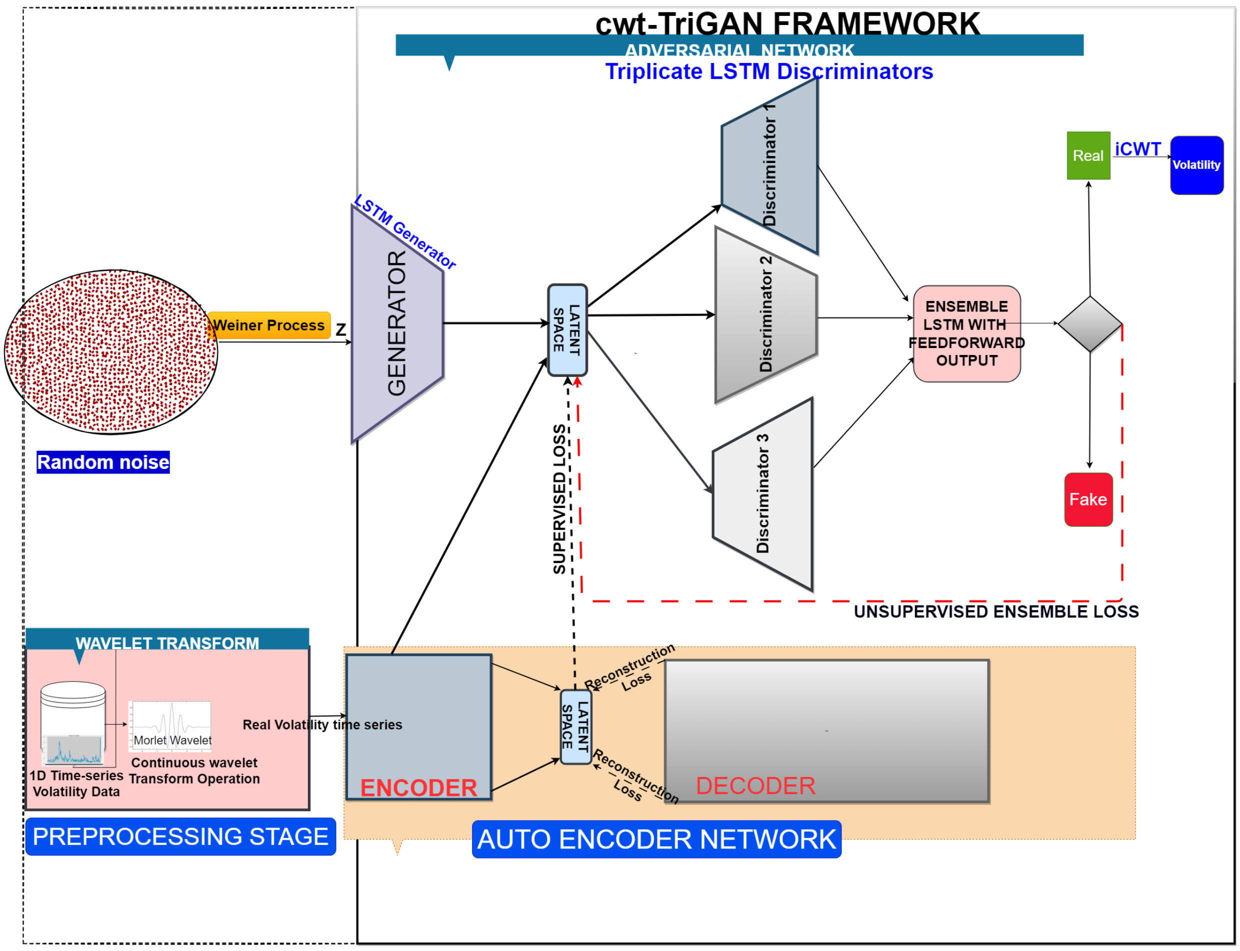


Figure 5.1: cwt-TriGAN Architecture showing an Auto-encoder component made of Encoder and Decoder networks, an adversarial component containing a Generator network, G responsible for the generation of the synthetic data and 3 Discriminator networks, D_1, D_2, D_3 responsible for discriminating and evaluating the output from the single Generator network.

The four separate components under our proposed cwt-TriGAN are:

1. Data preprocessing component that entailed the application of the continuous wavelet transform to the volatility signal as a de-noising and decomposition tool. This operation decomposed the volatility data into coefficients and resulted in an increase in the dimensionality of the 1-D volatility time-series data to a 2-D image, with an increase in the dimensions and hidden dimensions of the data based on the scale factor used in the continuous wavelet transform process.

2. Stacked Auto-encoders component: This has a deep architecture trained in an unsupervised manner and made up of Encoder and Decoder Networks. This worked as a function to provide a mapping between the features and latent representations to reduce the high dimensionality of the adversarial learning space.

3. LSTM-based GAN network component: This was made up of a single sequence Generator and Triple Discriminator network in a 1-versus-3 minimax game, where the Generator has the ability to generate a one-step-ahead output whilst the combined power of the triplicate discriminators is able to effectively distinguish the real data from generated data through a robust adversarial training process.

4. Reconstruction process that transforms the preprocessed data cwt back to the original volatility. This allowed the original signal to be recovered from its wavelet transform by integrating over all scales and locations. The original wavelet function was used, instead of its conjugate which is employed in the forward transformation of the equation. The CWT used in this study is non-orthogonal, making it highly redundant at large scales, and the wavelet spectrum at adjacent times is highly correlated. As suggested by Torrence and Compo (1998), this non-orthogonal transform is important for time series analysis, where smooth, continuous variations in wavelet amplitude are expected and the redundancy nature led to reconstruction loss, which was calculated using the mean squared error between the original volatility and the reconstructed signal.

Architecture of Proposed cwt-TriGAN

The cwt-TriGAN consists of eight functions which are: a wavelet transform function, an inverse wavelet transform function, an Encoder network, a Decoder network, a sequence Generator, and three sequence discriminators. The wavelet transform component is made up of the cwt and the inverse cwt functions. These two, respectively, act as a de-noising tool for the data at the preprocessing stage and map it back to the original volatility data as the last stage through a reconstruction procedure after the GAN operation. It also contains an auto-encoder component; made up of the encoding and decoding networks which are jointly trained with the Generator and the Triple Discriminator networks referred to as the adversarial components, in such a way that cwt-TriGAN simultaneously learns to denoise the data, encode the temporal features, generate samples, and iterate across time.

Data preprocessing stage using Continuous Wavelet Transform (CWT)

The CWT remains an excellent tool for mapping the changing properties of non-stationary signals. As a result of this, it can be used to preprocess the volatility dataset using a desired scale factor. The first step in the CWT is to find a choice of mother wavelet to use. The Morlet wavelet(Lee et al., 2019) was used in this procedure. The Morlet wavelet was used because of its excellent balance between frequency and time localisation, its' analytic nature and its' shape that is similar to financial signals, which can lead to effective feature extraction. Other reasons include its' smoothness, parameter flexibility, and robustness to non-stationarities. The wavelet is then taken and compared with a section at the start of the original volatility data. The next step involves the calculation of wavelet coefficients, which represent how correlated the wavelet is with the selected section of the original signal. The correlation between sections of the volatility data and scaled versions of the wavelet can then be quantified. The wavelet is shifted to the right and can be repeated until the entire length of the volatility signal is covered. The procedure is then repeated with several scaled versions of the wavelet as the next step. This results in the production of coefficients produced at different scales by different sections of the volatility signal. The coefficients represent the results of a regression of the original volatility signal performed on the Morlet wavelets. Through this operation, the data set is decomposed into coefficients and frequencies.

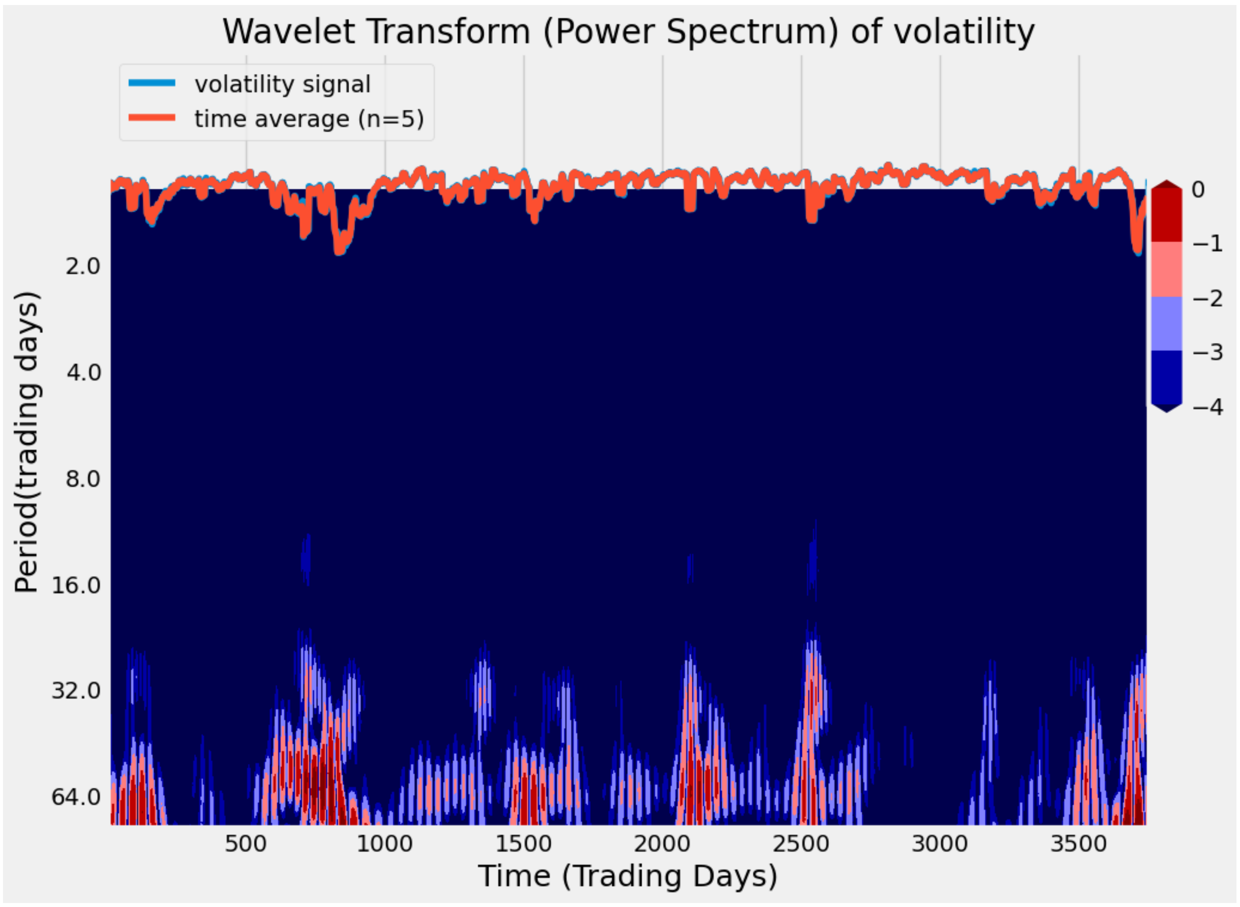


Figure 5.2: scaleogram of Google dataset

Each wavelet coefficient has elements of frequency (through scaling) and time (through shifting). Deriving scale- and time-dependent patterns from the time series requires a graphical and intuitive representation of the wavelet coefficients. This is achieved by plotting the information using three-dimensional time-scale-coefficient axes. Such a plot is referred to as a scaleogram as shown in Figure 5.2. The coefficients obtained can be used as input data for the deep learning model. Figure 5.2 shows the scaleogram image produced as a result of the CWT operations on the Google data sets used.

A summary of the Continuous wave Transform Algorithm in the form of a pseudo code is shown in algorithm 4

Algorithm 4 Continuous wave Transform Algorithm

```
    Require: For a given data set processed to 1-D Volatility, $\mathbf{v}$
    Ensure: The choice of mother wavelet is Morlet.
1: Split the Volatility data , $\mathbf{v}$ into training set , $\mathcal{V}$ and test set, $\mathcal{V}_{ts}$
2: Take the Morlet wavelet and compare it with a section at the start of the training set, $\mathcal{V}$
3: Calculate the correlation coefficient between the Morlet wavelet and training set, $\mathcal{V}$
4: Shift the Morlet wavelet to the right and repeat Steps 1 and 2 until the whole training set , $\mathcal{V}$ is covered.
5: Scale the Morlet wavelet and repeat steps 2 through 4
6: Repeat steps 2 through 5 for all scales
7: Return Coefficients
```

Auto-Encoder component

The Auto-encoder component of the cwt-TriGAN is composed of an Encoder network and a Decoder network, which collectively form a mapping between the wavelet coefficients and a latent space. This latent space enables the GAN to learn the underlying temporal dynamics of the coefficients through a reduced-dimensional representation. This section provides a comprehensive overview of the auto-encoder component, including its architectural details and mathematical representations. A diagrammatic representation of the auto-encoder component of the cwt-TriGAN is provided in Figure 5.3.

Auto-encoder Architecture

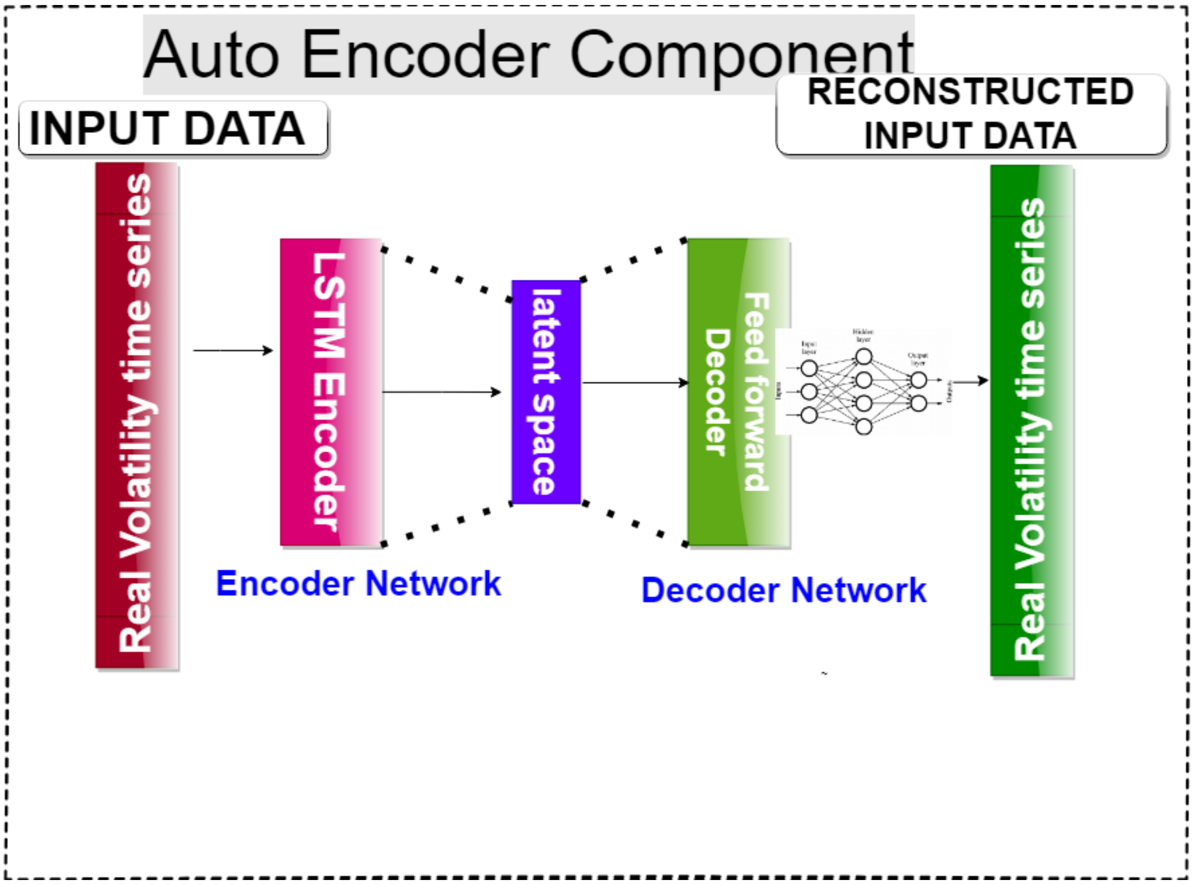


Figure 5.3: Diagramatic representation of the Auto-encoder component of cwt-TriGAN

The Encoder network consists of LSTM cells that encode the temporal features of the coefficients while the Decoder network consists of a simple feed-forward network that decodes the latent representation as reconstructed input data. Both the encoding and decoding networks are auto-regressive in nature to maintain the temporal dynamics of the coefficients of volatility time series data. The Encoder function used the recurrent encoding network for the coefficients feature in the data.

Conversely, the Decoder network performs the inverse operation, converting the temporal codes of the coefficients back into their original feature representation. The Encoder and Decoder networks establish a reversible mapping between the feature and latent spaces to enable accurate reconstructions of the original coefficients data from their latent representations. The **Reconstruction loss**, $\mathcal{L}_R$ that serves as the first objective function involved in this procedure.This loss function quantifies the difference between the actual coefficients and their reconstructions to show the accuracy of the learned latent representation in capturing the underlying data patterns.

Adversarial components of cwt-TriGAN

The Adversarial components of cwt-TriGAN encompass a Generator network and Triple Discriminators. The Generator network uses an LSTM architecture to produce synthetic latent codes in the embedding space. The Generator network accepts the coefficients of temporal random noise and uses the LSTM Generator network, G to map these random vectors into synthetic latent codes, $\hat{\mathbf{h}}_T = G(\mathbf{z}_T)$.

The Generator also receives two types of data input during the training phase. The first one is the synthetic embeddings coming from its own previous outputs, which can be used to generate the next synthetic vector in an auto-regressive manner, and the second is the sequence of the encoded coefficients from CWT operations received by the Generator to produce the latent vector.

Supervised Loss

A supervised loss function ($\mathcal{L}_S$) is introduced to aid in the learning of the distribution since the use of only the adversarial losses of the Triple Discriminators is not adequate for the Generator network to capture the step-wise conditional temporal distribution of the coefficients. The mathematical representation of the supervised loss ($\mathcal{L}_S$) obtained through the application of the maximum likelihood is demonstrated in equation (5.1):

$$\mathcal{L}_S = \mathbb{E}_{C_{T \sim p}} \left[\sum_t \| \mathbf{h}_t - G_{\mathcal{C}} \left(\mathbf{h_{t-1}}, \mathbf{Z_t} \right) \|_2 \right] \tag{5.1}$$

Generator Losses

The Generator loss ($\mathcal{L}_G$) is calculated as:

$$\mathcal{L}_G = w_1 \cdot \mathcal{L}_{G_1} + w_2 \cdot \mathcal{L}_{G_2} + w_3 \cdot \mathcal{L}_{G_3}$$

with hyper-parameters w_1, w_2, w_3 controlling the relative contribution of each component. Where: L_{G_1} is the cross entropy loss component of the Generator loss L_{G_2} is the least square loss component of the Generator loss L_{G_3} is the hinge loss component of the Generator loss. These are represented mathematically and shown in equations (5.2), (5.3) and (5.4) respectfully.

$$\mathcal{L}_{G_1} = -\mathbb{E}\left[\sum_t \log(\hat{y}_t)\right] \tag{5.2}$$

$$\mathcal{L}_{G_2} = \mathbb{E}\left[\sum_t (\hat{y}_t - 1)^2\right] \tag{5.3}$$

$$\mathcal{L}_{G_3} = -\mathbb{E}\left[\sum_t (\hat{y}_t)\right] \tag{5.4}$$

Hence the total Generator loss, $\mathcal{L}_{G_T}$ is calculated as shown in equation (5.5):

$$\mathcal{L}_{G_T} = \gamma \mathcal{L}_S + \mathcal{L}_G \tag{5.5}$$

Where γ is a hyper-parameter for combining the two losses.

Triple Discriminators

The Adversarial component involves a set of three discriminators, each operating from the embedding space. The purpose of these three discriminators is to maximise their ability to distinguish between real and fake data instances. This loss term encourages the Generator to produce data that are indistinguishable from real data, as shown in Figure 5.4

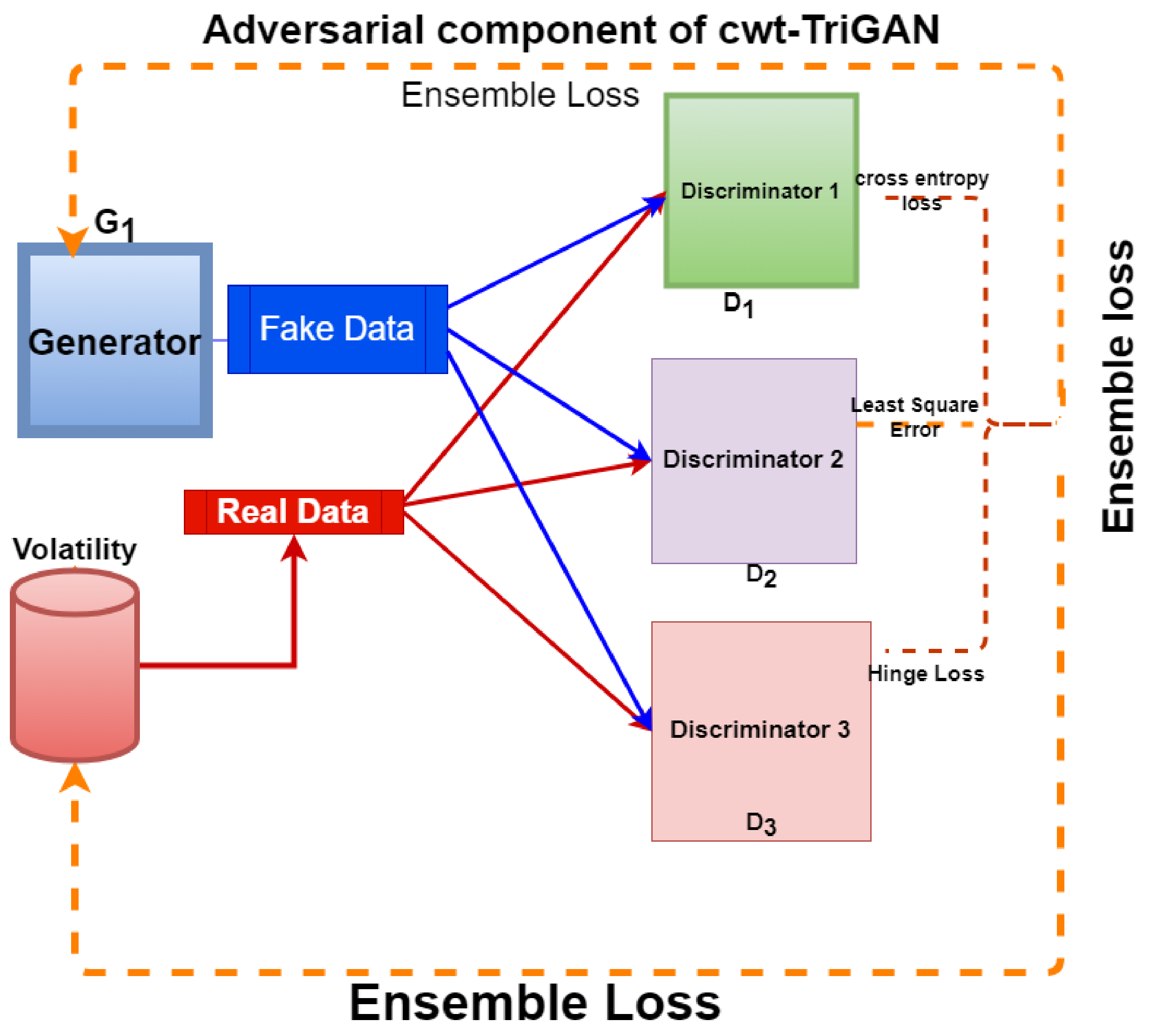


Figure 5.4: Adversarial component of the cwt-TriGAN showing Ensemble loss made up of weighting of w_1 for cross-entropy loss, w_2 for least square loss and w_3 for Hinge loss functions respectively for discriminators D_1, D_2 and D_3.

The discrimination functions denoted as $D_1, D_2,$ and D_3 for discriminators 1, 2 and 3 respectively are shown mathematically as follows:

$$D_1 : \prod_t \mathcal{H}_{\mathcal{C}1} \rightarrow [0,1]_1 \times \prod_{t_1} [0,1]_1 \,;$$
$$D_2 : \prod_t \mathcal{H}_{\mathcal{C}2} \rightarrow [0,1]_2 \times \prod_{t} [0,1]_2 \,;$$
$$D_3 : \times \prod_t \mathcal{H}_{\mathcal{C}3} \rightarrow [0,1]_3 \times \prod_{t} [0,1]_3$$

These three Discriminator networks receive the coefficient codes and classify them as real or fake for both the embeddings and data. Each of the components of the Triple Discriminator utilises a bidirectional LSTM in conjunction with a feed-forward network for classification and can be implemented mathematically as shown in equations (5.6)

$$\tilde{y}_t = \sum_{i=1}^{3} (D_{\mathcal{C}})_i \left(\overleftarrow{\mathbf{q}}_t, \overrightarrow{\mathbf{q}}_t \right) \text{ for } i = 1, 2, 3 \tag{5.6}$$

With:

$$\overrightarrow{\mathbf{q}}_t = \sum_{i=1}^{3} \overrightarrow{f}_{\mathcal{C}_i} \left(\tilde{\mathbf{h}}_t, \overrightarrow{\mathbf{q}}_{(t+1)} \right) \text{ for } \quad i = 1, 2, 3$$
$$\overleftarrow{\mathbf{q}}_t = \sum_{i=1}^{3} \overleftarrow{f}_{\mathcal{C}_i} \left(\tilde{\mathbf{h}}_t, \overleftarrow{\mathbf{q}}_{(t+1)} \right) \text{ for } \quad i = 1, 2, 3$$

where:
$D_{\mathcal{C}_i}$ are the respective classification layers for the output layers for Discriminators D_1, D_2, D_3
$\overrightarrow{\mathbf{q}}_t$ represents the sequence of forward states for Discriminators D_1, D_2, D_3 .
$\overleftarrow{\mathbf{q}}_t$ represents the sequence of backward states for Discriminators D_1, D_2, D_3 operations.
$\overrightarrow{f}_{\mathcal{C}_i}$ represents the LSTM of the forward states for discriminators D_1, D_2, D_3 operations.
$\overleftarrow{f}_{\mathcal{C}_i}$ represents the LSTM of the backward states for Discriminators D_1, D_2, D_3 operations.
Let $\boldsymbol{\theta}_r$ $\boldsymbol{\theta}_e$ be the parameters of the Encoder network $\boldsymbol{\theta}_r$ be the parameters of the Decoder network. $\boldsymbol{\theta}_g$ be the parameters of the Generator network. $\boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \text{and } \boldsymbol{\theta}_{d_3}$ be the parameters for the Triple Discriminator networks $D_1, D_2, \text{and } D_3$ respectively.

Gradients $\boldsymbol{\theta}_g, \boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \text{and } \boldsymbol{\theta}_{d_3}$ are computed on the Discriminator Losses, allowing the Triple discriminators to maximise their probability of providing the correct classifications for both the training data and the synthetic data from the Generator. The average of the parameters of the 3 discriminators, $\overline{\boldsymbol{\theta}}_d$ is computed as shown in Equation 5.7.

$$\overline{\boldsymbol{\theta}}_d = \left(\frac{\boldsymbol{\theta}_{d_1} + \boldsymbol{\theta}_{d_2} + \boldsymbol{\theta}_{d_3}}{3} \right) \tag{5.7}$$

Discriminator Losses

The Discriminator losses ($\mathcal{L}_{U_i}$) for each Discriminator D_1, D_2 and D_3 can be calculated as follows: The Cross entropy loss used for Discriminator D_1 was calculated as shown in equation 5.8:

$$\mathcal{L}_{U_1} = \mathbb{E} \left[\sum_t \log y_t \right] + \mathbb{E} \left[\sum_t \log(1 - \hat{y}_t) \right] \tag{5.8}$$

The $\mathcal{L}_{U_2}$ used for Discriminators D_2 computed by the least Squares loss was calculated using equation 5.9:

$$\mathcal{L}_{U_2} = \mathbb{E}\left[\sum_t (y_t - 1^2)\right] + \mathbb{E}\left[\sum_t (\hat{y}_t)^2\right] \tag{5.9}$$

The hinge loss used for Discriminators D_3 is given as shown in equation 5.10:

$$\mathcal{L}_{U_3} = \mathbb{E}\left[\sum_t \min(0, -1 + y_t)\right] + \mathbb{E}\left[\sum_t \min(0, -1 - \hat{y}_t)\right] \tag{5.10}$$

The discriminator loss $\mathcal{L}_U$ for Discriminators 1, 2 and 3 can be computed by finding the average of individual Discriminator losses ($\mathcal{L}_{U_i}$) as shown in Equation (5.11).

$$\mathcal{L}_U = \sum_{i=1}^{3} w_i \cdot \mathcal{L}_{U_i} \text{ for } \quad \text{i=1,2,3} \tag{5.11}$$

Where w_i is the proportional weights used for the $\mathcal{L}_{U_1}$ $\mathcal{L}_{U_2}$and $\mathcal{L}_{U_3}$ for Discriminators D_1, D_2, D_3.

Optimisation process for CWT-TriGAN Training

The cwt-TriGAN is trained in a closed-loop fashion, with the Generator receiving sequences of embeddings of actual volatility data. The Generator then generates the subsequent hidden vector, and gradients are computed based on the difference between the distributions. The training scheme ensures that the Generator learns to create realistic-looking volatility data through adversarial training while preserving accurate step-wise conditional temporal distribution patterns.

The Discriminator ensemble loss function ($\mathcal{L}_U$) forces the Generator to create realistic-looking coefficients of volatility data through adversary training, and the supervised loss, $\mathcal{L}_S$ also further ensures that it produces similar step-wise transitions evaluated by ground-truth targets. The optimisation and training scheme of the proposed method is diagrammatically represented in Figure 5.5.

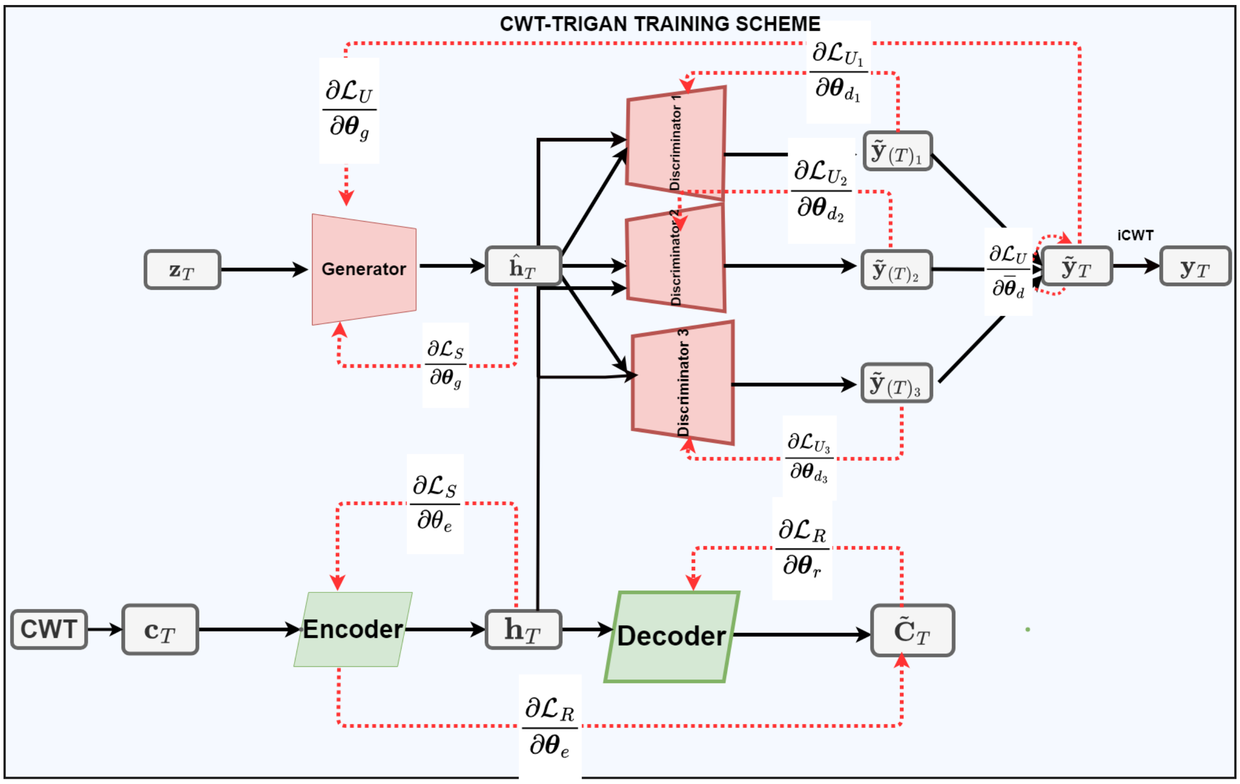

CWT-TRIGAN TRAINING SCHEME
$\frac{\partial \mathcal{L}_U}{\partial \boldsymbol{\theta}_g}$
$\frac{\partial \mathcal{L}_{U_1}}{\partial \boldsymbol{\theta}_{d_1}}$
$\frac{\partial \mathcal{L}_{U_2}}{\partial \boldsymbol{\theta}_{d_2}}$
$\frac{\partial \mathcal{L}_{U_3}}{\partial \boldsymbol{\theta}_{d_3}}$
$\frac{\partial \mathcal{L}_U}{\partial \bar{\boldsymbol{\theta}}_d}$
$\frac{\partial \mathcal{L}_S}{\partial \boldsymbol{\theta}_g}$
$\frac{\partial \mathcal{L}_S}{\partial \theta_e}$
$\frac{\partial \mathcal{L}_R}{\partial \boldsymbol{\theta}_r}$
$\frac{\partial \mathcal{L}_R}{\partial \boldsymbol{\theta}_e}$
$\mathbf{z}_T$
Generator
$\hat{\mathbf{h}}_T$
Discriminator 1
Discriminator 2
Discriminator 3
$\tilde{\mathbf{y}}_{(T)_1}$
$\tilde{\mathbf{y}}_{(T)_2}$
$\tilde{\mathbf{y}}_{(T)_3}$
$\tilde{\mathbf{y}}_T$
iCWT
$\mathbf{y}_T$
CWT
$\mathbf{c}_T$
Encoder
$\mathbf{h}_T$
Decoder
$\tilde{\mathbf{C}}_T$

Figure 5.5: The training scheme of the CWT-TriGAN where the solid lines represent forward propagation and dotted red lines represent backward propagation

Let $\boldsymbol{\theta}_e, \boldsymbol{\theta}_r, \boldsymbol{\theta}_g, \boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \boldsymbol{\theta}_{d_3}$ denote the parameters of the Encoder, Decoder, Generator and Triple Discriminator networks respectively.

The $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_r$ jointly train on the reconstruction loss and the supervised loss as shown as:

$$\min_{\theta_e,\theta_r} (\lambda \mathcal{L}_S + \mathcal{L}_R)$$

where $\lambda \geq 0$ is a hyper-parameter used to balance $\mathcal{L}_S$ and $\mathcal{L}_R$. Using the average of the parameters of the 3 discriminators, $\bar{\boldsymbol{\theta}}_d$ resulted in the adversarial training of the Generator and the Triple Discriminator networks as shown in Equation (5.12).

$$\min_{\boldsymbol{\theta}_g} \left(\gamma \mathcal{L}_S + \max_{\bar{\boldsymbol{\theta}}_d} \mathcal{L}_U \right) \tag{5.12}$$

where $\gamma \geq 0$ is a hyper-parameter that balances the $\mathcal{L}_S$and$\mathcal{L}_U$

The parameters $\boldsymbol{\theta}_e, \boldsymbol{\theta}_r, \boldsymbol{\theta}_g, \boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \boldsymbol{\theta}_{d_3}\overline{\boldsymbol{\theta}}_d$ are also updated as follows:

$$
\begin{aligned}
\boldsymbol{\theta}_e &= \boldsymbol{\theta}_e - \gamma \bigtriangledown_{\boldsymbol{\theta}_e} -[\lambda \mathcal{L}_S + \mathcal{L}_R] \\
\boldsymbol{\theta}_r &= \boldsymbol{\theta}_r - \Gamma \bigtriangledown_{\boldsymbol{\theta}_r} -[\lambda \mathcal{L}_S + \mathcal{L}_R] \\
\boldsymbol{\theta}_g &= \boldsymbol{\theta}_g - \gamma \bigtriangledown_{\boldsymbol{\theta}_g} -[\lambda \mathcal{L}_S + \mathcal{L}_U] \\
\overline{\boldsymbol{\theta}}_d &= \overline{\boldsymbol{\theta}}_d - \gamma \bigtriangledown_{\overline{\boldsymbol{\theta}}_d} -\lambda \mathcal{L}_U
\end{aligned}
$$

Algorithm 5 Pseudo-code of cwt-TriGAN optimisation Process

```
1: Input: For a given dataset processed to 1-D Volatility , $\mathcal{V}$ with $\gamma$, batch size $b_s$, learning rate $\lambda$
2: Initialize Parameters: $\boldsymbol{\theta_e}, \boldsymbol{\theta_r}, \boldsymbol{\theta_g}, \boldsymbol{\theta_{d_1}}, \boldsymbol{\theta_{d_2}}, \boldsymbol{\theta_{d_3}}, w_i$
3: for $n = 1, ..., b_s, t = 1, ..., T_n$ do
4:     $\tilde{y}_t = \sum_{i=1}^{3} D_{\mathcal{C}_i} \left( \overleftarrow{\mathbf{q}}_{t_i}, \overrightarrow{\mathbf{q}}_{t_i} \right)$ for $i = 1, 2, 3$
5:     Calculate Reconstruction ($\mathcal{L}_R$), Discriminator Losses ($\mathcal{L}_U$), Supervised Loss ($\mathcal{L}_S$):
       $\mathcal{L}_R = \frac{1}{b_s} \sum_{n=1}^{b_s} \left[ \sum_t \parallel \mathbf{C_{n,t}} - \tilde{\mathbf{C}}_{n,t} \parallel_2 \right]$
       $\mathcal{L}_U = \frac{1}{b_s} \sum_{n=1}^{b_s} \sum_{i=1}^{3} w_i \cdot \mathcal{L}_{U_i}$ Where $\mathcal{L}_{U_i}$ represent losses for Discriminators 1, 2 and 3
       $\mathcal{L}_S = \frac{1}{b_s} \sum_{n=1}^{b_s} [\sum_t \parallel \mathbf{h}_t - G_{\mathcal{C}} (\mathbf{h_{t-1}}, \mathbf{z_t}) \parallel_2]$
6:     while Not converged do
7:         Update the parameters Auto-encoder Component, $\boldsymbol{\theta}_e, \boldsymbol{\theta}_r$:
           $\boldsymbol{\theta}_e = \boldsymbol{\theta}_e - \gamma \bigtriangledown_{\theta_e} -[\lambda \mathcal{L}_S + \mathcal{L}_R]$
           $\boldsymbol{\theta}_r = \boldsymbol{\theta}_r - \gamma \bigtriangledown_{\theta_r} -[\lambda \mathcal{L}_S + \mathcal{L}_R]$
8:         Update parameters for Adversarial component, $\boldsymbol{\theta}_g, \boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \boldsymbol{\theta}_{d_3}$:
           $\boldsymbol{\theta}_g = \boldsymbol{\theta}_g - \gamma \bigtriangledown_{\theta_g} -[\lambda \mathcal{L}_S + \mathcal{L}_U]$
           $\overline{\boldsymbol{\theta}}_d = \overline{\boldsymbol{\theta}}_d - \gamma \bigtriangledown_{\overline{\boldsymbol{\theta}}_d} -\lambda \mathcal{L}_U$ where: $\overline{\boldsymbol{\theta}}_d = (\frac{\boldsymbol{\theta}_{d_1} + \boldsymbol{\theta}_{d_2} + \boldsymbol{\theta}_{d_3}}{3})$
9:     end while
10: end for
11: Return: Average optimal parameter, $\overline{\boldsymbol{\theta}}_d$.
```

Overall, the cwt-TriGAN algorithm effectively combines the Auto-encoder and adversarial components to learn the temporal dynamics and generate synthetic volatility data that preserves important statistical properties and temporal dependencies. A high-level representation of the proposed cwt-TriGAN Algorithm is represented as a pseudo-code in Algorithm 6 with a focus on achieving optimal performance in generating realistic and accurate volatility time series data.

Algorithm 6 Pseudo-code of cwt-TriGAN

```
1: **Input**: For a given data set processed to 1D Volatility , $\mathcal{V}$ with $\Gamma$, $\eta = 10$, batch size $b_s$, learning rate $\lambda$, coefficients, $C$ of volatility.
2: **Initialize Parameters**: $\boldsymbol{\theta}_e, \boldsymbol{\theta}_r, \boldsymbol{\theta}_g, \boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \boldsymbol{\theta}_{d_3}$
3: Call algorithm 4 to apply the continuous wavelet transform on $\mathcal{V}$ to obtain the coefficients, $C$.
4: Call Algorithm 5 To:
       a. Map between Latent Space and the Feature Space and sample a batch of real coefficients samples from the training data set with parameters $\boldsymbol{\theta}_e, \boldsymbol{\theta}_r$
       b. Generate a batch of fake coefficients samples using the current Generator (G) with parameters $\boldsymbol{\theta}_g$
       c. Classify the coefficients samples as real or fake using the Triple discriminator with parameters $\boldsymbol{\theta}_{d_1}, \boldsymbol{\theta}_{d_2}, \boldsymbol{\theta}_{d_3}$.
5: Reconstruct coefficients of the volatility, $C$ to original volatility using using inverse CWT
       $$v(t) = \int_{-\infty}^{\infty} \int_{0}^{\infty} C(v)(\phi, x)\phi^{\frac{-5}{2}}\psi\left(\frac{t-x}{\phi}\right) dx d\phi$$
6: **Return:** $(\mathcal{V}(t))_{n=1}^{N}$
```

In summary, the cwt-TriGAN algorithm involves the following key steps:

1. **Auto-encoder component:** Utilise the Encoder and Decoder networks to map between the wavelet coefficients and a latent space, highlighting accurate reconstruction.
2. **Adversarial Components:** Employ the Generator network to produce synthetic latent codes and use the Triple discriminators to classify coefficient codes as real or fake, guiding the Generator to produce indistinguishable data.
3. **Optimisation:** Balance the reconstruction and discriminator losses to train the Auto-encoder and the Generator, respectively by updating the corresponding network parameters.

The next section introduces the experiments carried out in our work and includes the results obtained.

5.4 Experiments

The setup of our experiments is described in this section. It covers a description of the features of the data sets used, a description of the performance metrics used to evaluate the individual models, the results obtained, and the discussions of the results.

Data Description

To ensure that the volatility of a wide range of financial data sets was captured and tested with the proposed cwt-TriGAN model; Apple, Google, Unilever and Amazon stock prices were used. Other data sources used were from financial market indices like the S& P 500 and FTSE ALL index. Data from foreign exchange (FX) rates for British Pounds (GBP) / US dollar (USD) and Ghanaian Cedis / US dollar (USD) Foreign Exchange (FX) rates were also used for the FX Index, while the spot rate for crude oil constituted data from the commodity markets. Each of the datasets used covered a period of 15 years ranging from 1st June 2005 to 1st June 2020 and consisted of an average trading days of about 3750 depending on the time step and the dataset used.

Figure 5.6 represents the closing prices of all the datasets used and shows the variation in asset prices during this period. Figure 5.7 also shows the distribution of the closing prices for each data set in the form of a box plot and shows their median values, interquartile range values, minimum values, maximum values and outlier values. Each box represents the interquartile range (IQR) of closing prices for each asset used, from the first quartile to the third quartile. The horizontal line in each box represents the median closing prices of each of the assets used, while the whiskers that extend from the boxes show the range of the closing prices. The closing prices that appear as individual dots or beyond the whiskers are the outliers for each asset class used. The presence of outliers for some of the assets used indicates that there were some days with closing prices significantly different from the typical range.

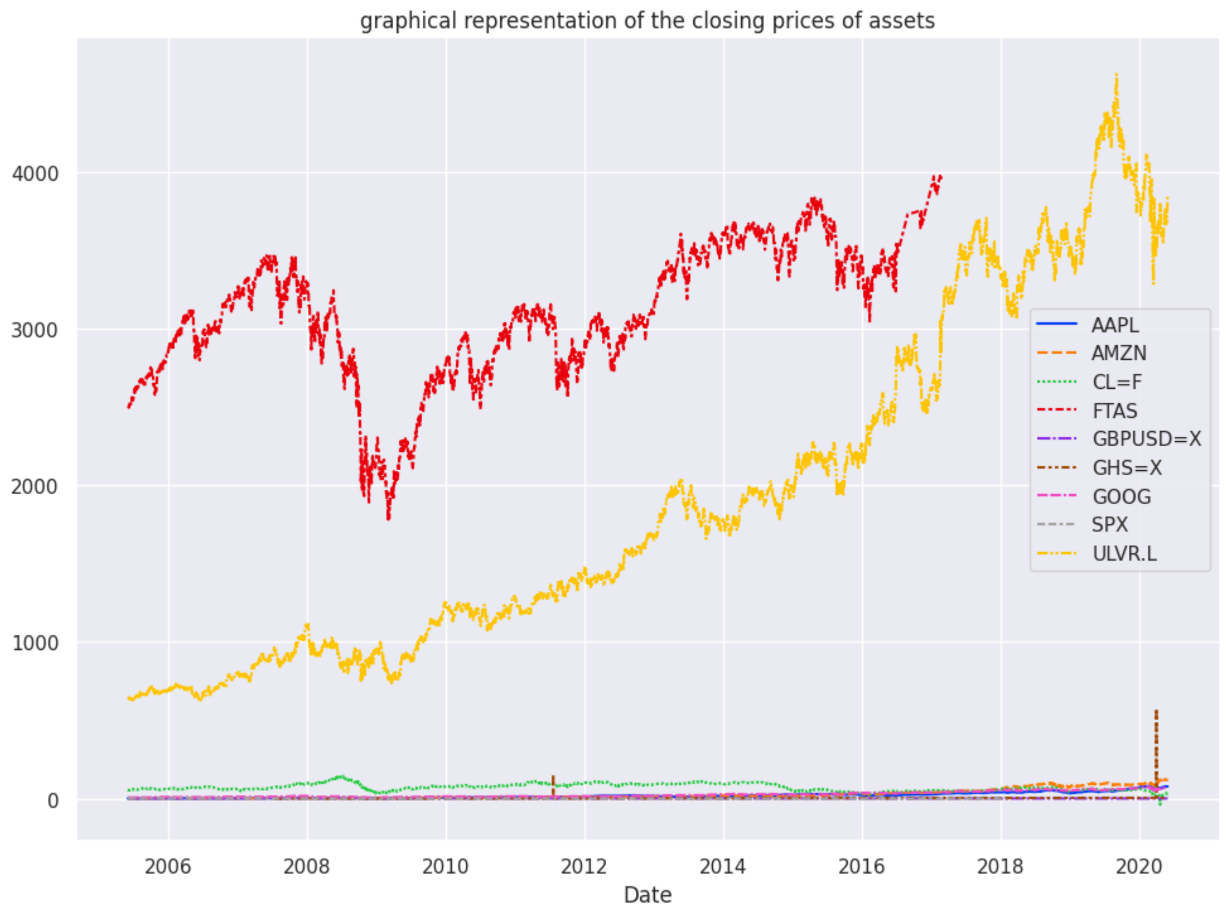


Figure 5.6: historical closing prices of all the 9 datasets, where AAPL, AMZN, CL=F, FTAS, GBPUSD=X, GHS=X, GOOG, SPX, ULVR.L represents Apple, Amazon, Crude oil, FTSE all, British Pounds (GBP) / US dollar (USD) FX, Ghanaian Cedis/US dollar (USD) FX, Google, S& P 500 index and Unilever datasets respectively.

The box plots in Figure 5.7 show that Amazon and Crude oil had the most stable closing prices during the observed period, while Unilever had the greatest variability. While British Pounds (GBP) / US dollar (USD) experienced several unusually low closing prices, as indicated by the outliers.

Figure 5.8 also captures the volatility distribution of the nine datasets in the form of a box plot, showing their median values, interquartile range values, minimum values, maximum values and outlier values. It can be seen that Unilever and FTSE all assets had the largest outliers and also had the greatest volatility variability, suggesting frequent and significant deviations from their median volatility levels. The box plot of Apple, crude oil, pound/dollar FX rate and Ghanaian Cedis/US dollar (USD) FX rate were the smallest, which shows that there was less variability in their volatility distribution with fewer and less extreme outliers, as shown in Figure 5.8.

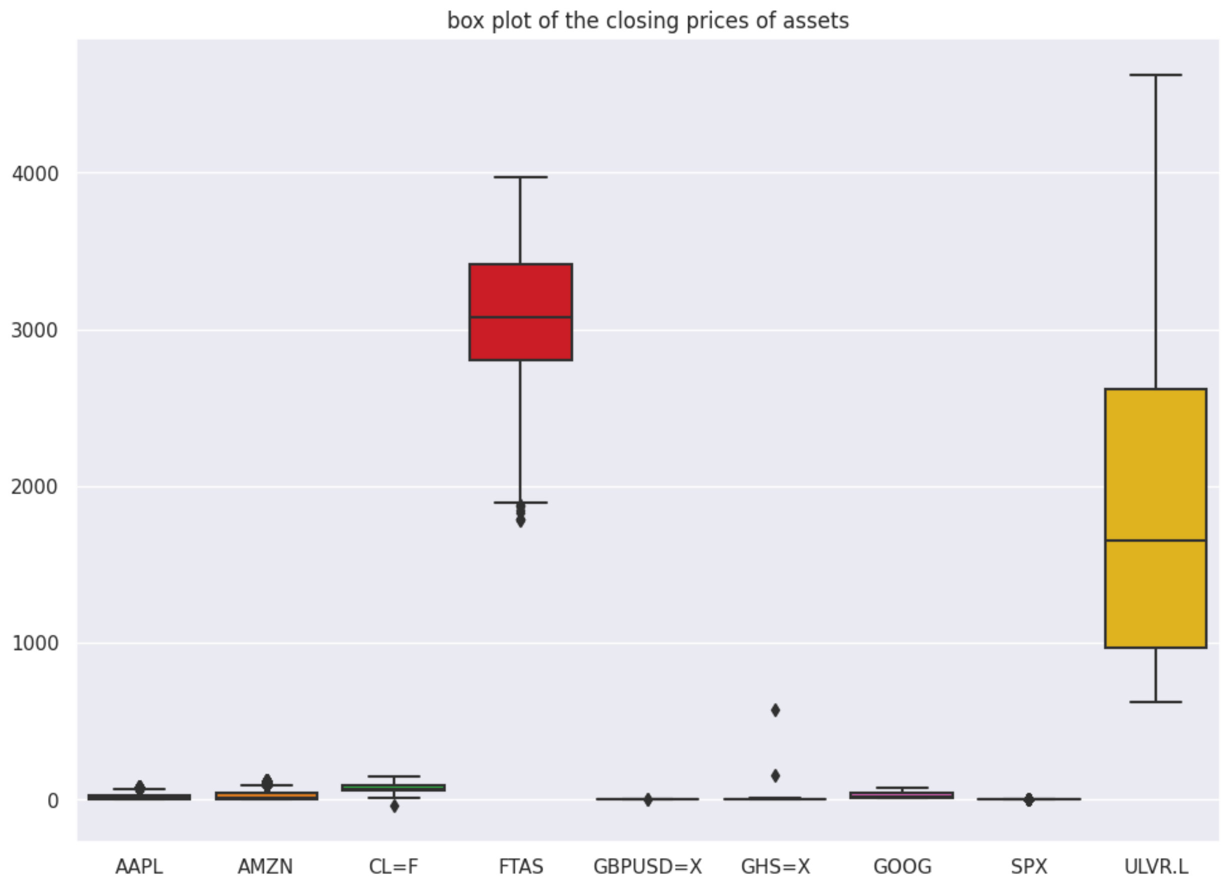


Figure 5.7: Boxplot of the closing prices of all the 9 datasets, where AAPL, AMZN, CL=F, FTAS, GBPUSD=X, GHS=X, GOOG, SPX, ULVR.L represents Apple, Amazon, Crude oil, FTSE all, British Pounds (GBP) / US dollar (USD) FX, Ghanaian Cedis/US dollar (USD) FX, Google, S& P 500 index and Unilever datasets respectively.

The scaleogram is the wavelet analogue of the spectrogram used to demonstrate the strength of the coefficient of the wavelet transform at a point in time (Wallisch et al., 2014) as shown in Figures 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15 and 5.16 for the Google, Apple, Amazon, Unilever, Ghana Cedis /US dollar FX rate, Pounds/US dollar FX rate, FTSE all index and crude oil datasets, respectively. The horizontal axis represents the trading period, and the vertical axis represents the scales that could be converted to their frequency equivalent in Hertz (Hz) using equation 5.13. The scales used in the experiments ranged from 1 to 64.

$$f = \frac{1}{S} \tag{5.13}$$

where f is the frequency and S is the scales or period. The intensity or colour of each point in the scaleogram represents the amplitude of the corresponding frequency component at a particular time. Darker colours indicate higher amplitudes and lighter colours indicate lower amplitudes.

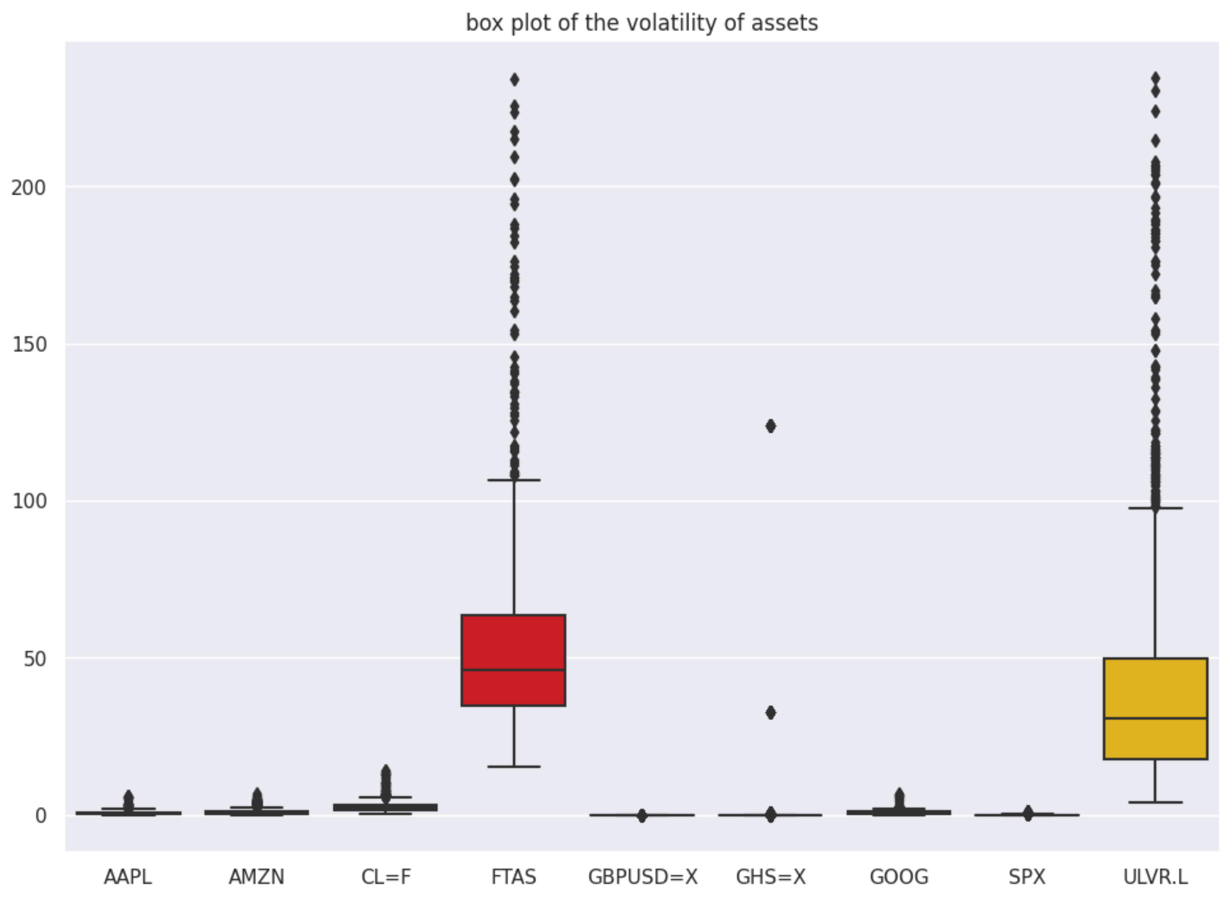
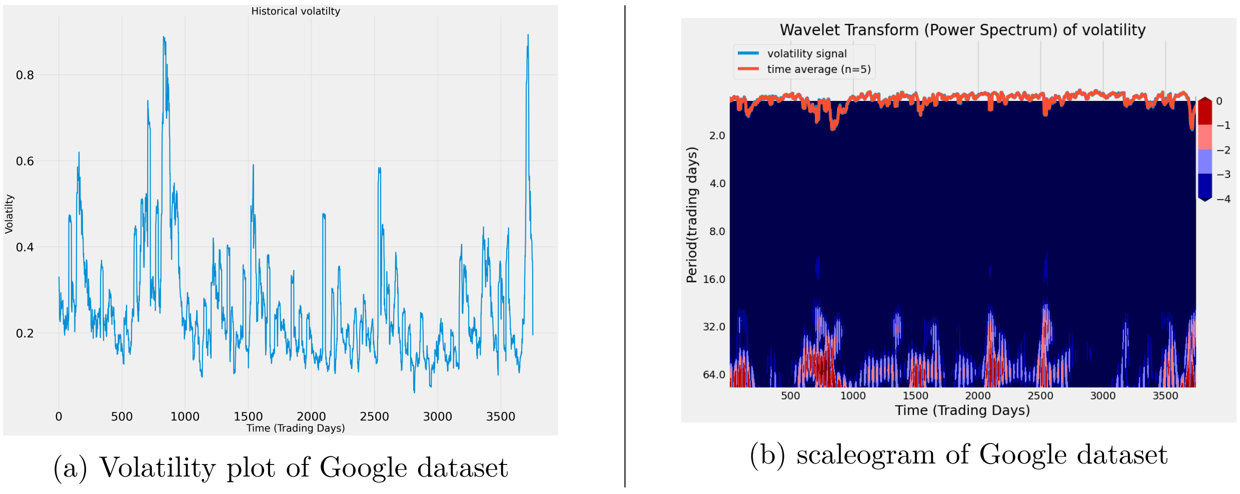


Figure 5.8: Boxplot of the Volatility of all the 9 datasets, where AAPL, AMZN, CL=F, FTAS, GBPUSD=X, GHS=X, GOOG, SPX, ULVR.L represents Apple, Amazon, Crude oil, FTSE all, British Pounds (GBP) / US dollar (USD) FX, Ghanaian Cedis/US dollar (USD) FX, Google, S& P 500 index and Unilever datasets respectively.

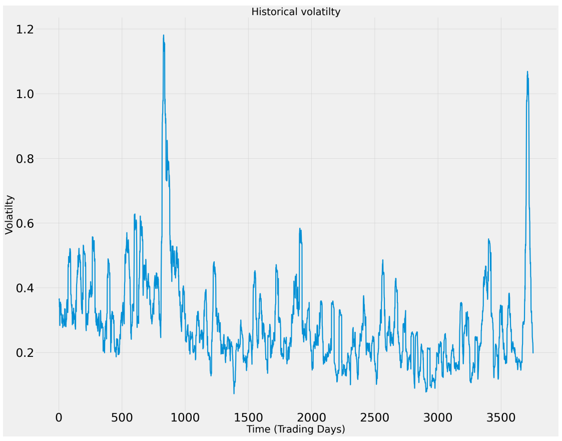


(a) Volatility plot of Google dataset

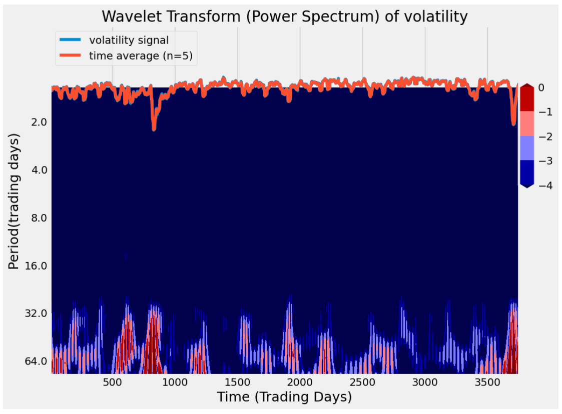
(b) scaleogram of Google dataset

Figure 5.9: Comparison of Volatility plots and scaleogram of Google dataset using the total trading days of 3754

As can be seen from Figure 5.9(a), the volatility of the Google data increased around the trading days of 750 to 1000 that corresponded to the 2008 to 2009 financial crisis. The higher volatility in 2020 corresponding to around 3,700 trading days caused by the Covid-19 pandemic can also be seen captured by the same figure. The scaleogram shown in Figure 5.10(b)) also captures the same trend, where it can be observed that the increase in the intensity of volatility for the Google data set is concentrated between the 48-64 trading period, which corresponds to a frequency equivalent of 0.016Hz - 0.02Hz. The trading periods shown in deep red represent the periods of high volatility. It can also be observed that there were many fluctuations, between the 500 and 1000 trading days (corresponding to 2007-2009), while there were not so many fluctuations in volatility between the 1600 - 2000 trading days (corresponding to the year 2011 to 2013). It can also be observed that there was a shift from shorter to longer periods as time progressed.



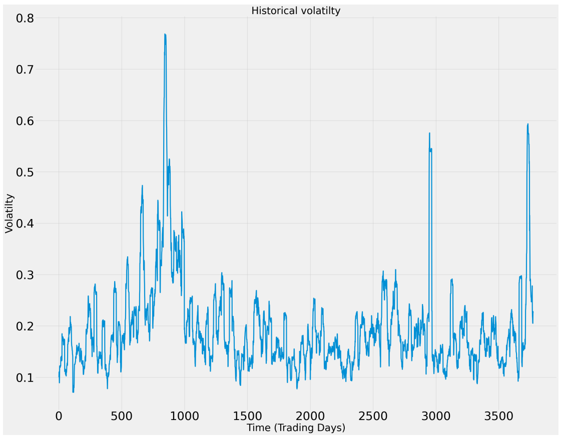
(a) Volatility of Apple data



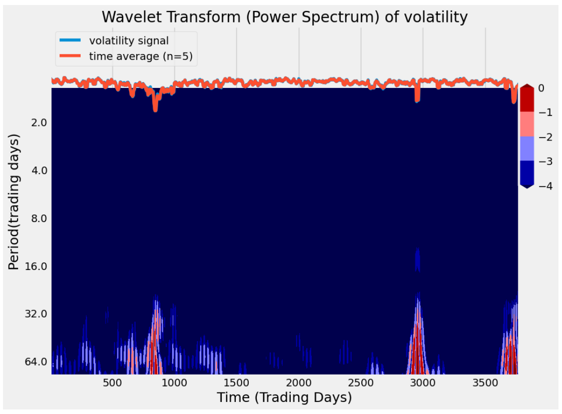
(b) scaleogram of Apple dataset

Figure 5.10: Comparison of Volatility plots and scaleogram of Apple dataset using the total trading days of 3754

As can be seen from Figure 5.10(a), the volatility of the Apple data also increased during the 2008 to 2009 financial crisis(750 to 1000 trading days). The higher volatility from 2019 to 2020 caused by the Covid-19 pandemic can also be seen captured. The scaleogram shown in Figure 5.10(b) captures the same trend. The trading periods shown in deep red colour represent the period of intense volatility, while the light red colour represents periods of low volatility, and the blue colour represents periods of absolute calmness with no volatility experienced at all. It can be observed that the increase in the intensity of volatility for the Apple data set was concentrated between the 32 to 64 trading period, with an equivalent frequency value of 0.016 - 0.03Hz. While there were high fluctuations in volatility between 500 and 1000 trading days (2007-2009), there were not so many fluctuations in volatility between 1300 and 3500 trading days (from 2010 to 2019).



(a) Volatility of Unilever data

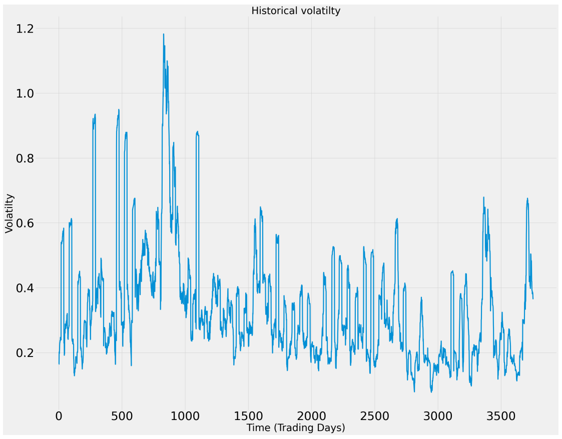


(b) scaleogram of Unilever dataset

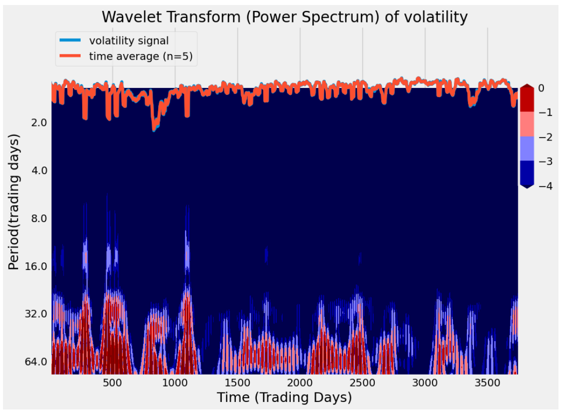
Figure 5.11: Comparison of Volatility plots and scaleogram of Unilever dataset using the total trading days of 3773

As can be seen from Figure 5.11(a) the volatility of the Unilever data also increased around the 2008 to 2009 financial crisis period (750 to 1000 trading days). The higher volatility from 2019 to 2020 caused by the Covid-19 pandemic can also be seen captured in the same figure. The scaleogram shown in Figure 5.11(b) captures the same trend, where it can be seen that the highest intensity of volatility for the Unilever dataset was concentrated between the 32-64 trading period, which corresponds to an amplitude of 0.016 - 0.03Hz when measured in frequency. From the same scaleogram, it can be observed that whereas there were not many fluctuations in the volatility prior to 2007, there were, however, significant increases in volatility, between 500 to 1000 trading days (2007-2008) and the 1000 - 3000 trading days (2009 to 2017). Also, volatility remained steady with relatively low figures between the 3000-3600 trading periods, while becoming intense from early 2020 (3700 trading days) again due to the covid-19 pandemic.

Like-wise, as can be observed from Figure 5.12(a), the volatility of the Amazon data increased around the year 2008 to 2009 financial crisis (750 to 1000 trading days). The higher volatility from 2019 to 2020 caused by the Covid-19 pandemic was also captured. The scaleogram shown in Figure 5.12(b) captures the same trend, where it can be observed that the increase in the highest intensity of volatility for the Amazon data set was concentrated between the 24-64 trading period, which corresponds to frequency values of 0.016 - 0.04Hz. Also, it can be seen that from 2005 to 2010 (0 to 1260 trading days) there was many high volatility, between 500 and 1000 trading days (2007-2008). The trading periods shown in deep red represent the period of high volatility.

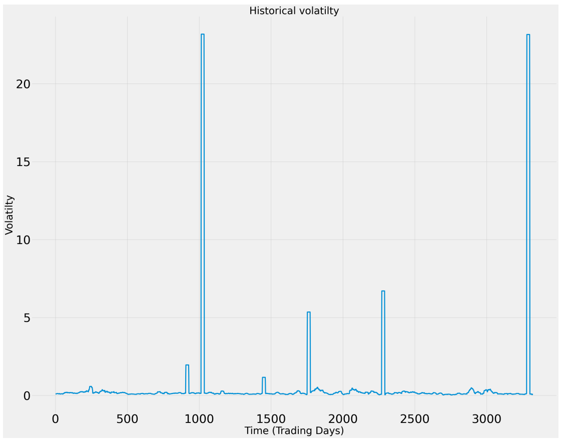


(a) Volatility of Amazon data

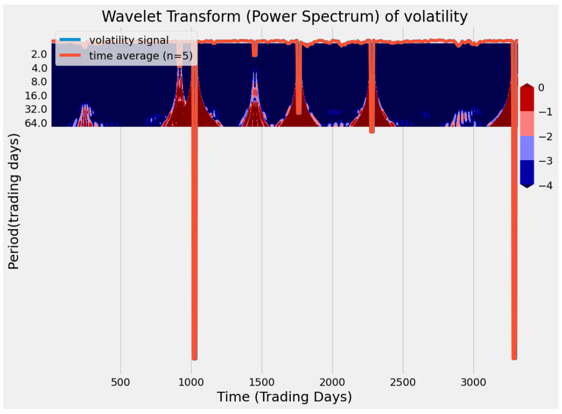


(b) scaleogram of Amazon dataset

Figure 5.12: Comparison of Volatility plots and scaleogram of Amazon dataset using the total trading days of 3754



(a) Volatility plots of US Dollar/Ghana cedis rate dataset

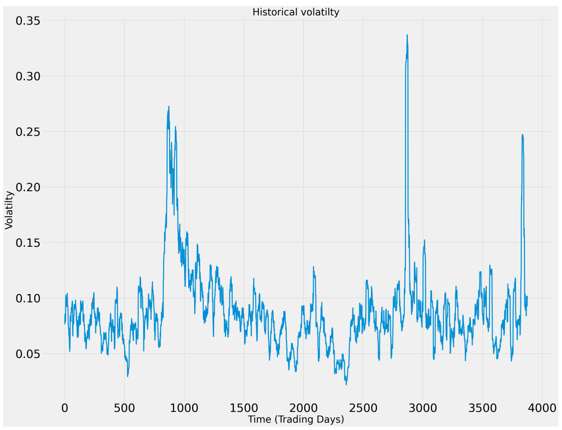


(b) scaleogram of US dollar/Ghana cedis rate dataset

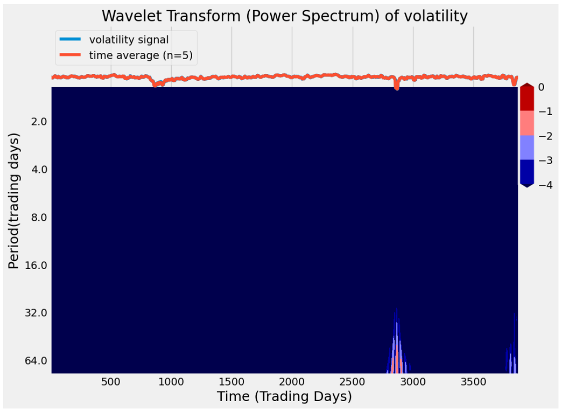
Figure 5.13: Comparison of Volatility plots and scaleogram of US Dollar/Ghana cedis rate dataset using the total trading days of 3322

As can be seen from Figure 5.13(a), the volatility of the US dollar/Ghana cedis FX rate data also increased during the 2008 to 2009 global financial crisis (750 to 1000 trading days). The higher volatility from 2019 to 2020 caused by the Covid-19 pandemic can also be seen captured. The scaleogram shown in Figure 5.13(b) captures the same trend, where it can be observed that the increase in the intensity of volatility for the US dollar/Ghana cedis FX rate dataset is concentrated between the 8-64 trading period, which corresponds to an amplitude of 0.016-0.125Hz when converted to frequency equivalence. It can also be observed that up to the year 2007, there was not much increase in volatility, as it was relatively calm for this period, but there were hikes in its value for 2011, 2014, 2016 and

2020 years, with the 2011 and 2020 extreme hikes occurring within scales of 8 days or equivalent amplitude value of 0.125hz during the 1000th trading days (2007-2008), while there were not so much between for the trading days in between these sudden hikes.



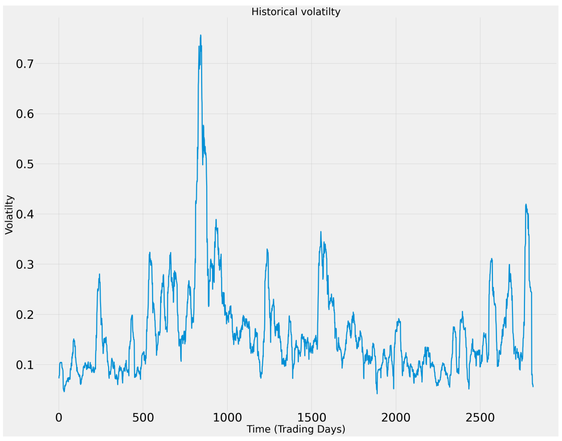
(a) Volatility of Pounds/US Dollar rate data



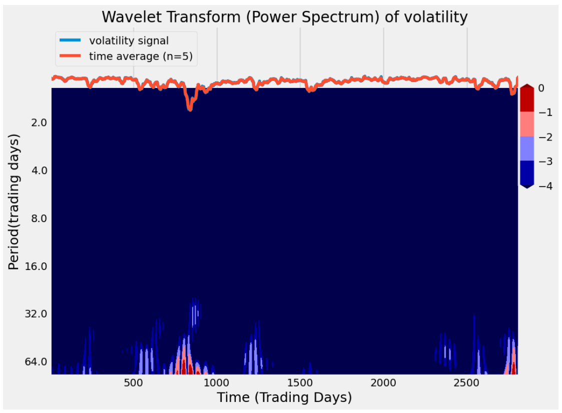
(b) scaleogram of Pounds/US Dollar rate dataset

Figure 5.14: Comparison of Volatility plots and scaleogram of Pounds/US Dollar rate dataset using the total trading days of 3876

As can be seen from Figure 5.14(a) the volatility of the Pounds/US Dollar rate data shot up around 2008 to 2009 during the financial crisis (750 to 1000 trading days). The higher volatility from 2019 to 2020 caused by the Covid-19 pandemic can also be seen captured. The scaleogram shown in Figure 5.14 (b) however did not capture any high volatility within the scales or frequency used in the experiments from 2005 to 2020, except around 2016 (2800 trading days) where there was some slight increase in volatility.



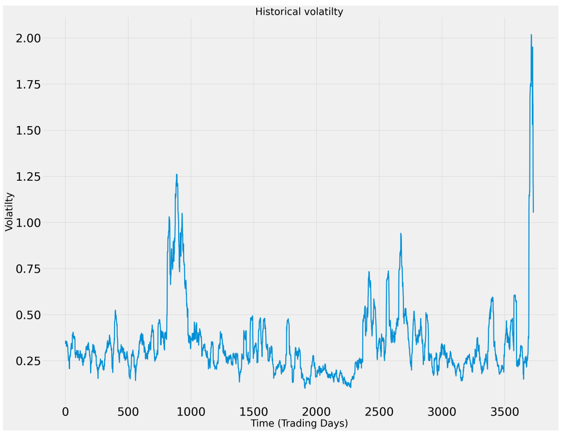
(a) Volatility of FTSE ALL index data



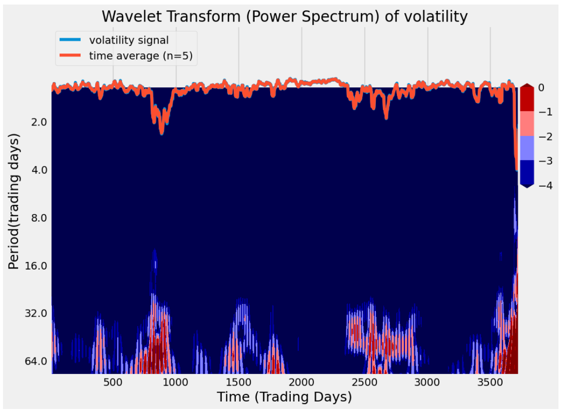
(b) scaleogram of FTSE ALL index dataset

Figure 5.15: Comparison of Volatility plots and scaleogram of FTSE All index dataset using the total trading days of 2814

As can be seen from Figure 5.15(a) the volatility of the FTSE ALL index data went up during the 2008 to 2009 financial crisis. The higher volatility from 2019 to 2020 caused by the Covid-19 pandemic can also be seen captured in the same figure. The scaleogram shown in Figure 5.15(b) however failed to capture any high volatility within the scales or frequency used in the experiments except around 2008-2009 and 2020.



(a) Volatility of Crude oil data



(b) scaleogram of Crude oil dataset

Figure 5.16: Comparison of Volatility plots and scaleogram of FTSE All index dataset using the total trading days of 3729

It can also be seen from Figure 5.16(a) that the volatility of Crude oil was high throughout the period of study from 2005 to 2020 except for some brief periods in between where it was relatively stable. The scaleogram shown in Figure 5.16(b) also shows high volatility within the scales or frequency used in the experiments shown in deep red colour between the 32-64 trading days with an equivalent amplitude of 0.016-0.03Hz.

Performance Evaluation Metrics

To evaluate the effectiveness of the proposed cwt-TriGAN model, the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) were used to compare the results of several state-of-the-art models such as timeGAN, RCGAN, ARIMA, Garch and LSTM. The adversarial component of our proposed cwt-TriGAN, which was composed of discriminators ensemble loss from three discriminators, was then used to replace the adversarial component of the timeGAN in order to assess the effectiveness of the ensemble loss proposed in the cwt-TriGAN. A cwt-GAN with only one Discriminator was also evaluated.

Root Mean Squared Error (RMSE)

It represents the square root of the mean square of all errors. It is considered an excellent metric for evaluating the performance of a prediction model. It shows how far predictions deviate from their true measured values using the Euclidean distance. It is computed by first calculating the difference between predicted values and true values for each data point and then normalising the residual for each data point. The final stage involves the calculation of the mean of the residuals and the square root of the mean. Mathematically, it is represented as equation (5.14)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \parallel y_{(i)} - \hat{y}_{(i)} \parallel^2}{N}} \tag{5.14}$$

Where N represents the number of data points, $y(i)$ is the true data point and $\hat{y}_{(i)}$ is the predicted data point.

Mean Absolute Error (MAE)

It refers to the magnitude of the difference between the predicted values and the true values. Each error is weighted equally, with the MAE values increasing linearly with increasing errors. The MAE value is measured as the mean of the absolute error values. Mathematically, it is represented as equation (5.15)

$$MAE = \frac{\sum_{i=1}^{N} \parallel y_{(i)} - \hat{y}_{(i)} \parallel^2}{N} \tag{5.15}$$

Where N represents the number of data points, $y_{(i)}$ is the true data point and $y_{(i)} - \hat{y}_{(i)}$ is the predicted data point.

Hyper-Parameters for cwt-TriGAN

The configuration of hyper-parameters for the cwt-TriGAN is documented in Table 5.1. These parameters govern critical aspects of the cwt-TriGAN network, such as batch size, learning rate, optimisation type and number of iterations (epochs).

Hyper-parameters	Description	size
batch size	size of mini-batch used	128
Learning Rate	Used for Adam Optimisation	$1 \times 10^{-}4$
Optimiser used	Type of optimiser used	Adam
Hidden dimension	Number of Hidden dimensions used	40
Number of layers	Number of layers used	3
latent dim	dimensionality of the latent space in GAN	10
Number of iterations	Number of epochs used	15,000

Table 5.1: Hyper-parameters used for the cwt-TriGAN Network

Results and Discussion

In this section, a comprehensive analysis of model performance is provided using the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics across nine data sets. Furthermore, the mean absolute error (MAE) values obtained for each model applied to the Google, Apple, Unilever, Amazon, S&P 500, FTSE All, Ghana / Dollar FX, Pounds / US Dollar FX and Crude oil datasets are investigated, showcasing the performance of these models across various forecast time steps.

RMSE Analysis

The results of the Root mean square error (RMSE) shown in the figure 5.17, highlight the different levels of performance of different models when applied to different data sets. It can be observed that the Garch model had the worst performance among all other models with an RMSE of 4.3715 when applied to the Ghana/US Dollar FX rate. The LSTM model followed closely with an RMSE of 3.4867 for the same dataset, while the ARIMA model exhibited a performance of 2.6724. These results underscore the challenges faced by traditional models in capturing the complex dynamics of financial markets. Our proposed cwt-TriGAN model stands out prominently in this analysis. Demonstrating the lowest RMSE values across all datasets, it underscores the effectiveness of the Continuous Wavelet Transform (CWT) and Triple Discriminator architecture in enhancing the predictive accuracy of models.

MAE Evaluation

Moving on to the Mean Absolute Error (MAE) evaluation, the outcomes of applying eight models to Google, Apple, Unilever, Amazon, S&P 500, FTSE All, Ghana/Dollar FX,

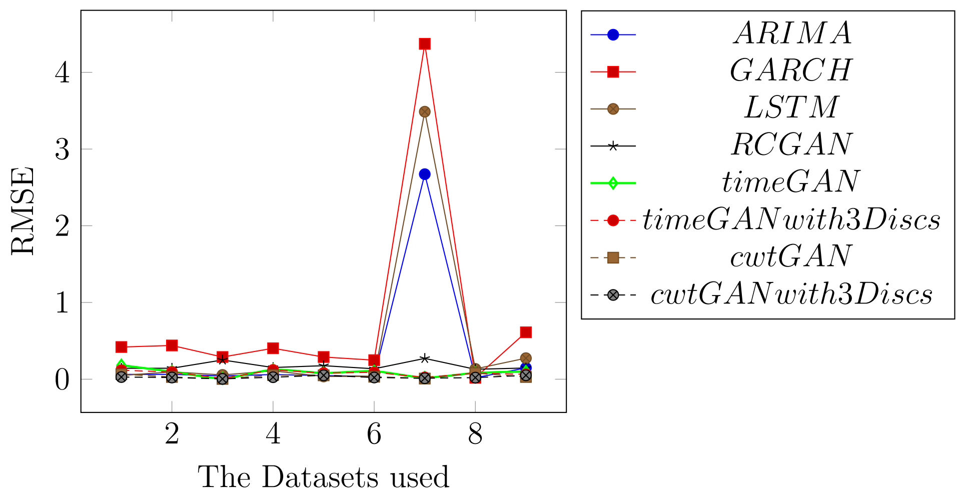


Figure 5.17: Results of RMSE of models for Google, Apple, Unilever, Amazon, S&P 500, FTSE All, Ghana/Dollar FX, pound/dollar FX and Crude oil

Pounds/US dollar FX and Crude oil datasets is presented as shown in Tables 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 and 5.10 respectively.

Model	**MAE OF GOOGLE**	
	T_s= 5	T_s=21
ARIMA	0.0386 ±0.0013	0.1575±0.0021
GARCH	0.3393±0.0002	0.3393 ±0.0011
LSTM	0.0391±0.0022	0.04583±0.0012
RCGAN	0.1315±0.0015	0.1272±0.0031
timeGAN	0.1807±0.0012	0.115±0.0003
timeGAN with 3 Discs	0.1151 ±0.0011	0.1185±0.0004
cwt-GAN	0.0579±0.0022	**0.0065**±0.0021
cwt-TriGAN	**0.0229**±0.0002	0.0345±0.0012

Table 5.2: model performance evaluation comparison for Google datasets. **Where T_s is the forecast time steps

Table 5.2 provides a detailed comparison of MAE values for the Google dataset across two forecasting time steps. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0386 and 0.1575 respectively. For the Garch model, there was no difference between the two forecasting time steps, where each of them achieved an MAE of 0.3393. Also, for the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.0391 compared to the 21-day forecasting steps, which obtained an MAE of 0.04583. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.1272 and 0.1315 respectively. The 21-day forecast time steps for the timeGAN model achieved an MAE of 0.115 which was superior to the 5-day forecast time steps of the same model with an MAE of 0.1807. Furthermore, the timeGAN with 3 Discriminator model with 5-day forecast time steps achieved an MAE of 0.1151 which was better than the MAE of the same model using a forecast time steps of 21 days, which had an MAE of 0.1185. For the cwt-GAN, the 5-day forecast time steps achieved an MAE of 0.0579 while that of the 21-day forecast time steps was 0.0065. Finally, the MAE of our proposed cwt-TriGAN was 0.0229 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0345.

Looking at the same table 5.2 and comparing the performance of each model with the other for each forecast time step, it can be observed that the MAE of 0.0229 obtained by our proposed cwt-TriGAN, which was 1381.7 % less than the MAE of 0.3393 obtained from the model with the worst performance known as the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.0579 obtained by the cwt-GAN model compared to the MAE of 0.0229 achieved by cwt-TriGAN indicates that the use of the Triple Discriminator was effective in improving the model's performance. Similarly, comparing the timeGAN and timeGAN with 3 discriminators models shows that the timeGAN with 3 Discriminators achievement of an MAE of 0.1151 as compared to the MAE of 0.1807 for the timeGAN shows the effectiveness of the triple discriminators. Therefore, this means that the use of the triple Discriminator was effective in improving the model's performance. Comparing the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN and that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. All the two models with CWT outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-GAN obtained the best performance among all the other models.

Table 5.3 presents a similar assessment for the Apple dataset. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0425 and 0.1846 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.3498. Also, for the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.0929 compared to the 21-day forecasting steps, which obtained an MAE of 0.0935. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.092 and 0.1401 respectively.

Model	**MAE OF APPLE**	
	T_s= 5	T_s=21
ARIMA	0.0425 ±0.012	0.1846 ±0.011
GARCH	0.3498±0.0150	0.3498 ±0.012
LSTM	0.0929±0.0014	0.0935±0.0019
RCGAN	0.1401±0.001	0.092±0.00
timeGAN	0.0963±0.0013	0.1399±0.002
timeGAN with 3 Discs	0.0907±0.002	0.0961±0.001
cwtGAN	**0.002**±0.01	0.047±0.005
cwtGAN with 3 Discs	0.0173±0.001	**0.0214**±0.010

Table 5.3: model performance evaluation comparison for APPLE datasets, **Where T_s represents the forecasting time steps**

The 21-day forecast time steps for the timeGAN model achieved an MAE of 0.1399, which was a lower performance compared to the 5-day forecast time steps of the same model with an MAE of 0.0963. Furthermore, the timeGAN with 3 Discriminator model with a 5-day forecast time steps achieved an MAE of 0.0907 which was better than the MAE of the same model using a forecast time steps of 21 days, which had an MAE of 0.1399. For the cwt-GAN, the 5-day forecast time steps achieved an MAE of 0.002 while that of the 21-day forecast time steps was 0.047. Finally, the MAE of our proposed cwt-TriGAN was 0.0173 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0214.

Looking at the same table 5.3 and comparing the performance of each model with the other for each forecast time step, it can be observed that the MAE of 0.0173 obtained by our proposed cwt-TriGAN, which was 1921.96 % less than the MAE of 0.3498 obtained from the model that performed the worst known as the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.000 obtained by the cwt-GAN model compared to the MAE of 0.0173 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was not effective in improving the model's performance. Similarly, comparing the timeGAN and timeGAN with 3 discriminators models shows that the timeGAN with 3 Discriminators achievement of an MAE of 0.0907 as compared to the MAE of 0.0963 for the timeGAN shows the effectiveness of the Triple Discriminator. Therefore, this means that the use of the Triple Discriminator was effective in improving the model performance. Comparing the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN and that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. The two models with CWT outperformed their corresponding models without CWT, respectively. The results obtained for the use of a 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-TriGAN obtained the best performance among all other models used with an MAE of 0.0214.

Model	**MAE of Unilever**	
	T_s= 5	T_s=21
ARIMA	0.0268 ±0.0025	0.0940 ±0.0013
GARCH	0.2352±0.0009	0.2352 ±0.0009
LSTM	0.0467±0.0021	0.0444±0.0023
RCGAN	0.2447±0.0014	0.1849±0.0025
timeGAN	0.0106±0.0013	0.0101±0.0023
timeGAN with 3 Discs	0.095 ±0.0017	0.089±0.0021
cwt-GAN	**0.0022**±0.0015	0.0197±0.0003
cwt-TriGAN	0.027 ±0.0019	**0.0022**±0.0023

Table 5.4: Model performance evaluation comparison for Unilever datasets, **Where T_s represents the forecasting time steps**

Table 5.4 compares the MAE of the Unilever dataset when applied with the 8 models using forecasting time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0268 and 0.0940 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.2352. Also, for the LSTM model, the 21-day forecast time steps model achieved superior results than the 5-day forecast time steps, with the 21-day forecast time steps achieving an MAE of 0.0444 compared to the 5-day forecasting steps, which obtained an MAE of 0.0467. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.1849 and 0.2447 respectively. The 21-day forecast time steps for the timeGAN model achieved an MAE of 0.0101, which was a higher performance compared to the 5-day forecast time steps of the same model with an MAE of 0.0106. Furthermore, the timeGAN with 3 Discriminator model with a 21-day forecast time steps achieved an MAE of 0.089 which was better than the MAE of the same model using a forecast time steps of 5 days, which had an MAE of 0.095. For the cwt-GAN, the 5-day forecast time steps achieved an MAE of 0.0022 while that of the 21-day forecast time steps was 0.0197. Finally, the MAE of our proposed cwt-TriGAN was 0.027 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0022.

Looking at the same table 5.4 and comparing the performance of each model with the other for each forecast time step, it can be observed that the MAE of 0.027 obtained by our proposed cwt-TriGAN, which was 771.1 % less than the MAE of 0.2352 obtained from the worst performing model known as the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.0022 obtained by the cwt-GAN model compared to the MAE of 0.027 achieved by the cwt-TriGAN shows that the use of the triple Discriminator was not effective in improving the model's performance. Similarly, comparing the timeGAN and timeGAN with 3 discriminators models shows that the timeGAN with 3 Discriminators achievement of an MAE of 0.095 as compared to the MAE of 0.0106 for the timeGAN shows the effectiveness of the triple discriminators. Therefore, this signifies that the use of the Triple Discriminator was effective in improving the model's performance. Comparing the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN and that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. The two models with CWT outperformed their corresponding models without CWT, respectively. The results obtained for the use of a 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-TriGAN obtained the best performance among all other models with an MAE of 0.0022.

Model	**MAE of Amazon**	
	T_s= 5	T_s=21
ARIMA	0.0376 ±0.0002	0.1262 ±0.0007
GARCH	0.3150±0.0015	0.3150 ±0.0021
LSTM	0.1057±0.0012	0.1107±0.0003
RCGAN	0.1491±0.0022	0.1312±0.0004
timeGAN	0.1184±0.00	0.1272±0.00
timeGAN with 3 Discs	0.1186 ±0.00	0.122±0.00
cwtGAN	0.0497±0.0013	0.0358±0.0021
cwt-TriGAN	**0.0372**±0.0011	**0.00574**±0.0020

Table 5.5: model performance evaluation comparison for Amazon datasets, **Where T_s represents the forecast time steps**

Table 5.5 compares the MAE of the Amazon dataset when applied with the 8 models using forecasting time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0376 and 0.1262 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.3150. For the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.1057 compared to the 21-day forecasting steps, which obtained an MAE of 0.1107. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.1312 and 0.1491 respectively. The 5-day forecast time steps for the timeGAN model achieved an MAE of 0.1184, which was a higher performance compared to the 21-day forecast time steps of the same model with an MAE of 0.1272. Furthermore, the timeGAN with 3 Discriminator model with a 5-day forecast time steps achieved an MAE of 0.1186 which was better than the MAE of the same model using a forecast time steps of 21 days, which achieved an MAE of 0.122. Similarly, for the cwt-GAN, the 5-day forecast time steps achieved an MAE of 0.0497 while that of the 21-day forecast time steps was 0.0358. Finally, the MAE of our proposed cwt-TriGAN was 0.0372 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.00574.

Looking at the same table 5.5 and comparing the performance of each model with the other, it can be observed that for each forecast time step, the MAE of 0.0372 obtained by our proposed cwt-TriGAN was 746.8% smaller than the MAE of 0.3150 obtained from the worst performing model, called the GARCH model, using the forecast time steps of 5 trading days. The MAE of 0.0497 obtained by the cwt-GAN model compared to the MAE of 0.0372 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was effective in improving the model performance. However, comparing the timeGAN and timeGAN with 3 Discriminator models shows that the timeGAN with 3 Discriminators attainment of an MAE of 0.1186 as compared to the MAE of 0.1184 for the timeGAN, which meant that the use of the Triple discriminators was not effective in improving the model's performance. Comparison of the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN with that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. The two models with the CWT component outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-TriGAN obtained the best performance among all other models with an MAE of 0.00574.

Table 5.6 compares the MAE of the S&P 500 datasets when applied to the 8 models using forecast time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0268 and 0.1391 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.2159. For the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.0331 compared to the 21-day forecasting steps, which obtained an MAE of 0.0343.

Model	**MAE of S & P 500**	
	T_s= 5	T_s=21
ARIMA	0.0268 ±0.0001	0.1391 ±0.0014
GARCH	0.2159±0.0011	0.2159 ±0.0021
LSTM	0.0331±0.0008	0.0343±0.0032
RCGAN	0.1741±0.0032	0.1475±0.0010
timeGAN	0.0792±0.0032	0.0768±0.0043
timeGAN with 3 Discs	0.0779 ±0.0056	0.111±0.0012
cwtGAN	0.0454±0.0014	**0.018**±0.0020
cwt-TriGAN	**0.0453**±0.0004	0.0272±0.0002

Table 5.6: model performance evaluation comparison for S&P 500 datasets, **Where T_s represents the forecast time steps**

However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.1741 and 0.1475 respectively. The 5-day forecast time steps for the timeGAN model achieved an MAE of 0.0792, which was a lower performance compared to the 21-day forecast time steps of the same model with an MAE of 0.0768. Furthermore, the timeGAN with 3 Discriminator model using the 5-day forecast time steps achieved an MAE of 0.0779 which was better than the MAE of the same model using a forecast time steps of 21 days, which achieved an MAE of 0.111. Similarly, for the cwt-GAN, the 21-day forecast time steps achieved an MAE of 0.018 while that of the 5-day forecast time steps was 0.0454. Finally, the MAE of our proposed cwt-TriGAN was 0.0453 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0272.

Looking at the same table 5.6 and comparing the performance of each model with the other, it can be observed that for each forecast time step, the MAE of 0.0453 obtained by our proposed cwt-TriGAN was 376.6% smaller than the MAE of 0.2159 obtained from the model that performed the worst, called the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.0454 obtained by the cwt-GAN model compared to the MAE of 0.0453 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was effective in improving the model performance. However, comparing the timeGAN and timeGAN with 3 Discriminator models shows that the timeGAN with 3 discriminators achieves an MAE of 0.0779 as compared to the MAE of 0.0792 for the timeGAN means that the use of the Triple discriminators was effective in improving the model's performance. Comparing the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN with that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. All two models with the CWT component outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-GAN obtained the best performance among all other models with an MAE of 0.018.

Model	**MAE of FTSE All**	
	T_s= 5	T_s=21
ARIMA	0.0239 ±0.0024	0.1167 ±0.0003
GARCH	0.2039±0.0023	0.2039 ±0.0014
LSTM	0.02571±0.0021	0.0270±0.0023
RCGAN	0.11861±0.0045	0.0913±0.0002
timeGAN	0.103±0.0023	0.1385±0.0012
timeGAN with 3 Discs	0.089 ±0.0001	0.0894±0.0003
cwtGAN	0.027±0.0003	0.0243±0.0025
cwt-TriGAN	**0.0176**±0.0014	**0.0164**±0.0021

Table 5.7: Model performance evaluation comparison for FTSE All datasets, **Where T_s represents the forecast time steps**

Table 5.7 compares the MAE of the FTSE All datasets when applied with the 8 models using forecast time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0239 and 0.1167 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.2039. For the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.02571 compared to the 21-day forecast time steps, which obtained an MAE of 0.0270. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.0913 and 0.11861 respectively. The 5-day forecast time steps for the timeGAN model achieved an MAE of 0.103, which was a higher performance compared to the 21-day forecast time steps of the same model with an MAE of 0.1385. Furthermore, the timeGAN with 3 Discriminator model using the 5-day forecast time steps achieved an MAE of 0.0890 which was better than the MAE of the same model using a forecast time steps of 21 days, which achieved an MAE of 0.0894. Similarly, for the cwt-GAN, the 21-day forecast time steps achieved an MAE of 0.0243 while that of the 5-day forecast time steps was 0.027. Finally, the MAE of our proposed cwt-TriGAN was 0.0176 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0164.

Looking at the same table 5.7 and comparing the performance of each model with the other, it can be observed that for each forecast time step, the MAE of 0.0176 obtained by our proposed cwt-TriGAN was 1058.5% smaller in value than the MAE of 0.2039 obtained from the model with the worst performance, known as the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.027 obtained by the cwt-GAN model compared to the MAE of 0.0176 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was effective in improving the model performance. However, comparing the timeGAN and timeGAN with Triple Discriminator models shows that the timeGAN with Triple discriminators achieves an MAE of 0.089 as compared to the MAE of 0.103 for the timeGAN means that the use of the Triple discriminators was effective in improving the model's performance. Comparison of the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN with that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. All the two models with the CWT component outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-TriGAN obtained the best performance among all other models with an MAE of 0.0164.

Model	**MAE of GHS/USD FX**	
	T_s= 5	T_s=21
ARIMA	0.3680 ±0.0012	1.3645 ±0.0023
GARCH	1.968±0.0043	1.968 ±0.0017
LSTM	0.7946±0.0028	0.8601±0.0102
RCGAN	0.2699±0.0004	0.0172±0.0029
timeGAN	0.0206±0.0072	0.0216±0.0061
timeGAN with 3 Discs	0.0224 ±0.0017	0.0203±0.0024
cwtGAN	0.0047±0.0062	0.0046±0.0009
cwt-TriGAN	**0.0046**±0.0045	**0.0039**±0.0021

Table 5.8: model performance evaluation comparison for Ghana Cedis/US Dollar FX rate datasets, **Where T_s represents the forecasting time steps**

Table 5.8 compares the MAE of the Ghana Cedis/US Dollar FX datasets when applied with the 8 models using forecast time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.3680 and 1.3645 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 1.968. For the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.7946 compared to the 21-day forecasting steps, which obtained an MAE of 0.8601. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.0172 and 0.2699 respectively. The 5-day forecast time steps for the timeGAN model achieved an MAE of 0.0206, which was a higher performance compared to the 21-day forecast time steps of the same model with an MAE of 0.0216. Furthermore, the timeGAN with 3 Discriminator model using the 21-day forecast time steps achieved an MAE of 0.0203 which was better than the MAE of the same model using a forecast time steps of 5 days, which achieved an MAE of 0.0224. Likewise, for the cwt-GAN, the 21-day forecast time steps achieved an MAE of 0.0046 while that of the 5-day forecast time steps was 0.0047. Finally, the MAE of our proposed cwt-TriGAN was 0.0046 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0039.

Looking at the same table 5.8 and comparing the performance of each model with the other, it can be observed that for each forecast time step, the MAE of 0.0046 obtained by our proposed cwt-TriGAN was 42682.6% smaller in value than the MAE of 1.968 obtained from the model with the worst performance, known as the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.0047 obtained by the cwt-GAN model compared to the MAE of 0.0046 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was effective in improving the model performance. However, comparing the timeGAN and timeGAN with three Discriminator models shows that the timeGAN with three discriminators achieved an MAE of 0.0224 as compared to the MAE of 0.0206 for the timeGAN means that the use of the Triple discriminators failed to show its effectiveness in improving the model's performance. Comparison of the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN with that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. All the two models with the CWT component outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-TriGAN obtained the best performance among all other models with an MAE of 0.0039.

Table 5.9 compares the MAE of the Pounds/ US dollar FX rate datasets when applied with the 8 models using forecast time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0131 and 0.0432 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.1147.

Model	**MAE of GBP/USD**	
	T_s= 5	T_s=21
ARIMA	0.0131 ±0.0027	0.0432 ±0.0035
GARCH	0.1147±0.0029	0.1147 ±0.0062
LSTM	0.114±0.0011	0.1141±0.0024
RCGAN	0.1261±0.0003	0.0748±0.0021
timeGAN	0.0784±0.0040	0.0972±0.0037
timeGAN with 3 Discs	0.0771 ±0.0016	0.0757±0.0009
cwtGAN	0.0704±0.0014	**0.0253**±0.0032
cwt-TriGAN	**0.0703**±0.0032	0.0612±0.0045

Table 5.9: Model performance evaluation comparison for Pounds/ US dollar FX rate datasets, **Where T_s represents the forecasting time steps**

For the LSTM model, the 5-day forecast time steps model and the 21-day forecast time steps both achieved an MAE of 0.114. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.0748 and 0.1261 respectively. The 5-day forecast time steps for the timeGAN model achieved an MAE of 0.0784, which was a higher performance compared to the 21-day forecast time steps of the same model with an MAE of 0.0972. Furthermore, the timeGAN with 3 Discriminator model using the 21-day forecast time steps achieved an MAE of 0.0757 which was better than the MAE of the same model using a forecast time steps of 5 days, which achieved an MAE of 0.0771. Similarly, for the cwt-GAN, the 5-day forecast time steps achieved an MAE of 0.00704 while that of the 21-day forecast time steps was 0.0253. Finally, the MAE of our proposed cwt-TriGAN was 0.0703 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an MAE of 0.0612.

Looking at the same table 5.9 and comparing the performance of each model with the other, it can be observed that for each forecast time step, the MAE of 0.0703 obtained by our proposed cwt-TriGAN was 63.16% smaller in value than the MAE of 0.1147 obtained from the model with the worst performance, known as the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.00704 obtained by the cwt-GAN model compared to the MAE of 0.0703 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was slightly effective in improving the model performance. Likewise, comparing the timeGAN and timeGAN with 3 Discriminator models shows that the timeGAN with 3 discriminators achieved an MAE of 0.0771 as compared to the MAE of 0.0784 for the timeGAN means that the use of the Triple discriminators was effective in improving the model's performance. Comparison of the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN with that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. All the two models with the CWT component outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-GAN obtained the best performance among all other models with an MAE of 0.0253.

Model	**MAE of Crude Oil**	
	T_s= 5	T_s=21
ARIMA	0.0626 ±0.0017	0.1843 ±0.0034
GARCH	0.459±0.0008	0.459 ±0.0042
LSTM	0.2301±0.0075	0.2345±0.0061
RCGAN	0.1414±0.0015	0.0562±0.0022
timeGAN	0.0991±0.0017	0.0762±0.0038
timeGAN with 3 Discs	0.0628 ±0.0014	0.0666±0.0053
cwtGAN	**0.0252**±0.0021	**0.0238**±0.0017
cwt-TriGAN	0.0467±0.0010	0.0402±0.0021

Table 5.10: Model performance evaluation comparison for Crude Oil datasets, **Where T_s represents the forecasting time steps**

Table 5.10 compares the MAE of the crude oil data sets when applied to the 8 models using forecast time steps of 5 and 21 trading days. It can be seen that for the ARIMA model, the 5-day forecasting time steps outperformed the 21-day forecast time steps with an MAE of 0.0626 and 0.1843 respectively. For the Garch model, there were no differences between the two forecasting time steps, where each of them achieved an MAE of 0.459. For the LSTM model, the 5-day forecast time steps model achieved superior results than the 21-day forecast time steps, with the 5-day forecast time steps achieving an MAE of 0.2301 compared to the 21-day forecasting steps, which obtained an MAE of 0.2345. However, for the RCGAN model, the 21-day forecast time steps outperformed the 5-day forecast time steps, which had MAE of 0.0562 and 0.1414 respectively. The 21-day forecast time steps for the timeGAN model achieved an MAE of 0.0762, which was a higher performance compared to the 5-day forecast time steps of the same model with an MAE of 0.0991. Furthermore, the timeGAN with 3 Discriminator model using the 5-day forecast time steps achieved an MAE of 0.0628 which was better than the MAE of the same model using a forecast time steps of 21 days, which achieved an MAE of 0.0666. Similarly, for the cwt-GAN, the 21-day forecast time steps achieved an MAE of 0.0238 while that of the 5-day forecast time steps was 0.0252. Finally, the MAE of our proposed cwt-TriGAN was 0.0467 for the forecast time steps of 5 days compared to that of the forecast time steps of 21 days, which had an

MAE of 0.0402.

Looking at the same table 5.10 and comparing the performance of each model with the other, it can be observed that for each forecast time step, the MAE of 0.0467 obtained by our proposed cwt-TriGAN was 882.87% smaller in value than the MAE of 0.459 obtained from the model with the worst performance, called the GARCH model using the forecast time steps of 5 trading days. The MAE of 0.0252 obtained by the cwt-GAN model compared to the MAE of 0.0467 achieved by the cwt-TriGAN shows that the use of the Triple Discriminator was not effective in improving the model performance. However, comparing the timeGAN and timeGAN with three Discriminator models shows that the timeGAN with three discriminators achieves an MAE of 0.0628 as compared to the MAE of 0.0991 for the timeGAN demonstrates that the use of the Triple discriminators was effective in improving the model's performance. Comparison of the MAE of the timeGAN model with that of cwt-GAN and the MAE of timeGAN with that of the cwt-TriGAN model indicates how effective the application of the continuous wavelet transform was. All the two models with the CWT component outperformed their corresponding models without CWT, respectively. The results obtained for the use of the 21-day forecast time steps can also be seen in the same table, where it can be observed that cwt-GAN obtained the best performance among all other models with an MAE of 0.0238.

Significance Test of models' MAE

The tables 5.11 and 5.12 contain the output of the pairwise analysis of all the forecasting models using the t-test results for the data with time-steps of 5 days and 21 days respectively. It compares the mean absolute error (MAE) for different time-series forecast models using a 5-day and 21-days time steps respectfully. The two tables include the comparison of one model with the other, t-statistics, p-values, and a column to indicate whether the null hypothesis of equal means is rejected or not using the significance level of 0.05.

5.11, it can be observed that there was no significant difference in Mean Absolute Error (MAE) between cwt-TriGAN and cwtGAN models where the t-statistics was 0.0452 and p-value was 0.9645. This suggests that the complexity of cwt-TriGAN, where three discriminators were used did not result in a statistically significant improvement in the forecast accuracy of volatility compared to that of the cwtGAN. On the other hand, timeGAN with 3 discriminators was found to be significantly different from the cwt-TriGAN, where it had a t-statistic of -4.5323 and p-value of 0.0005. This suggests that the addition of cwt component in the cwt-TriGAN enabled it to statistically perform better.

Comparison	t-Statistic	p-Value	Reject Null
cwtTriGAN vs cwtGAN	0.0452	0.9645	False
cwtTriGAN vs timeGAN with 3 Discs	-4.5323	0.0005	True
cwtTriGAN vs timeGAN	-3.0396	0.0119	True
cwtTriGAN vs RCGAN	-6.9474	0.0000	True
cwtTriGAN vs LSTM	-1.6205	0.1433	False
cwtTriGAN vs GARCH	-2.2788	0.0521	False
cwtTriGAN vs ARIMA	-1.0266	0.3329	False
cwtGAN vs timeGAN with 3 Discs	-4.1614	0.0008	True
cwtGAN vs timeGAN	-2.9411	0.0125	True
cwtGAN vs RCGAN	-6.7179	0.0000	True
cwtGAN vs LSTM	-1.6230	0.1424	False
cwtGAN vs GARCH	-2.2804	0.0519	False
cwtGAN vs ARIMA	-1.0293	0.3307	False
timeGAN with 3 Discs vs timeGAN	-0.1002	0.9217	False
timeGAN with 3 Discs vs RCGAN	-3.9278	0.0019	True
timeGAN with 3 Discs vs LSTM	-0.9652	0.3620	False
timeGAN with 3 Discs vs GARCH	-1.9979	0.0806	False
timeGAN with 3 Discs vs ARIMA	0.3703	0.7196	False
timeGAN vs RCGAN	-3.1776	0.0059	True
timeGAN vs LSTM	-0.9282	0.3784	False
timeGAN vs GARCH	-1.9824	0.0822	False
timeGAN vs ARIMA	0.3962	0.6994	False
RCGAN vs LSTM	0.0181	0.9860	False
RCGAN vs GARCH	-1.5698	0.1544	False
RCGAN vs ARIMA	2.2873	0.0419	True
LSTM vs GARCH	-1.4571	0.1735	False
LSTM vs ARIMA	1.0430	0.3189	False
GARCH vs ARIMA	2.0367	0.0735	False

Table 5.11: Comparison of the MAE Pairwise t-test results for the data rising a time-step of 5 days

The cwt-TriGAN architecture was able to provide a considerable advantage over that of RCGAN in the forcasting of the volatility time series with t-statistics of -6.9474 and p-value of 0.0000. Likewise, The cwtGAN model was found to be statistically more accurate in its prediction When compared to models such as timeGAN with 3 discs that produced a t-statistics of -4.1614 and p-value of 0.0008 and that of the RCGAN with t-statistics of -6.7179 and p-value of 0.0000. Also the addition of triple discriminators to the timeGAN model did not necessarily improve forecasting accuracy, as there was no significant difference in the MAE when the comparison was made between timeGAN with 3 discs and timeGAN where the t-statistics obtained was -0.1002 with p-value of 0.9217. Finally, the performance of the RCGAN model was found to have significantly outperform the ARIMA model when the forecasting horizon of 5 -days was used since a p-value of 0.0419 was obtained for this pairwise comparison.

As can be observed in table 5.12, a pairwise comparison of different forecasting models to determine their accuracy in predicting volatility has been carried out using the t-test. Given the p-value of 0.307, the results indicate that there is no statistically significance difference in the forecasting accuracy of cwtTriGAN and cwtGAN models for a data with 21-day forecasting time horizon. However, comparison of the cwt-TriGAN model, timeGAN with 3 Discriminators, timeGAN and RCGAN models, showed that the cwt-TriGAN model had a statistically significant higher Mean Absolute Error (MAE) than all the others. This suggests that the cwtTriGAN's performance exceeds that of the timeGAN with 3 discs, timeGAN and the RCGAN models.

Similarly, the cwtGAN model also showed statistically significant MAE performance over models such as timeGAN with 3 Discs, timeGAN, and RCGAN. This reinforces the conclusion that our proposed cwt-based models are more capable of capturing the volatility's complex distribution over a 21-day time-step when used for forcasting. However, within the same model family, the timeGAN with 3 Discs did not show a statistically significant improvement over the timeGAN model, meaning the inclusion of the triple discriminators with the TimeGAN did not translate into statistically significant improvement in the performance when the data with forecasting time horizon of 21 days was used.

Finally, the traditional statistical models such GARCH and ARIMA did not show significant differences in MAE when compared with LSTM and RCGAN models. However, the t-test between GARCH and ARIMA indicates no statistically significant difference between the two models. This suggests similar performance levels for these traditional models.

Comparison	t-Statistic	p-Value	Reject Null
cwtTriGAN vs cwtGAN	-0.6410	0.5307	False
cwtTriGAN vs timeGAN with 3 Discs	-5.1960	0.0002	True
cwtTriGAN vs timeGAN	-3.8039	0.0031	True
cwtTriGAN vs RCGAN	-4.3442	0.0014	True
cwtTriGAN vs LSTM	-1.6941	0.1283	False
cwtTriGAN vs GARCH	-2.3232	0.0486	True
cwtTriGAN vs ARIMA	-1.7687	0.1147	False
cwtGAN vs timeGAN with 3 Discs	-4.8531	0.0003	True
cwtGAN vs timeGAN	-3.5189	0.0054	True
cwtGAN vs RCGAN	-4.0776	0.0023	True
cwtGAN vs LSTM	-1.6312	0.1412	False
cwtGAN vs GARCH	-2.2939	0.0509	False
cwtGAN vs ARIMA	-1.7282	0.1221	False
timeGAN with 3 Discs vs timeGAN	-0.0810	0.9366	False
timeGAN with 3 Discs vs RCGAN	-0.7461	0.4686	False
timeGAN with 3 Discs vs LSTM	-0.9682	0.3605	False
timeGAN with 3 Discs vs GARCH	-1.9859	0.0821	False
timeGAN with 3 Discs vs ARIMA	-1.3029	0.2285	False
timeGAN vs RCGAN	-0.5720	0.5753	False
timeGAN vs LSTM	-0.9423	0.3720	False
timeGAN vs GARCH	-1.9739	0.0833	False
timeGAN vs ARIMA	-1.2868	0.2332	False
RCGAN vs LSTM	-0.7925	0.4494	False
RCGAN vs GARCH	-1.9035	0.0929	False
RCGAN vs ARIMA	-1.1899	0.2672	False
LSTM vs GARCH	-1.3941	0.1901	False
LSTM vs ARIMA	-0.5737	0.5755	False
GARCH vs ARIMA	0.8457	0.4114	False

Table 5.12: comparison of the MAE Pairwise t-test results for the data with time-step of 21 days

5.5 Summary

This chapter has explored the use of the continuous wavelet transform to improve the forecasting and generation ability of a volatility time series, which is noisy and non-stationary in nature. A comprehensive set of experiments has been conducted to evaluate the performance of our proposed cwt-TriGAN model for financial time series forecasting.

It has been established that the continuous wavelet transform can be beneficial when used as a data preprocessing technique when applied on a cwt-TriGAN. The proposed technique compared to other techniques such as ARIMA, GARCH, LSTM, RCGAN and timeGAN had superior performances in the nine different financial data sets used in the experiment when evaluated with the root mean square error (RMSE) and mean absolute error (MAE) metrics. The ability of our proposed cwt-TriGAN model to effectively capture volatility patterns and generate accurate predictions positions it as a valuable tool for researchers, financial analysts, traders, and decision-makers looking for reliable information on market trends. While the cwt-TriGAN is innovative with the ability to achieve superior results for volatility forecasting, it has some limitations just like any complex machine learning model. Here are some of the main limitations of our proposed model:

- The architecture of cwt-TriGAN is quite complex, which can result in difficulty in implementation with the requirement of significant computational resources which can limit its usage.
- Risk of overfitting: Because of the complex nature of the model, it might be prone to overfitting, especially when dealing with limited or noisy financial datasets.
- Since the proposed model contains multiple components with their own individual parameters, the fine-tuning of these parameters for optimal performance can be challenging and time-consuming.
- Lack of Interpretability: Like many deep learning models, our proposed cwt-TriGAN may suffer from a lack of interpretability or transparency in its decision-making process. This ”black box” nature of it can be a significant drawback, especially in the area of finance where understanding the reasoning behind predictions is crucial for trust and regulatory compliance.

Having established techniques for accurate volatility forecasting, After successfully developing methods for forecasting volatility with accuracy, we will now explore how these insights can be leveraged for optimising investment portfolios in the next chapter using CapsNet-based reinforcement learning.

Chapter 6

CapsNet-based Reinforcement Learning for Portfolio Optimisation

6.1 Introduction

The previous chapters provided a foundation in predicting credit risks and market volatility. This chapter integrates these elements into a comprehensive framework for portfolio optimisation using reinforcement learning. Portfolio optimisation is generally the process of allocating assets in a portfolio to maximise the expected return on investment while at the same time minimising financial risk (Soleymani and Paquet, 2020). It involves the continuous reallocation of assets in a portfolio through the attribution of weights to each asset class, such as bonds, stocks, or derivatives, to optimise a preferred performance metric. The Sharpe ratio is an indicator used to measure the risk-adjusted performance of an investment over time and helps investors measure the returns on their investments relative to the risks (Sharpe, 1964, 1994). It is the average return earned by a portfolio in excess of the risk-free rate for every unit of volatility or total risk taken. Volatility and correlation associated with the various assets must be considered when managing a portfolio (Soleymani and Paquet, 2020). This work involves the application of a simple deterministic policy gradient as a reinforcement learning algorithm to asset allocation problems to observe and independently learn from the market history without any prior knowledge of the financial markets. It does not directly predict the price of any single asset but rather predicts the optimal combination of weights of assets that yield the highest profit.

Researchers have investigated deep learning techniques that have proven to be successful in computer vision, Natural Language Processing (NLP), and board games in the dynamic financial markets domain (Yashaswi, 2021). Reinforcement learning (RL) is a branch of artificial intelligence that focuses on how agents take action in a dynamic environment to maximise a cumulative reward (Silver et al., 2014). The modelling capabilities of reinforcement learning (RL) can be utilised in domains such as finance (Dixon et al., 2020), where it has been used for portfolio optimisation but not as extensively as algorithmic trading; which refers to the use of a computerised system to automate one or more stages of the trading process, such as data analysis, the generation of trading signals (buy or sell recommendations) and the execution of trades (Nuti et al., 2011).

In recent years, advances have been made in the use of deep reinforcement learning to optimise portfolios with excellent results (Almahdi and Yang, 2017; Jiang et al., 2017; Liang et al., 2018; Yashaswi, 2021). Most of the current work on the use of deep reinforcement learning for portfolio optimisation has been inspired by the work of researchers such as (Almahdi and Yang, 2017; Jiang et al., 2017; Liang et al., 2018; Wang and Yang, 2019; Soleymani and Paquet, 2020), who leveraged the use of deep learning (LeCun et al., 2015) for continuous action space and utilised a model-free RL approach to model the dynamics of the market through an exploration strategy (Yashaswi, 2021).

Convolutional Neural Networks (CNN) (LeCun et al., 2015) consist of convolution layers, pooling layers, and various fully connected layers. CNN perform convolutional operations on input data and features using a set of kernels that result in various feature maps in the convolutional layer. In general, the pooling layer follows a convolutional layer, which is utilised to reduce the dimensions of feature maps and network parameters. Capsules Networks (CapsNet) (Hinton et al., 2011; Sabour et al., 2017) on the other hand, involve the grouping of neurons together to form capsules, where each neuron's output represents a different property of the same feature space. Training and inferencing are done on the group (capsules) as a single unit. The output of convolutional neural networks serves as the input of a capsule in CapsNet.

The selection of a reward function in reinforcement learning can significantly affect the performance of an RL algorithm, but the determination of a reward signal has been shown to be the most challenging part of the design of a reinforcement learning problem. While Jiang et al. (2017) and other researchers utilised the log-returns as the reward function to be maximized, their approach did not take into account the risk-related parameters in the formulation of the reward function. This approach can cause an RL algorithm to take riskier positions which could lead to a loss in portfolio value and eventually create downside opportunities. The maximisation of the expected cumulative reward over time, referred to as the reward hypothesis, is the main driving force that determines the success of an RL algorithm. Risk-adjusted reward functions such as the Sharpe Ratio (Sharpe, 1964, 1994) have been shown to be unsuitable candidates to be used as objective functions that are required to be maximized for adaptive episodic learning (Ziemba, 2005). This is because it is not additive in nature and cannot be used in Q-learning and vanilla policy gradient methods.

Experience Replay (Lin, 1992) has been shown to play a significant role in deep reinforcement learning since it enables the RL algorithm to memorise and re-use past experiences. Experience replay offers great stability to the training process of an RL algorithm and leads to improved sample efficiency by breaking the temporal correlations between the samples (Zha et al., 2019). Most of the experience replay techniques in use have been applied to off-policy RL algorithms which usually use a uniform sampling strategy to replay past experiences, including Jiang et al. (2017) who utilized it on a Deterministic Policy Gradient (DPG) algorithm for portfolio optimization. Existing approaches used a full exploiting technique as a reward function and utilised the explicit average of periodic logarithmic returns to calculate the set of portfolio weights. This leads to extreme weights, which often alternate between 0 and 1 in a short space of time. This implies network instability and results in the over-concentration of resources in only a small number of assets since it does not factor in the volatility or riskiness of the portfolio; hence, they are not penalised for taking on high risk. Although the approaches used by Jiang et al. (2017); Almahdi and Yang (2017); Liang et al. (2018); Soleymani and Paquet (2020) factored in transaction cost together with the application of the model-free RL algorithm made of single buffer memory, their approach did not consider the instability and risk associated with the non-stationary and noisy financial market environment.

Against this background, this chapter proposes a model-free simple Deterministic Policy Gradient Algorithm together with a Capsule Network (CapsNet) used as a deep learning architecture to learn and enforce investment policies that incorporate novel strategies that lead to better and more stable portfolio optimisation outcomes. The network is trained using gradient ascent through a dual online batch learning scheme where an exponential distribution sampling strategy is employed for scenarios with Agents' rewards greater than or equal to rewards from a baseline model, while a geometric distribution sampling strategy is used for instances where the agents' rewards are greater than or equal to rewards from a baseline model. It specifically involves an investigation of the use of reinforcement learning to optimise the portfolio of assets through the maximisation of its discounted cumulative Differential Sharpe Ratio and the exploration of their use in the learning of market patterns to make a profit.

The Mean Reward-to-CVaR Ratio (Tong and Wu, 2014) which is a risk-adjusted reward performance evaluation measure that uses the ratio of mean portfolio returns and the conditional Value at Risk (CVaR) to evaluate the performance of the proposed RL strategy, has been used as part of the performance evaluation strategy. Additionally, the final reward-to-VaR ratio (Alexander and Baptista, 2003a), which is also a risk-adjusted reward performance metric and uses the ratio of the final portfolio returns and the Value at Risk (VaR) to assess how well a model performs, was also used.

Our proposed techniques which serve as our main contributions for this chapter are:

- Multi-Memory Weight Reservoir (MMWR) training scheme using Capsules Neural Network: This facilitates and improves the optimisation process of the portfolio weights. It, therefore, helps in the sequential re-balancing of the portfolio throughout the trading period using a continuous action space.
- The discounted cumulative reward function, referred to as a Markov Differential Sharpe Ratio, provides stability and optimises the training process since it is based on the discounted cumulative Differential Sharpe Ratio reward function.

To the best of the author's knowledge, this is the first research work that leverages a multi-memory approach and the maximisation of the Markov Differential Sharpe Ratio derived from actual volatility and dynamic risk-free assets. It further integrates with a deep CapsNet RL Algorithm framework in the portfolio optimisation domain, from which Mean Portfolio value, Final Portfolio value, Sharpe Ratio, CVaR, VaR, Final Portfolio returns-to-VaR ratio, and average portfolio returns-to-VaR ratio are used as portfolio evaluation metrics to measure its effectiveness.

The rest of this chapter is organised as follows: The related work is provided in Section 6.2. Section 6.3 covers the background information on Capsules Network and Reinforcement Learning, including deep reinforcement learning and how it can be applied as a Markov Decision Process. The underlying mathematical modelling associated with Portfolio management is provided in Section 6.4 as the Problem formulation section. Our proposed Deep Reinforcement learning techniques using the Multi-Memory Weight Reservoir (MMWR) and discounted cumulative reward function using the Markov Differential Sharpe Ratio together and their architecture are introduced in Section 6.5. This is followed by our experimental setup and the results in section 6.6. The conclusion of this chapter is captured in Section 6.7.

6.2 Related Works

The application of deep Reinforcement learning models in Portfolio Optimisation has increased exponentially over the past years because of their increased complexity. Prior to this, most portfolio optimisation models were based on different versions of Modern Portfolio Theory(MPT) (Markowitz, 1952). These approaches were not fit for purpose as they were static in nature and dependent on linear computational methods. The dynamic approaches to overcome these issues, such as convex optimisation, concave optimisation, and dynamic programming were inefficient and lacked the ability to capture market information. This is because they required discrete action space-based models (Cai et al., 2013). A variety of Deep Neural network structures combined with deep deterministic policy gradients to optimise cryptocurrency portfolios have been introduced (Jiang et al., 2017). This work was extended by Ye et al. (2020) who incorporated different data sources, such as news, to improve robustness against market uncertainty. There has been the existence of deep machine learning-based techniques for financial trading, but most of these attempts have been used to predict the movement of prices or trends (Heaton et al., 2016). There exist some lapses in this approach because their performance depends on the degree of accuracy of the prediction, and it is now evident that the prediction of future market prices is difficult. Price predictions are not known to be market actions (Jiang et al., 2017), so the ability to convert them into market actions requires an extra logical layer. RL algorithms have been used to produce discrete trading signals in single assets and were not suitable for multiple asset trading operations such as portfolio management (Moody and Saffell, 2001; Dempster and Leemans, 2006; Deng et al., 2017). On the other hand, model-free and fully machine learning techniques have been utilised as an algorithmic trading technique suitable for portfolio management (Benhamou et al., 2021; Jiang et al., 2017).

A cumulative reward function that only maximises the average logarithmic accumulated return R factoring the risk involved in a portfolio creation has been used (Jiang et al., 2017). Moody et al. (1998) on the other hand, used the Differential Sharpe Ratio (D_t) as the main objective function, but was used as an immediate reward that lacked the ability to achieve optimality. A deep long–short-term-memory network has been trained using double Q-learning to achieve positive gains in a bearish cryptocurrency market (Bu and Cho, 2018). On the other hand, Q-learning and other value-based RL algorithms for stock trading have been evaluated (Pendharkar and Cusatis, 2018). An adversarial training technique that leads to an improvement in the performance of deep reinforcement learning methods has also been proposed (Liang et al., 2018). It utilised a deep residual network in its designs that led to positive returns after being tested the Chinese stock market. Adaptive stock trading strategies that utilised deep reinforcement learning methods such as Gated Deep Q-learning and Gated Deterministic Policy Gradient trading strategies have also been proposed (Wu et al., 2020). Betancourt and Chen (2021) proposed the use of deep reinforcement learning for cryptocurrency trading using a dynamic number of assets (DNA) that has the ability to consider all assets in the market, which automatically adapts when new cryptocurrencies are introduced to the market. In the work of Betancourt and Chen (2021), DNA-S was used to represent portfolio agents trained with the Differential Sharpe Ratio (Moody et al., 1998). The approach adopted in this work to compute the reward function used the average of the difference between the instantaneous reward of the RL algorithm and the instantaneous reward of the secured portfolio compared to the approaches used by Moody and Saffell (1998); Betancourt and Chen (2021) that used exponential moving estimates of the returns and standard deviation of the return R_t in their calculations.

The next section covers the preliminaries of the Capsule Network (CapsNets).

6.3 Preliminaries of Capsules Network (CapsNets)

A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity, such as an object or an object parts (Hinton et al., 2011; Sabour et al., 2017). CapsNets use dynamic routing shown in Algorithm 7 as a strategy to assign weights to neurons' connections, as opposed to pooling operations used in CNNs. This leads to a vector output in the CapsNet model. The features of a financial portfolio, such as high, low, open and close prices, used as input, $\mathbf{X_t}$ are divided into many capsules, which in turn contain neurons. A capsule is therefore a wrapper around a dedicated group of neurons (Dombetzki, 2018).

The operations and inner workings of a typical capsule are demonstrated in Figure 6.1.

The features of financial data are processed on the basis of the type of capsule employed. The output of a capsule is composed of the probability that the features of the financial data encoded by the capsule are present given a set of vector values commonly called instantiation parameters. The high-level structure can be considered as a parse tree, where each active capsule chooses a capsule in the layer above it as its parent capsule.

The total input of a capsule (o_j) is a weighted sum of all the prediction vectors, $\hat{\mathbf{u}}_{j|i}$ as

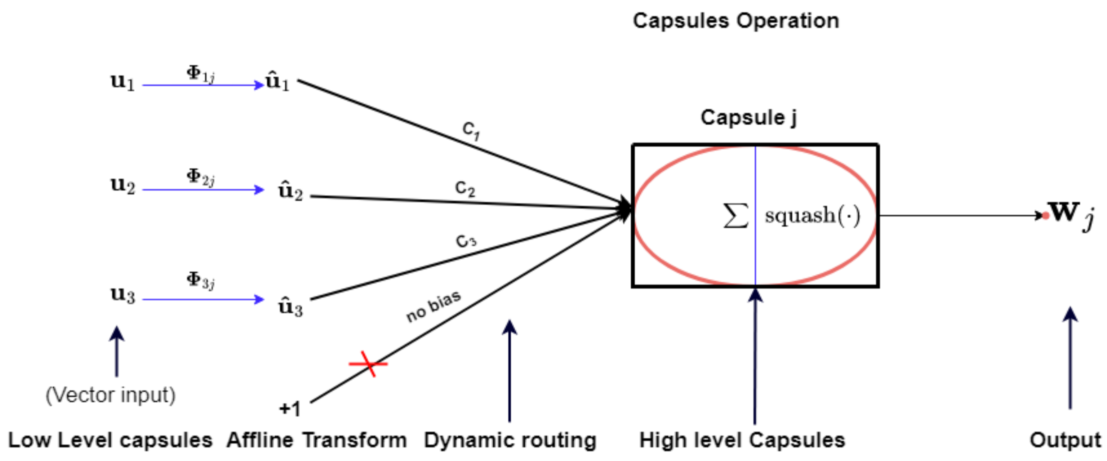


Figure 6.1: Schematic presentation of the operations and inner workings of Capsule

shown in equation 6.2

$$\hat{\mathbf{u}}_{j|i} = \Phi_{ij}\,\mathbf{u}_i$$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{6.1}$$

$$o_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i} \tag{6.2}$$

where Φ_{ij} is the weight coefficient, c_{ij} is the coupling coefficient shown mathematically in equation 6.1 and ensures that the prediction of capsule i in layer l is in agreement with that of capsule j in the layer above, $l+1$. The sum of the attention coefficient, c_{ij} is always equal to 1 through the use of a softmax function, whose input, b_{ij} is produced by the dynamic routing algorithm. The agreement is a scalar product given as $b_{ij} = \mathbf{w}_j \cdot \hat{\mathbf{u}}_{j|i}$ and added to the initial logit b_{ij} to compute new coupling coefficients to link the capsule i to higher-level capsules using $b_{ij} \leftarrow \mathbf{w}_j \cdot \hat{\mathbf{u}}_{j|i}$. A squash function represented in equation 6.3 is a non-linear activation function used on vectors to ensure that short vectors are squashed to almost zero, while long vectors are reduced to a length slightly below 1 (Sabour et al., 2017).

$$\mathbf{w}_j = \frac{\| o_j \|^2}{1+ \| o_j \|^2}\frac{o_j}{\| o_j \|} \tag{6.3}$$

where $\mathbf{w}_j$ represents the vector output of the capsule j and o_j represents its total input.

The dynamic routing algorithm used in CapsNet is represented in algorithm 7.

Algorithm 7 Dynamic routing algorithm

```
For all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$ : $b_{ij} \leftarrow o$
for r iterations do
    for all capsule $i$ in layer $l$: $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$ with softmax computing equation 6.1
    for all capsule $j$ in layer $(l+1)$: $\mathbf{o}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule $j$ in layer $(l+1)$: $\mathbf{w}_j \leftarrow \text{squash}(\mathbf{o}_j)$ where squash computes equation 6.3
    for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$ : $b_{ij} \leftarrow b_{ij} + \mathbf{w}_j \cdot \hat{\mathbf{u}}_{j|i}$
    return $\mathbf{w}_j$
end for
```

The structure of the Capsules Network consists of a standard convolution layer (Conv layer), a primary capsule layer (Primary-Caps layer), a secondary capsule layer (Outer layer) and a fully connected capsule layer (Sabour et al., 2017). The Conv layer in CapsNet is identical to the convolutional layer in CNN (Sabour et al., 2017), while Primary-Caps uses a convolution operation on the input and then reallocates the output to several capsules using a dynamic routing algorithm. The output shape of the Conv layer is made up of batch size, height, width, and number of channels, while the output shape of the Primary-Caps layer consists of batch size, height, width, number of capsules, and output vector length of the capsules. The outer layer then follows the Primary caps layer. The Decoder is made up of three fully connected networks, where the last layer uses sigmoid activation, and the other two layers use a ReLU activation function. The Decoder takes the vectors from the correct outer layer and then learns and maps them back to the original features to form $\hat{\mathbf{X}}_T$, serving as a regulariser in the process. The decoder forces the capsules to learn features that are important for the reconstruction of the original input data.

The problem formulation comprising the market assumptions and the mathematical model used is presented in the next section.

6.4 Problem Definition

As has been defined earlier, Portfolio management is the act of constantly reallocating capital resources to various financial assets with the aim of maximising profit. This section provides a mathematical background for a portfolio management problem using transaction costs in an approach similar to the one used by Jiang et al. (2017) which was initially introduced by Ormos and Urban (2013).

Trading days

It is known that stock exchanges and markets have one calendar day as the trading period with time divided into periods of equal lengths. In this work, the CapsNet-based RL algorithm performs a reallocation of the funds across the assets at the beginning of each trading period. It is known that the prices of financial trading assets do fluctuate during the trading day and are characterised by four main metrics which are Open, Low, High, and Close prices which respectively represent the price at the opening of the market, the lowest price reached during the day, the highest prices obtained during the day's trading and the price at the closing of the markets for the day.

Market assumptions

In this work, backtest trading is considered with the assumption that the RL algorithm does not know the future market information. As in any market environment, some assumptions are considered close to reality if the assets' trading volume is high. The two major assumptions underlying our work are:

- Zero Slippage: Due to high market liquidity, each trade can be executed and completed immediately at the last price when an order is placed.
- Zero Market Impact: The amount invested by our RL algorithm is insignificant to have an effect on the outcome of the market.

Problem statement

Our proposed deep-reinforcement learning process for portfolio optimisation is detailed in Section 6.5, and outlined below.

Let:

$$\mathbf{V}_t = [v_{t-l+1} \oslash v_t, v_{t-l+2} \oslash v_t, ..., v_{t-1} \oslash v_t, v_t \oslash v_t] \tag{6.4}$$

$$\mathbf{V}_t^{hi} = \left[v_{t-l+1}^{hi} \oslash v_t, v_{t-l+2}^{hi} \oslash v_t, ..., v_{t-1}^{hi} \oslash v_t, v_t^{hi} \oslash v_t\right] \tag{6.5}$$

$$\mathbf{V}_t^{lo} = \left[v_{t-l+1}^{lo} \oslash v_t, v_{t-l+2}^{lo} \oslash v_t, ..., v_{t-1}^{lo} \oslash v_t, v_t^{lo} \oslash v_t\right] \tag{6.6}$$

where: $\oslash$ is the element-wise division operator. Define the price tensor as a matrix $\mathbf{X}_t$ which is a stack of three normalised price matrices, $\mathbf{V}_t$, $\mathbf{V}_t^{hi}$ and $\mathbf{V}_t^{lo}$ given in equations

(6.4), (6.5) and (6.6) respectively.

To generate an initial non-optimised portfolio weight output, $\mathbf{w}_{t_{init}}$ at the start of the trading activities at any period and shown mathematically in Equation (6.7).

$$\mathbf{w}_{t_{init}} = \pi_{\theta_{init}}(\mathbf{X}_t, \mathbf{w}_0) \tag{6.7}$$

where θ_{init} represents the initial random set of parameters used by the Deep CapsNet to start the gradient update process and $\pi_{\theta_{init}}$ is the initial policy used at the start of the training process. Figure 6.2 shows how the reconstruction loss and the marginal loss operate together to update the policy network.

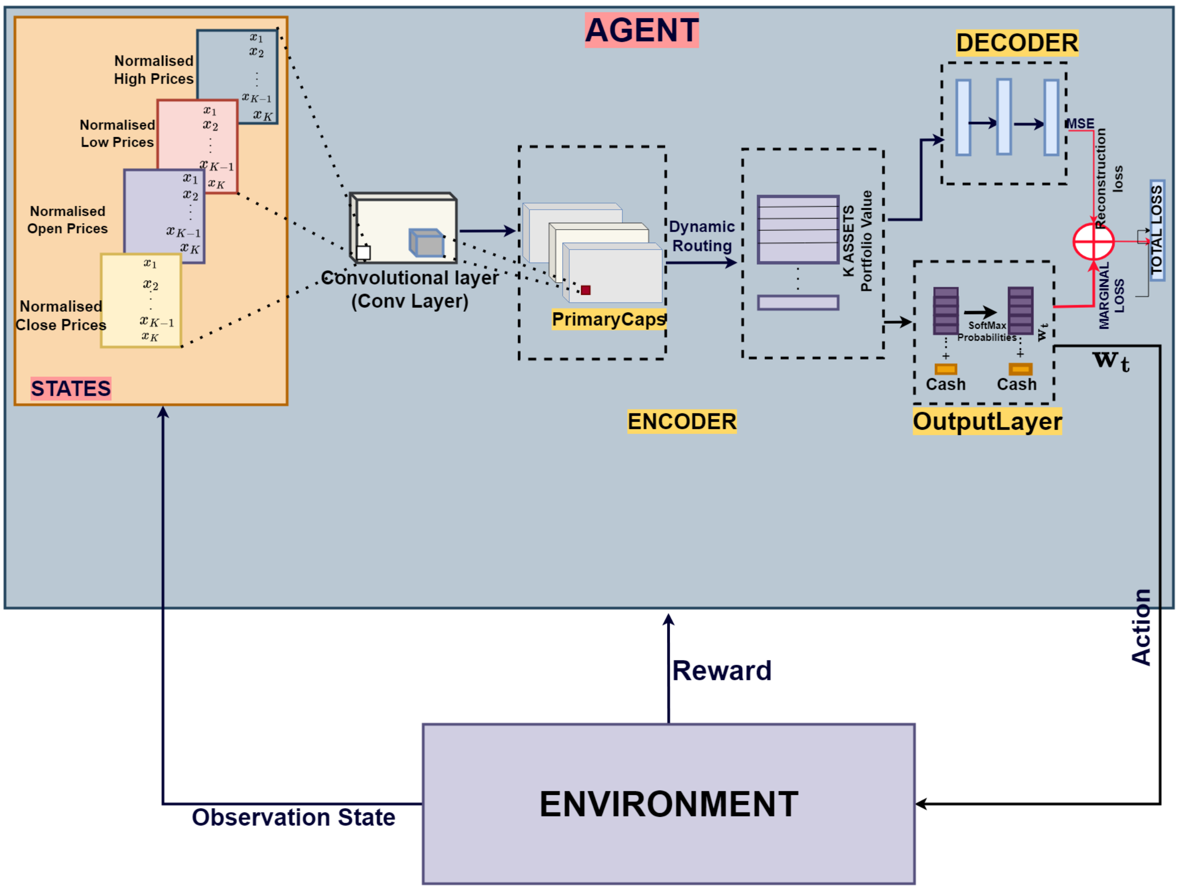


Figure 6.2: Capsules Network structure showing the Marginal Loss and Reconstruction loss

The policy π can be parameterised with a set of parameters of the CapsNet, θ which can be updated to improve the parameterised policy (π_θ). At each training step t, θ_t updates the policy π_{θ_t} which is in turn used to calculate $\mathbf{w}_t$ using equation (6.8).

$$\mathbf{w}_t = \pi_{\theta_t}(\mathbf{X}_t, \mathbf{w}_{t-1}) \tag{6.8}$$

The objectives of this chapter are two-fold; (i) to introduce a novel policy π using the parameters of Deep CapsNet, θ, which is optimised using a Markov Differential Sharpe Ratio reward function; (ii) to introduce a novel workflow of the MMWR that facilitates the online training of θ per time step to achieve an optimised $\mathbf{w}_t$.

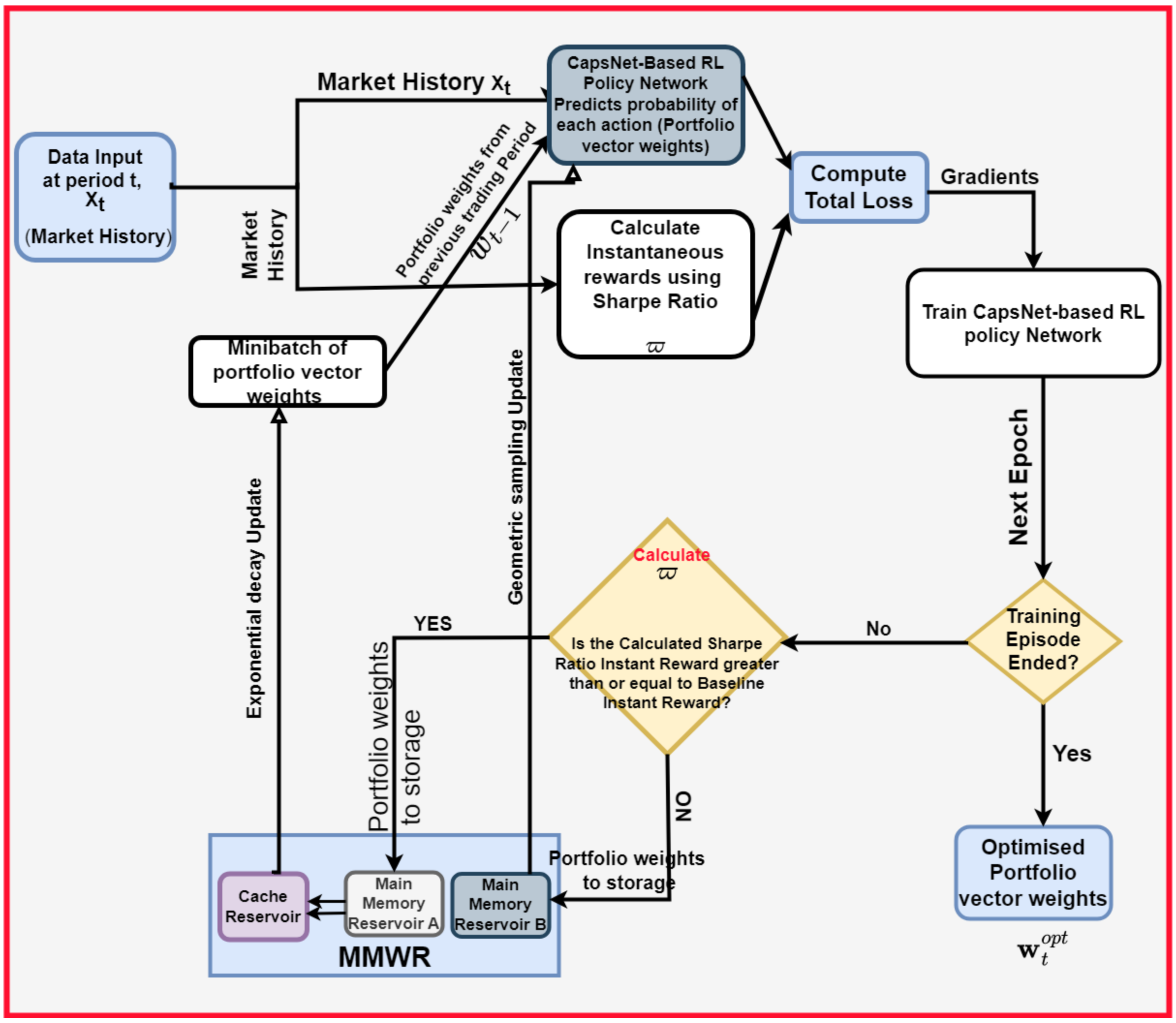

Data Input at period t, X_t (Market History)
Market History X_t
Market History
CapsNet-Based RL Policy Network Predicts probability of each action (Portfolio vector weights)
Portfolio weights from previous trading Period w_{t-1}
Calculate Instantaneous rewards using Sharpe Ratio ϖ
Compute Total Loss
Gradients
Train CapsNet-based RL policy Network
Minibatch of portfolio vector weights
Exponential decay Update
Geometric sampling Update
Next Epoch
Calculate ϖ
Is the Calculated Sharpe Ratio Instant Reward greater than or equal to Baseline Instant Reward?
YES
No
NO
Training Episode Ended?
Yes
Portfolio weights to storage
Portfolio weights to storage
Optimised Portfolio vector weights $\mathbf{w}_t^{opt}$
Cache Reservoir
Main Memory Reservoir A
Main Memory Reservoir B
MMWR

Figure 6.3: General workflow of Multi-Memory weight Reservoir-based RL for portfolio optimisation: The deep CapsNet network predicts the initial portfolio weight vector, $\mathbf{w}_t$ using Equation (6.7). Then uses the initial portfolio weight vector, $\mathbf{w}_t$ obtained in Equation (6.8), to interact with the environment to obtain an initial instantaneous reward and the next state. This reward is then used to calculate the discounted cumulative reward. The discounted return and portfolio vector weights are then used to compute the gradients used to train the deep CapsNet policy Network

The general workflow of the MMWR and the RL algorithm is demonstrated in Figure 6.3 and shows the computation of the gradients of the deep CapsNet network and its optimisation by the Markov Differential Sharpe Ratio reward through the use of the MMWR framework to achieve optimised portfolio vector weights, $\mathbf{w}_t^*$ at the end of the training session. Kindly refer to section 6.5 for the full details of the optimisations and training process involved.

The next section introduces our proposed Deep Reinforcement Learning Process for Portfolio Optimisation using the Deep Capsules Network.

6.5 Proposed Deep Reinforcement Learning Process for Portfolio Optimisation using Deep Capsules Network

Deep Capsules Network Architecture

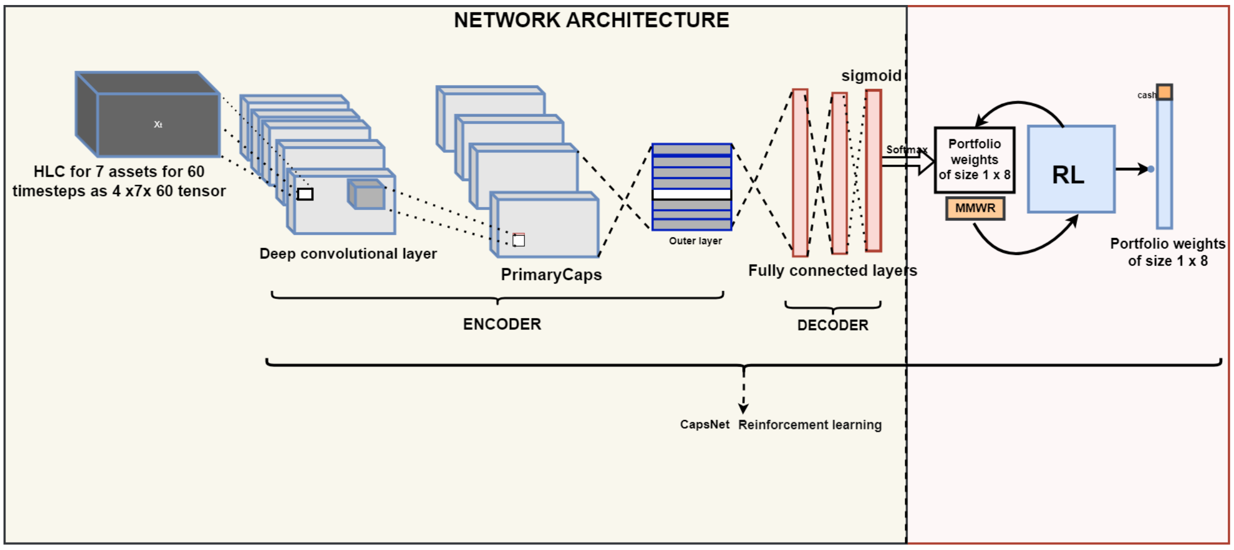

NETWORK ARCHITECTURE
Xt
HLC for 7 assets for 60 timesteps as 4 x7x 60 tensor
Deep convolutional layer
PrimaryCaps
Outer layer
sigmoid
Softmax
Fully connected layers
ENCODER
DECODER
Portfolio weights of size 1 x 8
MMWR
RL
cash
Portfolio weights of size 1 x 8
CapsNet Reinforcement learning

Figure 6.4: Structure of Deep CapsNet-based RL policy network: Made up of a deep convolutional layer, PrimaryCaps, Outer Layer, 3 fully connected layers and RL component. The structure of the CapsNet is of a shape of (f, t', k), where f is the number of features, K is the number of non-cash assets and t' is the number of input periods before t

In our framework, we introduce a Deep Capsules Neural Network for enforcing the policy, as depicted in figure 6.4. Refer to Section 6.5 for detailed information on the network setup and component implementation. The policy is initially learnt offline from historical data using Monte-Carlo policy gradient deterministically in a dual online mini-batch training process. The policy, π is then continuously updated as new information becomes available using the parameters of the Deep CapsNet, θ_t, optimised using our proposed reward function. The Deep CapsNet was used to observe each non-cash asset in the portfolio at a time to recommend and enforce the investment policy and used the softmax function to ensure non-negative output weights that sum up to one.

The next section introduces our proposed Markov Differential Sharpe Ratio, which is used as the main reward function in the work.

Proposed Markov Differential Sharpe Ratio as Reward function

The instantaneous reward function used was represented by a scalar value that indicated the performance of the CapsNet Policy agent in any given time period, t. The Sharpe ratio, ϖ was used as an instantaneous reward to maximise the final portfolio value, which served as an objective function. The Sharpe ratio represented mathematically in equation (6.9) was used as an instantaneous reward because of its effectiveness in measuring the performance of an investment portfolio since it utilises both risks and returns in its calculations.

$$\varpi = \frac{\mathbb{E}(r_t^r)}{\sigma(r_t^r)} \tag{6.9}$$

where $\mathbb{E}(r_t^r) = (1/T)\sum_{i=1}^{T} r_t^r$ and σ are the rate of returns and the standard deviation, respectively.

Our proposed Markov Differential Sharpe Ratio ($R_{(d)}$) is a modification and an extension of the work of Moody et al. (1998) on the Differential Sharpe Ratio, D_t which is represented using equation (6.10). Our reward function used a special value-based reward function that utilised the Markov Reward Process on the Differential Sharpe Ratio to form the Markov Differential Sharpe Ratio, as presented in equation (6.10). The Markov Reward process can be used under the Bellman optimality condition with a discount factor of 0.5 used in the Differential Sharpe Ratio, D_t to estimate the Discounted cumulative reward, the Markov Differential Sharpe Ratio $R_{(d)}$, which is a using a time period of 10. This can then be maximised by the CaspNet-based policy network during the mini-batch training scheme shown in equation (6.11).

$$\begin{aligned}
\Delta A_t &= \varpi - \varpi_s \\
\Delta B_t &= (\varpi)^2 - \varpi_s \\
D_{(t)} &= \frac{B_{(t-1)}\Delta A_{(t)} - \frac{1}{2}A_{(t-1)}\Delta B_{(t)}}{(B_{(t-1)} - A^2_{(t-1)})^{3/2}} \qquad (6.10) \\
A_{(t)} &= \eta\varpi_{(t)} + (1-\eta)A_{(t-1)} = A_{(t-1)} + \eta(\varpi_{(t)} - A_{(t-1)}) \\
B_{(t)} &= \eta\varpi^2_{(t)} + (1-\eta)B_{(t)} = B_{(t-1)} + \eta(\varpi^2_{(t)} - B_{(t-1)}) \\
R_{(d)} &= \mathbb{E}[G_t|S_t = s] \\
R_{(d)} &= \mathbb{E}[D_{(t+1)} + \gamma v(D_{(t+1)})|S_t = s] \qquad (6.11)
\end{aligned}$$

where:
ϖ represents the instantaneous reward from the CapsNet-based agent at time t.
ϖ_s represents instantaneous reward from the secured portfolio at time t.
A_t and B_t represent the average estimates of the first and second moments of the instant Reward ϖ respectively, and η represents the decay rate.

The next section introduces the framework of our proposed Multi-Memory Weight Reservoir (MMWR) and covers the basic its basic structure.

Proposed Multi-Memory Weight Reservoir (MMWR) framework

Any intelligent system with continuous interaction with the environment encounters the problem of continual learning, where new experiences are constantly being obtained while old experiences may still be relevant. The strategy of using experience replay memory (Mnih et al., 2016) to achieve continual reinforcement learning in this work was the inspiration behind its use.

The proposed memory framework is divided into three parts; these are the Main Reservoir A (M_A), the Main Reservoir B (M_B) and the Cache Reservoir (M_C) as demonstrated in Figure 6.5. The Main Reservoir A (M_A) is designed to store and stack only the portfolio weight vectors obtained from the Deep CapsNet-based RL algorithm with an instantaneous reward, ϖ at time step t greater than or equal to the reward from the baseline model, ϖ_{eq} during the training episode. On the other hand, Main Reservoir B (M_B) only stores the portfolio weight vectors from the Deep CapsNet-based RL algorithm with an instant reward, ϖ at time t less than the reward from the baseline model, ϖ_{eq}. The Main Reservoir A (M_A) serves as the feeder for the Cache reservoir (M_C).

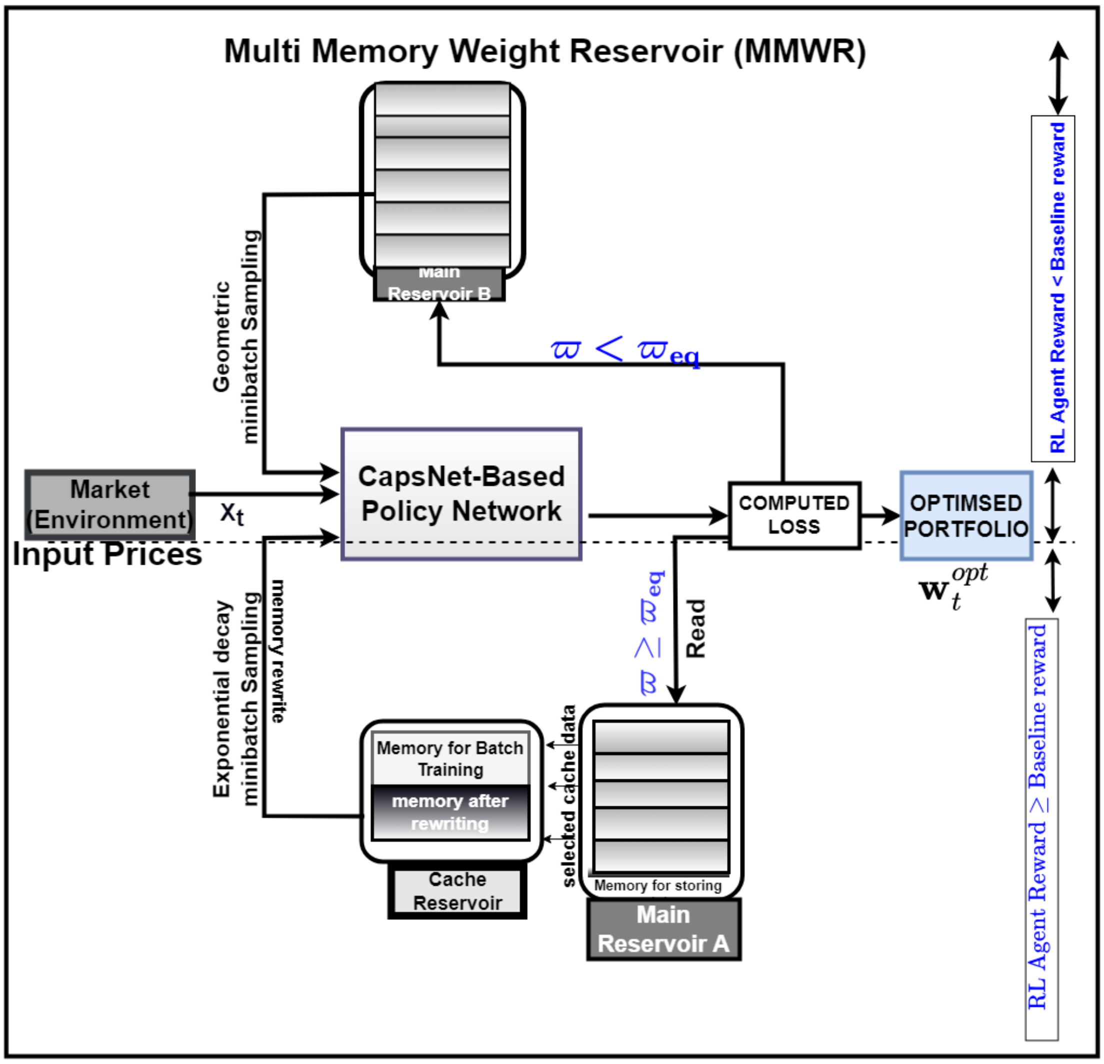

Multi Memory Weight Reservoir (MMWR)
Main Reservoir B
Geometric minibatch Sampling
$\varpi < \varpi_{eq}$
RL Agent Reward < Baseline reward
Market (Environment)
X_t
Input Prices
CapsNet-Based Policy Network
COMPUTED LOSS
OPTIMSED PORTFOLIO
$\mathbf{w}_t^{opt}$
Exponential decay minibatch Sampling
memory rewrite
$\varpi \geq \varpi_{eq}$
Read
Memory for Batch Training
memory after rewriting
Cache Reservoir
selected cache data
Memory for storing
Main Reservoir A
RL Agent Reward ≥ Baseline reward

Figure 6.5: Architecture of Multi-Memory weight Reservoir (MMWR): The Main Reservoir A (M_A) stores and stacks the $\mathbf{w}_t$ obtained from the Deep CapsNet-based RL algorithm with a reward, ϖ at time step t greater than or equal to the reward from the baseline model, ϖ_{eq} during the training episode. On the other hand, Main Reservoir B (M_B) stores only $\mathbf{w}_t$ from the Deep CapsNet-based RL algorithm with ϖ at time t less than ϖ_{eq}. The Main Reservoir A (M_A) serves as the feeder for the Cache reservoir (M_C).

The content of the Cache Reservoir (M_C) acts as a secondary storage for the randomly sampled portfolio weight vectors, $\mathbf{w}_t$ from which the contents are sampled to the RL algorithm for online mini-batch training to update the individual weights of the assets in each portfolio. The portfolio weight vectors in the Main Reservoir A (M_A) and Main Reservoir B (M_B) have holding capacities of T_A and T_B, respectively. Hence, the total portfolio weights of the Main reservoirs M_A and M_B can be expressed using equations (6.12) and (6.13) respectively:

$$M_A = \mathbf{w}_t^A(i), \text{where}\, i = 1, 2, ..., T_A \tag{6.12}$$

$$M_B = \mathbf{w}_t^B(i), \text{where}\, i = 1, 2, ..., T_B \tag{6.13}$$

Where:
-T_A and T_B are the maximum history held by M_A and M_B respectively.
-$\mathbf{w}_t^A$ $\mathbf{w}_t^B$ and represent the portfolio weight vectors stored in M_A and M_B respectively.

Likewise, the sampled portfolio weight vector in the Cache Reservoir has a holding capacity of H and the total portfolio weights vectors of the Cache reservoir can be represented using equation (6.14)

$$M_c = \mathbf{w}_t^C(i), \text{where}\, i = 1, 2, ...H \tag{6.14}$$

where:
-H is the holding capacity of the Cache Reservoir.
-$\mathbf{w}_t^C$ and represent the portfolio weight vectors stored in M_C.

Refer to Section 6.5 for details of how the proposed MMWR is used for the mini-batch training scheme and the pseudo-code it uses.

The next section introduces our proposed policy optimisation using the MMWR framework.

Policy Optimisation Using our proposed Deep CapsNet-based MMWR framework

A policy is deemed optimal if it leads to the maximisation of our proposed Markov Differential Sharpe Ratio $R_{(d)}$ used as a reward function. A policy gradient was deterministically used to train the CapsNet-based policy network. The policy $\pi_{\theta_\mathbf{t}}$, from the CapsNet-based policy network, used a mapping from state space to action space $\pi_{\theta_\mathbf{t}} : s \to A$. To achieve an optimal policy that maximises the returns, the parameters θ_t and $\mathbf{a}_t = \pi_{\theta_\mathbf{t}}(\mathbf{s}_t)$ are specified.

Our proposed CapsNet-based policy determines the best action taken by the RL algorithm learnt through the deep CapsNet framework in order to achieve a maximised Final Portfolio value. It uses the normalised historical data for Low, High, Open and Close prices to feed into the Deep CapsNet. The CapsNet has the ability to capture the spatial relationship between asset prices, portfolio weights, and rewards. The CapsNet receives the portfolio's state representations given by the Low, High, Open and Close prices of each asset and previous weight vector ($\mathbf{w}_{t-1}$) and returns the appropriate actions given by $\mathbf{w}_t$ and can predict the portfolio allocation. It consists of multiple layers of capsules, each capsule representing a set of features. The output of the CapsNet is a set of weights that represent the allocation of portfolio assets. The portfolio weights are calculated based on the output vectors of the capsules, which encode the pose of different portfolio allocations. The portfolio weights can be calculated using a softmax function, which normalises the output vectors of the capsules to produce a probability distribution over the possible portfolio allocations. Each element in the probability distribution represents the probability of assigning a certain weight to a certain asset or factor in the portfolio.

The dynamic routing agreement between the capsules shown in Algorithm 7 is utilised to distinguish the vectors according to the level of agreements between them. In a nutshell, the training process for the capsules can be defined as the extraction and refinement of the active routes from a preceding Primary-Caps to the successive output layer. It is used to optimise portfolio allocation based on the CapsNet output. It helps the capsules to learn the hierarchical structure of the input data $\mathbf{X}_t$, such as correlations between different asset classes, the interactions between different market factors, and the patterns of market volatility over time.

A loss function $\mathcal{L}_M$ can be used to measure the difference between the predicted portfolio allocation and the actual portfolio allocation and penalises CapsNet for generating allocations that deviate too much from the current portfolio allocations. The loss function, $\mathcal{L}_M$ is mathematically shown in Equation (6.15)

$$\mathcal{L}_M = \sum |\mathbf{w}_t - \mathbf{w}_{t-1}^{opt}| \tag{6.15}$$

where $\mathbf{w}_{t-1}^{opt}$ is the optimised portfolio weights for the previous period. The Reconstruction loss $\mathcal{L}_r$ measures how well the CapsNet can reconstruct the input data from the output vectors, and it serves as the regulariser of the network to ensure it does not overfit the training data. The Reconstruction loss, $\mathcal{L}_r$ associated with this is calculated using the Mean Square Error (MSE) as shown in equation (6.16).

$$\mathcal{L}_r = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{X}_t - \hat{\mathbf{X}}_t)^2 \tag{6.16}$$

where $\hat{\mathbf{X}}_t$ is the reconstructed input data.

The Total Loss, $\mathcal{L}_T$ used for the training of the CapsNet policy is shown in Equation (6.17).

$$\mathcal{L}_T = \mathcal{L}_M + \varrho \mathcal{L}_r \tag{6.17}$$

Where ϱ is the weighting parameter for the reconstruction loss and is always greater than 0. The total loss $\mathcal{L}_T$ can be subtracted from the reward function to guide the learning process of the reinforcement learning algorithm. They measure the difference between the predicted portfolio weights and the actual portfolio weights.

In the initial learning stage, the parameters of Capsnet, $\pi_{\theta_{\textbf{init}}}$ are initialised randomly. With the use of feedback from the markets through asset price changes, CapsNet uses the difference between the expected reward and the actual reward received to adjust its gradients accordingly to improve the reward profile. The gradient used to update the policy network training can be computed through the use of Total loss, $\mathcal{L}_T$ given in Equation (6.17). The performance metric of the policy parameterised by θ_t, π_θ for the time interval $[t_{bs_a}, t_{bs_b}]$ is defined as the corresponding Markov Differential Sharpe Ratio $R_{(d)}$ of the interval shown in equation (6.18).

Let us formally denote $J_{[t_{bs_a},t_{bs_b}]}(\pi_\theta)$ as the quantity of Markov Differential Sharpe Ratio cumulative reward ($R_{(d)}$) shown in equation 6.11 that needs to be directly maximised by the policy gradient using the mini-batch of size bs. The use of the CapsNet Policy network with parameter, π_θ generates trajectory,
$\varphi = [S_0, A_1, R_{d_1}, S_1, ..., S_f, A_f]$ for the trading environment to obtain a sequence of rewards.

$$J_{[t_{bs_a},t_{bs_b}]}(\pi_\theta) = \mathbb{E}_{\pi_\theta \sim \varphi}\left[\frac{1}{bs}\sum_{b=1}^{bs}\sum_{t=0}^{T_f}(R_{(d)} - \alpha\mathcal{L}_T)\right] \tag{6.18}$$

where α is a loss weighting parameter to determine the trade between the reward and the loss, b is the batch number. After random initialisation, the policy parameter θ can be updated by gradient ascent using the learning rate λ to maximise $J_{[t_{bs_a},t_{bs_b}]}(\pi_\theta)$. The gradients of the reward with respect to the network parameters are computed as follows:

$$\nabla_\theta J_{[t_{bs_a},t_{bs_b}]}(\pi_\theta) = \mathbb{E}_{\pi_\theta \sim \varphi}\left[\frac{1}{bs}\sum_{b=1}^{bs}\sum_{t=0}^{T_f}(R_{(d)} - \alpha\mathcal{L}_T)\nabla_\theta \log \pi_\theta\right] \tag{6.19}$$

The mini-batch training regime can be used to update the parameters.
Let bs represent the total batch size sampled from the Cache reservoir (M_C) and Main reservoir B (M_B) in any training episode.
Let bs_1 represent the mini-batch sample to be drawn for training from the Cache reservoir (M_C) at any time period as shown in Equation 6.20.

$$bs_1 = \varkappa * bs \tag{6.20}$$

Let bs_2 represent the mini-batch sample to be drawn for training from the Main Reservoir B (M_B) at the time period as demonstrated in equation 6.21.

$$bs_2 = (1 - \varkappa) * bs \tag{6.21}$$

where $\varkappa$ is the weighting parameter used for adjusting sample size from the Cache reservoir (M_C). Our proposed approach recommends using a minimum $\varkappa$ of 0.7 to be used for (M_C). The parameter updates of each of these respective mini-batches during training are given in equations (6.22) and (6.23) respectively. The time range of a mini-batch for M_C is $[t_{bs_{1_a}}, t_{bs_{1_b}}]$

$$\theta_{\mathbf{t+1}}{}^{bs_1} \leftarrow \theta_{\mathbf{t}}{}^{bs_1} + \lambda \nabla_\theta J_{[t_{bs_{1_a}}, t_{bs_{1_b}}]}(\theta) \tag{6.22}$$

The time range of a mini-batch for M_B is $[t_{bs_{2_a}}, t_{bs_{2_b}}]$

$$\theta_{\mathbf{t+1}}{}^{bs_2} \leftarrow \theta_{\mathbf{t}}{}^{bs_2} + \lambda \nabla_\theta J_{[t_{bs_{2_a}}, t_{bs_{2_b}}]}(\theta) \tag{6.23}$$

Algorithm 8 Implementation of MMWR-based deterministic policy gradient

```
Input: Market Price Vectors: Open prices, $\mathbf{V}_t^o$ , Close Prices, $\mathbf{V_t}$, High Prices, $\mathbf{V}_t^{hi}$, Low Prices, $\mathbf{V_t^{lo}}$
Require: $\mathbf{w}_t$- Selected portfolio vector weights at $t$, $\mathbf{w}_t^C$- Selected portfolio vector weights from $M_C$, $\mathbf{w}_t^B$- Selected portfolio vector weights from $M_B$
iteration = 0
initial $\epsilon \leq 0.8$
while $iteration \leq n$ do
  if random number $< \epsilon$ then
    $\mathbf{w}_t \leftarrow$ random portfolio weights
  else
    $\mathbf{w}_t \leftarrow \pi_\theta(\mathbf{X}_t, \mathbf{w}_{t-1})$
  end if
  for $b = 1, ..., bs$ do
    Maximise the objective function of a mini-batch, $bs$ with batch number, $b$ :
        $$J_{[0,t_f]}(\pi_\theta) = \mathbb{E}_{\pi_\theta \sim \varphi}\left[\frac{1}{bs}\sum_{b=1}^{bs}\sum_{t=0}^{T_f}(R_{(d)} - \alpha\mathcal{L}_T)\right] \quad (6.24)$$
    The gradients of the reward of the mini-batch with respect to the CapsNet policy network parameters is computed as:
        $$\nabla_\theta J_{[t_{bs_a}, t_{bs_b}]}(\pi_\theta) = \mathbb{E}_{\pi_\theta \sim \varphi}\left[\frac{1}{bs}\sum_{b=1}^{bs}\sum_{t=0}^{T_f}(R_{(d)} - \alpha\mathcal{L}_T)\nabla_\theta \log \pi_\theta\right] \quad (6.25)$$
    Update the Policy Network with Mini-batch from $M_C$:
        $\theta\mathbf{t+1}^{bs_1} \leftarrow \theta_\mathbf{t}^{bs_1} + \lambda\nabla_\theta J_{[t_{bs_{1_a}}, t_{bs_{1_b}}]}(\theta)$
    Update the Policy Network with Mini-batch from $M_B$:
        $\theta_\mathbf{t+1}^{bs_2} \leftarrow \theta_\mathbf{t}^{bs_2} + \lambda\nabla_\theta J_{[t_{bs_{2_a}}, t_{bs_{2_b}}]}(\theta)$
  end for
  $\mathbf{w}_t \leftarrow \mathbf{w}_t^B + \mathbf{w}_t^C$
end while
return $\mathbf{w}_t$
```

The pseudo-code for the implementation of the MMWR-based deterministic policy gradient training procedure is demonstrated in algorithms 8.

The next section covers the details of the training process involved in our proposed MMWR-based mini-batch training scheme.

Proposed dual Stochastic mini-batch Batch training scheme using MMWR

At each time period of each episode, samples of M_A, estimated by the number of assets in the portfolio K are selected and chronologically stored in the Cache Reservoir (M_C). At the time, t the selection process from M_A to M_C involved the application of a decay function $t \times (1 - \tau)$ to obtain the time step from which the stored weights in M_A can be selected and added to M_C.

For the training episode where the agent's instantaneous reward, ϖ is greater than or equal to the instantaneous reward of the equi-weighted baseline model, ϖ_{eq}, the following mini-batch training procedure occurs:

- The batch of Portfolio weights vectors, $\mathbf{w}_t^C$ is selected to be transferred from the cache reservoir M_C for training at period t.
- At the end of the t^{th} training period, the price movement for that period is added to the training set.
- In the next time period, $t+1$, the CapsNet-based policy network is used to train some randomly chosen bs_1 number of mini-batches from the Cache memory M_C.
- A mini-batch, bs_1 starting with time period $t_s \leq t - n_{bs_1}$ is selected using an exponentially distributed probability, $p_\Omega(t_s)$, shown in equation 6.26

  $$p_{\Omega(t_s)} = \Omega e^{-\Omega t_s - n_{bs_1}} \tag{6.26}$$

  where $\Omega \in (0, 1)$ represents the probability-decaying rate used to determine the shape of the probability distribution and n_{bs_1} represents the number of periods in each mini-batch from M_C.

Whilst for the training episode where the CapsNet-based policy agent Instantaneous reward, ϖ is less than the Instantaneous reward of the equi-weighted baseline model, ϖ_{eq}, the following mini-batch training procedure occurs:

- The batch of Portfolio weights vectors, $\mathbf{w}_t^B$ is selected to be transferred from the Main reservoir B, (M_B).

- At the end of the t^{th} training period, the price movement for that period is added to the training set.
- In the next time period, $t+1$, the CapsNet-based policy network trains some randomly chosen bs_2 number of mini-batches from the Main reservoir B (M_B).
- A batch starting with time period $t_s \leq t - n_{bs_2}$ is selected using a geometrically distributed probability, $p_\beta(t_s)$, shown in equation 6.27

  $$p_\beta(t_s) = \beta(1-\beta)^{t-t_s-n_{bs_2}} \tag{6.27}$$

  where $\beta \in (0,1)$ represents the probability-decaying rate used to determine the shape of the probability distribution and n_{bs_2} represents the number of periods in each mini-batch.

The training regime of our proposed MMWR is such that a mixture of Cache memory M_C and Main memory M_B is fed back into the policy network in a proportion to a minimum $\varkappa$ of 0.7 for M_C using equations 6.20 and 6.21. At the end of the trading period t, and after going through the reinforcement learning procedure, the optimised portfolio vector weight, $\mathbf{w}_t$ for the trading period can be calculated using information from the price tensor matrix, $\mathbf{X}_t$ and the previous optimal weight vector,$\mathbf{w}_{t-1}$ at the preceding time step using the specified policy, π to do the computation. This is represented mathematically in equation 6.28.

$$\mathbf{w}_t = \pi(\mathbf{X}_t, \mathbf{w}_{t-1}) \tag{6.28}$$

The pseudo-code for our Proposed Multi-memory weight Reservoir (MMWR) framework is shown in Algorithm 9.

Algorithm 9 The pseudo-code for Proposed Multi-memory weight Reservoir (MMWR)

```
Require: $\varpi$- Instant reward for a RL algorithm, $\varpi_{eq}$ - Instant reward for an equi-weighted (baseline) portfolio, $w_t$- Selected portfolio vector weights at $t$, $\mathbf{w}_t^C$- Selected portfolio vector weights from $M_C$ at $t$, $\mathbf{w}_t^B$- Selected portfolio vector weights from $M_B$ at $t$
Initialize MMWR: Main Reservoir A ($M_A$), Main Reservoir B ($M_B$) and Cache Reservoir ($M_C$ )
iteration = 0
while $iteration \leq n$ do
    Call Algorithm 6 which is the MMWR-based policy gradient computation.
    if $\varpi \geq \varpi_{eq}$ then
        Store $\mathbf{w}_t^A$ in $M_A$
        Sample $K'$ number of samples from $M_A$ and store in $M_C$ using exponential decay operation.
        Sample mini-batch of $bs_1$ number of $\mathbf{w}_t^C$ from $M_c$ using exponential distribution, $p_{\Omega(t_s)} = \Omega e^{-\Omega t_s - n_{bs_1}}$.
    else if $\varpi < \varpi_{eq}$ then
        Store $\mathbf{w}_t^B$ in $M_B$
        Sample mini-batch of $bs_2$ number of $\mathbf{w}_t^B$ from $M_B$ using geometric distribution, $p_\beta(t_s) = \beta(1-\beta)^{t-t_s-n_{bs_2}}$.
    end if
    Use training mixture formula $\varkappa * bs + (1-\varkappa) * bs$ to obtain the mini-batch numbers of $\mathbf{w}_t^C$ and $\mathbf{w}_t^B$ respectively for training in any training episode
end while
return $\mathbf{w}_t^C$ and $\mathbf{w}_t^B \in \mathbf{w}_t$
```

The pseudo code showing the high-level implementation of the Deep CapsNet policy network is shown in Algorithm 10.

Algorithm 10 Algorithm showing the High-level implementation of the Deep CapsNet Policy Network

```
Require:$f, t', K, c$ : Number of Features $f$, Number of input periods $t'$, Number of pre-selected non-cash assets $K$, Cash basis $c$, $n_{bs}$ represents the number of periods in each mini-batch.
Input: Market Price Vectors: Open prices, $\mathbf{V}_t^o$ , Close Prices, $\mathbf{V_t}$, High Prices, $\mathbf{V}_t^{hi}$, Low Prices, $\mathbf{V_t^{lo}}$.
Initialization: Input: $\mathbf{X}_t = [\mathbf{V_t}, \mathbf{V_t^{hi}}, \mathbf{V_t^{lo}}]^T$
while $i \leq n_{bs}$ do
    1: Deep Convolution layer:
    first Convolution:
    Second Convolution:
    Third Convolution:
    2: Primary Capsules:
    3: Dynamic Routing:
    Call Algorithm 7
    4: Portfolio allocation:
    5: Masking
    6: Reconstruction of features using fully connected layers
    7: Adding Cash: Concatenate Cash Bias
    8: Apply Soft-max :
    return  Distributed Portfolio Weights $\leftarrow w = [w_{t,k}]^T$ where $k = 1, ..., K$
    9: Train the network using the MMWR-based policy gradient;
    Call Algorithm 9
end while
Output: $\mathbf{w}_t^B + \mathbf{w}_t^C \in \mathbf{w}_t^{opt}$
return  $\mathbf{w}_t^{opt}$
```

Summary of the Set Up Process

For each asset class, historical prices made of High, Low and Close prices were used as the initial input for the constructed Deep Capsnet, at trading period t that generated the portfolio weight vector as the output. The output of the neural network was the vector of the actions that the agent took. The created environment was such that it could compute the new vector of weights, the new value of the portfolio, and an instant reward. The Deep CapsNet-based RL portfolio agent was set up as follows:

1. The policy Network was designed with a deep Capsule Network(deep CapsNet) which had an input tensor shape $K \times t' \times f$, where K is the number of assets, t' is the previous trading time period and f is the number of features. The four matrix channels were made of the closing, low, open, and high prices of all assets, and time steps based on the number of trading days used and the number of assets.

2. A first convolution through to a fifth convolution was applied to obtain a smaller tensor. A kernel size of 1×3 and a stride of 1 were used without padding to shrink the data size after each convolutional.

3. The output was reshaped to obtain a 4D vector that represented the output of the primary capsules. The output of the second convolutional layer (conv2) was an array containing 32×8=128 feature maps for each instance, where each feature map was 4×7. Hence the shape of this output became (batch size, 4, 7, 128). However, since this first capsule layer was fully connected to the next capsule layer, the primary capsules were reshaped to (batch size, 4×7×40).

4. The output vectors were then squashed using the squash() function, based on equation (6.1) to obtain the output $\mathbf{u}_i$ of each primary capsule i.

5. To compute the output of the PortfolioWeightsCaps, the predicted output (one for each primary capsule or PortfolioWeightsCaps pair) was first computed. The routing by agreement algorithm was then applied. For each capsule, i in the first layer, the output of each capsule j in the second layer was predicted.

6. The softmax function was applied to compute the routing weights $c_i = \text{softmax}(b_i)$. This was then used to compute the weighted sum of all the predicted output vectors for each second-layer capsule, $s_j = \sum_i c_{i,j} \hat{u}_{j|i}$

7. The decoder network was then added to reconstruct the network. Then a fully connected 3-layer neural network was used to learn the reconstruction of the input data based on the output of the capsule network.

8. Instead of sending all the outputs of the capsule network to the decoder network during the training phase, masking was undertaken on the output vectors to obtain shape (batch size, 4, 40, 40).

9. Since the aim is to maximise the final portfolio value, the Sharpe ratio, ϖ, which served as an instantaneous reward at each time step, was calculated using equation 6.9. This was then used to calculate our proposed Discounted cumulative reward ($R_{(d)}$) as shown in equation (6.11).

10. training the policy network using Adams optimizer to maximise the $R_{(d)}$
11. The routing algorithm in the CapsNet was used to compute a set of attention coefficients to determine which prediction vectors can combine to form the output of the network.
12. The attention coefficients were computed using the agreement between the prediction vectors and the output of the previous layer using the equation (6.2).
13. Softmax was used in the last hidden layers to obtain the weighted voted scores for the non-cash assets used in the portfolio construction process. .

The next section covers the experiments carried out in this work and includes a description of the data used, the performance metrics, the results obtained, and the discussions.

6.6 Experiments

The setup of our experiments is described in this section. It also includes: a description of the features of the datasets used, metrics used to evaluate individual models, the results obtained and the discussions.

Data Description

To construct a reasonably diversified portfolio and maximise portfolio risk-adjusted returns, 7 different assets from different asset classes such as equities, fixed income, currencies, and commodities were obtained through the Yahoo Finance API interface and used as data sets for this project. These were Google, Unilever, Amazon, Gold, crude oil, US dollar/GBP exchange rate, and HICOX, also known as Colorado Bond shares; a tax-exempt fund. The total trading period used in the experiment started from 30th June 2005 to 28 August 2019. The sequential data set was then divided into a 60% training set, a 20% validation set, and a 20% test set. The Test ID for each of the periods used for the training data set, the back-test data sets, and the validation sets are shown in Tables 6.1 and 6.2, respectively.

Test ID	**Training Set(60%)**	**Test Set(20%)**	
		All	**100 Trading days**
Test 1	30-06-2005 to 14-09-2006	10-02-2007 to 09-07-2007	10-02-2007 to 09-07-2007
Test 2	30-06-2005 to 29-11-2007	23-09-2008 to 14-07-2009	23-09-2008 to 17-02-2009
Test 3	30-06-2005 to 17-02-2009	06-05-2010 to 22-07-2011	06-05-2010 to 28-09-2010
Test 4	30-06-2005 to 05-05-2010	15-12-2011 to 02-08-2013	15-12-2011 to 14-05-2012
Test 5	30-06-2005 to 22-07-2011	03-08-2013 to 10-08-2015	03-08-2013 to 27-12-2013
Test 6	30-06-2005 to 09-10-2012	17-03-2015 to 15-08-2017	17-03-2015 to 10-08-2015
Test 7	30-06-2005 to 27-012-2013	25-10-2016 to 22-08-2019	25-10-2016 to 21-03-2017

Table 6.1: Training and Test sets Trading periods

Test ID	Validation Set(20%)
Test 1	15-09-2006 to 09-02-2007
Test 2	30-11-2007 to 22-09-2008
Test 3	18-02-2009 to 05-05-2010
Test 4	06-05-2010 to 14-12-2011
Test 5	23-07-2011 to 02-08-2013
Test 6	10-10-2012 to 16-03-2015
Test 7	28-012-2013 to 24-10-2016

Table 6.2: Validation sets trading periods

The box-plot representation of the closing prices of the 7 assets used in the portfolio construction is represented in figure 6.6:

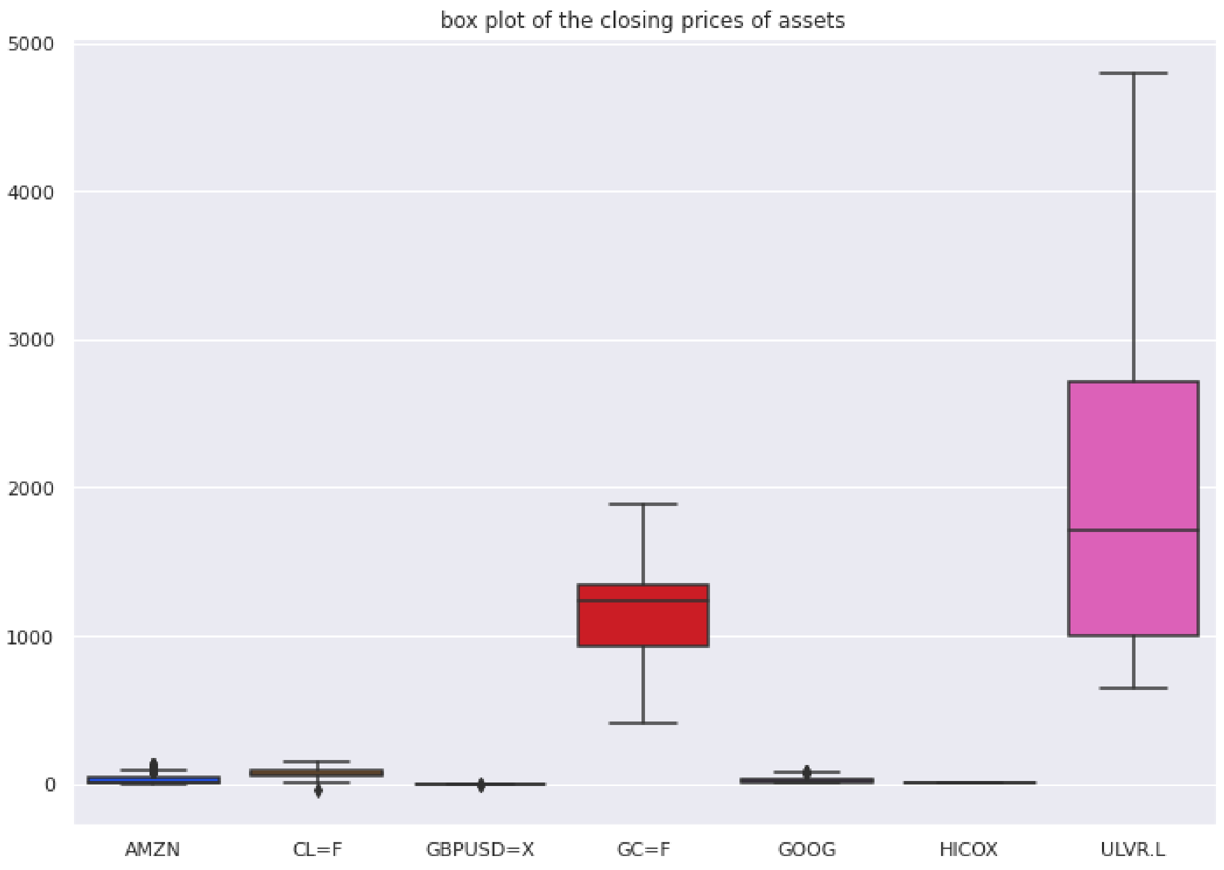


Figure 6.6: Box-plot of the closing prices of the 7 assets used

Hyper-Parameters

The configuration of hyper-parameters for the Reinforcement Learning (RL) Agent is documented in Table 6.3. These parameters govern critical aspects of portfolio construction and optimisation, such as portfolio size, kernel size, batch size, regularisation, learning rate, optimisation type, trading cost, interest rate, initial cash bias, window length, sample bias, and number of episodes. The optimal number of episodes, which is 3 was chosen after a series of experiments were carried out to the evaluate and monitor the final portfolio value for each of the chosen number of episodes. These hyper-parameters collectively shape the behaviour of the RL algorithm, influencing its learning dynamics and decision-making processes during the portfolio optimisation process.

Hyper-parameters	Description	size
portfolio size, K	Number of assets in portfolio	7
kernel size	size of kernel used	(1,3)
batch size	size of mini-batch used	60
ϖ	weight selection of mini-batch from M_C	0.8
Regularisation	L2 regularisation applied	1×10^-8
Learning Rate	Used for Adam Optimisation	1×10^-8
Trading cost	total cost to trade	0.1%
Interest Rate	The interest rate used	0.02/250
Initial Cash Bias	initial cash	0.5
Window Length	Number of trading periods	10
Sample Bias	Beta used for geometric distribution	5×10^-5
Number of episodes	Number of interactions between an agent and the environment from initial state to final state	3

Table 6.3: Hyper-parameters used for the CapsNet RL Network

Performance Evaluation Metrics

The metrics used in this study to evaluate the performance of the models were categorised into three broad groups that are: Reward-only metrics, risk-only metrics, and risk-adjusted reward metrics. The reward-only metrics used are Mean Portfolio value(MPV), Final Portfolio Value (P_f), % Change in Mean Portfolio value (%ΔM) and % Change in Final Portfolio value (%ΔF). These are described below:

- **Mean Portfolio value(MPV):** This measures the average value of the portfolio over the investment time period. During the training of the RL algorithm, the value of the portfolio takes on new values at each trading time step, where it can increase or decrease in value. The average portfolio value is measured using MPV.
- **Final Portfolio Value (P_f):** This is the Accumulated final value of the portfolio at the end of the training period, t.
- **% Change in Mean Portfolio value (%ΔM):** This is the percentage change in the Mean portfolio value. It is equal to the average returns produced by the portfolio at the end of the trading period. It is calculated as follows:

$$\%\Delta M = \frac{MPV_t - P_0}{P_0}$$

  Where: MPV_t is the Mean portfolio value at time t and P_0, is the initial portfolio value, which in this work represents the initial amount invested.

- % **Change in Final Portfolio value (%ΔF):** This is the percentage change in the final portfolio value. It is equal to the returns yielded by the portfolio at the end of the trading period. It is calculated as follows:

$$\%\Delta F = \frac{P_f - P_0}{P_0}$$

Where: P_f is the final portfolio value at time t and P_0, is the initial portfolio value, which in this work represents the initial amount invested.

The Risk-only metrics used are Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR). For both VaR and CVaR, the smaller the value, the better it is as a performance evaluation metric. Figure 6.7 shows the diagrammatic demonstration of VaR and CVaR.

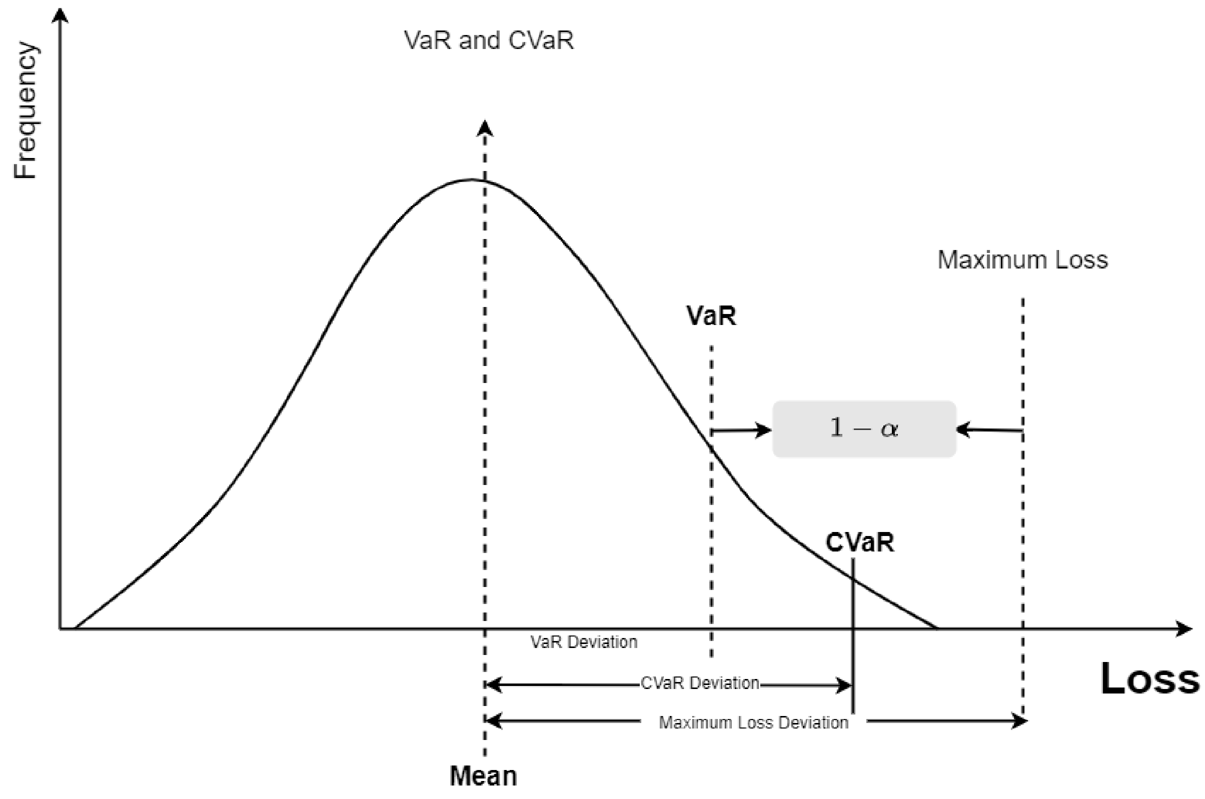


Figure 6.7: representation of VaR and CVaR

- Value-at-Risk (VaR): It is a percentile measure of risk in portfolios held by financial institutions that are continuously exposed to credit, market, and operational risk. Although the deployment of hedging strategies can mitigate this risk, the first stage in managing any risk is by measuring it. As opined by Banihashemi and Navidi (2017), VaR is an important risk measure that concentrates on adverse events and their probability of occurrence. It is a statistical technique defined as the maximum amount of investment that may be lost in a specified time frame. It is used in financial regulations such as Basel I and Basel II to measure the width of the daily loss distribution of a portfolio (Sarykalin et al., 2008). Given a random variable loss value of X with a cumulative distribution function.

$$F_X(z) = p\{X \leq z\}$$

The VaR of X with a confidence interval $\alpha \in [0, 1]$ is shown in Equation 6.29

$$VaR_\alpha(X) = min\{z \,|\, F_X(z) \geq \alpha\} \tag{6.29}$$

For normally distributed loss variables, VaR is proportional to the standard deviation.

- Conditional Value-at-Risk (VaR): Conditional Value-at-Risk (CVaR), introduced by Rockafellar and Uryasev (2000) is the expected shortfall or mean expected loss if the VaR is exceeded and is comparatively an effective financial risk management tool used for the percentile measure of risk. As stated by (Sarykalin et al., 2008), CVaR represents the mean of the percentage of worst-case loss scenarios and is similar to the value-at-risk (VaR) risk measure, which is a percentile of a loss distribution. It is obtained by finding the weighted average of the extreme losses in the tail of the returns' distribution past the VaR cut-off point as shown in Figure 6.7.

  For random variables with continuous distribution functions, $CVaR_\alpha(X)$ is equal to conditional expectation X subject to $X \geq VaR_\alpha(X)$. The CVaR of X with a confidence interval $\alpha \in [0, 1]$ is represented in equation 6.30

  $$CVaR_\alpha(X) = \int_{-\infty}^{\infty} z \, d\, F_X^\alpha(z) \tag{6.30}$$

  Where:

  $$F_X^\alpha(z) = \begin{cases} 0, & \text{when } z < VaR_\alpha(X), \\ \\ \frac{F_X(Z)-\alpha}{1-\alpha}, & \text{when } z \geq VaR_\alpha(X) \end{cases}$$

The risk-adjusted Reward metrics used are Sharpe Ratio (SR), Final Reward-to-VAR Ratio using % Change in Final Portfolio value%ΔF and Mean Reward-to-CVAR Ratio using % Change in Mean Portfolio value%ΔM.

- **Sharpe Ratio (SR):** It is a metric used to measure the risk-adjusted return on a portfolio and was proposed by Sharpe (1964, 1994). The Sharpe ratio is used to evaluate the overall performance of an investment portfolio or the performance of an individual asset. It shows how well an equity investment performs compared to the rate of return on a risk-free investment, such as government treasury bonds or bills. It is the risk-adjusted average return, defined as the average of the risk-free return divided by its standard deviation, and is mathematically represented in Equation 6.9. From 2005 to 2019, the average risk-free rate was derived from the Bank of England Sterling Overnight Inter-bank Average Rate (SONIA). It is the effective overnight interest rate paid by banks for unsecured transactions in the British sterling market. It is used for overnight funding for trades that occur in off-hours and represents the depth of overnight business in the marketplace.

- **Final Reward-to-VAR Ratio using % Change in Final Portfolio value %ΔF:** This is related to the reward-to-Var ratio proposed by Alexander and Baptista (2003b). It measures the ratio of the % change in the final portfolio value to the VAR. If risk-free assets were available for portfolio construction, then it can also be defined as an additional % change in the final portfolio value that would have been earned if an additional percentage point of VAR had been taken by reinvesting the amount invested in risk-free assets in the selected risky portfolio. Under the normality assumption, the ranking of portfolio performance gives the same ranking as the ranking under the Sharpe ratio. If this ranking becomes different, then the issue of non-normality must be further investigated (Alexander and Baptista, 2003b).

- **Mean Reward-to-CVAR Ratio using % Change in the Mean Portfolio value%ΔM:** The use of this is motivated by the mean-CVaR approach used by Banihashemi and Navidi (2017) in their work. It involves finding the ratio of the % Change in Mean Portfolio value to the CVaR.

The experimental setup and detailed evaluation metrics enable us to analyse the performance of each of the models effectively. In the following sections, the experimental results and insights derived from the evaluation of our proposed models using the aforementioned metrics are presented.

Results and Discussion

Evolution of Portfolio weights

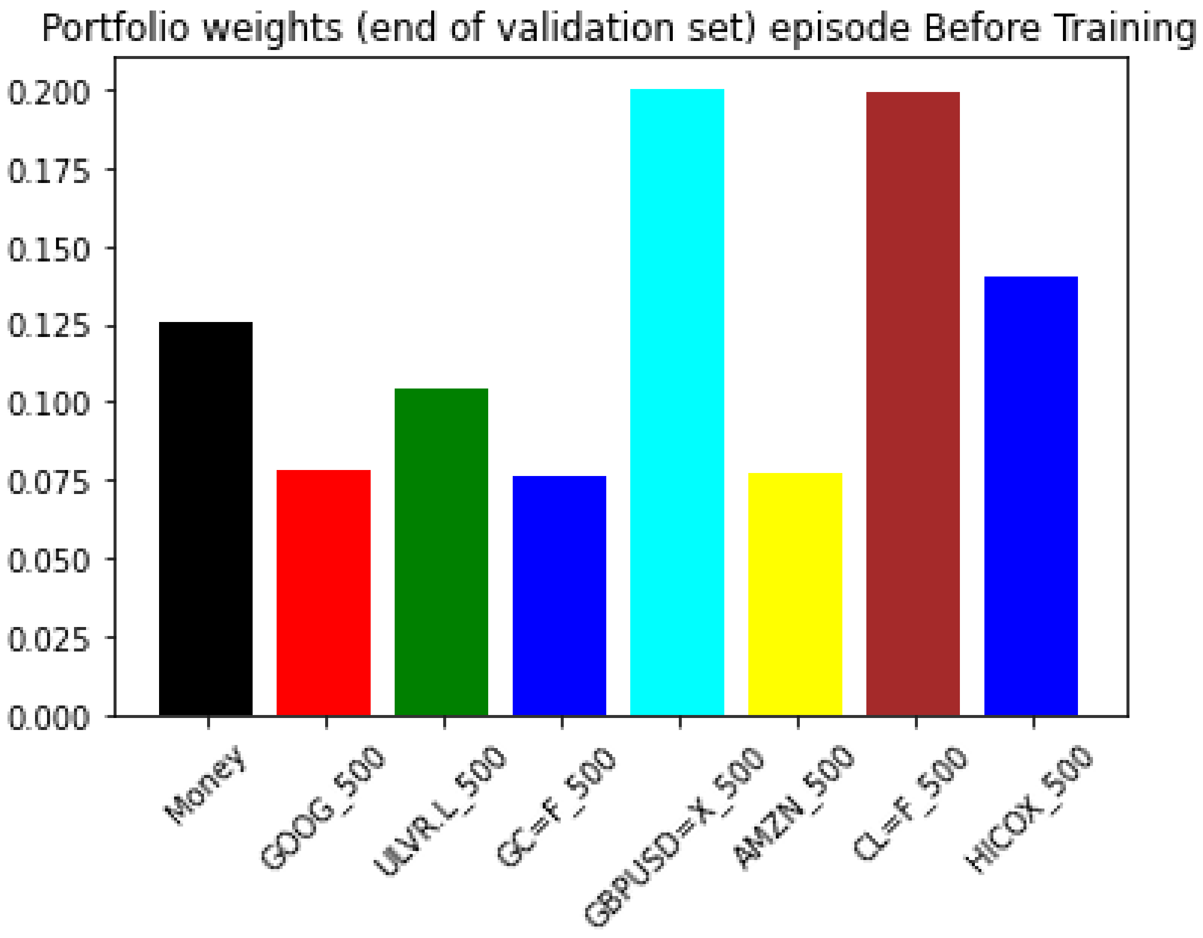


Figure 6.8: weight evolution at beginning of episode 1

The portfolio weights undergo dynamic changes during the training process, influencing the asset allocation strategy. Figure 6.8 represents the weight at the beginning of training episode 1 using the test ID: Test 1, while 6.9 and 6.10 respectively, represent the evolution, the weights go through at the beginning of training episode 2 and the end of training episode 3 using the Test ID: Test 1 set.

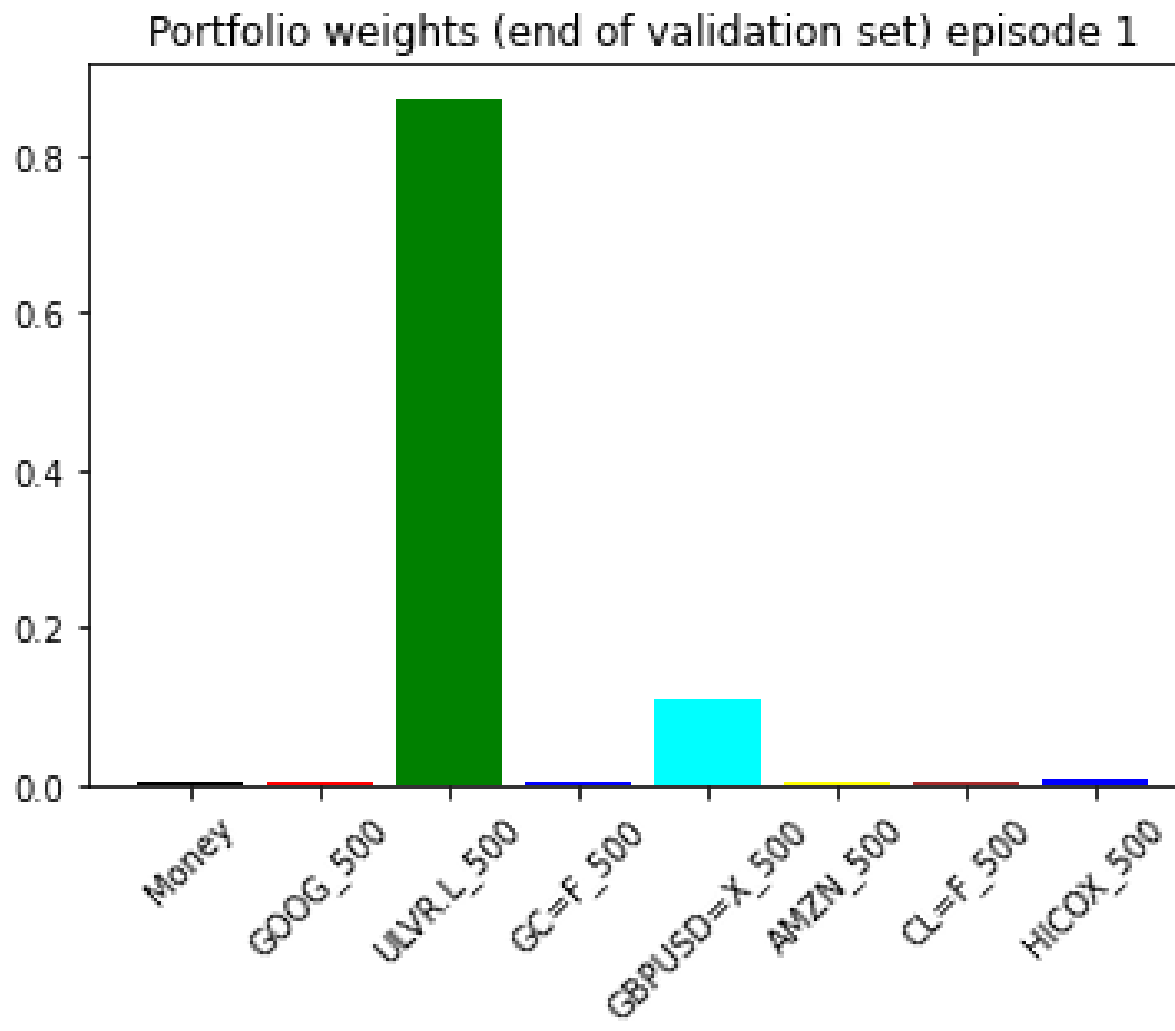


Figure 6.9: weight evolution at beginning of episode 2

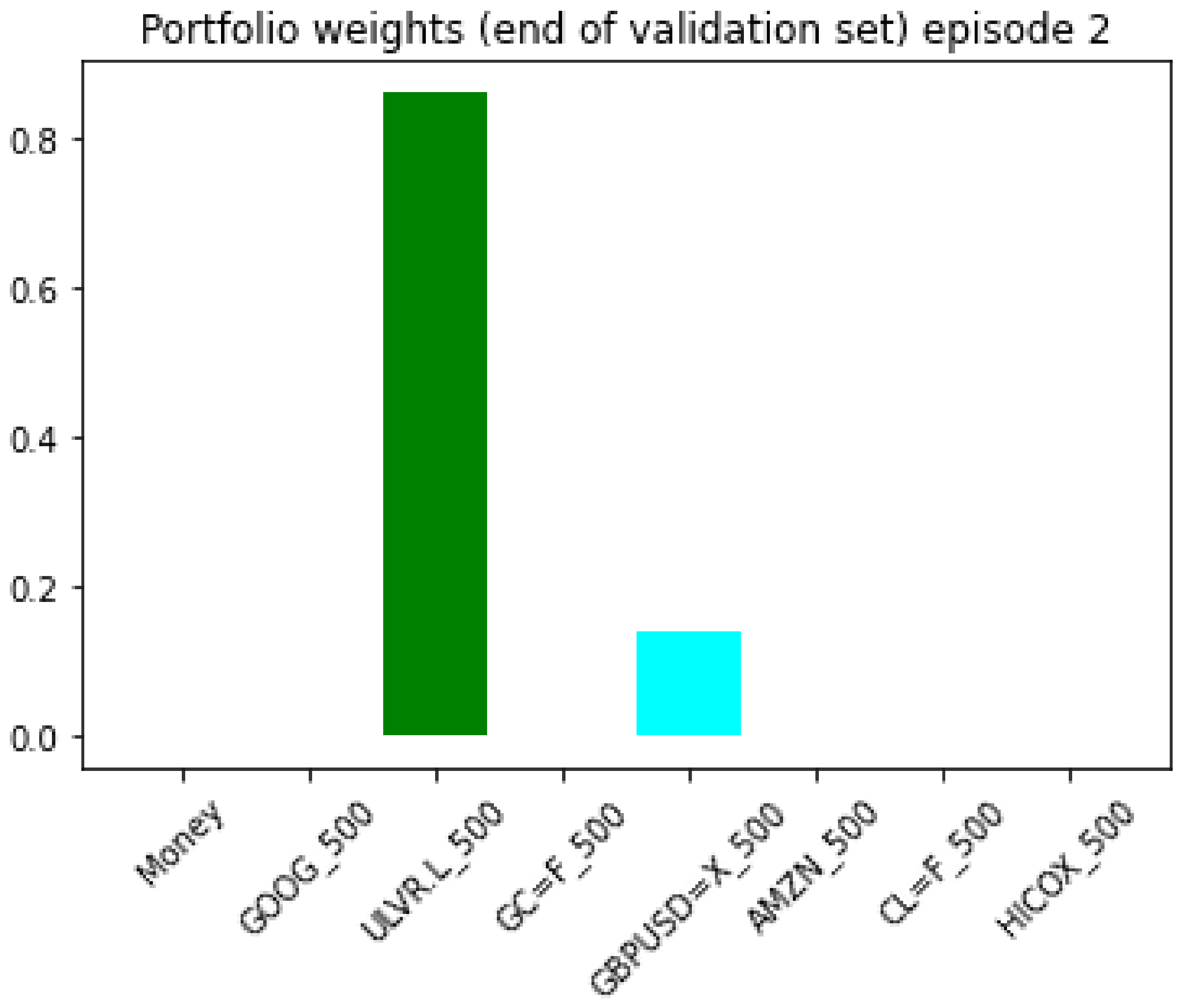


Figure 6.10: final weight evolution at the end of episode 3 training

The results of the experiments: This has been categorised into reward-only metrics, risk-only metrics, and risk-adjusted reward metrics. Tables 6.4, 6.5, 6.6, 6.7, 6.8, 6.9 and 6.10 respectively represent the results for the reward-only metrics for Test IDs; Test 1, Test 2, Test 3, Test 4, Test 5, Test 6 and Test 7.

Tables 6.11, 6.12, 6.13, 6.14, 6.15, 6.16, and 6.17 respectively, show the risk-adjusted reward metrics for the following Test IDs: Test 1, Test 2, Test 3, Test 4, Test 5, Test 6, and Test 7.

Finally, the results for the risk-only metrics for Test IDs: Test 1, Test 2, Test 3, Test 4, Test 5, Test 6, and Test 7 are represented in Tables 6.22, 6.23, 6.24, 6.25, 6.26, 6.27, and 6.28.

It can be seen from the results that our proposed approaches achieved superior performance when the average risk-adjusted reward metrics were used. This was expected since our approach sought to maximise the risk-adjusted returns used as its objective function.

Rewards-only Metrics

Figure 6.11 graphically represents the final portfolio value for each test ID used. It can be observed that our proposed approach achieved a superior Final Portfolio Value across the majority of the Test IDs used.

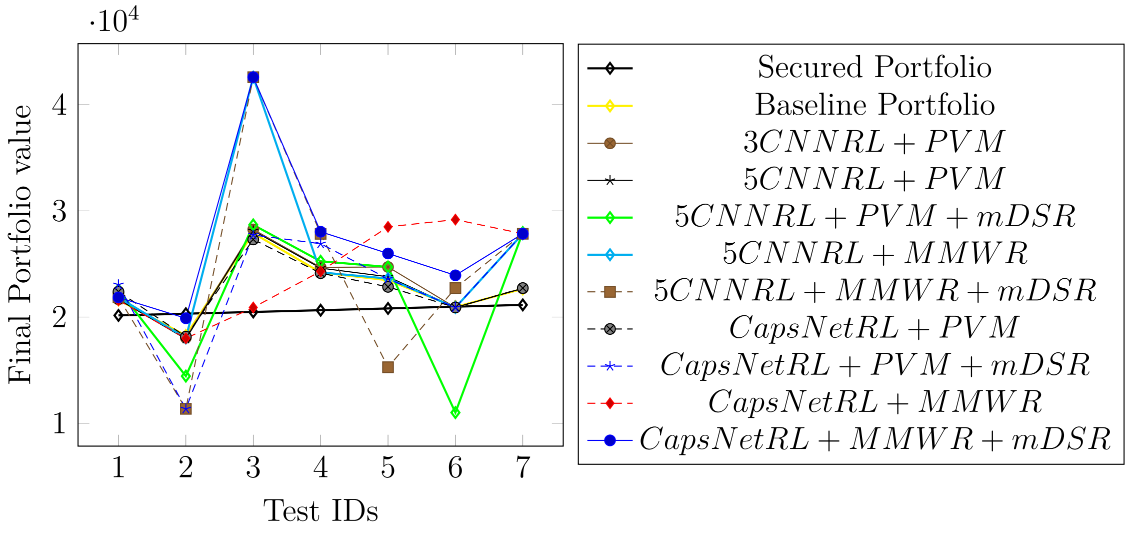


Figure 6.11: Comparison of the model performance on the Final portfolio value (P_f) for each Test ID used

It can be observed from Table 6.4 that for the test 1 set, the application of CapsNet with the Portfolio vector single memory and our Markov cumulative Differential Sharpe Ratio (CapsNetRL + PVM +mDSR) achieved the best overall results in all the rewards-only metrics used. It achieved a Mean Portfolio value of £23,126 which represented a gain of 15.63% relative to the initial amount of £20,000 initial investment made. The final portfolio value gained for the same model was £23,049 with a percentage increase of 15.24% on the initial invested amount of £20,000. The secured portfolio had the least performance in all the reward-only metrics used, with an MPV of £20077, an FPV of £20154, a percentage increase in the mean portfolio value (%ΔM) of 0.39% and a percentage increase in the final portfolio value (%ΔF) of 0.77%. The metrics of the baseline model were; an MPV of £21008, P_f of £21609, a percentage increase in changes in the mean portfolio value (%ΔM) of 5% and a percentage increase in changes in the final portfolio value (%ΔF) of 8%.

Also, for Test ID 2 shown in table 6.5, which corresponded with the global recession of the 2008-2009 subprime market, had a decline in all the metrics for all models used except

Test ID	Model	Metric			
		MPV	P_f	%ΔM	%ΔF
	Secured port	20077	20154	0.00385	0.0077
	Baseline Portfolio	21008	21609	0.050	0.0805
	3CNNRL + PVM	21030	21664	0.052	0.0832
	5CNNRL + PVM	21027	21660	0.051	0.083
	5CNNRL + PVM + mDSR	20795	22564	0.040	0.1282
Test 1	5CNNRL + MMWR	21138	21848	0.057	0.0924
	5CNNRL + MMWR+ mDSR	21138	21848	0.057	0.0924
	CapsNetRL + PVM	20743	22396	0.037	0.1198
	CapsNetRL + PVM + mDSR	**23126**	**23049**	**0.156**	**0.1525**
	CapsNetRL + MMWR	20973	21573	0.049	0.0787
	CapsNetRL + MMWR+ mDSR	21138	21848	0.057	0.0924

Table 6.4: Performance evaluation of all models using Test ID of test 1 data for portfolio Reward-only metrics

the secured portfolio model because the amount invested was not exposed to the harsh market conditions at the time. For all the remaining 10 models that were exposed to the market, our proposed Markov Differential Sharpe Ratio on the CapsNetRL model and the Multi-Memory Weight Reservoir ($5CNNRL + DMWR + mDSR$) achieved the best overall results in all metrics used. The application of CapsNet with the Multi-Memory Weight Reservoir and our Markov Differential Sharpe Ratio (CapsNetRL + MMWR + mDSR) achieved the best MPV and %ΔM of £20,909 and 0.045 respectively. Apart from the secured portfolio model, the final portfolio value gained for the CapsNetRL + MMWR + mDSR model was £19,882, which was the highest gained for all other models and was 75.15% in excess of the comparative 5CNNRL + MMWR+ mDSR model. It was also 8.83% larger than the FPV of the baseline model. Likewise, It can be observed from the test 3 set shown in table 6.6 that; the application of either the CapsNet or 5CNNRL with the Multi-Memory Weight Reservoir and our Markov Differential Sharpe Ratio (CapsNetRL + MMWR +mDSR or 5CNNRL + MMWR +mDSR) both achieved the highest MPV, P_f, %ΔM and %ΔF with values £33,004, £42,601, 65% and 113.01% respectively. This corresponded to an increase of 31 % and 52.70% above the MPV and P_f for the baseline model, respectively. It can be observed from the test 4 set shown in table 6.7 that; the application of our proposed CapsNetRL + MMWR + mDSR achieved an MPV, P_f, %ΔM and %ΔF values of £26,631, £28,037, 33.2% and 40.19% respectively as compared to £22587, £24,201, 12.9% and 21.01% respectively for the Baseline model.

It can be observed from the test 5 set shown in table 6.8 that the Secured Portfolio's MPV of £20,402 and P_f of £20,810 indicate relatively stable performance. This model achieved %Δ M and %Δ F values of 2. 01% and 4. 05%, respectively. It can also be seen that the

Test ID	**Model**	**Metric**			
		MPV	P_f	%ΔM	%ΔF
	Secured port	20158	**20316**	0.0079	**0.0158**
	Baseline Portfolio	19534	18268	-0.023	-0.0866
	3CNNRL + PVM	19436	18044	-0.028	-0.0978
	5CNNRL + PVM	19401	17939	-0.030	-0.1031
	5CNNRL + PVM +mDSR	16925	14434	-0.154	-0.2783
Test 2	5CNNRL + MMWR	19431	18028	-0.028	-0.0986
	5CNNRL + MMWR+ mDSR	15037	11351	-0.248	-0.4325
	CapsNetRL + PVM	19493	18156	-0.025	-0.0922
	CapsNetRL + PVM + mDSR	15037	11351	-0.248	-0.4325
	CapsNetRL + MMWR	19412	17994	-0.029	-0.1003
	CapsNetRL + MMWR+ mDSR	**20909**	19882	**0.045**	-0.0059

Table 6.5: Performance evaluation of all models using test 2 Test ID data using Portfolio Reward-only metrics

Test ID	**Model**	**Metric**			
		MPV	P_f	%ΔM	%ΔF
	Secured port	20239	20479	0.0119	0.02395
	Baseline Portfolio	25191	27899	0.260	0.3950
	3CNNRL + PVM	25410	28252	0.271	0.4126
	5CNNRL + PVM	25333	28134	0.267	0.4067
	5CNNRL + PVM + mDSR	25708	28683	0.285	0.4342
Test 3	5CNNRL + MMWR	33004	42601	0.650	1.1301
	5CNNRL + MMWR+ mDSR	**33004**	**42601**	**0.650**	**1.1301**
	CapsNetRL + PVM	24762	27312	0.238	0.3656
	CapsNetRL + PVM + mDSR	25172	27729	0.259	0.3865
	CapsNetRL + MMWR	21827	20899	0.091	0.0445
	CapsNetRL + MMWR+ mDSR	**33004**	**42601**	**0.650**	**1.1301**

Table 6.6: Performance evaluation of all models using Test 3 Test ID data using portfolio reward-only metrics

Baseline Portfolio model was also able to achieve an MPV of £21,560 and P_f of £23,468, with %Δ M and %Δ F values of 7. 80% and 17. 34% respectively. The 3CNNRL + PVM model also obtained an MPV of £21,930 and P_f of £24,717, with %ΔM and %ΔF values of 9.70% and 23.59% respectively. The 5CNNRL + PVM with an MPV of £21,672 and P_f of £23,795, this model shows %ΔM and %ΔF values of 8.40% and 18.98%. The 5CNNRL + PVM + mDSR achieved an MPV of £21,930 and P_f of £24,717, the values of this model %Δ M and %Δ F are 9. 70% and 23. 59%. The 5CNNRL + MMWR model produced an MPV of £21,603 and a final portfolio value, P_f of £23,625, with %ΔM and %ΔF values of 8.00% and 18.13%. The 5CNNRL + MMWR+ mDSR model This model has an MPV of £18,773 and P_f of £15,270, resulting in %ΔM and %ΔF values of -6.10% and -23.65%.

Test ID	**Model**	**Metric**			
		MPV	P_f	%ΔM	%ΔF
Test 4	Secured port	20320	20644	0.016	0.0322
	Baseline Portfolio	22587	24201	0.129	0.2101
	3CNNRL + PVM	22882	24684	0.144	0.2342
	5CNNRL + PVM	22836	24611	0.142	0.2306
	5CNNRL + PVM +mDSR	22188	25255	0.109	0.2628
	5CNNRL + MMWR	22579	24207	0.129	0.2104
	5CNNRL + MMWR+ mDSR	**28055**	27843	0.403	0.3922
	CapsNetRL + PVM	22501	24135	0.125	0.2068
	CapsNetRL + PVM + mDSR	24815	26908	0.241	0.3454
	CapsNetRL + MMWR	22645	24310	0.132	0.2155
	CapsNetRL + MMWR+ mDSR	26631	**28037**	**0.332**	**0.4019**

Table 6.7: Performance evaluation of all models using Test 4 Test ID data using portfolio reward-only metrics

Likewise, the CapsNetRL + PVM model obtained an MPV of £21,342 and P_f of £22,843, with %ΔM and %ΔF values of 6.70% and 14.22% respectively. The CapsNetRL + PVM + mDSR: With an MPV of £21,594 and P_f of £23,602, this model's %ΔM and %ΔF values are 8.00% and 18.01%. The CapsNetRL + MMWR model was the best performer with an MPV of £23,138 and P_f of £28,499, with corresponding %ΔM and %ΔF values of 15.70% and 42.50% respectively. Our proposed CapsNetRL + MMWR+ mDSR model was also able to achieve an MPV of £22,316 and P_f of £25,989, resulting in %ΔM and %ΔF values of 11.60% and 29.95% respectively. Comparing the models based on their MPV and P_f values, we can see that the CapsNetRL + MMWR model consistently outperforms other models in terms of generating profits. It achieved the highest MPV and P_f values, indicating better portfolio performance and growth. Furthermore, comparing the %ΔM and %ΔF values, the "CapsNetRL + MMWR" model also stands out. It achieves the highest positive percentage changes in both metrics, indicating substantial average returns and cumulative profits over the investment period. It is noteworthy that the 5CNNRL + MMWR+ mDSR model demonstrates negative %ΔM and %ΔF values, implying that it may have experienced losses during the trading period.

It can be observed from the results of test 6 set shown in table 6.9 that; the application of CapsNet with the Multi-Memory Weight Reservoir and our Markov Differential Sharpe Ratio (CapsNetRL + MMWR + mDSR) achieved the best MPV and %ΔM of £20,909 and 0.045 respectively. Apart from the secured portfolio model, the final portfolio value gained for the CapsNetRL + MMWR + mDSR model was £19,882, which was the highest gain for all other models and was 75.15% in excess of the comparative 5CNNRL + MMWR+ mDSR model. It was also 8.83% larger than the FPV of the baseline model.

Test ID	**Model**	**Metric**			
		MPV	P_f	%ΔM	%ΔF
	Secured port	20402	20810	0.0201	0.0405
	Baseline Portfolio	21560	23468	0.078	0.1734
	3CNNRL + PVM	21930	24717	0.097	0.2359
	5CNNRL + PVM	21672	23795	0.084	0.1898
	5CNNRL + PVM +mDSR	21930	24717	0.097	0.2359
Test 5	5CNNRL + MMWR	21603	23625	0.080	0.1813
	5CNNRL + MMWR+ mDSR	18773	15270	-0.061	-0.2365
	CapsNetRL + PVM	21342	22843	0.067	0.1422
	CapsNetRL + PVM + mDSR	21594	23602	0.080	0.1801
	CapsNetRL + MMWR	**23138**	**28499**	**0.157**	**0.4250**
	CapsNetRL + MMWR+ mDSR	22316	25989	0.116	0.2995

Table 6.8: Performance evaluation of all models using Test 5 Test ID data using portfolio reward-only metrics

Test ID	**Model**	**Metric**			
		MPV	P_f	%ΔM	%ΔF
	Secured Port	20484	20977	0.0242	0.04885
	Baseline Portfolio	21725	20945	0.086	0.0473
	3CNNRL + PVM	21772	20898	0.089	0.449
	5CNNRL + PVM	21748	20903	0.087	0.0444
	5CNNRL + PVM +mDSR	22777	10998	0.139	0.1389
Test 6	5CNNRL + MMWR	21973	20898	0.099	0.0447
	5CNNRL + MMWR+ mDSR	22326	22726	0.116	0.1363
	CapsNetRL + PVM	21726	20903	0.086	0.0452
	CapsNetRL + PVM + mDSR	21738	20902	0.087	0.451
	CapsNetRL + MMWR	**26458**	**29182**	**0.323**	**0.4591**
	CapsNetRL + MMWR+ mDSR	23146	23928	0.157	0.1964

Table 6.9: Performance evaluation of all models using Test 6 Test ID data using portfolio Reward-only metrics

From the results of the test set 7 shown in table 6.10, the Secured Portfolio, achieved an MPV of £20,567 and a Final Portfolio Value (P_f) of £21,145. The Baseline Portfolio also achieved an MPV of £20,194 and a Final Portfolio Value (P_f) of £22,653. Also, the MPV of £20,1283 obtained by the CNNRL + PVM model and the final portfolio value (P_f) of £22,696 were lower than the respective value of the baseline model. The percentage change metrics indicate marginal improvements in both MPV and P_f. It can also be seen that the %ΔM and %ΔF for this model were 0.006 and 0.1348 respectively.

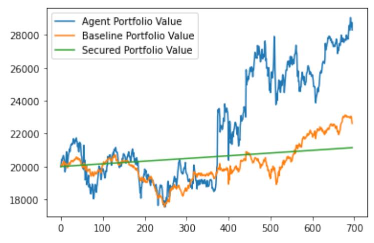
On the other hand, the 5CNNRL + PVM model achieved an MPV of £20,124 and a Final Portfolio Value (P_f) of £22,710. It can also be seen that the %ΔM and %ΔF for this model were 0.006 and 0.1355 respectively. The 5CNNRL + PVM + mDSR model achieved an

MPV of £22,134 and a Final Portfolio Value (P_f) of £27,968. The corresponding %ΔM and %ΔF associated with this model were 0.107 and 0.3984, respectively. The 5CNNRL + MMWR model outperformed other variants, achieving the highest MPV of £22,349 and the highest Final Portfolio Value (P_f) of £27,849. Similar to the previous model, the 5CNNRL + MMWR+ mDSR model achieved the same highest MPV of £22,349 and the highest final portfolio value (P_f) of £27,849. Meanwhile, the corresponding CapsNetRL + PVM model achieved an MPV of £20,116 and a Final Portfolio Value (P_f) of £22,731. Finally, the CapsNetRL + PVM + mDSR, the CapsNetRL + MMWR and CapsNetRL + MMWR + mDSR models achieved the same highest MPV of £22,349 and the highest Final Portfolio Value (P_f) of £27,849 as the 5CNNRL + MMWR variants. This suggests that the CapsNet architecture, when combined with mDSR, can achieve competitive returns.

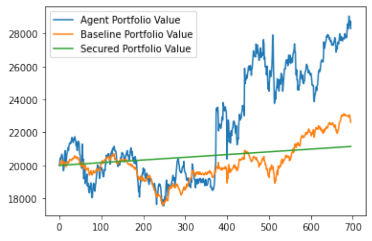
Test ID	**Model**	**Metric**			
		MPV	P_f	%ΔM	%ΔF
	Secured Portfolio	20567	21145	0.02835	0.05725
	Baseline Portfolio	20194	22653	0.010	0.1327
	3CNNRL + PVM	20128	22696	0.006	0.1348
	5CNNRL + PVM	20124	22710	0.006	0.1355
	5CNNRL + PVM +mDSR	22134	27968	0.107	0.3984
Test 7	5CNNRL + MMWR	**22349**	**27849**	**0.117**	**0.3925**
	5CNNRL + MMWR+ mDSR	**22349**	**27849**	**0.117**	**0.3925**
	CapsNetRL + PVM	20116	22731	0.006	0.1366
	CapsNetRL + PVM + mDSR	**22349**	**27849**	**0.117**	**0.3925**
	CapsNetRL + MMWR	**22349**	**27849**	**0.117**	**0.3925**
	CapsNetRL + MMWR+ mDSR	**22349**	**27849**	**0.117**	**0.3925**

Table 6.10: Performance evaluation of all models using Test 7 Test ID data using portfolio reward-only metrics

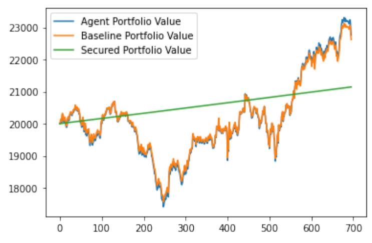
The results of the test ID: Test set 7 discussed above is pictorially represented in Figure 6.12 to highlight and compare the performance of the models with the highest MPV and the highest P_f together with that of the Secured and Baseline portfolio models. These models are CapsNetRL + MMWR + mDSR and CapsNetRL + MMWR versus 5CNNRL + MMWR + mDSR and 5CNNRL + MMWR. The results of the performance on CapsNet + PVM +m mDSR versus 5CNNRL + PVM + mDSR were also added to the plots. It can be seen that the application of the Markov Differential Sharpe Ratio and the multi-memory weight reservoir contributed to the increase in the performance of both the CNN-based and CapsNet-based models. These enhancements contribute to higher returns and improved reward-only performance.



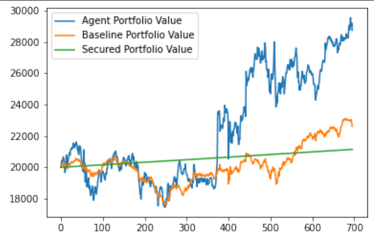
(a) comparison of the performance of CapsNet+MMWR+mDSR with secured and Baseline Portfolios



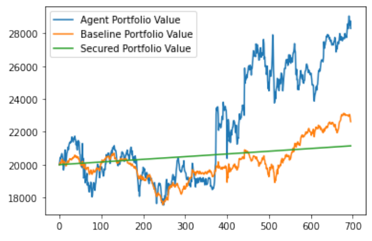
(b) comparison of performance of CapsNet+MMWR model with secured and Baseline portfolios



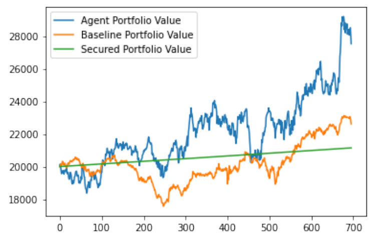
(c) comparison of performance of CapsNet+PVM+mDSR model with secured and Baseline portfolios



(d) comparison of the performance of 5CNNRL+MMWR+mDSR with secured and Baseline portfolios



(e) Comparison of performance of 5CNNRL+MMWR model with secured and Baseline portfolios



(f) Comparison of performance of 5CNNRL+PVM+mDSR model with secured and Baseline portfolios

Figure 6.12: Comparison of the performance of our proposed model using Reward only the Test ID: Test Set 7. The top row models are the CapsNet-based RL model and the bottom roles are the CNN-based models

Analysis of the effectiveness of Models on MPV: For the Test 1 set in Table 6.4, the mDSR used in the 5CNNRL + PVM + mDSR model did not outperform the baseline reward function used in the 5CNNRL + PVM model, since the total MPV obtained was £20,795 compared to the baseline line reward, which was able to obtain an MPV of £21,027 while there was no difference in the MPV value when the 5CNNRL + MMWR and 5CNNRL + MMWR+ mDSR models were compared with each other. Both achieved an MPV of £21,138. However, there was a significant increase in MPV when the mDSR used in the CapsNetRL + PVM + mDSR model was compared with the baseline reward model used in the CapsNetRL + PVM model. The CapsNetRL + PVM + mDSR model obtained an MPV of £23,126, which is 11. 49% higher than that of the CapsNetRL + PVM model, which had an MPV of £20,743. Again, MPV increased when the mDSR used in the CapsNetRL + MMWR + mDSR model was compared with the baseline reward model used in the CapsNetRL + MMWR model. The CapsNetRL + MMWR + mDSR model obtained an MPV of £21,138, which is 0. 78% higher than that of the CapsNetRL + MMWR model, which had an MPV of £20,973.

For the Test 2 set in Table 6.5, the mDSR used in the models 5CNNRL + PVM + mDSR, 5CNNRL + MMWR+ mDSR and 5CNNRL + PVM+ mDSR unperformed all the baseline reward functions used in model 5CNNRL + PVM, 5CNNRL + MMWR, CapsNetRL + PVM, respectively, except the model, CapsNetRL + MMWR + mDSR that obtained an MPV of £20,909 which was 7.7 % bigger than that of the corresponding model, CapsNetRL + MMWR 19,412.

For Test 3 set in Table 6.6, the mDSR used in model 5CNNRL + PVM + mDSR outperformed the baseline reward function used in model 5CNNRL + PVM, obtaining a total MPV of £25,708 compared to the baseline reward which had an MPV of £25,333. There was also no difference in the MPV value when the models 5CNNRL + MMWR and 5CNNRL + MMWR+ mDSR were compared. Both achieved an MPV of £33,004. There was, however, an increase in MPV when the mDSR used in the CapsNetRL + PVM + mDSR model was compared with the baseline reward model used in the CapsNetRL + PVM model. The CapsNetRL + PVM + mDSR model obtained an MPV of £25,172 which is 1.66% higher than that of the CapsNetRL + PVM model, which had an MPV of £24,762. Again, the MPV increased when the mDSR used in the CapsNetRL + MMWR + mDSR model was compared with the baseline reward model used in the CapsNetRL + MMWR model. The CapsNetRL + MMWR + mDSR model obtained an MPV of £33,004 which is 51. 21% higher than that of the CapsNetRL + MMWR model, which had an MPV of £21,827.

For Test 4 set in Table 6.7, the mDSR used in model 5CNNRL + PVM + mDSR failed to outperform the baseline reward function used in model 5CNNRL + PVM since the total MPV obtained was £22,188 compared to the baseline line reward which was able to obtain an MPV of £22,836. There was an increase in the value of MPV when models 5CNNRL + MMWR and 5CNNRL + MMWR+ mDSR were compared; with the model 5CNNRL + MMWR+ mDSR achieving an MPV of £28,055 while that of 5CNNRL + MMWR achieved an MPV of £22,579. There was also an increase in MPV when the $mDSR$ used in the CapsNetRL + PVM + mDSR model was compared with the baseline reward model, CapsNetRL + PVM. The CapsNetRL + PVM + mDSR model obtained an MPV of £24,815, which is 10. 28% in excess of that of the CapsNetRL + PVM model, which had an MPV of £22,501. Finally, MPV improved when the mDSR used in the CapsNetRL + MMWR + mDSR model was compared with the baseline reward model used in the CapsNetRL + MMWR model. The CapsNetRL + MMWR + mDSR model obtained an MPV of £26,631, which is 17. 60% greater than that of the CapsNetRL + MMWR model, which had an MPV of £22,645.

For the Test 5 set in Table 6.8, the mDSR used in model 5CNNRL + PVM + mDSR outperformed the baseline reward function used in model 5CNNRL + PVM, obtaining a total MPV of £21,930 compared to the baseline reward which had an MPV of £21,672. The mDSR in the 5CNNRL + MMWR + mDSR under-performed the 5CNNRL + MMWR by 15.07% with a decrease in MPV of £18,773. There was, however, a slight increase in the MPV when the mDSR used in the CapsNetRL + PVM + mDSR model was compared with the baseline reward model used in the CapsNetRL + PVM model. The CapsNetRL + PVM + mDSR model obtained an MPV of £21,594, which is 1. 18% higher than that of the CapsNetRL + PVM model, which had an MPV of £21,342. Finally, the MPV for the CapsNetRL + MMWR + mDSR model underperformed by 3. 6 % compared to the baseline reward model used in the CapsNetRL + MMWR model, which had an MPV of £23,138.

For Test 6 set in Table 6.9, the mDSR used in model 5CNNRL + PVM +mDSR outperformed the baseline reward function used in model 5CNNRL + PVM, obtaining a total MPV of £22,777 compared to the baseline reward, which had an MPV of £21,748. The mDSR in the 5CNNRL + MMWR+ mDSR again outperformed the 5CNNRL + MMWR by 1.61% with an MPV of £22,326. There was also a slight increase in MPV when the mDSR used in the CapsNetRL + PVM + mDSR model was compared with the baseline reward model used in the CapsNetRL + PVM model. The CapsNetRL + PVM + mDSR model obtained an MPV of £21,738, which is 0. 055% higher than that of the CapsNetRL + PVM model, which had an MPV of £21,726. Finally, the MPV for the CapsNetRL + MMWR + mDSR model underperformed by 14. 31 % compared to the baseline reward model used in model $CapsNetRL + MMWR$, which had an MPV of £26,458.

Finally, for Test 7 set in Table 6.10, the mDSR used in model 5CNNRL + PVM +mDSR outperformed the baseline reward function used in model 5CNNRL + PVM obtaining a total MPV of £22,134 compared to the baseline reward which had an MPV of £20,124. There was also no difference in the value of MPV when models 5CNNRL + MMWR and 5CNNRL + MMWR+ mDSR were compared. Both achieved an MPV of £22,349. There was, however, an increase in MPV when the mDSR used in the CapsNetRL + PVM + mDSR model was compared with the baseline reward model used in the CapsNetRL + PVM model. The CapsNetRL + PVM + mDSR model obtained an MPV of £22,349 which is 11.10% in excess of that of the CapsNetRL + PVM model, which had an MPV of £20,116. There was no difference in the MPV used with the CapsNetRL + MMWR + mDSR model and the baseline reward model CapsNetRL + MMWR, both of them achieved an MPV of £22,349.

Risk-adjusted rewards Metrics

As can be seen in Table 6.11 for the Test 1 set, the CapsNetRL + PVM + mDSR model had the highest Sharpe ratio of 2.41, which is a remarkable improvement compared to that of the baseline portfolio of 1.806 and is 38. 510% in excess of the Sharpe ratio obtained from the 3CNNRL + PVM model used by Jiang et al. (2017). The CapsNetRL + MMWR model also had the largest Mean Reward-to-CVAR Ratio (ΔM-CVaR) of 8.5 which is 232% and 106. 31% higher than the corresponding metric from the 5CNNRL + MMWR and 3CNNRL + PVM models, respectively.

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 1	Baseline Portfolio	1.806	**8.74**	4.20
	3CNNRL + PVM	1.740	8.490	4.12
	5CNNRL + PVM	1.742	8.47	4.11
	5CNNRL + PVM + mDSR	0.932	6.46	1.59
	5CNNRL + MMWR	1.67	5.25	2.56
	5CNNRL + MMWR+ mDSR	1.67	5.25	2.56
	CapsNetRL + PVM	0.92	6.34	1.55
	CapsNetRL + PVM + mDSR	**2.41**	5.33	4.32
	CapsNetRL + MMWR	1.74	4.14	**8.50**
	CapsNetRL + MMWR + mDSR	1.67	2.56	5.25

Table 6.11: Performance evaluation using risk-adjusted rewards Metrics Test ID; Test 1

For Test ID 2 in Table 6.12, the proposed CapsNetRL + MMWR + mDSR achieved the highest Sharpe ratio of 0.650, ΔF-VaR ratio of -0.22 and highest ΔM-CVaR ratio of 1.34, outperforming other models. The corresponding 5CNNRL + MMWR +mDSR had a Sharpe ratio of -2.73, ΔF-VaR ratio of -10.95 and ΔM-CVaR ratio of -5. Test ID: Test 3 shown in Table 6.13 had the 5CNNRL + PVM + mDSR model with the highest Sharpe ratio of 2.00. This model excelled in risk-adjusted returns, showcasing its adaptability to dynamic market conditions. The application of the Markov Differential Sharpe Ratio (mDSR) significantly improved the performance when applied to the 5CNNRL + PVM model, as seen in the higher Sharpe ratio and the higher Δ M-CVaR ratio of 15.68. No other model was able to surpass the Δ F-VaR ratio of 30.38 achieved by the baseline model. As can be seen in Table 6.14 for Test ID 4, the CapsNetRL + MMWR model achieved the highest Sharpe ratio of 1.630, indicating that MMWR integration enhances risk-adjusted returns. The CapsNetRL + MMWR+ mDSR model also exhibited the highest ΔM-CVaR ratio of 10.51, highlighting its effectiveness in providing higher returns relative to CVaR. For Test ID 5 in Table 6.15, the CapsNet + MMWR model achieved the highest ΔF-VaR ratio of 21.25 and highest ΔM-CVaR ratio of 6.21, signifying its strong risk-adjusted performance.

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 2	Baseline Portfolio	-0.72	-5.97	-1.28
	3CNNRL + PVM	-0.79	-6.27	-1.43
	5CNNRL + PVM	-0.81	-6.28	-1.45
	5CNNRL + PVM + mDSR	-2.10	-5.29	-7.73
	5CNNRL + MMWR	-0.79	-6.24	-1.46
	5CNNRL + MMWR+ mDSR	-2.73	-10.95	-5.00
	CapsNetRL + PVM	-0.730	-5.99	-1.31
	CapsNetRL + PVM + mDSR	-2.374	-10.95	-5.39
	CapsNetRL + MMWR	-0.809	-6.27	-1.46
	CapsNetRL + MMWR + mDSR	**0.650**	**-0.22**	**1.34**

Table 6.12: Performance evaluation using risk-adjusted rewards Metrics Test ID; Test 2

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 3	Baseline Portfolio	1.91	**30.38**	15.54
	3CNNRL + PVM	1.891	29.68	15.46
	5CNNRL + PVM	1.893	29.69	15.48
	5CNNRL + PVM + mDSR	**2.00**	30.15	**15.68**
	5CNNRL + MMWR	1.610	23.79	10.85
	5CNNRL + MMWR+ mDSR	1.610	23.79	10.85
	CapsNetRL + PVM	1.88	29.48	15.24
	CapsNetRL + PVM + mDSR	1.9	28.63	15.18
	CapsNetRL + MMWR	1.80	2.99	4.87
	CapsNetRL + MMWR + mDSR	1.61	23.79	10.85

Table 6.13: Performance evaluation using Risk-adjusted Rewards Metrics Test ID; Test 3

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 4	Baseline Portfolio	1.629	16.80	8.19
	3CNNRL + PVM	1.601	16.26	7.92
	5CNNRL + PVM	1.602	**17.21**	8.39
	5CNNRL + PVM + mDSR	1.025	6.49	4.79
	5CNNRL + MMWR	1.606	7.96	2.62
	5CNNRL + MMWR+ mDSR	1.520	16.47	8.01
	CapsNetRL + PVM	1.578	16.41	7.87
	CapsNetRL + PVM + mDSR	1.500	15.92	8.82
	CapsNetRL + MMWR	**1.630**	16.58	8.02
	CapsNetRL + MMWR + mDSR	1.560	16.07	**10.51**

Table 6.14: Performance evaluation using risk-adjusted rewards Metrics Test ID; Test 4

The proposed CapsNet + MMWR+ mDSR followed closely as the second best-performing model with ΔF-VaR ratio of 17.01 and the highest ΔM-CVaR ratio of 5.23. The 5CNNRL + PVM + mDSR model achieved the highest Sharpe Ratio of 1.171, indicating the effectiveness of the proposed Markov Differential Sharpe Ratio. Similarly, for Test ID 6

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 5	Baseline Portfolio	1.700	14.57	5.20
	3CNNRL + PVM	**1.171**	6.11	1.99
	5CNNRL + PVM	1.340	14.06	4.92
	5CNNRL + PVM + mDSR	**1.171**	6.11	1.99
	5CNNRL + MMWR	1.350	14.22	4.99
	5CNNRL + MMWR+ mDSR	-0.750	-11.26	-2.34
	CapsNet + PVM	1.380	12.52	4.69
	CapsNet + PVM + mDSR	1.353	14.24	5.01
	CapsNet + MMWR	1.030	**21.25**	**6.21**
	CapsNet + MMWR + mDSR	1.105	17.01	5.23

Table 6.15: Performance evaluation using risk-adjusted rewards Metrics Test ID; Test 5

in Table 6.16, the CapsNet + MMWR model achieved the highest Sharpe ratio of 2.200, with the corresponding highest ΔF-VaR ratio of 20.87 and the highest ΔM-CVaR ratio of 11.78. This suggests that our proposed MMWR integration enhances risk-adjusted returns. The proposed CapsNet + MMWR + mDSR closely followed as the second best performing model with a Sharpe ratio of 1.881, ΔF-VaR ratio of 11.16, and the highest Δ M-CVaR ratio of 7.05. Finally, for Test ID 7 in Table 6.17, the CapsNetRL + PVM + mDSR model

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 6	Baseline Portfolio	1.878	4.14	5.95
	3CNNRL + PVM	1.815	3.62	5.68
	5CNNRL + PVM	1.795	3.39	5.30
	5CNNRL + PVM + mDSR	1.770	6.20	4.92
	5CNNRL + MMWR	1.810	3.55	0.62
	5CNNRL + MMWR+ mDSR	1.770	6.08	4.12
	CapsNet + PVM	1.830	3.78	5.72
	CapsNet + PVM + mDSR	1.83	3.74	5.72
	CapsNet + MMWR	**2.200**	**20.87**	**11.78**
	CapsNet + MMWR + mDSR	1.881	11.16	7.05

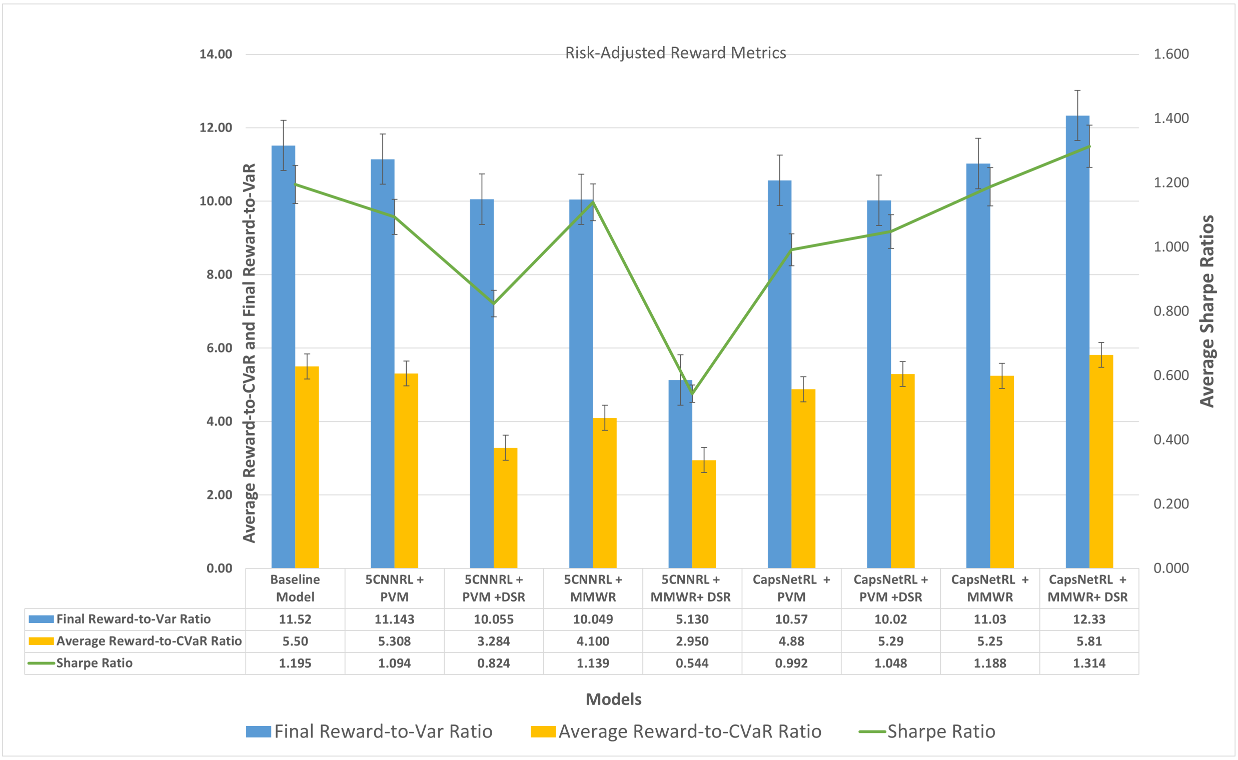
Table 6.16: Performance evaluation using Risk-adjusted rewards Metrics Test ID; Test 6

achieved the highest Sharpe ratio of 0.970, with the corresponding highest ΔF-VaR ratio of 18.79 and the highest ΔM-CVaR ratio of 3.91, indicating strong risk-adjusted returns. The introduction of the Markov Differential Sharpe Ratio (mDSR) 5CNNRL + PVM model significantly improved model performance.

Test ID	**Model**	**Sharpe Ratio**	Ratios (0.95)	
			ΔF-VaR	ΔM-CVaR
Test 7	Baseline Portfolio	0.157	11.95	0.49
	3CNNRL + PVM	0.096	11.56	0.44
	5CNNRL + PVM	0.092	11.48	0.42
	5CNNRL + PVM + mDSR	**0.970**	**18.79**	**3.91**
	5CNNRL + MMWR	0.720	13.30	3.13
	5CNNRL + MMWR + mDSR	0.720	13.26	3.16
	CapsNet + PVM	0.085	11.43	0.38
	CapsNet + PVM + mDSR	0.720	13.26	3.16
	CapsNet + MMWR	0.720	13.26	3.16
	CapsNet + MMWR + mDSR	0.720	13.26	3.16

Table 6.17: Performance evaluation using risk-adjusted rewards Metrics Test ID; Test 7

To determine the overall performance of each model on all the test sets used and rank them appropriately, the average of each risk-adjusted metric used, which are Sharpe ratio, Mean Reward-to-CVaR Ratio and Final Reward-to-VaR Ratio for Test IDs; Test 1, Test 2, Test 3, Test 4, Test 5, Test 6 and Test 7 were plotted as shown in Figure 6.13. The figure plots three different risk-adjusted return metrics on the same graph, allowing for a direct comparison of how each model performs across these metrics. From the figure, it can be observed that the CapsNet + MMWR + mDSR exhibited superior performance over all the other models with an average Sharpe ratio of 1.314, Mean Reward-to-CVaR Ratio of 5.81 and Final Reward-to-VaR Ratio of 12.33 across all the 7 test IDs. At the other end of the spectrum is the 5CNNRL + MMWR + mDSR model, which achieved the poorest average performance across all 7 test IDs with an average Sharpe ratio of 0.544, Mean Reward-to-CVaR Ratio of 2.950 and Final Reward-to-VaR Ratio of 5.130.



	Baseline Model	5CNNRL + PVM	5CNNRL + PVM +DSR	5CNNRL + MMWR	5CNNRL + MMWR+ DSR	CapsNetRL + PVM	CapsNetRL + PVM +DSR	CapsNetRL + MMWR	CapsNetRL + MMWR+ DSR
Final Reward-to-Var Ratio	11.52	11.143	10.055	10.049	5.130	10.57	10.02	11.03	12.33
Average Reward-to-CVaR Ratio	5.50	5.308	3.284	4.100	2.950	4.88	5.29	5.25	5.81
Sharpe Ratio	1.195	1.094	0.824	1.139	0.544	0.992	1.048	1.188	1.314

Figure 6.13: comparison of the average performance of the models for the Risk-adjusted returns metrics for CapsNet-based and CNN-based models using the Average across all the 7 Test IDs for Sharpe ratio, Mean Reward-to-CVaR Ratio and Final Reward-to-VaR Ratio

Significance Test on Sharpe Ratio metric for the models

As part of the analysis of the performance of various models used, a pairwise comparisons were conducted to determine if there are statistically significant differences between their Sharpe ratios shown in tables 6.18, 6.19, 6.20 and 6.21 respectively. The t-test, a statistical hypothesis test, was used for this purpose where it compared the models that used our proposed techniques with the models that did not use our proposed techniques.

The results from the t-test statistics shown in Tables 6.18, 6.19, 6.20 and 6.21 respectively provide important insights into the effectiveness of all the models including our proposed CapsNetRL + MMWR+ mDSR against each other. None of the pairwise comparisons between the models resulted in a p-value of less than 0.05, making the results from the Sharpe ratio statistically insignificant. As a result, a null hypothesis that states that there is no statistical difference between the mean Shape Ratio from each of the models could not be rejected for any of the paired models including those containing our proposed Multi-memory Weight Reservoir and mDSR.

Models Comparison	**t-stats**	**p-value**	**Reject null**
Baseline Portfolio vs 3CNNRL + PVM	0.217705	0.831318	No
Baseline Portfolio vs 5CNNRL + PVM	0.181525	0.858986	No
Baseline Portfolio vs 5CNNRL + PVM +mDSR	0.573946	0.577289	No
Baseline Portfolio vs 5CNNRL + MMWR	0.105591	0.917677	No
Baseline Portfolio vs 5CNNRL + MMWR+ mDSR	0.865804	0.406973	No
Baseline Portfolio vs CapsNetRL + PVM	0.375416	0.713927	No
Baseline Portfolio vs CapsNetRL + PVM +mDSR	0.636079	0.538596	No
Baseline Portfolio vs CapsNetRL + MMWR	0.524983	0.609183	No
Baseline Portfolio vs CapsNetRL + MMWR+ mDSR	-0.259967	0.801085	No
3CNNRL + PVM vs 5CNNRL + PVM	-0.035606	0.972182	No
3CNNRL + PVM vs 5CNNRL + PVM +mDSR	0.388534	0.704899	No
3CNNRL + PVM vs 5CNNRL + MMWR	-0.124458	0.903039	No

Table 6.18: Part 1 of the results of the pairwise comparison of the models' performance using the Sharpe Ratio metric using t-test statistics

Models Comparison	**t-stats**	**p-value**	**Reject null**
3CNNRL + PVM vs 5CNNRL + MMWR+ mDSR	0.707425	0.495618	No
3CNNRL + PVM vs CapsNetRL + PVM	0.153122	0.880853	No
3CNNRL + PVM vs CapsNetRL + PVM +mDSR	0.470415	0.647913	No
3CNNRL + PVM vs CapsNetRL + MMWR	0.315499	0.757823	No
3CNNRL + PVM vs CapsNetRL + MMWR+ mDSR	-0.54203	0.601663	No
5CNNRL + PVM vs 5CNNRL + PVM +mDSR	0.418086	0.683757	No
5CNNRL + PVM vs 5CNNRL + MMWR	-0.086459	0.932547	No
5CNNRL + PVM vs 5CNNRL + MMWR+ mDSR	0.732292	0.480898	No
5CNNRL + PVM vs CapsNetRL + PVM	0.189054	0.853221	No
5CNNRL + PVM vs CapsNetRL + PVM +mDSR	0.493491	0.634156	No
5CNNRL + PVM +mDSR vs 5CNNRL + MMWR	-0.508776	0.621348	No

Table 6.19: Part 2 of the results of the pairwise comparison of the models performance using the Sharpe Ratio metric using t-test statistics

Models Comparison	**t-stats**	**p-value**	**Reject null**
5CNNRL + PVM +mDSR vs 5CNNRL + MMWR+ mDSR	0.342527	0.738168	No
5CNNRL + PVM +mDSR vs CapsNetRL + PVM	-0.265231	0.795764	No
5CNNRL + PVM +mDSR vs CapsNetRL + PVM +mDSR	0.112005	0.912722	No
5CNNRL + PVM +mDSR vs CapsNetRL + MMWR	-0.106842	0.916755	No
5CNNRL + PVM +mDSR vs CapsNetRL + MMWR+ mDSR	-0.889381	0.401364	No
5CNNRL + MMWR vs 5CNNRL + MMWR+ mDSR	0.816493	0.434719	No
5CNNRL + MMWR vs CapsNetRL + PVM	0.289795	0.776938	No
5CNNRL + MMWR vs CapsNetRL + PVM +mDSR	0.578027	0.576572	No
5CNNRL + MMWR vs CapsNetRL + MMWR	0.451591	0.659851	No
5CNNRL + MMWR vs CapsNetRL + MMWR+ mDSR	-0.427558	0.678928	No

Table 6.20: Part 3 of the results of the pairwise comparison of the models performance using the Sharpe Ratio metric using t-test statistics

Models Comparison	t-stats	p-value	Reject null
5CNNRL + MMWR+ mDSR vs CapsNetRL + PVM	-0.605611	0.558812	No
5CNNRL + MMWR+ mDSR vs CapsNetRL + PVM +mDSR	-0.217993	0.83111	No
5CNNRL + MMWR+ mDSR vs CapsNetRL + MMWR	-0.460099	0.655033	No
5CNNRL + MMWR+ mDSR vs CapsNetRL + MMWR+ mDSR	-1.145958	0.289603	No
CapsNetRL + PVM vs CapsNetRL + PVM +mDSR	0.361163	0.725533	No
CapsNetRL + PVM vs CapsNetRL + MMWR	0.174782	0.864201	No
CapsNetRL + PVM vs CapsNetRL + MMWR+ mDSR	-0.763914	0.464966	No
CapsNetRL + PVM +mDSR vs CapsNetRL + MMWR	-0.21689	0.832406	No
CapsNetRL + PVM +mDSR vs CapsNetRL + MMWR+ mDSR	-0.904631	0.39535	No
CapsNetRL + MMWR vs CapsNetRL + MMWR+ mDSR	-0.911005	0.388102	No

Table 6.21: Part 4 of the results of the pairwise comparison of the models performance using the Sharpe Ratio metric using t-test statistics

Risk-only Metric

This section provides an evaluation of different models used for each of the test IDs based on VaR and CVaR at a confidence level of 95%. The Tables 6.22 to 6.28 below provide valuable information on the risk profiles of different models in various trading time frames. For Test ID 1 in Table 6.22, the Baseline Portfolio achieved the lowest VaR of 0.0092 and CVaR of 0.012, indicating a relatively low risk. Among the other models used, the CapsNetRL + MMWR model was the best rated with the lowest VaR of 0.0092 and CVaR of 0.01176, suggesting superior risk management. The 5CNNRL + PVM + mDSR model has the highest VaR (0.01983) and CVaR (0.02506), indicating higher risk levels compared to other models. For Test ID 2 in Table 6.23, the Baseline Portfolio obtained the lowest

Test ID	Model	Risk (0.95)	
		VaR	**CVaR**
Test 1	Baseline Portfolio	0.0092	0.012
	3CNNRL + PVM	0.0098	0.01249
	5CNNRL + PVM	0.0098	0.0125
	5CNNRL + PVM + mDSR	0.01983	0.02506
	5CNNRL + MMWR	0.01761	0.0222
	5CNNRL + MMWR+ mDSR	0.0176	0.0222
	CapsNetRL + PVM	0.0189	0.0239
	CapsNetRL + PVM + mDSR	0.0286	0.03617
	CapsNetRL + MMWR	**0.0092**	**0.01176**
	CapsNetRL + MMWR + mDSR	0.0176	0.0222

Table 6.22: Performance evaluation using Risk-only metric for Test ID; Test 1

Test ID	Model	Risk (0.95)	
		VaR	**CVaR**
Test 2	Baseline Portfolio	**0.0145**	**0.01822**
	3CNNRL + PVM	0.0156	0.0197
	5CNNRL + PVM	0.0164	0.0206
	5CNNRL + PVM + mDSR	0.0526	0.0199
	5CNNRL + MMWR	0.0158	0.01944
	5CNNRL + MMWR+ mDSR	0.0395	0.0496
	CapsNetRL + PVM	0.0154	0.0194
	CapsNetRL + PVM + mDSR	0.0395	0.046
	CapsNetRL + MMWR	0.016	0.02012
	CapsNetRL + MMWR + mDSR	0.027	0.034

Table 6.23: Performance evaluation using Risk-only metric for Test ID; Test 2

VaR of 0.0145 and CVaR of 0.01822 among all the models used, indicating their ability to minimise losses in the portfolios. For Test ID 3 in Table 6.24, the CapsNetRL + PVM performed exceptionally well with the lowest VaR of 0.0124 and CVaR of 0.01562 achieved. The 5CNNRL + MMWR and 5CNNRL + MMWR + mDSR models achieved the highest VaR and CVaR values of 0.0475 and 0.05994 respectively. For Test ID 4 in Table 6.25, the Baseline Portfolio achieved the best results in terms of the VaR and CVaR where the lowest values of (0.0125 and 0.0158 were obtained.

In Test ID 5 shown in 6.26, CapsNetRL + PVM exhibited the lowest VaR of 0.01135 and CVaR of 0.0143, indicating a lower risk compared to the models. Furthermore, the baseline model was the second-best performing model with a VaR of 0.0119 and CVaR of 0.015. For this test ID, the 3CNNRL + PVM model exhibited high VaR and CVaR of 0.0386 and 0.0484 respectively, suggesting a higher potential for portfolios to incur losses when used.

Test ID	**Model**	**Risk (0.95)**	
		VaR	**CVaR**
Test 3	Baseline Portfolio	0.013	0.0167
	3CNNRL + PVM	0.0139	0.0175
	5CNNRL + PVM	0.0137	0.01723
	5CNNRL + PVM + mDSR	0.0144	0.0182
	5CNNRL + MMWR	0.0475	0.05994
	5CNNRL + MMWR+ mDSR	0.0475	0.05994
	CapsNetRL + PVM	**0.0124**	**0.01562**
	CapsNetRL + PVM + mDSR	0.0135	0.01703
	CapsNetRL + MMWR	0.01487	0.01875
	CapsNetRL + MMWR + mDSR	0.0475	0.05994

Table 6.24: Performance evaluation using Risk-only metric for Test ID; Test 3

Test ID	**Model**	**Risk (0.95)**	
		VaR	**CVaR**
Test 4	Baseline Portfolio	0.0125	0.0158
	3CNNRL + PVM	0.0144	0.0182
	5CNNRL + PVM	0.0134	0.0169
	5CNNRL + PVM + mDSR	0.033	0.0417
	5CNNRL + MMWR	0.01277	0.0161
	5CNNRL + MMWR+ mDSR	0.0448	0.05642
	CapsNetRL + PVM	0.0126	0.01588
	CapsNetRL + PVM + mDSR	0.0217	0.0273
	CapsNetRL + MMWR	0.013	0.0165
	CapsNetRL + MMWR + mDSR	0.025	0.03155

Table 6.25: Performance evaluation using Risk-only metric for Test ID; Test 4

Test ID	**Model**	**Risk (0.95)**	
		VaR	**CVaR**
Test 5	Baseline Portfolio	0.0119	0.015
	3CNNRL + PVM	0.0386	0.0484
	5CNNRL + PVM	0.0135	0.017
	5CNNRL + PVM + mDSR	0.0386	0.0484
	5CNNRL + MMWR	0.01275	0.01605
	5CNNRL + MMWR+ mDSR	0.021	0.0262
	CapsNetRL + PVM	**0.01135**	**0.0143**
	CapsNetRL + PVM + mDSR	0.01265	0.0159
	CapsNetRL + MMWR	0.02	0.02525
	CapsNetRL + MMWR + mDSR	0.0176	0.02216

Table 6.26: Performance evaluation using Risk-only metric for Test ID; Test 5

In Test ID 6 shown in 6.27, the Baseline Portfolio achieved the lowest VaR of 0.0114

Test ID	Model	Risk (0.95)	
		VaR	**CVaR**
Test 6	Baseline Portfolio	**0.0114**	**0.0145**
	3CNNRL + PVM	0.0124	0.0156
	5CNNRL + PVM	0.01309	0.01649
	5CNNRL + PVM + mDSR	0.0224	0.0282
	5CNNRL + MMWR	0.01258	0.01585
	5CNNRL + MMWR + mDSR	0.0224	0.0282
	CapsNetRL + PVM	0.01195	0.0151
	CapsNetRL + PVM + mDSR	0.01206	0.01519
	CapsNetRL + MMWR	0.022	0.0274
	CapsNetRL + MMWR+ mDSR	0.0176	0.0223

Table 6.27: Performance evaluation using Risk-only metric for Test ID; Test 6

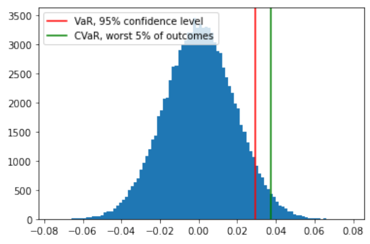
and CVaR of 0.0145. The 5CNNRL + PVM + mDSR and 5CNNRL + MMWR + mDSR models exhibited relatively high-risk levels with a VaR and CVaR of 0.0224 and 0.0282 respectively.

Test ID	Model	Risk (0.95)	
		VaR	**CVaR**
Test 1	Baseline Portfolio	0.0111	0.014
	3CNNRL + PVM	0.01166	0.0147
	5CNNRL + PVM	0.0118	0.0148
	5CNNRL + PVM + mDSR	0.0212	0.0273
	5CNNRL + MMWR	0.0295	0.0372
	5CNNRL + MMWR+ mDSR	0.0296	0.0372
	CapsNetRL + PVM	0.01195	0.0151
	CapsNetRL + PVM + mDSR	0.0296	0.0372
	CapsNetRL + MMWR	0.0296	0.0372
	CapsNetRL + MMWR+ mDSR	0.0296	0.0372

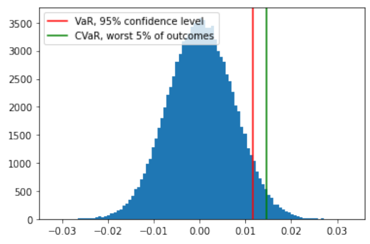
Table 6.28: Performance evaluation using Risk-only metric for Test ID; Test 7

Finally, for Test ID 7 shown in 6.28, the Baseline Portfolio model obtained the lowest VaR of 0.0111 and CVaR of 0.014. The CapsNetRL + PVM achieved the second lowest VaR of 0.0111 and CVaR of 0.014.

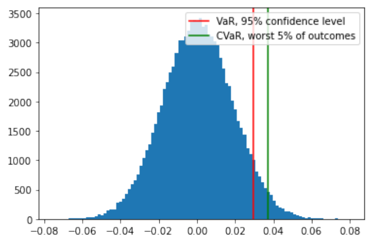
Figure 6.14 serves as a visual representation of the comparative evaluation of the VaR and CVaR metrics used as the risk-only metrics in this thesis. It shows a detailed plot of the VaR and CVaR values for both the CapsNet-based and CNN-based reinforcement learning (RL) models. The top row is dedicated to CapsNet-based models and their variants: CapsNetRL +MMWR+ mDSR, CapsNetRL+PVM+mDSR and CapsNetRL + MMWR models, while the bottom row represents CNN-based RL models and their respective variations: 5CNNRL + MMWR + mDSR, 5CNNRL + PVM + mDSR and 5CNNRL + MMWR models. Comparing the performance of the VaR and CVaR models within each row can be used to assess how effective our proposed MMWR and mDSR are, in comparison to the models with PVM and standard rewards respectively.



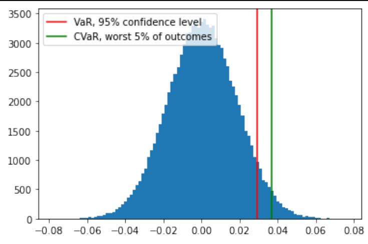
(a) demonstration of VaR and CVaR Values obtained from CapsNetRL + MMWR + mDSR model



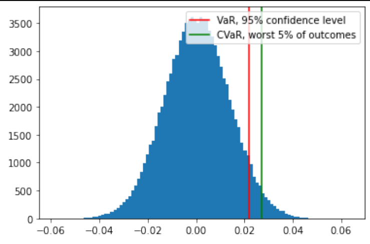
(b) demonstration of VaR and CVaR Values obtained from CapsNetRL + PVM + mDSR model



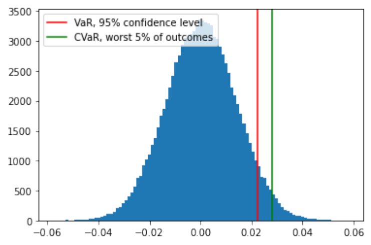
(c) demonstration of VaR and CVaR Values obtained from CapsNetRL + MMWR model



(d) demonstration of VaR and CVaR Values obtained from 5CNNRL + MMWR + mDSR model



(e) demonstration of VaR and CVaR Values obtained from 5CNNRL + PVM + mDSR model



(f) demonstration of VaR and CVaR Values obtained from 5CNNRL + MMWR model

Figure 6.14: Comparison of the Risk-only metrics showing VAR and CVAR values obtained from our proposed CapsNet-based models in the upper rows and CNN-based RL models in the lower row using Test ID 7.

6.7 Summary

This chapter has proposed effective techniques to improve the practice of financial risk management using the reinforcement learning technique for portfolio optimisation and asset allocation. CapsNet was used to enforce and implement the investment policy using an effective Dual Memory Weight Reservoir system and a specialised value-based deferential Sharpe ratio-based reward function to achieve superior performance over existing approaches. The introduction of the Differential Sharpe Ratio has been shown to be suitable for use as a reward function. The use of the Markov Differential Sharpe Ratio as a reward function has resulted in better strategies because of its ability to lead to multi-step maximisation of the Sharpe ratio that balances risk and returns. It is advantageous for online learning because of its ability to converge with speed and also its ability to adapt to changing market conditions during live trading. Our proposed RL algorithm can consistently produce better portfolio performance when risk has been taken into account. Even though our proposed CapsNetRL is innovative and achieved superior performance, it has some limitations which are:

- The integration of Deep Capsules Networks with RL adds significant complexity, which can make the implementation of the model to be difficult and understand and requires specialised knowledge in both deep learning and financial market dynamics.
- Due to the complex nature of the model, there is the requirement of substantial computational resources for training and inference.
- The model's performance may depend on the availability of high-quality data

Chapter 7

Conclusion and Future Works

7.1 Recap of Research Objectives

In this final chapter, the main objectives of this thesis are revisited with a reflection on how each chapter contributed to achieving these objectives. The overarching aim of this thesis was to explore and investigate the problems inherent in the application of machine learning techniques to financial risk management tasks. In this thesis, a comprehensive exploration of the application of machine learning techniques to the complex area of financial risk management has been carried out. By delving into the challenges posed by imbalanced data sets in credit risk assessment, the challenges of volatility forecasting, and venturing into the domain of portfolio optimisation through reinforcement learning, significant contributions have been made to the field. This final chapter encapsulates the essence of our research journey and outlines potential avenues for future research.

7.2 Summary of Thesis Achievements

The work has proposed the effective techniques necessary to overcome the problems of data unavailability, imbalanced datasets, and noisy datasets inherent in the adaptation of machine learning algorithms to financial risk management.

In Chapters 2 through 6, a comprehensive exploration of various machine learning techniques and models aimed at improving portfolio optimisation and financial risk management has been conducted. Let us summarise the key findings from each chapter:

In Chapter 2, a strong foundation was established by reviewing the existing literature in the machine learning landscape. This included the background information on Logistic Regression, background information on density estimation using the Gaussian Mixture Model (GMM), background information on the use of deep learning models such as the use of Long Short Term Memory (LSTM) and Generative Adversarial Networks (GAN) for time-series forecasting, the background information on Continuous Wavelet Transform and the basic concepts of reinforcement learning.

Chapter 3 also introduced the background of the financial risk management techniques used in this thesis. It provided the relevant financial background on credit risk modelling, market risk, volatility forecasting, and portfolio optimisation concepts used. In a nutshell, chapters 2 and 3 highlighted the gaps in current approaches and set the stage for our proposed novel methodologies.

In Chapter 4, a Hybrid Dual Resampling with Cost-Sensitive Technique (HDRCS) that can be used to overcome the problem of insufficient and imbalanced credit datasets was proposed. It captured and evaluated the novel Hybrid Dual Resampling with Cost-sensitive Technique to model the credit risk of imbalanced data sets. The technique involved the simultaneous use of cluster-based under-sampling on the majority class, the use of the Gaussian Mixture Model to over-sample the minority class and a Cost-sensitive learning algorithm.

In Chapter 5, detailed information on the novel triple-discriminator used within the proposed Continuous Wavelet Transform Triple Discriminator Generative Adversarial Network (cwt-TriGAN) for volatility estimation was covered. A detailed architecture of the proposed cwt-TriGAN that includes the in-depth operation of the eight component functions that make up our proposed cwt-TriGAN was captured.

In Chapter 6, the use of CapsNet-based reinforcement learning for portfolio optimisation was covered. It covered our proposed Multi-Memory Weight Reservoir (MMWR) training scheme using Capsules Neural Network and also captured the proposed Markov differential Sharpe ratio for portfolio optimisation.

7.3 Summary of Contributions to the Field

This research has made several significant contributions that enhance our understanding of machine learning's applicability and effectiveness in financial risk management. The contributions are as follows:

Improved credit Risk Assessment through HDRCS

One of the primary objectives of this research was to improve the credit risk assessment on imbalanced datasets. The proposed Hybrid Dual Resampling with Cost-Sensitive Technique (HDRCS) has offered a comprehensive solution to address the problem of data imbalance through a combination of oversampling, under-sampling techniques, and cost-sensitive learning. The empirical formula for optimal cluster numbers further refines this approach, resulting in a powerful tool for financial institutions to make more accurate credit decisions, thus minimising the risk of misjudgments and incorrect valuations. This technique showcases promising results in mitigating the challenges posed by imbalanced financial datasets.

Advantages:

- Precise Credit Decisions: HDRCS enhances the precision of credit risk assessment, reducing the likelihood of misclassifying credit applicants.
- Risk Mitigation: The technique helps financial institutions mitigate the risks associated with incorrect valuations and credit judgments, potentially reducing financial losses.

Limitations:

- Data Dependency: The HDRCS algorithm's performance depends on the quality and the ability of the GMM to accurately capture the distributions of the minority class data used.
- Complexity in Implementation: The hybrid nature of the model, combining resampling and cost-sensitive approaches, can lead to increased complexity in implementation and parameter tuning.
- Class Imbalance Sensitivity: While designed to address class imbalance, the technique might still be sensitive to extreme imbalances.

- Computational Cost: Due to the hybrid approach and the multiple steps and algorithms involved, the HDRCS model might incur higher computational costs compared to simpler models.

Enhanced Volatility Forecasting using cwt-TriGAN

Leveraging the power of the proposed continuous wavelet transform triple discriminator GAN network (cwt-TriGAN) results in a significant improvement in resolving the problem of modal collapse. Through the utilisation of an innovative ensemble loss function, this technique provides robust volatility forecasts and a powerful tool for understanding and anticipating market fluctuations. The empirical investigation into the effect of time steps further underlines the potential to refine volatility forecasting.

Advantages:

- Accurate Volatility Predictions: The cwt-TriGAN method provides accurate and reliable volatility forecasts, aiding investors in making well-informed decisions.
- Robust Insights: The ensemble loss function enhances the robustness of the model, ensuring reliable market insights

Limitations:

- The architecture of cwt-TriGAN is quite complex, which can result in difficulty in implementation with the requirement of significant computational resources, with the potential of limiting its accessibility and scalability.
- Risk of overfitting: Because of the complex nature of the model, it might be prone to overfitting, especially when dealing with limited or noisy financial datasets.
- Since the proposed model contains multiple components with their own individual parameters, the fine-tuning of these parameters for optimal performance can be challenging and time-consuming.
- Lack of Interpretability: Like many deep learning models, our proposed cwt-TriGAN may suffer from a lack of interpretability or transparency in its decision-making process. This ”black box” nature of it can be a significant drawback, especially in finance where understanding the reasoning behind predictions is crucial for trust and regulatory compliance.

CapsNet-based Reinforcement Learning for Portfolio Optimisation

The Capsules Neural Network-based Multi-Memory Weight Reservoir (MMWR) training scheme introduces a novel approach to portfolio optimization. By addressing challenges such as sample efficiency and exploration-exploitation trade-offs, this scheme empowers investors to navigate complex and dynamic market conditions more effectively. Additionally, the proposed Markov differential Sharpe ratio for the design of reward functions significantly contributes to improving the stability and optimisation of asset allocation in a portfolio.

Advantages:

- Adaptive Investment Strategies: The CapsNet-based approach equips investors with tools to build adaptive investment strategies that navigate complex market conditions effectively.
- Enhanced Stability: The Markov differential Sharpe ratio improves the stability and optimisation of asset allocation within portfolios.

Limitations:

- The integration of Deep Capsules Networks with RL adds significant complexity, which can make the implementation of the model difficult to understand and requires specialised knowledge in both deep learning and financial market dynamics.
- Due to the complex nature of the model, there is the requirement of substantial computational resources for training and inference.
- The model's performance may depend on the availability of high-quality data

7.4 Practical Implications and Relevance of Integrated Financial Risk Management

In the domain of credit risk, our proposed novel techniques lead to improved credit risk assessments, which directly influence the stability and performance of financial portfolios. Through the integration of these assessments into the broader context of financial risk management, an improved prediction and mitigation of potential losses due to bad credit events can be achieved. Volatility forecasting through the use of machine learning, offers a clearer understanding of market dynamics. This understanding is crucial for portfolio optimisation since it leads to the strategic allocation of assets to balance risk and return effectively.

The practical implications of our research offer numerous advantages for various stakeholders within the financial landscape. These are:

Financial Institutions: By adopting the HDRCS technique, financial institutions can significantly enhance their credit risk assessment processes. The reduction in misjudgments and incorrect valuations directly translates into reduced financial losses and improved portfolio quality.

Investors: Enhanced volatility forecasting provided by the cwt-TriGAN empowers investors with more precise insights into market fluctuations. This, in turn, enables them to make better-informed investment decisions, reduce uncertainty, and potentially maximize returns.

Portfolio Managers: The CapsNet-based reinforcement learning model offers a novel approach to portfolio optimisation. Portfolio managers can harness its' capabilities to build more robust and adaptive investment strategies that navigate complex market conditions effectively.

7.5 Future Works

Although this thesis has successfully achieved its primary objectives, there are still opportunities for further research and exploration. Conducting comprehensive investigations into the future works planned below will contribute to the advancement of knowledge and development of the effectiveness and enhancement of the models proposed in this thesis.

HDRCS

Future work will extend the HDRCS technique to address multi-class imbalanced credit dataset Zhu et al. (2022). This is because multi-class credit data often exhibit more complex imbalance patterns, and designing effective solutions for them will be a significant achievement. The multi-class credit data that will be used is known as the Kaggle Credit Risk dataset and categorises customers into three categories, which are high-risk, medium risk and low-risk customers. Our approach will be used to balance this imbalanced multi-class data with the aim of improving the prediction accuracy.

The HDRCS method will also be applied to deep learning models such as Convolutional Neural Networks, Long Short-Term Memory Networks and Auto-encoders, where it will be extended to use large datasets compared to the size of the data used in this work. Scaling up the dataset size will allow the HDRCS's performance to be assessed in real-world large-scale credit risk prediction scenarios.

cwt-TriGAN

Future work under this will involve exploring advanced GAN architectures and incorporating additional data preprocessing techniques to build on the cwt-TriGAN to further improve the accuracy of volatility forecast. Other tasks planned for the future involve the incorporation of external data such as economic indicators and market news into the cwt-TriGAN framework to explore how external information can improve the accuracy of volatility forecasts and decision-making. The development of online learning techniques for the cwt-TriGAN to adapt to changing market conditions in real-time will also be investigated. This will involve an investigation into how the model can continuously update its parameters and adapt its forecasting strategy based on new incoming data.

CapsNet-based reinforcement learning model

Future work under this proposed approach will involve the expansion of the CapsNet-based reinforcement learning models to encompass more complex portfolio optimisation scenarios and incorporate additional market factors that have the potential to produce more sophisticated and adaptable investment strategies. The use of actor-critic techniques will also be investigated to further improve the results. It will also involve the investigation into the use of transformer models to enforce investment policies.

7.6 Summary and Final Remarks

In conclusion, this thesis has significantly contributed to the field of financial risk management through the development and application of novel and advanced machine learning techniques. The insights obtained from addressing data imbalances, accurate forecasting of market volatility, and advanced portfolio optimisation techniques provide a robust framework for financial risk management. Specifically, the Hybrid Dual Resampling Cost-sensitive technique used for credit risk prediction on imbalanced datasets, cwt-TriGAN used for improved volatility forecasting and Capsules Neural Network-based used for improved portfolio optimisation represent innovative approaches to address the complex challenges in the field. These contributions have the potential to reshape risk assessment, decision-making and investment strategies in the financial sector. This thesis has therefore made significant contributions to the field of financial risk management through the development and application of novel and advanced machine learning techniques. These innovative approaches address the complex challenges in the field of finance and machine learning and have the potential to improve credit risk assessment, decision-making, and investment strategies in the financial sector. As we move forward, the continued exploration of these techniques will further enrich the understanding and management of financial risks to contribute to better financial risk assessment and decision-making.

References

1. Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An. A Bradford Book, Cambridge, MA, 2018. ISBN 9780262039246.

2. Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. SN Computer Science, 2(3):160, Mar 2021. ISSN 2661-8907. doi: 10.1007/s42979-021-00592-x. URL https://doi.org/10.1007/s42979-021-00592-x.

3. MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Machine learning towards intelligent systems: applications, challenges, and opportunities. Artificial Intelligence Review, 54(5):3299–3348, Jun 2021. ISSN 1573-7462. doi: 10.1007/s10462-020-09948-w. URL https://doi.org/10.1007/s10462-020-09948-w.

4. M. Crouhy, D. Galai, and R. Mark. The essentials of risk management. McGraw-Hill Education, New York, 2nd edition, 2014a.

5. F. Fabozzi, M. Crouchy, R. Mark, and D. Galai. Financial Risk Manager: Foundations of Risk Management. Pearson, New York, 2021.

6. Farzan Soleymani and Eric Paquet. Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—deepbreath. Expert Systems with Applications, 156:113456, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113456. URL https://www.sciencedirect.com/science/article/pii/S0957417420302803.

7. M.F. Dixon, I. Halperin, and P. Bilokon. Applications of reinforcement learning. In Machine Learning in Finance. Springer, 2020.

8. James A. Nichols, Hsien W. Herbert Chan, and Matthew A. B. Baker. Machine learning: applications of artificial intelligence to imaging and diagnosis. Biophysical Reviews, 11 (1):111–118, Feb 2019. ISSN 1867-2469. doi: 10.1007/s12551-018-0449-9. URL https://doi.org/10.1007/s12551-018-0449-9.

9. Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. Electronic Markets, 31(3):685–695, Sep 2021. ISSN 1422-8890. doi: 10.1007/s12525-021-00475-2. URL `https://doi.org/10.1007/s12525-021-00475-2`.

10. Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer-Verlag New York Inc., 2006. ISBN 0387310738. URL `https://www.ebook.de/de/product/5324937/christopher_m_bishop_pattern_recognition_and_machine_learning.html`.

11. Lyn Thomas. Modelling the credit risk for portfolios of consumer loans: Analogies with corporate loan models. Mathematics and Computers in Simulation, 79(8):2525 – 2534, 2009. ISSN 0378-4754. doi: https://doi.org/10.1016/j.matcom.2008.12.006. URL `http://www.sciencedirect.com/science/article/pii/S0378475408004199`. Nonstandard Applications of Computer Algebra Applied Scientific Computing VI: Numerical Grid Generation, Approximation and Visualization Modelling and Managing Financial Risk.

12. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

13. Pal Biprodip and Kumar Paul Mahit. A gaussian mixture based boosted classification scheme for imbalanced and oversampled data. In 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE). IEEE, feb 2017. doi: 10.1109/ecace.2017.7912938.

14. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

15. David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/silver14.html`.

16. Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. ArXiv, abs/1706.10059, 2017.

17. Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management, 2018.

18. Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning, 110(9):2419–2468, April 2021. doi: 10.1007/s10994-021-05961-4. URL https://doi.org/10.1007/s10994-021-05961-4.

19. Yuxi Li. Reinforcement learning in practice: Opportunities and challenges, 2022.

20. Tyler Lu, Dale Schuurmans, and Craig Boutilier. Non-delusional q-learning and value-iteration. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5fd0245f6c9ddbdf3eff0f505975b6a7-Paper.pdf.

21. David Abel, Will Dabney, Anna Harutyunyan, Mark K. Ho, Michael L. Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward, 2022.

22. Ayman Chaouki, Stephen Hardiman, Christian Schmidt, Emmanuel Sérié, and Joachim de Lataillade. Deep deterministic portfolio optimization. The Journal of Finance and Data Science, 6:16–30, 2020. ISSN 2405-9188. doi: https://doi.org/10.1016/j.jfds.2020.06.002. URL https://www.sciencedirect.com/science/article/pii/S2405918820300118.

23. Tianxiang Cui, Nanjiang Du, Xiaoying Yang, and Shusheng Ding. Multi-period portfolio optimization using a deep reinforcement learning hyper-heuristic approach. Technological Forecasting and Social Change, 198:122944, 2024. ISSN 0040-1625. doi: https://doi.org/10.1016/j.techfore.2023.122944. URL https://www.sciencedirect.com/science/article/pii/S0040162523006297.

24. Giorgio Lucarelli and Matteo Borrotti. A deep q-learning portfolio management framework for the cryptocurrency market. Neural Computing and Applications, 32:17229 – 17244, 2020. URL https://api.semanticscholar.org/CorpusID:224862422.

25. Uta Pigorsch and Sebastian Schäfer. High-dimensional stock portfolio trading with deep reinforcement learning, 2021.

26. Michael L. Littman, Ufuk Topcu, Jie Fu, Charles Isbell, Min Wen, and James MacGlashan. Environment-independent task specifications via gltl, 2017.

27. Saud Almahdi and Steve Y. Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. Expert Syst. Appl., 87:267–279, 2017.

28. Agresti Alan. Categorical Data Analysis. John Wiley and Sons, 1990. ISBN 0471853011.

29. Hyunjeong Kwon, Mingyu Woo, Young Hwan Kim, and Seokhyeong Kang. Statistical leakage analysis using gaussian mixture model. IEEE Access, 6:51939–51950, 2018. doi: 10.1109/access.2018.2870528.

30. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9 (8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

31. Yang Liu. Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. Expert Systems with Applications, 132:99 – 109, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2019.04.038. URL `http://www.sciencedirect.com/science/article/pii/S0957417419302635`.

32. Ruoxuan Xiong, Eric P. Nichols, and Yuan Shen. Deep learning stock volatility with google domestic trends. Papers, arXiv.org, 2016. URL `https://EconPapers.repec.org/RePEc:arx:papers:1512.04916`.

33. Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. Neural Computation, 12(10):2451–2471, 2000. doi: 10.1162/089976600300015015. URL `https://doi.org/10.1162/089976600300015015`.

34. Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2):654 – 669, 2018. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.11.054. URL `http://www.sciencedirect.com/science/article/pii/S0377221717310652`.

35. Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/reed16.html`.

36. S. Takahashi, Y. Chen, and Kumiko Tanaka-Ishii. Modeling financial time-series with generative adversarial networks. Physica A-statistical Mechanics and Its Applications, 527:121261, 2019.

37. Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

38. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

39. Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 5508–5518. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8789-time-series-generative-adversarial-networks.pdf.

40. J.B. Ramsey. The contribution of wavelets to the analysis of economic and financial data. Philosophical Transactions of the Royal Society B Biological Sciences, 357(357):2593–606, 1999.

41. Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron Leary. Pywavelets: A python package for wavelet analysis. Journal of Open Source Software, 4(36): 1237, 2019. doi: 10.21105/joss.01237. URL https://doi.org/10.21105/joss.01237.

42. Michel Misiti. Wavelet Toolbox for Use with MATLAB: User's Guide; Version 2; Computation, Visualization, Programming. MathWorks Incorporated, 2000.

43. Albert Cohen, Ingrid Daubechies, and Pierre Vial. Wavelets on the interval and fast wavelet transforms. Applied and Computational Harmonic Analysis, 1(1):54–81, 1993. ISSN 1063-5203. doi: https://doi.org/10.1006/acha.1993.1005. URL https://www.sciencedirect.com/science/article/pii/S1063520383710055.

44. James B. Ramsey and Zhifeng Zhang. The analysis of foreign exchange data using waveform dictionaries. Journal of Empirical Finance, 4(4):341–372, 1997. ISSN 0927-5398. doi: https://doi.org/10.1016/S0927-5398(96)00013-8. URL https://www.sciencedirect.com/science/article/pii/S0927539896000138.

45. Zhaojie Luo, Jinhui Chen, Xiao Jing Cai, Katsuyuki Tanaka, Tetsuya Takiguchi, Takuji Kinkyo, and Shigeyuki Hamori. Oil price forecasting using supervised gans with continuous wavelet transform features. In 2018 24th International Conference on Pattern Recognition (ICPR), pages 830–835, 2018. doi: 10.1109/ICPR.2018.8546240.

46. Ramazan Gencay, Faruk Selcuk, and Brandon Whitcher. Differentiating intraday seasonalities through wavelet multi-scaling. Physica A: Statistical Mechanics and its Applications, 289(3):543–556, 2001. ISSN 0378-4371. doi: https://doi.org/10.1016/S0378-4371(00)00463-5. URL https://www.sciencedirect.com/science/article/pii/S0378437100004635.

47. Sam Subbey, Kathrine Michalsen, and Geir Kjetil Nilsen. A tool for analyzing information from data storage tags: the continuous wavelet transform (CWT). Reviews in Fish Biology and Fisheries, 18(3):301–312, November 2007.

48. A. Popoola and K. Ahmad. In Testing the Suitability of Wavelet Preprocessing for TSK Fuzzy Models. IEEE International Conference on Fuzzy Systems, 2006.

49. M.L. Menendez, J.A. Pardo, L. Pardo, and M.C. Pardo. The jensen-shannon divergence. Journal of the Franklin Institute, 334(2):307–318, 1997. ISSN 0016-0032. doi: https://doi.org/10.1016/S0016-0032(96)00063-4. URL `https://www.sciencedirect.com/science/article/pii/S0016003296000634`.

50. Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. IEEE Transactions on Information Theory, 60(7):3797–3820, 2014.

51. David Silver. Lectures on reinforcement learning. URL: `https://www.davidsilver.uk/teaching/`, 2015.

52. M. Crouhy, D. Galai, and R. Mark. The Essentials of Risk Management, Second Edition. McGraw-Hill Education, 2014b. ISBN 9780071818513. URL `https://books.google.co.uk/books?id=qs1ulwEACAAJ`.

53. Richard Apostolik, Christopher Donohue, Peter Went, et al. Foundations of banking risk: an overview of banking, banking risks, and risk-based banking regulation. John Wiley, 2009.

54. Ameni Ghenimi, Hasna Chaibi, and Mohamed Ali Brahim Omri. The effects of liquidity risk and credit risk on bank stability: Evidence from the MENA region. Borsa Istanbul Review, 17(4):238–248, dec 2017. doi: 10.1016/j.bir.2017.05.002.

55. Naeem Siddiqi. Intelligent Credit Scoring : Building and Implementing Better Credit Risk Scorecards. John Wiley and Sons,, illustrated edition, 2017. ISBN 9781119279150. URL `https://books.google.co.uk/books?id=q-qoDQAAQBAJ`.

56. Marcin Borsuk. Climate-policy-relevant sectors and credit risk. Journal of Credit Risk, 19(1), 2023. URL `https://www.risk.net/journal-of-credit-risk/volume-19/number-1`.

57. Philippe Jorion. Value at risk: the new benchmark for managing financial risk. The McGraw-Hill Companies, Inc., 2007.

58. K. Kennedy, B. Mac Namee, S.J. Delany, M. O'Sullivan, and N. Watson. A window of opportunity: Assessing behavioural scoring. Expert Systems with Applications, 40(4): 1372–1380, mar 2013. doi: 10.1016/j.eswa.2012.08.052.

59. Feng-Hui Yu, Lu Jiejunn, Gu Jia-Wen, and Ching Wai-Ki. Modeling credit risk with hidden markov default intensity. Computational Economics, Nov 2018. ISSN 1572-9974. doi: 10.1007/s10614-018-9869-7. URL `https://doi.org/10.1007/s10614-018-9869-7`.

60. Stephen Zamore, Kwame Ohene Djan, Ilan Alon, and Bersant Hobdari. Credit risk research: Review and agenda. Emerging Markets Finance and Trade, 54(4):811–835, 2018. doi: 10.1080/1540496X.2018.1433658. URL `https://doi.org/10.1080/1540496X.2018.1433658`.

61. Franco Modigliani and Merton H. Miller. The cost of capital, corporation finance and the theory of investment. The American Economic Review, 48(3):261–297, 1958. ISSN 00028282. URL `http://www.jstor.org/stable/1809766`.

62. Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. Journal of Political Economy, 81(3):637–54, 1973. URL `https://EconPapers.repec.org/RePEc:ucp:jpolec:v:81:y:1973:i:3:p:637-54`.

63. Robert C. Merton. On the pricing of corporate debt : Trisk structure of interest rates. The Journal of Finance, 29(2):449–470, may 1974. doi: 10.1111/j.1540-6261.1974.tb03058.x.

64. Robert A. Jarrow and Stuart M. Turnbull. Pricing derivatives on financial securities subject to credit risk. The Journal of Finance, 50(1):53–85, mar 1995. doi: 10.1111/j.1540-6261.1995.tb05167.x.

65. Dilip B. Madan and Haluk Unal. Pricing the risks of default. Review of Derivatives Research, 2(2-3):121–160, dec 1998. doi: 10.1007/bf01531333.

66. T. Benzschawel. Credit Risk Modelling: Facts, Theory and Applications. Risk Books, 2012. ISBN 9781906348588. URL `https://books.google.co.uk/books?id=Nat6MAEACAAJ`.

67. Bernd Engelmann and Ha Pham. Measuring the performance of bank loans under basel ii/iii and ifrs 9/cecl. Risks, 8(3), 2020. ISSN 2227-9091. doi: 10.3390/risks8030093. URL `https://www.mdpi.com/2227-9091/8/3/93`.

68. BCBS. Basel i: International convergence of capital measurement and capital standards. 1988. URL `https://www.bis.org/publ/bcbs04a.pdf[accessed07.03.2023]`.

69. Haibin Zhu. Capital regulation and banks' financial decisions. International Journal of Central Banking, 4(1):165–211, March 2008. URL `https://ideas.repec.org/a/ijc/ijcjou/y2008q1a5.html`.

70. BCBS. Basel iii: International regulatory framework for banks. 2020. URL https://www.bis.org/bcbs/basel3.htm?m=3|14|572[accessed25.4.2020].

71. IASB. IFRS Standard 9: Financial Instruments. International Accounting Standards Committee Foundation (IASCF), 2015.

72. FASB. Accounting standards update no. 2016-13. financial instruments–credit losses (topic 326): Measurement of credit losses on financial instruments. 2016.

73. Sandar Win. What are the possible future research directions for banks' credit risk assessment research? a systematic review of literature. International Economics and Economic Policy, 15(4):743–759, Oct 2018. ISSN 1612-4812. doi: 10.1007/s10368-018-0412-z. URL https://doi.org/10.1007/s10368-018-0412-z.

74. John B. Caouette, Edward I. Altman, Paul Narayanan, and Robert Nimmo. Front Matter, pages i–xxvi. John Wiley and Sons, Ltd, 2008. ISBN 9781118266236. doi: https://doi.org/10.1002/9781118266236.fmatter. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118266236.fmatter.

75. M. Kabir Hassan, Jennifer Brodmann, Blake Rayfield, and Makeen Huda. Modeling credit risk in credit unions using survival analysis. International Journal of Bank Marketing, 36 (3):482–495, may 2018. doi: 10.1108/ijbm-05-2017-0091.

76. Ioan Trenca, Andreea Maria Pece, and Ioana Sorina Mihut. The assessment of market risk in the context of the current financial crisis. Procedia Economics and Finance, 32:1391–1406, 2015. ISSN 2212-5671. doi: https://doi.org/10.1016/S2212-5671(15)01516-6. URL https://www.sciencedirect.com/science/article/pii/S2212567115015166. Emerging Markets Queries in Finance and Business 2014, EMQFB 2014, 24-25 October 2014, Bucharest, Romania.

77. Nicolás Magner, Jaime F Lavin, Mauricio Valle, and Nicolás Hardy. The predictive power of stock market's expectations volatility: A financial synchronization phenomenon. PLoS One, 16(5):e0250846, May 2021.

78. Zhe Chen. The impact of trade and financial expansion on volatility of real exchange rate. PLoS One, 17(1):e0262230, January 2022.

79. Evan W. Anderson, Eric Ghysels, and Jennifer L. Juergens. The impact of risk and uncertainty on expected returns. Journal of Financial Economics, 94(2):233–263, 2009. ISSN 0304-405X. doi: https://doi.org/10.1016/j.jfineco.2008.11.001. URL https://www.sciencedirect.com/science/article/pii/S0304405X09001275.

80. Ser-Huang Poon and Clive Granger. Practical issues in forecasting volatility. Financial Analysts Journal, 61(1):45–56, 2005. doi: 10.2469/faj.v61.n1.2683. URL https://doi.org/10.2469/faj.v61.n1.2683.

81. Linlan Xiao and Abdurrahman Aydemir. 1 - volatility modelling and forecasting in finance. In John Knight and Stephen Satchell, editors, Forecasting Volatility in the Financial Markets (Third Edition), Quantitative Finance, pages 1 – 45. Butterworth-Heinemann, Oxford, 3rd edition, 2007. ISBN 978-0-7506-6942-9. doi: https://doi.org/10.1016/B978-075066942-9.500030. URL http://www.sciencedirect.com/science/article/pii/B9780750669429500030.

82. Christian T. Brownlees, Robert F. Engle, and Bryan T. Kelly. A practical guide to volatility forecasting through calm and storm. August 2011. doi: http://dx.doi.org/10.2139/ssrn.1502915. URL https://ssrn.com/abstract=1502915.

83. Robert F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. Econometrica, 50(4):987–1007, 1982. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1912773.

84. Christian Schlag, Julian Thimme, and Rüdiger Weber. Implied volatility duration: A measure for the timing of uncertainty resolution. Journal of Financial Economics, 140(1):127–144, 2021. ISSN 0304-405X. doi: https://doi.org/10.1016/j.jfineco.2020.11.003. URL https://www.sciencedirect.com/science/article/pii/S0304405X20302877.

85. Rongjun Yang, Lin Yu, Yuanjun Zhao, Hongxin Yu, Guiping Xu, Yiting Wu, and Zhengkai Liu. Big data analytics for financial market volatility forecast based on support vector machine. International Journal of Information Management, 50:452–462, 2020. ISSN 0268-4012. doi: https://doi.org/10.1016/j.ijinfomgt.2019.05.027. URL https://www.sciencedirect.com/science/article/pii/S0268401218313604.

86. Wenbo Ge, Pooia Lalbakhsh, Leigh Isai, Artem Lenskiy, and Hanna Suominen. Neural network–based financial volatility forecasting: A systematic review. ACM Comput. Surv., 55(1), jan 2022. ISSN 0360-0300. doi: 10.1145/3483596. URL https://doi.org/10.1145/3483596.

87. Ravi. Bansal and Amir Yaron. Risks for the long run: A potential resolution of asset pricing puzzles. The Journal of Finance, 59:1481–1509, 2004. doi: doi:10.1111/j.1540-6261.2004.00670.x.

88. Valentina Corradi, Walter Distaso, and Antonio Mele. Macroeconomic determinants of stock volatility and volatility premiums. Journal of Monetary Economics, 60(2):203 –

220, 2013. ISSN 0304-3932. doi: https://doi.org/10.1016/j.jmoneco.2012.10.019. URL http://www.sciencedirect.com/science/article/pii/S0304393212001341.

89. Robert F. Engle, Eric Ghysels, and Bumjean Sohn. Stock market volatility and macroeconomic fundamentals. The Review of Economics and Statistics, 95(3):776–797, 2013. doi: 10.1162/REST_a_00300. URL https://doi.org/10.1162/REST_a_00300.

90. CHARLOTTE CHRISTIANSEN, MAIK SCHMELING, and ANDREAS SCHRIMPF. A comprehensive look at financial volatility prediction by economic variables. Journal of Applied Econometrics, 27(6):956–977, 2012. ISSN 08837252, 10991255. URL http://www.jstor.org/stable/23355909.

91. Zhe Lin. Modelling and forecasting the stock market volatility of sse composite index using garch models. Future Generation Computer Systems, 79:960 – 972, 2018. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2017.08.033. URL http://www.sciencedirect.com/science/article/pii/S0167739X17313067.

92. Bradley S. Paye. 'Déjà vol': Predictive regressions for aggregate stock market volatility using macroeconomic variables. Journal of Financial Economics, 106(3):527–546, 2012. doi: 10.1016/j.jfineco.2012.06. URL https://ideas.repec.org/a/eee/jfinec/v106y2012i3p527-546.html.

93. Kim Young Min, Han Sung Kwon, Kim Tae Yoon, Oh Kyong, Joo Luo, and Zhiming Kim Chiho. Intelligent stock market instability index: Application to the korean stock market. Intelligent Data Analysis, 19(4):879–895, 2015. ISSN 0304-4076. doi: https://doi.org/10.3233/IDA-150749.

94. Wei Liu and Bruce Morley. Volatility forecasting in the hang seng index using the garch approach. Asia-Pacific Financial Markets, 16:51–63, 03 2009. doi: 10.1007/s10690-009-9086-4.

95. John C. Hull. Options, Futures, and Other Derivatives (International Edition). Pearsons, 2002. ISBN 13: 9780130465924.

96. Philippe Masset. Volatility stylized facts. SSRN Electronic Journal, 09 2011. doi: 10.2139/ssrn.1804070.

97. Ashutosh Kolte, Jewel Kumar Roy, and László Vasa. The impact of unpredictable resource prices and equity volatility in advanced and emerging economies: An econometric and machine learning approach. Resources Policy, 80:103216, 2023. ISSN 0301-4207. doi: https://doi.org/10.1016/j.resourpol.2022.103216. URL https://www.sciencedirect.com/science/article/pii/S0301420722006596.

98. Eugene F. Fama. The behavior of stock-market prices. The Journal of Business, 38(1): 34–105, 1965. ISSN 00219398, 15375374. URL http://www.jstor.org/stable/2350752.

99. Benoit Mandelbrot. The variation of certain speculative prices. The Journal of Business, 36(4):394–419, 1963. ISSN 00219398, 15375374. URL http://www.jstor.org/stable/2350970.

100. Fischer Black. The pricing of commodity contracts. Journal of Financial Economics, 3(1): 167 – 179, 1976. ISSN 0304-405X. doi: https://doi.org/10.1016/0304-405X(76)90024-6. URL http://www.sciencedirect.com/science/article/pii/0304405X76900246.

101. George Tauchen, Harold Zhang, and Ming Liu. Volume, volatility, and leverage: A dynamic analysis. Journal of Econometrics, 74(1):177 – 208, 1996. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(95)01755-0. URL http://www.sciencedirect.com/science/article/pii/0304407695017550.

102. Daniel B. Nelson. Conditional heteroskedasticity in asset returns: A new approach. Econometrica, 59(2):347–370, 1991. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/2938260.

103. Muhammad Surajo Sanusi. Investigating the sources of black's leverage effect in oil and gas stocks. Cogent Economics & Finance, 5(1):1, 2017. doi: 10.1080/23322039.2017.1318812. URL https://doi.org/10.1080/23322039.2017.1318812.

104. Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3):307 – 327, 1986. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(86)90063-1. URL http://www.sciencedirect.com/science/article/pii/0304407686900631.

105. Michel Dacorogna, Ramazan Gencay, Ulrich Muller, Richard Olsen, and Olivier Pictet. An Introduction to High-Frequency Finance. 01 2001.

106. Ramazan Gencay, Nikola Gradojevic, Faruk Selcuk, and Brandon Whitcher. Asymmetry of information flow between volatilities across time scales. Quantitative Finance, 10 (8):895–915, 2010. doi: 10.1080/14697680903460143. URL https://doi.org/10.1080/14697680903460143.

107. Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep learning for portfolio optimization. The Journal of Financial Data Science, 2(4):8–20, aug 2020. doi: 10.3905/jfds.2020.1.042. URL https://doi.org/10.3905\%2Fjfds.2020.1.042.

108. Bart Van Liebergen et al. Machine learning: a revolution in risk management and compliance? Journal of Financial Transformation, 45:60–67, 2017.

109. Mihaly Ormos and Andras Urban. Performance analysis of log-optimal portfolio strategies with transaction costs. Quantitative Finance, 13(10):1587–1597, 2013. doi: 10.1080/14697688.2011.570368. URL `https://doi.org/10.1080/14697688.2011.570368`.

110. Harry Markowitz. Portfolio selection. The Journal of Finance, 7(1):77–91, 1952. ISSN 00221082, 15406261. URL `http://www.jstor.org/stable/2975974`.

111. Eugene F. Fama and Kenneth R. French. The capital asset pricing model: Theory and evidence. Journal of Economic Perspectives, 18(3):25–46, September 2004. doi: 10.1257/0895330042162430. URL `https://www.aeaweb.org/articles?id=10.1257/0895330042162430`.

112. William F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk*. The Journal of Finance, 19(3):425–442, 1964. doi: https://doi.org/10.1111/j.1540-6261.1964.tb02865.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1964.tb02865.x`.

113. John Lintner. The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. The Review of Economics and Statistics, 47(1):13–37, 1965. ISSN 00346535, 15309142. URL `http://www.jstor.org/stable/1924119`.

114. R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. Journal of Risk, 2:21–41, 2000.

115. F.A Sortino and R van der Meer. Downside risk: Capturing what's at stake in investment situations. Journal of Portfolio Management, 17(4):27–31, 1991. ISSN 0095-4918. doi: 10.3905/jpm.1991.409343.

116. M. J. Ershadi and D. Omidzadeh. Customer validation using hybrid logistic regression and credit scoring model. Calitatea, 19:59–62, 2018. URL `https://en.irandoc.ac.ir/sites/fa/files/attach/article/q-asvol19no167december-2018p59-62.pdf`.

117. Luigi D'Arco, Haiying Wang, and Huiru Zheng. Deephar: A deep feed-forward neural network algorithm for smart insole-based human activity recognition. Neural Computing and Applications, March 2023. ISSN 0941-0643. doi: 10.1007/s00521-023-08363-w. Funding Information: Luigi D'Arco was funded by Ulster University Beitto Research Collaboration Programme. This research was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant agreement No. 823978. Publisher Copyright: © 2023, The Author(s).

118. Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 40(12):3358–3378,

2007. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2007.04.009. URL https://www.sciencedirect.com/science/article/pii/S0031320307001835.

119. Bee Wah Yap, Khatijahhusna Abd Rani, Hezlin Aryani Abd Rahman, Simon Fong, Zuraida Khairudin, and Nik Nik Abdullah. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In Tutut Herawan, Mustafa Mat Deris, and Jemal Abawajy, editors, Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013), pages 13–22, Singapore, 2014. Springer Singapore. ISBN 978-981-4585-18-7.

120. Ginny Y. Wong, Frank H. F. Leung, and Sai-Ho Ling. A hybrid evolutionary preprocessing method for imbalanced datasets. Information Sciences, 454-455:161–177, jul 2018. doi: 10.1016/j.ins.2018.04.068.

121. R. Laza, R. Pavon, M. Reboiro-Jato, and F. Fdez-Riverola. Evaluating the effect of unbalanced data in biomedical document classification. Journal of integrative bioinformatics, 8:177, 2011. doi: 10.1515/jib-2011-177.

122. Iain Brown and Christophe Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. Expert Systems with Applications, 39(3):3446–3453, February 2012. doi: 10.1016/j.eswa.2011.09.033. URL https://eprints.soton.ac.uk/204741/.

123. Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. World Wide Web, 16 (4):449–475, Jul 2013. ISSN 1573-1413. doi: 10.1007/s11280-012-0178-0. URL https://doi.org/10.1007/s11280-012-0178-0.

124. N. N. Rahman and D. N. Davis. Addressing the class imbalance problems in medical datasets. International Journal of Machine Learning and Computing, 3:224–228, 2013. doi: 10.7763/ijmlc.2013.v3.307.

125. Tom Au, Meei-Ling Ivy Chin, and Guangqin Ma. Mining rare events data by sampling and boosting: A case study. In Sushil K. Prasad, Harrick M. Vin, Sartaj Sahni, Mahadeo P. Jaiswal, and Bundit Thipakorn, editors, Information Systems, Technology and Management, pages 373–379, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12035-0.

126. Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl., 6(1):20–29, June 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007735. URL http://doi.acm.org/10.1145/1007730.1007735.

127. Nitesh Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: Special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl., 6(1):1–6, jun 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007733. URL `https://doi.org/10.1145/1007730.1007733`.

128. Qing Chen, Anguo Zhang, Tingwen Huang, Qianping He, and Yongduan Song. Imbalanced dataset-based echo state networks for anomaly detection. Neural Computing and Applications, Oct 2018. ISSN 1433-3058. doi: 10.1007/s00521-018-3747-z. URL `https://doi.org/10.1007/s00521-018-3747-z`.

129. Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. Journal of Big Data, 6(1):27, March 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0192-5. URL `https://doi.org/10.1186/s40537-019-0192-5`.

130. Hongyu Guo and Herna L. Viktor. Learning from imbalanced data sets with boosting and data generation: The databoost-im approach. SIGKDD Explor. Newsl., 6(1):30–39, June 2004. ISSN 1931-0145. doi: 10.1145/1007730.1007736. URL `http://doi.acm.org/10.1145/1007730.1007736`.

131. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16:321–357, jun 2002. doi: 10.1613/jair.953.

132. Zehao Liu, Emmanuel Osei-Brefo, Siyuan Chen, and Huizhi Liang. UoR at SemEval-2020 task 8: Gaussian mixture modelling (GMM) based sampling approach for multi-modal memotion analysis. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pages 1201–1207, Barcelona (online), December 2020. International Committee for Computational Linguistics. doi: 10.18653/v1/2020.semeval-1.159. URL `https://aclanthology.org/2020.semeval-1.159`.

133. Nathalie Japkowicz. The class imbalance problem: Significance and strategies. In In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI, pages 111–117, 2000.

134. Xu Han, Runbang Cui, Yanfei Lan, Yanzhe Kang, Jiang Deng, and Ning Jia. A gaussian mixture model-based combined resampling algorithm for classification of imbalanced credit data sets. International Journal of Machine Learning and Cybernetics, 10(12): 3687–3699, may 2019. doi: 10.1007/s13042-019-00953-2.

135. Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pages 1322–1328, 2008.

136. S. Chen, H. He, and E.A. Garcia. Ramoboost: Ranked minority oversampling in boosting. IEEE Transactions on Neural Networks, 21(10):1624–1642, 2010. doi: 10.1109/TNN.2010.2066988. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-77957793322&doi=10.1109\%2fTNN.2010.2066988&partnerID=40&md5=30f549487686c986fdf40927eaca2b91`. cited By 109.

137. Liangxiao Jiang, Chen Qiu, and Chaoqun Li. A novel minority cloning technique for cost-sensitive learning. International Journal of Pattern Recognition and Artificial Intelligence, 29(04):1551004, may 2015. doi: 10.1142/s0218001415510040.

138. E.R. Davies. Training sets and a priori probabilities with the nearest neighbour method of pattern recognition. Pattern Recognition Letters, 8(1):11–13, 1988. doi: 10.1016/0167-8655(88)90017-7. URL `https://www.scopus.com/inward/record.uri?eid=2-s2.0-0024255125&doi=10.1016\%2f0167-8655\%2888\%2990017-7&partnerID=40&md5=74aba008cfc6512a7549996cab800a72`. cited By 14.

139. Foster Provost and Tom Fawcett. Robust classification for imprecise environments. Mach. Learn., 42(3):203–231, March 2001. ISSN 0885-6125. doi: 10.1023/A:1007601015854. URL `http://dx.doi.org/10.1023/A:1007601015854`.

140. Peng Li, Pei-Li Qiao, and Yuan-Chao Liu. A hybrid re-sampling method for SVM learning from imbalanced data sets. In 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery. IEEE, oct 2008. doi: 10.1109/fskd.2008.407.

141. Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. Machine Learning, 46:191–202, 01 2002. doi: 10.1023/A:1012406528296.

142. Ibomoiye Domor Mienye and Yanxia Sun. Performance analysis of cost-sensitive learning methods with application to imbalanced medical data. Informatics in Medicine Unlocked, 25:100690, 2021. ISSN 2352-9148. doi: https://doi.org/10.1016/j.imu.2021.100690. URL `https://www.sciencedirect.com/science/article/pii/S235291482100174X`.

143. Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence, 5(4):221–232, 2016.

144. Jerzy Błaszczyński and Jerzy Stefanowski. Neighbourhood sampling in bagging for imbalanced data. Neurocomputing, 150:529–542, feb 2015. doi: 10.1016/j.neucom.2014.07.064.

145. Zhongbin Sun, Qinbao Song, Xiaoyan Zhu, Heli Sun, Baowen Xu, and Yuming Zhou. A novel ensemble method for classifying imbalanced data. Pattern Recognition, 48(5):1623–1637, may 2015. doi: 10.1016/j.patcog.2014.11.014.

146. I Tomek. Two modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(11):769–772, nov 1976. doi: 10.1109/tsmc.1976.4309452.

147. Robert E. Schapire. The strength of weak learnability. Machine Learning, 5(2):197–227, jun 1990. doi: 10.1007/bf00116037.

148. Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In Knowledge Discovery in Databases: PKDD 2003, pages 107–119. Springer Berlin Heidelberg, 2003. doi: 10.1007/978-3-540-39804-2_12.

149. Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 40:185–197, 2010.

150. Monika Arya, Hanumat Sastry G, Anand Motwani, Sunil Kumar, and Atef Zaguia. A novel extra tree ensemble optimized DL framework (ETEODL) for early detection of diabetes. Frontiers in Public Health, 9, February 2022. doi: 10.3389/fpubh.2021.797877. URL `https://doi.org/10.3389/fpubh.2021.797877`.

151. Changming Zhu and Zhe Wang. Entropy-based matrix learning machine for imbalanced data sets. Pattern Recognition Letters, 88:72–80, mar 2017. doi: 10.1016/j.patrec.2017.01.014.

152. Thomas Oommen, Laurie G. Baise, and Richard M. Vogel. Sampling bias and class imbalance in maximum-likelihood logistic regression. Mathematical Geosciences, 43(1): 99–120, oct 2010. doi: 10.1007/s11004-010-9311-8.

153. Haibo He and E. A. Garcia. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9):1263–1284, sep 2009. doi: 10.1109/tkde.2008.239.

154. Yidan Wang and Liming Yang. A robust loss function for classification with imbalanced datasets. Neurocomputing, 331:40–49, feb 2019. doi: 10.1016/j.neucom.2018.11.024.

155. Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, Machine Learning: ECML 2004, pages 39–50, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30115-8.

156. Qiuyan Yan, Shixiong Xia, and Fan-Rong Meng. Optimizing cost-sensitive svm for imbalanced data : Connecting cluster to classification. CoRR, abs/1702.01504, 2017.

157. Enislay Ramentol, Yailé Caballero, Rafael Bello, and Francisco Herrera. Smote-rsb: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. Knowledge and Information Systems, 33(2):245–265, dec 2011. doi: 10.1007/s10115-011-0465-6.

158. Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. Clustering-based undersampling in class-imbalanced data. Information Sciences, 409-410:17–26, oct 2017. doi: 10.1016/j.ins.2017.05.008.

159. Aytuğ Onan. Consensus clustering-based undersampling approach to imbalanced learning. Scientific Programming, 2019:1–14, mar 2019. doi: 10.1155/2019/5901087.

160. Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65, nov 1987. doi: 10.1016/0377-0427(87)90125-7.

161. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

162. Emmanuel Osei-Brefo. Credit risk modelling for small datasets. Master's thesis, University of Southampton, 2015.

163. Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. URL `DOI:https://doi.org/10.24432/C5NC77`.

164. Give Me Some Credit. Give me some credit data set, 2020. URL `https://www.kaggle.com/c/GiveMeSomeCredit/data`.

165. Lending Club. Lending club data set. URL `https://www.openintro.org/data/index.php?data=loans_full_schema`.

166. Li Liu and Zhiyuan Pan. Forecasting stock market volatility: The role of technical variables. Economic Modelling, 84:55 – 65, 2020. ISSN 0264-9993. doi: https://doi.org/10.1016/j.econmod.2019.03.007. URL `http://www.sciencedirect.com/science/article/pii/S0264999318309398`.

167. Ehsan Lotfi, M. Darini, and M. R. Karimi-T. Cost estimation using anfis. The Engineering Economist, 61(2):144–154, 2016. doi: 10.1080/0013791X.2015.1104568. URL `https://doi.org/10.1080/0013791X.2015.1104568`.

168. Dong Ha Kim, Suk Jun Lee, Kyong Joo Oh, and Tae Yoon Kim. An early warning system for financial crisis using a stock market instability index. Expert Systems, 26(3): 260–273, 2009. doi: 10.1111/j.1468-0394.2009.00485.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0394.2009.00485.x.

169. Jae Joon Ahn, Dong Ha Kim, Kyong Joo Oh, and Tae Yoon Kim. Applying option greeks to directional forecasting of implied volatility in the options market: An intelligent approach. Expert Syst. Appl., 39:9315–9322, 2012.

170. W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE, 12(7):0180944, 2017. doi: 10.1371/journal.pone.0180944. URL https://doi.org/10.1371/journal.pone.0180944.

171. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. Neural Comput., 18(7):1527–1554, July 2006.

172. Xingyu Zhou, Zhisong Pan, Guyu Hu, Siqi Tang, and Cheng Zhao. Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering, 2018:1–11, 2018. doi: 10.1155/2018/4907423.

173. Adriano Koshiyama, Nick Firoozye, and Philip Treleaven. Generative adversarial networks for financial trading strategies fine-tuning and combination. 2019.

174. Ugo Fiore, Alfredo Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Information Sciences, 479, 12 2017. doi: 10.1016/j.ins.2017.12.030.

175. Cesar F. Caiafa, Zhe Sun, Toshihisa Tanaka, Pere Marti-Puig, and Jordi Solé-Casals. Machine learning methods with noisy, incomplete or small datasets. Applied Sciences, 11 (9), 2021. URL https://www.mdpi.com/2076-3417/11/9/4132.

176. Tsung-Jung Hsieh, Hsiao-Fen Hsiao, and Wei-Chang Yeh. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. Applied Soft Computing, 11(2):2510–2525, 2011. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2010.09.007. URL https://www.sciencedirect.com/science/article/pii/S1568494610002565. The Impact of Soft Computing for the Progress of Artificial Intelligence.

177. Davood Zabihzadeh. Ensemble of loss functions to improve generalizability of deep metric learning methods, 2021.

178. Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. ArXiv, abs/1912.02757, 2019.

179. Hamideh Hajiabadi, Vahide Babaiyan, Davood Zabihzadeh, and Moein Hajiabadi. Combination of loss functions for robust breast cancer prediction. Computers and Electrical Engineering, 84:106624, 2020. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2020.106624. URL `https://www.sciencedirect.com/science/article/pii/S0045790620304791`.

180. Christopher Torrence and Gilbert P. Compo. A practical guide to wavelet analysis. Bulletin of the American Meteorological Society, 79(1):61 – 78, 1998. doi: 10.1175/1520-0477(1998)079⟨0061:APGTWA⟩2.0.CO;2. URL `https://journals.ametsoc.org/view/journals/bams/79/1/1520-0477_1998_079_0061_apgtwa_2_0_co_2.xml`.

181. Ismail Fawaz Hassan, Forestier Germain, Weber Jonathan, Idoumghar Lhassane, and Muller Pierre-Alain. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery, 33(4):917–963, 2019. doi: 10.1007/s10618-019-00619-1. URL `https://doi.org/10.1007/s10618-019-00619-1`.

182. Frank Z. Xing, Erik Cambria, and Yue Zhang. Sentiment-aware volatility forecasting. Knowledge-Based Systems, 176:68 – 76, 2019. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.03.029. URL `http://www.sciencedirect.com/science/article/pii/S0950705119301546`.

183. Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2980–2988. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5653-a-recurrent-latent-variable-model-for-sequential-data.pdf`.

184. Marcel Prokopczuk and Chardin Simen. The importance of the volatility risk premium for volatility forecasting. Journal of Banking & Finance, 40, 03 2014. doi: 10.2139/ssrn.2236370.

185. Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. ArXiv, abs/1611.09904, 2016.

186. Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. ArXiv, abs/1706.02633, 2017.

187. Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. Stock market prediction based on generative adversarial network. Procedia Computer Science, 147: 400 – 406, 2019. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2019.01.256. URL http://www.sciencedirect.com/science/article/pii/S1877050919302789. 2018 International Conference on Identification, Information and Knowledge in the Internet of Things.

188. Y Zhang, Z Gan, and L Carin. Generating text via adversarial training. NIPS workshop on Adversarial Training, 2016. Query date: 2021-03-11 11:12:31.

189. Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. Biosignal data augmentation based on generative adversarial networks. In 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018.

190. Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. 2019.

191. Alireza Koochali, Peter Schichtel, Andreas Dengel, and Sheraz Ahmed. Probabilistic forecasting of sensory data with generative adversarial networks – forgan. IEEE Access, 7:63868–63880, 2019. doi: 10.1109/ACCESS.2019.2915544.

192. Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. CoRR, abs/1411.1784, 2014.

193. Milena Vuletić, Felix Prenzel, and Mihai Cucuringu. Fin-GAN: Forecasting and classifying financial time series via generative adversarial networks. SSRN Electronic Journal, 2023. doi: 10.2139/ssrn.4328302.

194. Rao Fu, Jie Chen, Shutian Zeng, Yiping Zhuang, and Agus Sudjianto. Time series simulation by conditional generative adversarial net. International Journal of Mechanical and Industrial Engineering, 14(6):458 – 471, 2020. ISSN eISSN: 1307-6892. URL https://publications.waset.org/vol/162.

195. Divya Saxena and J. Cao. D-gan: Deep generative adversarial nets for spatio-temporal prediction. ArXiv, abs/1907.08556, 2019.

196. Raphaël Dang-Nhu, Gagandeep Singh, Pavol Bielik, and Martin Vechev. Adversarial attacks on probabilistic autoregressive forecasting models. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 2356–2365. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/dang-nhu20a.html.

197. Tomer Golany, Kira Radinsky, and Daniel Freedman. SimGANs: Simulator-based generative adversarial networks for ECG synthesis to improve deep ECG classification. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 3597–3606. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/golany20a.html.

198. Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. Deeptrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. Proceedings of the AAAI Conference on Artificial Intelligence, 35 (1):643–650, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16144.

199. Roni Mittelman. Time-series modeling with undecimated fully convolutional neural networks. ArXiv, abs/1508.00317, 2015.

200. Pascal Wallisch, Michael E. Lusignan, Marc D. Benayoun, Tanya I. Baker, Adam S. Dickey, and Nicholas G. Hatsopoulos. Chapter 13 - wavelets. In Pascal Wallisch, Michael E. Lusignan, Marc D. Benayoun, Tanya I. Baker, Adam S. Dickey, and Nicholas G. Hatsopoulos, editors, MATLAB for Neuroscientists (Second Edition), pages 245–252. Academic Press, San Diego, second edition edition, 2014. ISBN 978-0-12-383836-0. doi: https://doi.org/10.1016/B978-0-12-383836-0.00013-8. URL https://www.sciencedirect.com/science/article/pii/B9780123838360000138.

201. William F. Sharpe. The sharpe ratio. The Journal of Portfolio Management, 21(1):49–58, 1994. ISSN 0095-4918. doi: 10.3905/jpm.1994.409501. URL https://jpm.pm-research.com/content/21/1/49.

202. Kumar Yashaswi. Deep reinforcement learning for portfolio optimization using latent feature state space (lfss) module, 2021.

203. Giuseppe Nuti, Mahnoosh Mirghaemi, Philip Treleaven, and Chaiyakorn Yingsaeree. Algorithmic trading. Computer, 44(11):61–69, 2011. doi: 10.1109/MC.2011.31.

204. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553): 436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

205. Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21, pages 44–51. Springer, 2011.

206. Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. Advances in neural information processing systems, 30, 2017.

207. William T. Ziemba. The symmetric downside-risk sharpe ratio. The Journal of Portfolio Management, 32(1):108–122, 2005. ISSN 0095-4918. doi: 10.3905/jpm.2005.599515. URL `https://jpm.pm-research.com/content/32/1/108`.

208. Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning, 8(3–4):293–321, 1992. URL `http://www.cs.ualberta.ca/~sutton/lin-92.pdf`.

209. Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience replay optimization. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19, page 4243–4249. AAAI Press, 2019. ISBN 9780999241141.

210. Xiaojiao Tong and Felix Wu. Robust reward–risk ratio optimization with application in allocation of generation asset. Optimization, 63(11):1761–1779, 2014. doi: 10.1080/02331934.2012.672419. URL `https://doi.org/10.1080/02331934.2012.672419`.

211. Gordon Alexander and Alexandre Baptista. Portfolio performance evaluation using value-at-risk. The Journal of Portfolio Management, 29:93–102, 06 2003a. doi: 10.3905/jpm.2003.319898.

212. Y. Cai, Kenneth L. Judd, and Rong Xu. Numerical solution of dynamic portfolio optimization with transaction costs. Econometrics: Mathematical Methods & Programming eJournal, 2013.

213. Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. Proceedings of the AAAI Conference on Artificial Intelligence, 34(01): 1112–1119, Apr. 2020. doi: 10.1609/aaai.v34i01.5462. URL `https://ojs.aaai.org/index.php/AAAI/article/view/5462`.

214. J.B. Heaton, Nick Polson, and Jan Witte. Deep learning for finance: Deep portfolios. Business and Industry, 33(1):3–12 „ 2016. doi: 10.2139/ssrn.2838013. URL `https://ssrn.com/abstract=2838013`. Available at SSRN:.

215. John E. Moody and Matthew Saffell. Learning to trade via direct reinforcement. IEEE transactions on neural networks, 12 4:875–89, 2001.

216. M.A.H. Dempster and V. Leemans. An automated fx trading system using adaptive reinforcement learning. Expert Systems with Applications, 30(3):543552, 2006.

ISSN 09574174. doi: https://doi.org/10.1016/j.eswa.2005.10.012. URL `https://www.sciencedirect.com/science/article/pii/S0957417405003015`. Intelligent Information Systems for Financial Engineering.

217. Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. IEEE Transactions on Neural Networks and Learning Systems, 28:653–664, 2017.

218. Eric Benhamou, David Saltiel, Jean Jacques Ohana, Jamal Atif, and Rida Laraki. Deep reinforcement learning (drl) for portfolio allocation. In Yuxiao Dong, Georgiana Ifrim, Dunja Mladenić, Craig Saunders, and Sofie Van Hoecke, editors, Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track, pages 527–531, Cham, 2021. Springer International Publishing. ISBN 978-3-030-67670-4.

219. J. Moody, L. Wu, Y. Liao, and M. Saffell. Performance functions and reinforcement learning for trading systems and portfolios. Journal of Forecasting, 17(5-6):441–470, 1998. URL `www.scopus.com`. Cited By :133.

220. Seok-Jun Bu and Sung-Bae Cho. Learning optimal q-function using deep boltzmann machine for reliable trading of cryptocurrency. In Hujun Yin, David Camacho, Paulo Novais, and Antonio J. Tallón-Ballesteros, editors, Intelligent Data Engineering and Automated Learning – IDEAL 2018, pages 468–480, Cham, 2018. Springer International Publishing. ISBN 978-3-030-03493-1.

221. Parag C. Pendharkar and Patrick Cusatis. Trading financial indices with reinforcement learning agents. Expert Systems with Applications, 103:1–13, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.02.032. URL `https://www.sciencedirect.com/science/article/pii/S0957417418301209`.

222. Xing Wu, Haolei Chen, Jianjia Wang, Luigi Troiano, Vincenzo Loia, and Hamido Fujita. Adaptive stock trading strategies with deep reinforcement learning methods. Information Sciences, 538:142–158, 2020.

223. Carlos Betancourt and Wen-Hui Chen. Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. Expert Systems with Applications, 164: 114002, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.114002. URL `https://www.sciencedirect.com/science/article/pii/S0957417420307776`.

224. John E. Moody and Matthew Saffell. Reinforcement learning for trading systems and portfolios. In KDD, 1998.

225. Luca Alessandro Dombetzki. An overview over capsule networks. Network Architectures and Services, 2018.

226. Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/mniha16.html`.

227. Shokoofeh Banihashemi and Sarah Navidi. Portfolio performance evaluation in mean-cvar framework: A comparison with non-parametric methods value at risk in mean-var analysis. Operations Research Perspectives, 4:21–28, 2017. ISSN 2214-7160. doi: https://doi.org/10.1016/j.orp.2017.02.001. URL `https://www.sciencedirect.com/science/article/pii/S2214716016300665`.

228. Sergey Sarykalin, Gaia Serraino, and Stan Uryasev. Value-at-risk vs. conditional value-at-risk in risk management and optimization. In State-of-the-Art Decision-Making Tools in the Information-Intensive Age, pages 270–294. INFORMS, September 2008.

229. Gordon J. Alexander and Alexandre M. Baptista. Portfolio performance evaluation using value at risk. The Journal of Portfolio Management, 29(4):93–102, 2003b. ISSN 0095-4918. doi: 10.3905/jpm.2003.319898. URL `https://jpm.pm-research.com/content/29/4/93`.

230. Fangting Zhu, Xi Chen, and Gang Li. Multi-classification assessment of personal credit risk based on stacking integration. Procedia Computer Science, 214:605–612, 2022. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2022.11.218. URL `https://www.sciencedirect.com/science/article/pii/S1877050922019287`. 9th International Conference on Information Technology and Quantitative Management.