University of Reading

Department of Computer Science

Sensitivity Analysis of Convolutional Neural Networks: A Ranking of Hyper-parameter Influence and its Practical Application

Rhian Taylor

Supervisor: Pat Parslow

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Doctor of Philosophy in *Computer Science*

January 10, 2023

Declaration

I, Rhian Taylor, of the Department of Computer Science, University of Reading, confirm that all the sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Rhian Taylor
January 10, 2023

Abstract

The aim of this work is to better understand and quantify the influence of training hyper-parameters on Convolutional Neural Networks (CNN) test accuracy using Sensitivity Analysis (SA). The results of the SA will produce a general ranking of influence that will be able to inform the reduction of the parameter search space during Hyper-Parameter Optimisation (HPO) in an effort to increase the efficiency of the process without compromising model performance. Additionally, a novel metric, Accuracy Gain, was developed to better estimate tuning efficiency and facilitate the comparison of parameter group performance.

The methodology of this research can be summarised in three parts. Firstly, the creation of a framework for SA of Deep Learning (DL) models, SADL, which perform two state of the art SA methods, Sobol Indices and Morris Method, on CNN hyper-parameters. The resulting sensitivity measures indicating hyper-parameter influence produce a ranking that informs which parameters should be targeted during HPO. Bayesian Optimisation was performed for parameter groups of various influence, and the accuracy gain metric calculated for each to quantify tuning efficiency. Finally, these results were applied to a real world scenario in a case study on the classification of colo-rectal cancer images.

They key findings of this work were the development of a robust framework of SA applied to DL and that it is possible to provide empirically based guidance on which parameters to optimise. The SA highlighted batch size, learning rate decay and learning rate decay step as most influential, where batch size was significantly more influential than other hyper-parameters. Conversely, learning rate did not achieve the influence rank expected based on the literature. Tuning a subset of influential parameters was more efficient than tuning all parameters, which was confirmed in the case study where tuning the top three parameters was quicker and achieved higher accuracy than not only all training parameters but was also a significant improvement on the parameters explored in the original work.

The implications of this work for practitioners are that they can use this information to guide hyper-parameter tuning efforts, reducing the parameter search space to work within time and resource constraints without compromising model accuracy. Ultimately, these results facilitate the efficient development of optimal DL models. Furthermore, this work provides a framework and clear methodology that future work in this area can follow. Future directions of this work would focus on expanding the scope with additional model architectures, training datasets, hyper-parameters and performance metrics.

Keywords: Sensitivity Analysis, Sobol Indices, Morris Method, Deep Learning, Hyper-parameter tuning, Ranking hyper-parameters, Bayesian Optimisation, Convolutional Neural Networks, Framework

Acknowledgements

I would like to thank the following people, without whom I would not have completed this research and without whom I would not have made it through my PhD. Thank you to my supervisor, Pat Parslow, for supporting me on my PhD journey and providing guidance and feedback throughout this project. Thanks also to my partner, Ed, for being there for me during this turbulent time, for always providing a shoulder to cry on and being my number one cheerleader. I also could not have done this without the support of my family so thanks to my mum, Helen, dad, Ian, sister, Megan and grandad, Hopkin. I don't know how you put up with me for the past few years but I am so grateful to all of you. No matter the outcome I hope I have made you proud. To my friends, Megan and Amy, thank you for for accepting my weird schedules and believing in me when I didn't believe in myself. I would also like to thank the IT team in the department, Nick Gurr and Anthony Worall. Without their support and expert knowledge my work would not have been completed in time.

Dedication

This is for all the women who are fighting to to make their contributions known.

We have a voice, our work is worthy and we are just as capable as any man.

Don’t be silenced.

Contents

List of Figures

List of Tables

List of Abbreviations

SA	Sensitivity Analysis
DL	Deep Learning
ML	Machine Learning
HPO	Hyper-parameter Optimisation
LSA	Local Sensitivity Analysis
GSA	Global Sensitivity Analysis
OAT	One-at-a-Time
FAST	Fourier Amplitude Sensitivity Test
ANOVA	Analysis of Variance
EE	Elementary Effects
SOTA	State of the Art
RSA	Regional Sensitivity Analysis
STEM	Science, Technology, Engineering and Mathematics
FNN	Feed-forward Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Netowrk
MLP	Multi-Layer Perceptron
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
RNN	Recurrent Neural Network

LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
DRNN	Deep Residual Neural Network
SADL	Sensitivity Analysis Deep Learning Framework
RA	Robustness Analysis
NSATP	Noise Sensitivity Analysis Test Prioritisation
AI	Artificial Intelligence
LRP	Layer-wise Relevance Propogation
PaD	Partial Derivatives
SVM	Support Vector Machine

Chapter 1

Introduction

The advent of high-performance computing has facilitated the growth observed in the fields of Machine Learning (ML) and Deep Learning (DL) with Neural Networks (NN) growing in size and complexity to solve a vast array of problems. A major challenge facing practitioners in these fields surround the efficiency of Hyper-Parameter Optimisation (HPO), a time and resource consuming process to find optimal settings for model parameters.

Although there are resources which explain hyper-parameters and the importance of the role they play within the network, there is no definitive ranking of those parameters from most to least influential that practitioners can use to guide them through the HPO process. This research aims to use Sensitivity Analysis (SA) to understand and rank hyper-parameter influence on DL model accuracy and show that the HPO process can be more efficient when concentrated on the most influential parameters.

This chapter will provide an introduction to this work by firstly providing some background and context, stating the aims, objectives and research questions and presenting the solution developed.

1.1 Background

DL models are being applied in all manner of industries from healthcare to security to agriculture. Once a model has been optimised for the designated purpose they can be extremely efficient, however getting to that point is a time consuming process. Hyper-parameter selection is a significant part of model optimisation. A popular approach is to tune based on expert knowledge, manual tuning, which often results in a workable, sub-optimal solution (4). This is the most time efficient approach as it is a decision made by the practitioner, however if any tuning algorithm were to be employed, such as Bayesian Optimisation, then this becomes time consuming. Furthermore, there seems to be little action that can be taken to make this more efficient without compromising the performance of the model. This is supported by the No Free Lunch theorem, that no general-purpose optimisation strategy is possible (5), suggesting that HPO can only be optimised for the specific situation it is being used in, and there can be no general guidance that could improve HPO of NNs.

The hyper-parameters of NNs themselves are the parameters which cannot be modified once training has begun, as such HPO can be considered the final step of model design, where the parameters effect model structure, and the first step of model training, where the parameters effect learning speed and model accuracy (4). The automation of this process simply trades the human effort expended in manual tuning for computational resources and effort, which can be extensive if the parameter search space is large. However, for models to perform optimally and achieve high accuracies this task is unavoidable, clearly indicating a need for research in this area.

Sensitivity Analysis has seen much success in its application in many industries, similar to the use of DL models. Despite its success, SA is often overlooked or not performed in a way to maximise the quality of the findings (6). A decade long review of SA saw that the majority of SA literature relates to the medical field, followed by chemistry whereas the adoption in mathematics and computer science remained relatively low during that time (7). Though SA has been applied to the DL space, it is often to the input data features or, where applied to the hyper-parameters the methods used leave the results open to the interpretation and bias of the researcher. Applying the state of the art SA methods, Sobol Indices (8) and Morris Method (9), would produce reproducible, quantifiable results that could contribute towards a generalisable understanding of hyper-parameter influence on DL model.

1.2 Problem statement

This section highlights the gap in current literature that motivated this study, outlines the aims and objectives of the work and confirms the scope of the research.

1.2.1 Motivation

The consistent growth of DL problems and solutions is increasing pressure on HPO approaches, creating a potential bottleneck in the process of creating high-performing models (10). The lack of scalability of current methods means that additional levels of complexity and dimensionality will only worsen this tuning bottleneck. A reduction of the search space could reduce the resources and time required for tuning or target the tuning more specifically to increase the likelihood of finding an optimal solution. There is little guidance for this reduction of parameter search space outside of the concept of manually choosing which parameters to apply automated HPO to based on expert knowledge. If a parameter ranking existed which quantified the influence of each parameter on model accuracy, this could clearly show practitioners the value of directing their tuning in a specific direction.

Current applications of SA applied to DL focus on the input space and reducing dimensionality akin to PCA (11, 12, 13, 14). Where there are works that look at the model hyper-parameters the methods used are less formalised, looking at the average accuracy (15), inference from plots which cannot account for parameter groups (16, 17) or the focus of the SA was so context specific the findings were not easily generalisable to other applications. In terms of influential parameters, the most common existing guidance is to prioritise the learning rate when tuning (18). As a result, there is limited understanding of parameter influence on model performance and inadequate guidance on reducing the parameter search space for HPO. DL practitioners, especially beginners, find themselves ill-equipped to complete efficient HPO that results in a high-performing model in this ever-changing, complex ML landscape.

1.2.2 Aims and objectives

The aim of this work concentrates on producing a general ranking of Convolutional Neural Network (CNN) training hyper-parameters via SA that could inform HPO search spaces to improve tuning efficiency. The Research Objectives (RO) identified to achieve that aim are listed below:

- RO1: Create a framework to facilitate the calculation of sensitivity measures for the hyper-parameters of DL models which can be used for future work.
- RO2: Rank the influence of a hyper-parameter on model performance, taking into consideration two state of the art sensitivity measures.
- RO3: Discover relationships between parameter influence and network architecture or input data.
- RO4: Demonstrate any potential of reducing the hyper-parameter search space to influential parameters for HPO.
- RO5: Produce measure of hyper-parameter tuning efficiency that considers computation time and model accuracy to facilitate comparisons.
- RO6: Measure hyper-parameter tuning performance of influential parameter groups versus other parameter groups.
- RO7: Apply findings in case study to demonstrate real world scenarios and validate results.
- RO8: Provide guidance and a robust methodology to machine learning and deep learning practitioners in choosing what hyper-parameters to tune and the application of SA to DL.

1.2.3 Research Questions

This work aims to answer the following Research Questions (RQ):

- RQ1: Is it possible to quantify and rank the influence of a hyper-parameter on model performance using SA?
- RQ2: Are there any relationships between parameter influence and network architecture or input data?
- RQ3: Is it possible to make HPO more efficient, without compromising model accuracy, by reducing the parameter search space to the most influential parameter?
- RQ4: Can these theories be successfully applied to a real world scenario?

1.2.4 Scope

Due to time and resource constraints the scope of this work, which has the potential to be extremely broad, had to be narrowed so that it could be completed in time but still produce results that are generaliseable. Of the DL model possibilities, the CNN architecture was chosen as it is popular in the literature and widely adopted. Similar reasoning was behind the decision to concentrate on image classification data, Bayesian Optimisation HPO method and Sobol Indices and Morris Method approaches to SA. The hyper-parameter scope was reduced to the training parameters as state of the art model architectures would be used. The final decision was to explore hyper-parameter influence on model test accuracy, rather than another performance metric, as this gives the best indication of model accuracy and generalisability.

1.3 Solution

The solution and experimentation designed to answer the above research questions and fulfil the aim of this work is broken down into three distinct areas:

1. SA of CNN hyper-parameters
2. Rank informed Bayesian Optimisation
3. Case Study

1.3.1 SA of CNN hyper-parameters

The solution to apply SA to CNN hyper-parameters centres on the creation of a framework for SA of DL models, SADL. This includes sampling of the parameter space, compiling and building the architecture, training, recording the parameter settings and model performance and calculating the sensitivity measures for both Sobol Indices and the Morris method. Finally, both measures are combined to produce a generalised ranking across several CNN architectures and image classification datasets.

1.3.2 Rank Informed Bayesian Optimisation

The second element of the solution takes the resulting ranking from the SADL framework and uses it to inform parameter groups for tuning. Bayesian optimisation is conducted on influential parameter groups and compared to the performance of tuning all training parameters. A novel metric, Accuracy Gain, was developed to quantify the tuning efficiency and facilitate the comparison between parameter groups.

1.3.3 Case Study

The final stage of the solution applies the findings from part one and two to a real world scenario. A study will be replicated, using the same dataset and model architecture and tuning will be conducted on the parameters reported in the original paper, all parameters and influential parameter groups. The tuning efficiency of these groups will be compared to determined whether the influential parameter groups improve on the results reported in the original work. This is done to increase the robustness and validity of any findings.

1.4 Summary of contributions and achievements

The chronological contributions and achievements of this thesis are as follows:

1. Creation of the SADL framework for the application of SA to CNN hyper-parameters.
2. Successful use of the SADL framework to quantify hyper-parameter influence on CNN accuracy.
3. Production of generalised rank of hyper-parameter influence on CNN accuracy and publication of early results in the IEEE conference (ICTAI) (19).
4. Novel measure, Accuracy Gain, which quantifies tuning efficiency.
5. Guidance on which hyper-parameters to tune for practitioners.
6. Theoretical framework for conducting experimentation of SA applied to DL.

Firstly, the SADL framework successfully quantified hyper-parameter influence producing a generalised ranking for CNN hyper-parameters, the early results of which were published in the IEEE International Conference on Tools of Artificial Intelligence (ICTAI) (19). Batch size, learning rate decay and learning rate decay step proved highly influential whereas learning rate did not live up to the reputation of its importance conveyed in the literature.

A novel measure, Accuracy Gain, was developed to quantify tuning efficiency and showed that HPO conducted on influential parameter groups was more efficient, saving time without compromising accuracy. This was confirmed by the case study where the influential parameter group outperformed the parameters reported in the original work whilst also completing HPO in less time.

These results challenge the No Free Lunch theory, as they show a general way for improving HPO for CNN architectures. The implication here is that practitioners can make an informed decision on reducing the parameter space to the most influential parameters, knowing that this should be the most efficient approach to producing a high performing CNN model.

Finally, a general contribution is a robust approach and framework for conducting future work exploring SA applied to DL.

1.5 Organisation of the report

The next chapter of this thesis will present a literature review of relevant recent works, followed by an in depth presentation of the methodology. The next three chapters will present the three areas of the solution developed, the SA of CNN hyper-parameters, rank informed Bayesian Optimisation and the application of the findings in a case study. The results of these three areas are then discussed, culminating in the conclusion of this thesis.

Chapter 2

Literature Review

Sensitivity Analysis (SA) has the potential to shine a new light on Deep Learning (DL) models, offering a new perspective which could aid in DL explainability.

SA has a long history in areas such as environmental modelling and yet it is only now making its debut in areas such as Machine Learning (ML) (20). By definition, SA allows for better understanding of uncertainty in a systems outputs in terms of uncertainty in the system inputs (21). Various SA methodologies exist that allow for the quantification of input influence on a system with potential that is yet to be fully realised.

DL is an ever-evolving discipline within ML where continuous challenges are emerging at a rapid pace. Major challenges facing DL practitioners and researchers include the computational power and resources required, the 'black box' nature of DL networks, design complexity and the high computational cost associated with hyper-parameter tuning (22).

Hyper-parameter tuning is a resource intensive process, the cost of which is growing alongside the size of the DL models and datasets. The more complex the DL task the longer the tuning process will be. This presents a major challenge to DL practitioners as tuning is an essential step to maximising model performance and currently there is little guidance on increasing the efficiency of this vital process.

This chapter explores the potential of SA applied to the hyper-parameter space of DL models. A systematic review of the use of SA in machine learning was conducted and areas of improvement identified. Finally, potential applications are explored.

2.1 Sensitivity Analysis

SA has been a technique that has been applied in various sectors for decades and can be defined as an analysis of the uncertainty in a systems output and the relation to the uncertainty in the systems inputs (21). This basic premise of SA is represented in Fig. 2.1. For a given system or model, the SA method constructs a set of inputs from a given parameter space and then uses the outputs that the system generates from each input to quantify the influence of inputs on those outputs. Box (III) in Fig. 2.1 represents the importance assigned to each input through SA. Despite its history, SA has only recently gained visibility and status as an essential tool in areas such as environmental modelling and is making its debut in areas such as machine learning (20).

Two popular approaches to SA are variance-based methodologies, such as Sobol Indices, and screening-based methodologies, such as the Morris or Elementary Effects Method. Variance-based SA quantifies the uncertainty in a systems inputs and outputs as probability distributions and decomposes the variance in the output and attributes it to the inputs (8). On the other hand, screening-based SA methods rely on the sampling of input values to identify contribution to output and tend to require less computational resource compared to variance-based methods (9).

A recent review of the current state of SA from the perspectives of various researchers identified machine learning and DL as an area of growth for SA (1). They identified three areas of compatibility where SA and DL could be conjoined;

1. Feature Selection
2. Model Interpretability
3. Machine learning powered SA

The second area identified, model interpretability, suggests not only considering the effect of the features in the input data but also taking into consideration the structure of the network and taking a data-driven, "process-informed" approach to parameter settings. Exploring this in relation to its potential in a hyper-parameter tuning context is the aim of our work.

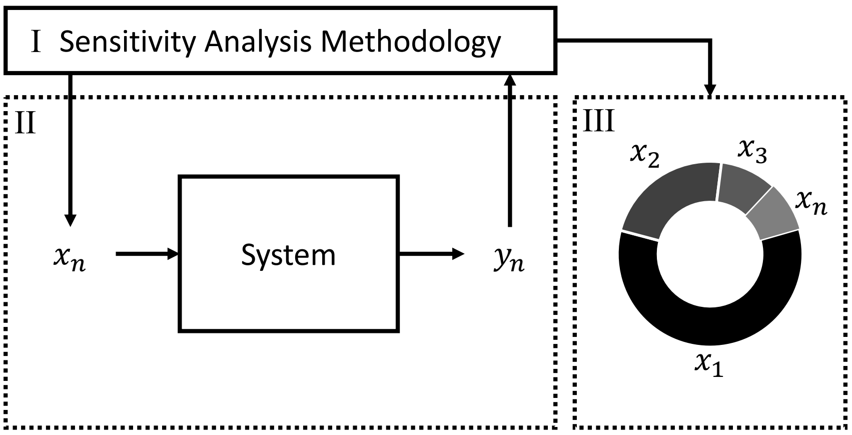


Figure 2.1: A high-level SA workflow. Box (I) represents an SA methodology that generates system inputs based on defined search space, $x_1, ..., x_n$ and receives outputs y_n. Box (II) represents the system being analysed. Box (III) represents an outcome of SA. The influence of each input on the output is quantified. (After Razavi (1))

2.1.1 Local vs Global

There are two main themes within Sensitivity Analysis (SA); Local (LSA) and Global (GSA). The various SA methods fall within these categories which are used to highlight the parameters encompassed in the analysis.

LSA most commonly refers to one-at-a-time (OAT) methods (23) where a single parameter is chosen for exploration. A set of possible values for the chosen parameter is created and they are 'local' in that they come from a neighbourhood of potential, realistic values. Analysis is completed by iteratively running the model, varying the value given to the chosen parameter and noting the effect on the output. LSA offers an understanding of the importance of a single parameter, lending insight into the effects of specific points of the model. Generally, derivative based sensitivity measures and the first-order sensitivity index are considered to be LSA. LSA has its advantages; it is often quicker to compute and easier to implement than GSA. However, due to its nature, the results are heavily affected by the area of exploration in the feature space and this must be considered in their interpretation.

On the other hand, GSA methods are computed based on a sample of representative locations from the parameter's entire distribution (24). This provides a more general insight into model sensitivity and includes popular methods such as variance based SA and screening (23). To determine a more general sensitivity measure which can be considered global, the influence of a parameter is averaged on its own distribution and the distributions of all input parameters (25). A key aspect of GSA is its ability to quantify the importance of inputs (26) which allows modellers to rank and analyse them quantitatively. The most common global sensitivity measure is Sobol's total-sensitivity index. This is a variance-based method, which are generally considered to be the most sophisticated methods available (27). Global methods of SA are less vulnerable to type 2 errors, unlike LSA, as more of the parameter space is explored thereby reducing the potential for critical parameters or combinations thereof to be omitted from the analysis. A disadvantage of this approach is the computational cost associated with it. However, unlike traditional hyper-parameter tuning the insights gleaned from this analysis will provide a more general insight to parameter importance which can be applied repeatedly as opposed to expensive HPO applicable to a single, specific problem.

2.1.2 Sensitivity Analysis Methodologies

First-Order Indices and Total Effects

The output of a model, y, can be written as:

$$y = f(x_n) \tag{2.1}$$

$$S_i = \frac{(\delta y)}{(\delta x_i)}_{(x_n^*)} \tag{2.2}$$

In equation 2.1 x_n are the input parameters to the model where $n = 1, ..., k \quad n \in \mathbb{N}$ and k is the maximum number of input parameters. SA is traditionally applied to the input space of a model however in this thesis the hyper-parameter space will be considered. A sensitivity measure, S_i, is used to enumerate the effect of an individual parameter, x_i, on the model's output, y, and can be represented as a simple derivative where x_n^* is a specific base point, from which the effect of x_i on y can be calculated (27). It is recommended, however, by the Intergovernmental Panel for Climate Change that a normalised version of this measure is more appropriate (21):

$$S_i = \frac{(\sigma_{x_i} \delta y)}{(\sigma_y \delta x_i)} \tag{2.3}$$

where σ, the standard deviation of x_i or y, is used as a normalising factor. Calculating the sensitivity measure in this way ensures that x_i is normalised to one which provides a more consistent ranking of parameters and allows for the direct comparison of S_i for different values of x_i. Despite the additional consistency the normalised derivative measure offers, it can only give an indication of the influence of the parameter in question at the specific point where it was analysed. This cannot be generalised to indicate a parameters general effect on the model, this task requires a more thorough measure.

Our model was defined as $y = f(x_n)$, and so a model's variance can be written as $V(y)$. When a parameter, x, is fixed to a value, x_i^*, then the conditional variance can be written as:

$$V_{x\sim i}(V|X_i = x_i^*) \tag{2.4}$$

which gives the variance taken over all potential values, except x_i, which is represented as $V_{x\sim i}$. This gives an indication of the relative importance of x_i to the overall variance, $V(y)$. However, this measure is completely dependent on the position of x_i, a problem shared with the derivative based measure, and does not always yield a value of less than the total variance, $V(y)$, and so the contribution remains unclear. To overcome this the average of all potential values for x_i provides a more sensible measure:

$$E_{xi}(V_{x\sim i}(y|x_i)) \tag{2.5}$$

So that E is the expected values of x. Using this measure the conditional variance will always be less than $V(y)$ because:

$$V(y) = E_{xi}(V_{x\sim i}(y|x_i)) + V_{xi}(E_{x\sim i}(y|x_i)) \tag{2.6}$$

Therefore, it is possible to determine the contribution of a parameter to the variance in the model output. $V_{xi}(E_{x\sim i}(y|x_i))$ is known as the first-order effect of x_i on y and is formally known in the literature as the first-order sensitivity index:

$$S_i = \frac{V_{xi}(E_{x\sim i}(y|x_i))}{V(y)} \tag{2.7}$$

Calculating the fraction of the total variance that the first-order effect represents produces a number between zero and one which quantifies the importance of x_i. The higher the value of S_i the more the parameter under observation contributes to the model output. Unfortunately, in most cases, this measure is still too simplistic. Models have more than one hyper-parameter and the interactions of these hyper-parameters can often play a part in the outcome, however this is not reflected in the first-order sensitivity index. Additionally, to continue to calculate the sensitivity in this way for a high number of indexes would be incur a high algorithmic cost. To completely understand the sensitivity of a model then all terms would need to be computed; a long and arduous process considering there are higher orders of interactions that would need to be calculated for most models. To avoid the exponential complexity of computing indexes beyond for all interactions, a separate measure can be used to understand a parameters total influence, aptly named the total effects:

$$ST_i = 1 - \frac{V_{xi}(E_{x\sim i}(y|x_{x\sim i}))}{V(y)} \tag{2.8}$$

where $V_{xi}(E_{x\sim i}(y|x_{x\sim i}))$ represents the first order effects of $x_{\sim i}$, representing everything but x_i. Removing this from the total possible effects, 1, leaves the value that can be attributed to the total effects of x_i.

FAST

The Fourier Amplitude Sensitivity Test (FAST) was created by Cukier et al as a variance-based method for SA(28). They developed this methodology to better understand the sensitivity of a models output to the uncertainty in the input parameters. The aim was to propose a technique that was more time efficient than OAT, brute force methods. Model outputs undergo Fourier analysis where the Fourier coefficients represent the output average over all input variations, where each parameter is assigned a frequency, w. The sampling approach used in FAST is similar to trajectory based search in that a search curve is constructed. In the literature, the main criticisms of FAST, despite its efficiency(27), is its difficulty to encode and that higher-order indices are not able to be calculated, basing the sensitivity on first-order effects only(21). However, in this methods partial variance measure, S^*_{wi}, we see the inspiration of what will become one of the most popular measures for SA: Sobol's Indices.

Sobol Indices

Sobol's Indices are variance-based sensitivity measures and provide global insight into the sensitivity of a model. Variance-based measures are considered to be state of the art in the literature as they are model independent, consider parameter interactions whilst representing the global search space(21). The main flaw with this methodology is the computational cost associated with calculating the variance-based sensitivity measures in addition to the cost of a number of model simulations. Sobol's Indices are the first-order effects (S_i), equation 2.7, and the total-effects (ST_i), equation 2.8, explained earlier in this thesis. By considering the decomposition of variance in a models output as an ANOVA (Analysis of Variance) decomposition(8) Sobol aimed to rank parameters based on their effect on the variance of the output of a model.

To compute Sobol's Indices using a Monte Carlo method the following approach(29) can be taken:

Algorithm 1 Sobol's Indices Monte Carlo Approach

```
$A \leftarrow RandomSample(n)$
$B \leftarrow RandomSample(n)$
$f_0 \leftarrow EstimatedMean(A, B)$
for $i = 1, ..., n$ do
    $C_i = (B_1, ..., B_{i-1}, A_i, B_{i+1}, ..., B_k)$
    $y_A = f(A)$
    $y_B = f(B)$
    $y_{C_i} = f(C_i)$
    $S_i = \frac{y_a \cdot y_{c_i} - f_0}{y_a \cdot y_a - f_0}$
    $ST_i = 1 - \frac{y_B \cdot y_{c_i} - f_0}{y_a \cdot y_a - f_0}$
end for
    return $(S_i, ST_i)$
```

where n is the number of samples and the estimated mean of (A, B), f_0, is calculated as:

$$f_0 = \frac{1}{n} \sum_{j=1}^{n} y_K^i \tag{2.9}$$

where K is a placeholder for either A, B or C_i depending on which calculation is taking place. A and B are matrices comprising of n randomly selected points from the sample space. C_i is a matrix formed from B except for the i^{th} column which comes from A. The model output is then computed for each of the matrices; A, B and C_i, which is then used along with the mean, f_0, to calculate the sensitivity measures.

Morris Method

The Morris Method (9) or Elementary Effects (EE) method applies local SA across the feature space, Ω, to create a global measure and is considered to be a screening method. The aim is to determine the effect of input parameters to a model whether they be negligible, linear and additive, nonlinear or involved in interactions with other parameters (21). The input space can be considered to be a grid of p-levels with n, the number of model inputs, dimensions. Knowing this, the EE for a given input, x_i, can be calculated as:

$$EE_i = \frac{y(x_1, ..., x_{i-1}, x_i + \Delta, ..., x_n) - y(x_1, ..., x_n)}{\Delta} \tag{2.10}$$

where Δ is the step-size chosen from $(\frac{1}{p-1}, ..., 1 - \frac{1}{p-1})$. The distribution of EE is denoted as $EE_i \sim F_i$ and is obtained through the random sampling of x values from Ω. A useful example given in Sensitivity Analysis a global primer (21), visualises Ω where $n = 2, p = 4$ and $\Delta = \frac{2}{3}$. As dictated in the example, there are two parameters being analysed, x_1 and x_2, with 4 levels, p, and the intervals are $\frac{2}{3}$, as given by the value of Δ. This example is easily visualised as it is 2-dimensional, when n is large then the dimensionality of Ω also increases. There are two sensitivity measures associated with the EE method: μ and σ. The measure μ represents the overall influence of a parameter, whereas the measure σ provides a degree of independence. Both sensitivity measures, μ and σ, range between 0 and 1 with a higher value of μ signifying a higher level of sensitivity to the parameter. A higher value of σ indicates that the influence of the parameter is independent of other parameters. To compute the sensitivity measures for x_i then EE can be defined as;

$$EE_i^j(x^l) = \frac{y(x^{l+1}) - y(x^l)}{\Delta} \tag{2.11}$$

where j is the trajectory of parameter space exploration and l is the sample point. Having calculated the EE for each trajectory the sensitivity measures are calculated as;

$$\mu_i = \frac{1}{r}\sum_{j=1}^{r} EE_i^j \tag{2.12}$$

$$\sigma_i = \sqrt{\frac{1}{r-1}\sum_{j=1}^{r}(EE_i^j - \mu)^2} \tag{2.13}$$

where EE_i^j is the relative EE for the parameter whose sensitivity is being calculated, i, along trajectory, j, for a given number of samples, r.

Modified Morris

Campolongo et al proposed a modification to the traditional Morris Method where an additional sensitivity measure was introduced (30). The aim was to allow the EE method to handle groups of parameters, producing a total sensitivity measure, and combatting Type 2 errors, which the μ measure is prone to. The proposed modified measure, μ^*, is the mean of the distribution of absolute values, $|EE_i^d(x)| \sim G_i$. The absolute EE for a group, $u = (x_{i1}, x_{i2})$, is calculated as;

$$|EE^{d_u(x)}| = \frac{|y(x) - y(x)|}{\Delta} \tag{2.14}$$

So the measure μ^* can be written as:

$$\mu_i^* = \frac{1}{r}\sum_{j=1}^{r} |EE_i^j| \tag{2.15}$$

This method was tested against variance-based measures in Campolongo's paper as they are considered to be state of the art in terms of SA methodologies. Comparisons were made through obtaining μ^* and ST_i for some models and then comparing the ranking of parameters determined by the separate methodologies measures which determined that μ^* was an effective substitute for ST_i. This is promising in terms of efficiency as computing μ^* does not change the time complexity of the method and can be computed alongside the original EE sensitivity measures. Variance-based measures are known to be computationally expensive, especially with larger models and therefore having a substitute measure for total sensitivity that is possible to calculate more efficiently is an advantage.

Regional Sensitivity Analysis

The purpose of Regional Sensitivity Analysis (RSA) (31) is to identify regions in the input space which correspond to particular values in the output. From this it is possible to produce a mapping, for example parameters with these values produce outputs above a certain threshold, which can be used to better understand model behaviour. An advantage of RSA is that it can be applied to non-numerical outputs, widening its range of potential applications (2).

Correlation and Regression Sensitivity Analysis

The basis of these methods of SA derive from statistical analysis of the input and output to a model or system. In the literature they are often used in conjunction with one another.

Correlation-based SA makes use of existing statistical measure of correlation such as the Pearson correlation coefficient and Spearman rank correlation coefficient. Furthermore, this form of SA is usually assessed visually through the use of charts such as scatter plots which provide a visual indication of the influence of parameters based on the shape produced and can aid in the identification of complex relationships. As a result, producing scatter plots to understand any correlations is usually taken as an initial step in any SA (32).

On the other hand, Regression-based SA takes a probabilistic approach (33) to describe the relationships between parameters and presents a simple method of global SA. The method consists of regressing the output parameters with respect to a set of input parameter forming a regression model. The estimated outputs of the regression model are described in terms of linear combination of the input parameters producing a sensitivity measure in the form of the standardised regression coefficient (34).

2.1.3 Sensitivity Analysis Settings

The "setting" of the SA is the formal definition of the SA objective, the reason behind investigating the parameters. Having a clear idea of the goal of the SA is key to avoiding inconclusive results. The definition of a "setting" in terms of SA is given by Saltelli et al. (21) as "a way of framing the sensitivity analysis quest" and there are three popular settings:

1. Ranking (Factor Prioritisation)
2. Screening (Factor Fixing)
3. Factor Mapping

The first setting proposed by Saltelli et al. was Factor Prioritisation, or "ranking" as it will be referred to in this thesis to simplify models. This setting identifies which parameters can be fixed to an arbitrary value within a range without affecting the output. This offers the opportunity to reduce the parameter space to the influential parameters only.

The third setting was Factor Mapping, referred to as simply "mapping" in this thesis. This setting concentrates on a particular region of the output which could be defined by

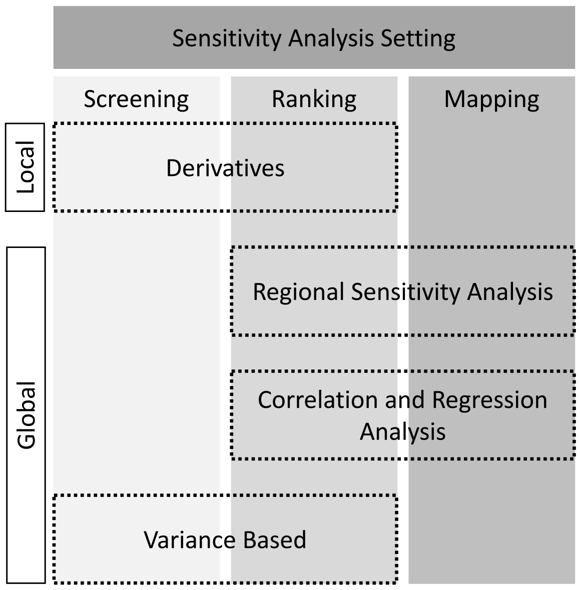

Sensitivity Analysis Setting
Screening
Ranking
Mapping
Local
Global
Derivatives
Regional Sensitivity Analysis
Correlation and Regression Analysis
Variance Based

Figure 2.2: SA methods in relation to Sensitivity Analysis Settings they are associated with and highlighting whether they are local or global methods. For more information on the SA methods see Section 2.1.2 . (After Pianosi (2))

some threshold. This threshold is often used to define a desirable and undesirable outcome by which the various model outputs can be grouped. A mapping is created between parameter values and outputs which makes it possible to see what values of the parameter lead to a desirable output. This setting is usually applied to influential parameters identified by earlier settings.

To understand where the SA methods described in Section 2.1.2 fall within the larger context of SA settings Fig 2.2 was created, a more complex version can be found in Pianosi's paper (2). As shown in Fig 2.2, all of the SA methods explored address the ranking setting, whereas half also overlap with either screening or the mapping setting. Furthermore, the majority of the SA methods were global. SA settings will colour the application of SA to DL hyper-parameter tuning as they describe the motivation of the SA and the results vary based on the setting chosen.

2.1.4 Current Applications of Sensitivity Analysis

SA has been applied in several fields in Science, Technology, Engineering and Mathematics (STEM). The varied use of models in STEM make it the perfect application area for SA.

The importance of SA is demonstrated in the field of Biomedical Sciences where mathematical models are frequently used for hypothesis testing of biological systems. There are software packages that have been built to conduct SA on biomedical data such as Dakota (35) which was created to apply SA on immunology data. SA methodologies have been applied to parameters identified as causing cancer and showed that cell division and mutation rate caused the most variance in model output suggesting the most affect on the cancer being analysed (36). The SA findings were supported by physiological evidence, reflecting the power of SA in the Biomedical field.

Within the Engineering discipline of STEM, SA is often applied to building systems. An extensive review of SA in building performance analysis was conducted in 2019 which identified several applications of SA including building design, building evaluation and model calibration (37). Similar to DL models, the challenges associated with building performance analysis surrounded increasing complexity, the "curse of dimensionality" which made it difficult for practitioners to calibrate and work with the models to produce intelligent building specifications. SA was adopted in this area as it combats these challenges by enabling model simplification.

Environmental modelling is a key application area of SA in the literature, with a long history of utilising SA to better inform modellers. Complex models are used within this subject to produce simulations of various environmental scenarios. Similarly to the building performance analysis models, calibration is a key aspect of the environmental models where SA is applied to provide insights into the influence of uncertain parameters on a performance metric (2).

The growing popularity of SA has led to the development of software tools to facilitate the application of SA. An overview of these tools was compiled by Douglas-Smith et al. (38) where they highlighted SA applications frameworks organised by the programming language they were developed in. The three packages that implemented the most varied SA methods were SimLab (Matlab), R sensitivity (R), and SALib (Python). All three packages have implementations of the Morris Method, Sobol Indices and FAST. As these three languages are also popular in ML and DL methods these SA packages present usable options to DL practitioners to conduct SA.

2.2 Deep Learning

As a result of the advent of high-performance computing DL has grown in popularity. The success of DL can be attributed in part to their ability to process large amounts of data with high dimensionality (39) and their potential to be applied to any domain (40). DL has evolved into a complex field with several areas of research contained within it. Prominent areas of exploration are DL architecture, popular approaches being Feed-forward Neural Networks (FNN) and Convolutional Neural Network (CNN), hyper-parameter tuning methodologies and the hyper-parameters themselves.

2.2.1 Deep Learning Network Architectures

Feed-Forward Neural Networks

The FNN is “the quintessential deep learning model” (18). The premise of the FNN is that it is composed of layers of neurons and the output of a layer becomes the input to the next layer. The prime example of a FNN is a Multi Layer Perceptron (MLP) or simple Deep Neural Network (DNN). FNN’s presented many advantages over traditional machine learning methods such as their ability to adapt to the problem without user interference and their ability to cope with non-linearity in the input data. However, training times were high and the black-box nature of FNNs were clear disadvantages of the models (41). They also offered an improvement over statistical models as they made no assumptions regarding data distributions and required no hypothesis to test (42). The recent improvements to FNNs compared to earlier attempts can be attributed to two factors (18):

1. Larger datasets available for training, aiding model generalisation.
2. Greater computational resources, allowing for larger models.

The applications of FNNs are vast with many industries adopting them to solve complex problems including science, finance and security (42). Medicine has adopted FNNs in several areas such as cancer recognition (43, 44, 45, 46), medical signal processing (47, 48), heart disease diagnosis (49, 50) and modeling depression (51). The potential of FNNs in the medical field is well documented, and with the progress within DL and computational resources constantly developing the applications of deep learning are also diversifying.

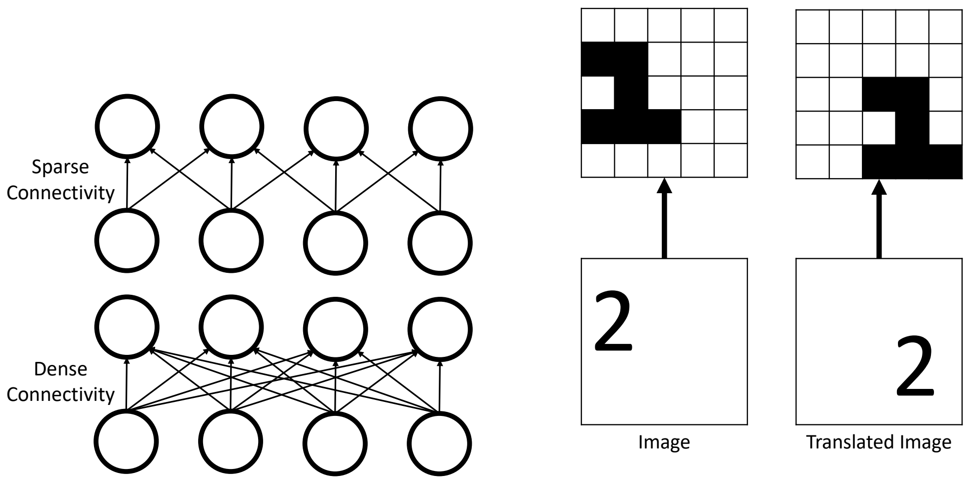


Figure 2.3: Pictorial examples of the three features utilised in CNNs: Sparse Connectivity, Parameter Sharing and Equivariant Representations.

Convolutional Neural Networks

A more complex example of an FNN is a Convolutional Neural Network (CNN) which operates similarly to a human visual processing system (40). As a result CNN architectures are popular in a sub-field of DL, Computer Vision. CNNs handle variance in input training examples better than FNNs, which would require additional training to recognise images that only vary slightly such as handwritten numbers (52). There are three features of CNNs that make them an improvement compared to the FNN architectures (18):

1. Sparse Interactions.
2. Parameter Sharing.
3. Equivariant Representations.

CNNs do not comply to the traditional fully connected architecture of typical FNNs. This is an improvement as it reduces the memory requirements to store the model and increases the efficiency of the model by reducing the number of operations needed. Parameter sharing also reduced the memory requirements of a CNN architecture compared to an FNN by learning one set of parameters rather than separate parameter sets at each location in the network. Another benefit of parameter sharing is that the layers become equivarient to translation, meaning changes in the output are consistent with those in the input. Examples of these features are visualised in Fig.2.3.

A timeline of developments in CNN architecture is shown in Table 2.2.1, starting in 1989

Table 2.1: Influential CNN Architecture Timeline

1989	ConvNet
1998	LeNet (53)
2012	AlexNet (54)
2014	GoogleNet (55), VGG (56)
2015	ResNet (58)
2016	DenseNet (61)
2018	Channel Boosted CNN (59)

with the original CNN architecture ConvNet. Almost a decade later LeCun et al. developed LeNet (53), a basic CNN which consists of seven layers. Ahead of its time, LeNet was limited by the computational resources available and so the potential of these DL approaches were not fully appreciated. Despite the existence of these early architectures CNNs are considered to have truly taken off in 2012 with the AlexNet architecture (54). AlexNet made history achieving state of the art accuracy compared to traditional ML approaches on the ImageNet dataset in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This demonstration of CNN potential can be considered the advent of the interest in DL, leading to more rapid developments. In 2014, the top two architectures that competed in ILSVRC were GoogleNet (55) and VGG (56). GoogleNet was built with the aim to conserve computational resources, and, despite being deeper than earlier architectures, was comprised of fewer parameters reducing its complexity (57). Runner up to GoogleNet, VGG demonstrated the importance of network depth in classification accuracy of CNN architectures. The 2015 winner of ILSRVC was ResNet (58), an ultra-deep network that overcame the vanishing gradient problem (57). As suggested by its name, DenseNet consists of densely connected CNN layers increasing the efficiency of feature reuse resulting in a reduction of network parameters (57). 2018 saw the introduction of the Channel Boosted CNN (59) which aimed to exploit transfer learning capabilities of CNNs and their channel dimensions. This architecture specifically targeted churn data which is an increasing issue in many sectors as a result of big data (60).

The popularity of these architectures suggested they would be a good application area of SA as better understanding how hyper-parameters effect the performance of this architecture type would benefit a high-proportion of DL practitioners across a variety of application areas.

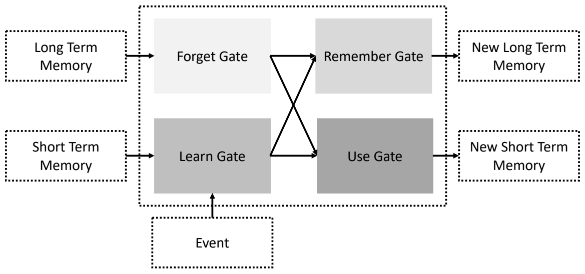


Figure 2.4: High-Level representation of LSTM architecture. (Based on Udacity Deep Learning video tutorial created by Luis Serrano)

Recurrent Neural Networks

Another popular DL architecture is the Recurrent Neural Network (RNN). A limitation of both FNN and CNN architecture is the lack of persistence in the learning process. When humans learn, it is a continuous process that is coloured by various personal experiences. When humans approach tasks we do not reset our brains and start from scratch. The RNN architecture is designed with this concept in mind, attempting to replicate a more human learning process by implementing a system that allows operations over time (57). There are two popular RNN architectures: the Long Short Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

Introduced in 1997, LSTMs are a gradient-based approach to DL over extended time intervals (62). The problem with conventional RNNs is that they are affected by the vanishing gradient problem which is addressed in the LSTM architecture by allowing constant error flow. Gate units are employed to protect the contents of the memory cells and network units from perturbations, be they irrelevant inputs or irrelevant memory contents. A high level representation of this concept is shown in Fig. 2.4.

The GRU architecture was proposed in 2014 by Cho et al. (63) as an approach for machine translation tasks. As shown in Fig. 2.5, GRU networks are similar to LSTMs, however they do not have separate memory cells. A comparison of traditional RNN, LSTM

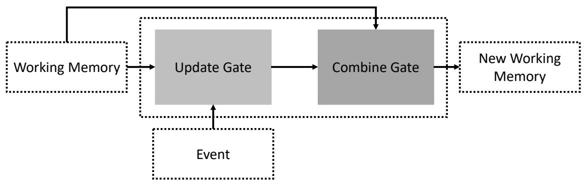


Figure 2.5: High-Level representation of GRU architecture. (Based on Udacity Deep Learning video tutorial created by Luis Serrano)

and GRU performance showed a clear advantage of gating units over traditional recurrent units whereas the dataset would influence whether the LSTM or GRU would have better performance (64).

Deep Learning Hyper-parameters

There are hyper-parameters associated with machine learning whose reputations precede them. Activation functions are generally considered to be the most important hyper-parameter as they allow models to solve more complex, non-linear problems (65). The impact of choosing the optimal activation function can have a significant impact on model performance. When comparing five activation functions on two MLP's, a 10-layer and a 40-layer, the accuracy ranged between 30%-95% and 19%-99% highlighting the influence of the activation function (66).

The importance of learning rate is also well documented. Too big a learning rate can be responsible for poor generalisation and an unthorough exploration of the error landscape (67). Additionally, the positive correlation between learning rate and a models ability to generalise was found to be statistically significant (68) further supporting the importance of the learning rate in training DL models. Bengio states in their book Deep Learning (18) that the learning rate significantly affects model performance, making it difficult to set as the setting chosen can be the difference between the success and failure of the model. These hyper-parameters live large in the literature and are, generally, the main targets of hyper-parameter tuning.

An ANOVA analysis of how hyper-parameters effect the accuracy of residual neural networks concurred that learning rate was amongst the most important parameters. However, it also showed that for a certain range of learning rate values there was little effect on the model suggesting that there may be other factors at play (69). Weight decay and momentum were also highlighted by the ANOVA analysis further illustrating that other parameters may be playing more of a role in deep learning models' ability to learn. Implementing a structured analysis of parameter influence would highlight whether some parameters are potentially being overlooked during tuning in favour of activation function and learning rate as they are perceived to be the most influential.

SA is used in this research to better understand the influence of specific hyper-parameters, producing rankings of the most sensitive to the least sensitive hyper-parameter. Thus, presenting the opportunity to use this ranking to inform practitioners to tune the hyper-parameters that models are most sensitive to, conserving resources.

2.2.2 Hyper-Parameter Tuning

Hyper-parameter tuning or Hyper-Parameter Optimization (HPO) is a critical step in the machine learning process and research into HPO approaches for DL is increasing in popularity (4). As model size and complexity grows ever larger, the need for more efficient HPO that finds optimal parameter settings is demanding the attention of researchers. There are three popular approaches to HPO in the literature; grid search, random search and Bayesian optimisation (70).

Manual Search

There are various techniques that have been applied to HPO. The simplest approach being manual search which consists of the practitioner manually setting the parameter values. Despite the ease of manual search, it does have many drawbacks. The first being that it relies completely on the expertise of the practitioner which, in the case of an expert, could result in good results but probably not the best that could be achieved. Additionally, it would be difficult to reproduce the results. Furthermore, the increasing complexity of DL problems can create difficulty in interpreting the hyper-parameters.

Grid Search

Grid search is a technique that is widely used as it is quick to implement and allows reproducible results as it systematically explores the parameter space. However, this systematic approach is responsible for the disadvantages of grid search. It is inefficient at search parameter spaces with high dimensionality and the rigidity of the approach can result in the explorations of parameter values that are unimportant. Grid search is a victim of the curse of dimensionality, the more parameters there are to search the less efficient the method.

Random Search

Random search offers an improvement on grid search as it is more efficient (71). As the name suggests, this method implements a random search over the parameter space. The same issue surrounding computational resource persists with random search, as the search space increases in size the resources required to assess the parameter values increases. Paradoxically, the efficiency curve of random search shows that the larger the search space the better the results and so there is a trade-off between the probability of finding optimal settings and the efficiency of the tuning process. The introduction of random search also highlighted that only a subset of hyper-parameters actually affect model performance and that the efficiency of the method can be increased by removing the non-influential parameters from the search space (72).

Bayesian Optimisation

Bayesian optimisation aims to find a global optimum solution and an advantage of this approach is that the search space is influenced based on the results of earlier trials (4). A probabilistic model is used to determine the values that are explored for the hyper-parameter settings. As a result, Bayesian optimisation is more computationally efficient than grid search and random search as it requires less trials to find the optimum. Unfortunately, the resource consumption of this method is also greater than that of grid and random search.

Addressing Hyper-parameter Tuning Challenges

Falkner (73) put forward a criteria for HPO methods stating that they should perform well at all times, find the optimal settings, make effective use of parallel resources and deal with different hyper-parameter set. Additionally, they state that methods should be robust and flexible allowing for the various challenges presented by sub-fields in machine learning. The first two criteria present a "catch 22" for the methodologies put forward in the literature as they are naturally opposing of each other. The tuning can either be performance conscious and potentially not find optimal solutions or the tuning can continue until an optimal solution is reached at the expense of performance.

To combat this, rather than change the HPO method, understanding the impact of specific hyper-parameters on the accuracy of a model could help reduce search space of a method such as Bayesian optimisation and thus improving the performance. Hutter made the point that a reliance on HPO methodologies, such as Bayesian optimisation, can hinder insight to the important of individual hyper-parameters and their interactions as they cannot determine their effect on model performance (74). Furthermore, as HPO methods are trial and error per case there is no way to gather information on the importance of hyper-parameters beyond that specific configuration (75).

2.3 Sensitivity Analysis in Deep Learning

A summary of works from the last two decades, sorted chronologically, is presented in Table 2.2 highlighting the SA methodology, DL application and DL parameter space. Four trends were identified in the related works; number of publications by year, SA methodologies used, the DL architectures adopted and the parameter spaces explored. These four trends are highlighted in Fig. 2.6. Chart a in Fig. 2.6 shows the increase in work applying SA to DL, with published works in this area more than doubling in 2021 compared to 2017. Chart b highlights the various methodologies reported in these works as SA.

The most commonly adopted methods are calculating the Partial Derivatives (PaD) or taking inferences from plots. There are instances of more complex variance-based measures being calculated and some novel SA methodologies being proposed. The state of the art sensitivity methods Sobol Indices and Morris are used in one instance.

The most widely used architecture explored in the literature, as shown in Chart c of Fig. 2.6, was a simple Deep Neural Network (DNN), followed by the CNN architecture. This is to be expected, as discussed in Section 2.2 CNNs are one of the most popular architectures in DL and are widely adopted in various sectors. The final trend explored, shown in Chart d, was the parameter space the SA was applied to in the context of DL.

For the majority, SA was applied to the input space, being used to reduce dimensional-

Table 2.2: Summary of related works SA methodologies, application of SA and parameter space explored. The first row in the table highlights the areas explored in this thesis.

Paper	SA Methodology	Application	Parameter Space
Gullmar 2022 (11)	Plots	CNN	Input Space
Taylor 2021 (19)	Sobol Indices and Morris Method	DNN & CNN(x3)	Network Training
Moussa 2021 (12)	Variance	DRNN	Input Space
Davis 2021 (76)	Variance	Transformer	Network Architectue
Pizarroso 2021 (13)	PaD	DNN	Input Space
Nagasato 2021 (77)	Nash-Sutcliffe Estimation of Accuracy	CNN	Network
Novello 2021 (78)	Hilbert-Schmidt Independence Criterion	NN	Network
Shu 2019 (79)	Perturbation method	CNN(x2)	Network Architecture
Dudek 2019 (17)	Plot model loss vs model input	FNN	Network Architecture
Zhang 2019 (80)	Noise SA test prioritisation	CNN	Input Space
Ithapu 2017 (16)	Gradient plots	NN	Network Architecture
Samek 2017 (81)	PaD and plots	CNN & SVM	Input Space
Zhang 2015 (15)	Average accuracy	CNN	Network
Gevrey 2006 (82)	PaD	DNN	Input Space
Hunter 2000 (14)	Model performance vs missing data	DNN	Input Space

ity in a way akin to principal component analysis and to gain insight into which parameters in the input space most influence model performance. There were three categories of model related parameters explored; Network Training parameters, Network Architecture parameters or a combination of both, represented simply as Network in Chart d. The least explored parameter space was network training parameters.

Having given a general overview of the literature as it relates to DL and SA, we now explore the recent works in more detail. The latest work considered was published in 2022 and explored how SA could be applied to the semantic segmentation of medical images (11). SA was applied to the performance of a CNN architecture and the parameters being observed were related to the augmentation applied to the input images. Heat maps were produced and referred to as Sensitivity maps as they illustrated the affect of the various changes to the input data on their chosen measure of performance, the Dice Similarity Coefficient. This simple method of SA was sufficient to discuss the relationship between the input data and the models segmentation performance. However, these plots are open to interpretation of the practitioner and may not reflect the interactions of the parameters being studied.

In (19) a framework, SADL, was created, which applied two state of the art SA methods, Sobol and Morris, to various DL architectures on 3 image classification datasets. The aim was to use SA to produce a general ranking of DL training hyper-parameters that can inform practitioners when conducting hyper-parameter tuning. By using formal SA measures, it allowed comparisons to be drawn across architectures, datasets and with any future work. The next stages will be to apply the ranking and evaluate the effect on the tuning process.

It was previously established in Section 2.1 that SA is utilised in the engineering sector, specifically in building. It is also true that the application of DL methodologies is also becoming popular. To save time in testing specific material properties, a Deep Residual Neural Network (DRNN), based on ResNet, was proposed and the input data was preprocessed with variance based SA (12). The first-order sensitivity measure multiplied by a normalising factor was used to analyse which input parameters were most influential on the accuracy of the DRNN. This a very context specific example of the application of SA to DL input parameters which highlights the potential for more general knowledge to be gained.

Transformer architectures are also subject to the hyper-parameter tuning process and can be affected by perturbations in parameter values. The idea of 'sensitivity' in the form of variance was adopted to measure the extent a transformer architecture is affected by random perturbations of its parameters. Furthermore, the observations made related to 'sensitivity' inspired a new transformer architecture, boasting increased stability in the face of parameter perturbations (76). In this case, 'sensitivity' is used in place of 'stability', and where the usual SA approaches sample the parameter space in a structured way this work is more interested in the robustness of the model against random perturbations. Despite naming the metric 'sensitivity', it could be argued that this work conducts Robustness Analysis (RA) rather than formal SA. Having said this, formal SA is still applicable here and could be conducted on the new, robust transformer to better understand the effect of the input parameters or to influence network parameter tuning.

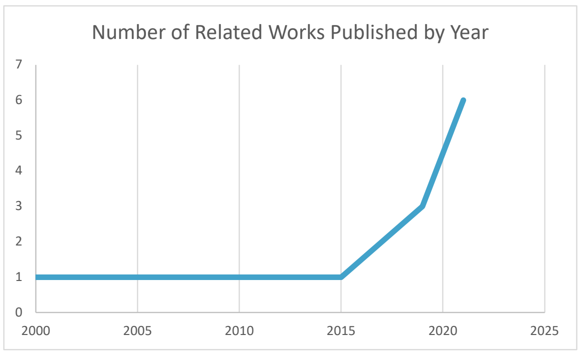
A novel metric, Noise SA Test Prioritisation (NSATP), was proposed to take a SA approach to model sensitivity to adversarial examples of input data (80). The early CNN architecture, LeNet, was used along with four image classification datasets. The aim was to produce a ranking of input examples based on their noise sensitivity which could lead to the development of more robust DNNs. The aim is similar to that in the previous work (76), however there is a clear link made here with the Ranking SA setting, demonstrating how SA can be applied to increasing model robustness through the analysis of adversarial input data.

Explainable Artificial Intelligence (AI) is a field of study aiming to crack the black-box nature of ML and DL models to better understand what contributes to their final outputs. Two methods, SA and Layer-wise Relevance Propagation (LRP), were adopted with the aim of explaining the predictions of various ML models in terms of their input parameters (81). Perturbation analysis was conducted on three scenarios to compare the two methodologies: CNN on and image classification task, a CNN on a text classification task and an SVM on an object detection task. Local SA in the form of the derivatives was calculated to produce a sensitivity measure which was supplemented by heat-maps for the image classification experiment to identify influential pixels in the input data. LRP explains model output through decomposition and assigns inputs relevance scores to quantify the contribution of each input towards the output. The explanation of LRP is very similar to the state of the art variance-based SA methodologies which were developed based on the ANOVA decomposition. The conclusion found that the heat-maps based on the LRP were more informative than those based on the SA, which would be expected as the measure of SA employed was much simpler. A more comparable method for LRP would be the Sobol's Indices. This work does, however, highlight that SA has the potential to be applied to better understand the inner-workings of the black-box DL models.

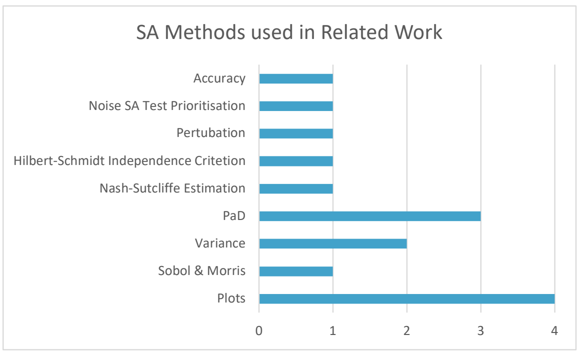
Simpler methods, such as calculating PaD, and analysing the input space encompass several works in the literature. The application of PaD to the input space is a common approach across application areas, be it the input to a Multi-Layer-Perceptron (82) or environmental simulation (83). Traditionally, SA is conducted on numerical data which is used to calculate derivatives. To overcome this, rather than calculate the derivatives Hunter (14) introduced missing data and inferred influence based on the deterioration in model performance, a simple yet effective example of SA. Some CNN hyper-parameters are categorical and, therefore, face the same issue that Hunter encountered. Introducing missing data in our work was not an option as the network required parameter values to operate and it would be beneficial to have insight into the most influential values of a parameter. The approach taken in our case was to one-hot encode the categorical values, representing them numerically to calculate the sensitivity measures.

A study of the influence of varying network structure was conducted where the lack of guidance surrounding hyper-parameter tuning for DL prompted Ithapu et al. (16) to take a loose SA approach. Inference based on the relationship between network architecture, hyper-parameter convergence and input data statistics was used as an indication of sensitivity. The main drawback of this work is that no comparable measure was produced that could be used in future work. A comparison of CNN performance on sentence classification tasks was conducted to understand hyper-parameter influence (15). Once again, the main drawback is the lack of a measure that can be used to compare against other work across architectures and tasks. By comparison, using the SADL framework this analysis of CNN performance on image classification task produces recognised SA measures.

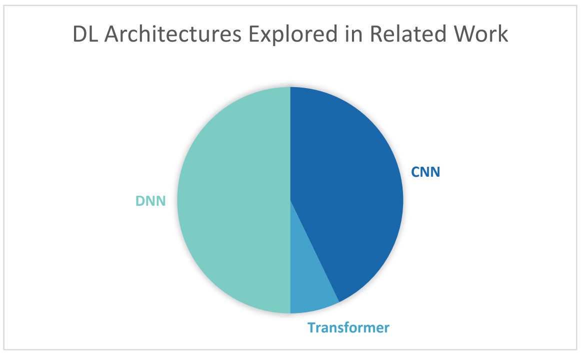
Four tasks were considered by Shu (79); outlier detection, SA of network architecture, SA of test and training sets and vulnerable region detection. They proposed a method of SA, perturbation manifold, and applied it to two networks ResNet50 and DenseNet on two datasets MNIST and CIFAR10. The experiments conducted explore the network's sensitivity to layer-wise perturbations and seem to be exploring the robustness of the architec-



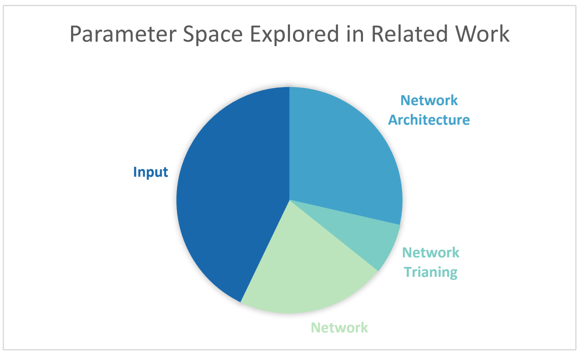
((a)) By Year



((b)) SA methodologies



((c)) DL Architecture



((d)) Parameter Space

Figure 2.6: Trends in the SA DL related works over the last two decades. More specifics on the works included can be found in Table 2.2.

tures. By comparison, our work explores the sensitivity of specific training hyper-parameters of five network architectures on three image classification datasets to offer more generaliseable insights.

2.4 Potential Applications of Sensitivity Analysis in Deep Learning

The need for Sensitivity Analysis (SA) was summarised by Furbinger in 1966 (84):

> "Sensitivity Analysis for modellers? Would you go to an orthopaedist who didn't use X-ray?"

With DL growing in complexity and being applied to serious problems surrounding disease detection, building stability and more, that have the potential to affect lives it begs the question: are we doing everything we can to understand and build the most efficient and accurate models possible? And, if SA in modelling can be compared to the necessity of x-ray to orthopaedists, should we be utilising SA in ML and DL as standard practise?

French proposed eight contexts for the use of SA(24):

1. To build and explore models
2. To explore science and models relationships
3. To determine influential inputs
4. To develop efficient algorithms
5. To design experiments
6. To guide decision making
7. To build consensus
8. To gain understanding

and several of these could also be applied to DL, once again highlighting the compatibility between SA and DL methods. Potential applications of SA to DL will be described in relation to a subset of the contexts mentioned above, linking back to the main focus of this work in hyper-parameter tuning.

To Build and Explore Models

DL Models of complex behaviours are often complex themselves, and this in part contributes to the black-box nature of DL models and drives the explainable AI community to explore their inner workings. One area of potential exploration surrounds the relationship between model hyper-parameters and model outputs, specifically the relationship between model parameters and accurate model outputs. Furthermore, whether these relationships are model specific, application specific or whether there are more general links between specific parameters and model accuracy. Conducting exploration into the general influence of DL hyper-parameters could impact how future models are built depending on the relationships that are observed. For instance, if hyper-parameter influence is found to be architecture specific then when developing improvements on CNN architectures, for example, researchers can concentrate on optimising the related influential parameters. If the influence is found to be application related, affected more by the nature of the input data, then that would guide researchers when developing models for computer vision or object detection or natural language processing. Any general findings would benefit all DL practitioners and could fill a gap in hyper-parameter tuning advice which is currently lacking.

To Determine Influential Inputs

This context focuses the work on the inputs that matter. In terms of DL hyper-parameter tuning, this would be the influential parameters. Better understanding of the parameters also reduces the potential that the model output is a product of imprecise parameters (24). The product of imprecise parameters in the tuning process could be taking additional time and resources to find optimal settings because non-influential parameters were being explored unnecessarily. Bayesian optimisation embodies this context as it searches for the optimal parameter values, pursuing areas of exploration based on previous trials to reduce the number of iterations needed. A SA informed approach to Bayesian optimisation would reduce the parameter search space to influential parameters only, increasing the efficiency of the search. Furthermore, SA can provide reassurance to practitioners that removing a subset of parameters from the tuning scope would not have detrimental affects on model performance. Additionally, SA presents the opportunity and evidence to adjust approaches and ideas that are favoured *a priori*. The importance of the learning rate hyper-parameter is well documented in the literature, with influential DL practitioners recommending to only tune the learning rate if you had to pick one parameter to tune (18). However, through the application of SA methods to DL model training parameters, learning rate was ranked as having low influence compared to batch size and learning rate decay (19). This finding presents the possibility that SA can lead to new insights and understanding of the importance of popular hyper-parameters.

To Develop Efficient Algorithms

Despite the current advancements and growth in both computational power and resources, efficiency is still a key metric that DL practitioners aim to maximise. SA itself is a computationally expensive process, especially when applied to complex models. However, a robust analysis of DL hyper-parameters using SA would produce general knowledge that can be adopted by practitioners to maximise the efficiency of the hyper-parameter tuning process. Additionally, the application of SA to DL architectural parameters could aid in removing redundant features, making the models shallower or narrower without compromising the performance. Simply put, SA can help practitioners find the least complex, most efficient version of a model that will still achieve high accuracy.

To Design Experiments

One of the most important elements of experimental design surrounds the fundamental choice of which points to collect data at, which are most informative and can reduce uncertainty (24). SA methodologies take very systematic approaches to sampling often employing Latin Hyper-Cube or Monte-Carlo sampling to produce the most representative analysis of parameter influence. Too often DL practitioners have to choose to tune more parameters less thoroughly due to computational or time restrictions and the lack of available guidance that sufficiently reassures them in their choices of which parameters should be explored further. By reducing the depth of tuning of the parameter's values in favour of breadth of model parameters under investigation, practitioners are potentially missing optimal solutions. As previously discussed SA would be useful here to reduce the parameter set that is considered for tuning. Alternatively, if practitioners wish to explore a larger parameter set, SA mapping could highlight which ranges of values for each parameter are most influential. This would allow the tuning process to explore that range more thoroughly and increasing the possibility of finding optimal values for more parameters whilst being mindful of external constraints.

To Guide Decision Making

When considering decision making, reassurance, robustness and future direction come to mind. The use of SA to reassure practitioners has been discussed at length in the contexts above. In terms of robustness, the insights provided by SA can lead to an awareness of which hyper-parameters the model would be robust against perturbations or variance in and could highlight any risks to model performance that modellers can then take steps to address. The emphasis in this paper has been placed on the potential SA presents to the hyper-parameter tuning process of DL models and how the information gleaned can aid practitioners in choosing future direction within that process. Not only understanding the influence of hyper-parameters, but quantifying and ranking it, provides clear information on which parameters effect model accuracy and the relationships that parameter influence has with DL factors such as task/application or architecture. These results could influence decisions regarding which subsets of parameters to focus on when tuning or which ranges in each parameter space should be explored, all with the aim of reducing time or complexity whilst retaining high performance.

To Gain Understanding

Each of the contexts put forward by French share a common goal: to build understanding (24). In this case, the application of SA to DL hyper-parameters would produce quantitative results reflecting influence on model performance that will inform practitioners. This information will lead to better understanding of the inner workings of DL models in regards to parameter importance and their contribution to the decision making process that is contained within the black-box. The various charts associated with SA can be used to communicate these insights to a wider community that may not have expert knowledge of DL models but is steadily growing thanks to the countless DL libraries which is making ML more accessible. The wide-spread adoption of DL only strengthens the need for general, reassuring advice that can be put into practice and help people understand the influencing factors in their work.

2.5 Conclusions

SA provides a lens which offers a view inside the black-box of DL models. Despite this potential and inherent compatibility, there are few examples of formal SA methodologies being applied to the hyper-parameters of DL models themselves, with the majority of works concentrating on the model input data. This chapter contributed a thorough review of SA methodologies, DL practices, SA applications to DL and finished with suggestions and recommendation on how SA should be applied to DL model hyper-parameters. By highlighting the benefits to efficiency and insights into explainability SA could offer DL practitioners, we hope to encourage the adoption of SA within the ML community.

2.6 Summary

Sensitivity Analysis (SA) has the potential to be applied to Deep Learning (DL) the results of which could improve the explainability of DL models. DL hyper-parameters are key to model performance however there is little understanding of the extent of their influence on model output. An extensive review of SA and DL was conducted to better understand their compatibility and explore how SA and DL have been previously connected in the literature. This culminated in recommendations on how SA can be used in a DL context to better inform practitioners regarding DL hyper-parameter tuning. SA can aid practitioners in increasing the efficiency of the hyper-parameter tuning process by providing information that will reduce the parameter search space. The results of the review suggest that DL is a domain that would benefit from the application of SA as it can offer insight into DL model explainability.

The key points of this review are summarised as follows:

1. Despite the popularity and success of formal Sensitivity Analysis techniques in some STEM sectors, there has been little adoption of Sensitivity Analysis in the Machine Learning community, specifically in its application to Deep Learning.

2. Efficient use of time and computational resources is becoming increasingly challenging as Deep Learning models and datasets grow in size and complexity. Model hyper-parameter tuning, finding their optimal values, is a key step in improving model performance, the increasing cost of which is placing practitioners in an impossible position. Applying Sensitivity Analysis to the hyper-parameters and quantifying their importance to model performance can aid and reassure practitioners in their reduction of the parameter set, allowing them to improve efficiency without compromising performance.

3. Explainability is a major focus of DL research, aiming to better understand the models which are currently considered to be black-boxes. Sensitivity Analysis presents an opportunity to better understand the contribution DL model hyper-parameters make to the decision process.

Chapter 3

Methodology

The experimentation carried out explored hyper-parameter tuning from the novel angle of using SA to reduce the parameter search space by establishing generally influential parameters. These influential hyper-parameters should be prioritised when tuning and can reduce the search time without compromising model performance. The data required to compute the sensitivity measures was collected by systematically changing the value of a single parameter whilst controlling the rest and using the resulting model accuracy to calculate the sensitivity measures. Two measures from state of the art SA methods contribute to a general ranking of hyper-parameters. SA is used successfully in other areas and industries, as discussed in Chapter 2, to quantify importance of parameters. Additionally, within machine learning SA has been used to reduce input data features. This history makes these methods well prepared for this application. To increase the validity and reliability of the resulting parameter ranking, which is explored further in tuning experiments, SA is conducted on a variety of CNN architectures and image classification data-sets.

The aim of this work has three key focus areas, which have shared and distinct methodology:

1. SA of CNN Hyper-parameters
2. Rank Informed Hyper-parameter Tuning
3. Case Study

This chapter, therefore, can be split into three sections, shown in Fig. 3.1. This work began with the development of a framework which facilitated the calculation of sensitivity measures for DL CNN models, SADL, which will be discussed first. The second set of experiments built on the results from the framework and explored how the resulting ranking could be used to inform hyper-parameter tuning. Finally, the culmination of this work tested the results in a case study.

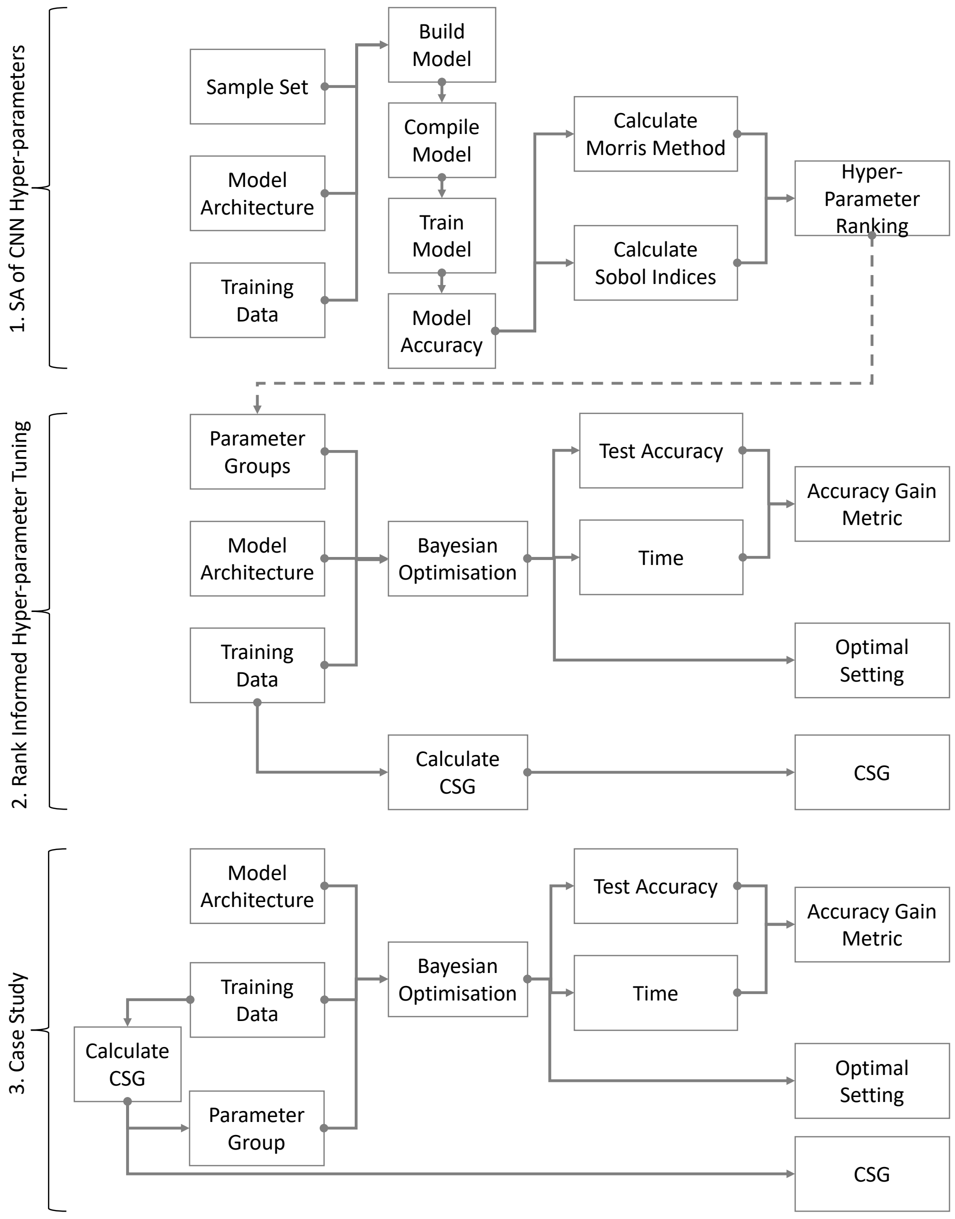


Figure 3.1: Thesis focus areas, a graphical overview of the methodology.

3.1 SA of CNN Hyper-parameters

The first phase of work encompasses the development of a framework that facilitates the calculation of sensitivity measures for DL models. These measures can be used to determine the influence of hyper-parameters and produce a general ranking. Additionally, any relationships between hyper-parameter influence and model architecture or dataset can be identified. This section was planned to target the work aim to better understand parameter influence on model performance and will contribute a formalised, quantified rank of CNN hyper-parameters.

3.1.1 Sensitivity Analysis Framework for Deep Learning Development

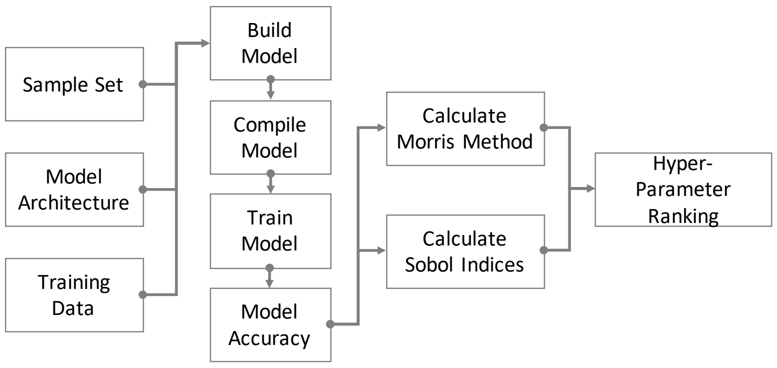


Figure 3.2: Sensitivity Analysis Deep Learning Framework (SADL) overview.

The Sensitivity Analysis Deep Learning Framework (SADL) was developed to produce the sensitivity measures of two state of the art SA methods, Sobol and Morris, for the training hyper-parameters of CNN models. The framework can be broken down into four stages; inputs, training, SA calculation and rank generation. A high-level interpretation of the framework is shown in Fig. 3.2 and early results were published in the International Conference of Tools of Artificial Intelligence (ICTAI)(19).

3.1.2 Framework Inputs

As shown in Fig. 3.3, there are three inputs to SADL: a sample set of hyper-parameter ranges, model architecture and training dataset. The sample set, discussed further in Section 3.1.3, consisted of the values for each hyper-parameter. A sample was generated for each hyper-parameter under investigation where the values of that parameter were varied and the oth-

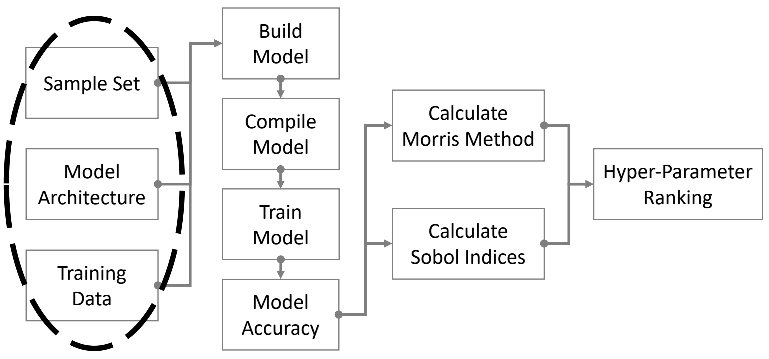


Figure 3.3: Sensitivity Analysis Deep Learning Framework (SADL) inputs.

ers remained consistent. Five model architectures were explored, and three image classification datasets. For each set of experiments the architecture and dataset had to be set to calculate the sensitivity measures of the hyper-parameters for each combination.

3.1.3 Hyper-parameter Sample Set Generation

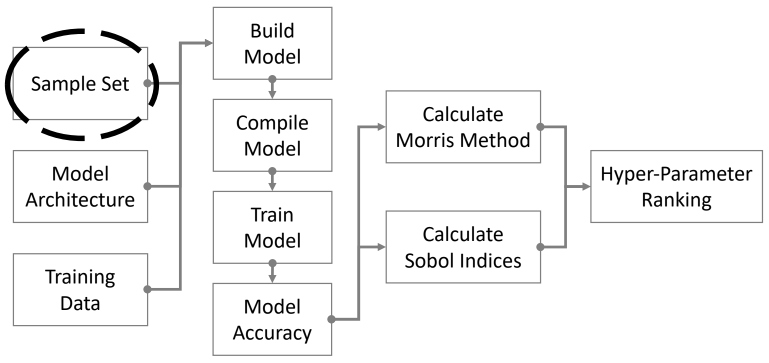


Figure 3.4: Sensitivity Analysis Deep Learning Framework (SADL) sampling.

In the literature, SA approaches have an associated sampling method to produce the required sample set shown in Fig.3.4. In Morris' case, a trajectory-based sampling methodology is used and in Sobols' a Monte Carlo approach is taken (21). If SA was to be conducted on the basis of simple, random sampling then there is no guarantee that the sample used did not contain clusters of points or areas of the domain space that were underrepresented. Employing a stratified sampling approach reduces these issues by slicing the sample space into regions from which points are chosen for exploration. This results in a well-proportioned sample. Informative SA is completely reliant on the number and distribution of sample points available for analysis which is why more detailed approaches to sampling are applied.

Rather than implementing two separate sampling methods, one sample was created and used to calculate all sensitivity measures. Latin Hyper-cube Sampling (LHS) is associated with SA in the literature and provides a more structured approach than random sampling by offering a higher probability of covering the whole sample space (85). By combining desirable factors from stratified and random sampling techniques, LHS has the additional benefit of simple implementation, making it an ideal method for large models (86). A sample of 100 variations was created for each hyper-parameter being explored. Morris' method does not require a certain size of sample whereas Sobol requires a larger sample to provide a more correct measure of influence. This need for a large sample size increases the computational cost of the method and so 100 was chosen as it provides the balance between a sufficiently large sample and resource usage. A sample less than 100 would not produce robust sensitivity measures and samples more than 100 do not increase the robustness of the measures sufficiently to justify the cost (87).

3.1.4 Hyper-Parameters

Seven core training parameters were included in the scope of this work that are listed in Table 3.1. The default values for the parameters and their ranges were decided based on recommendations in the literature. Learning rate was highlighted in many works as being a significant hyper-parameter which should be prioritised for tuning (18). This influences the hyper-parameter direction in this work as the set had to include learning rate, the parameter reported as one of the most important. By concentrating on training parameters in these experiments the architectures could remain consistent and therefore, introduced less resource constraints than experimenting with network depth and width would have. Keeping the scope of hyper-parameters as training only also presents opportunities for future work to explore architectural parameters and draw comparisons.

Table 3.1: Deep Learning Hyper-parameters descriptions, symbols, ranges and default values used for SA experimentation.

Parameter	Description	Range	Default Value
Optimiser	List of gradient descent (GD) algorithms.	Category*	Adam
Learning rate (α)	Initial GD step controller.	[1x10^{-7}, 0.5]	0.001
Momentum (β)	Acceleration factor for GD.	[0, 0.99]	0.6
Learning rate decay (α_{decay})	Reduction rate of (α).	[0, 1]	0.9
Learning rate decay step (α_{d-step})	Number of epochs between Learning Rate Decay.	[1,100]	10
Batch size	Size of training subset for GD update.	Category*	32
Epochs	Number of training cycles.	[5, 1000]	100

Note: Category* indicate that there are two hyper-parameters with categorical ranges (88): (i) optimiser, Adam, SGD, RMSprop, ADAdelta, ADAgrad and ADAmax; and (ii) batch size, 1, 32, 64 and 128.

3.1.5 Model Architecture

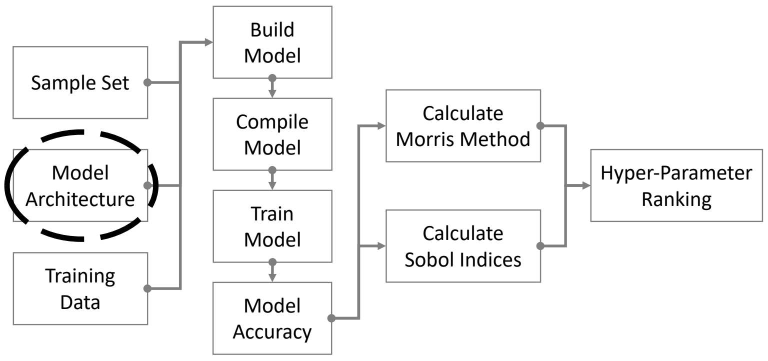


Figure 3.5: Sensitivity Analysis Deep Learning Framework (SADL) model architecture.

Model Descriptions

A single architecture is explored at a time using the framework as shown in Fig.3.5. A DNN, and four state of the art CNN architectures were investigated in this study: ResNet18 (58), AlexNet (54), VGG16 (56) and GoogleNet (55). Including the DNN allowed for initial insight into the application of SA to DL hyper-parameters that could be applied to CNNs. The state of the art CNNs: ResNet18, Alexnet, VGG16 and GoogleNet are represented in Fig. 3.6. This shows that the chosen architectures vary in depth and complexity. Using CNN's with a variety of compositions in the experiments allowed for relationships between hyper-parameter influence and model architecture features to come to light.

The DNN architecture was comprised of a three layer network with 64 units, based on a MLP which is a general recommendation to practitioners starting out in DL and machine learning. This architecture was chosen to show the influence of hyper-parameters on a small, simple network. Additionally, due to the shallower architecture, training the DNN's was quicker, allowing for the development of the framework (19).

The winner of ILSVRC 2012, AlexNet is a CNN which consists of 8 layers and 60 million trainable parameters. The development of AlexNet influenced several aspects of DL model design. With the advantage of increasing training speeds, Alexnet popularised the use of ReLU over Tanh. Furthermore, overlapping pooling layers were introduced through this architecture, decreasing the training error and the potential for over fitting. On the other hand, dropout is required to decrease the risk of over-fitting caused by the high number of trainable parameters resulting in an increase in training time required. As the shallowest

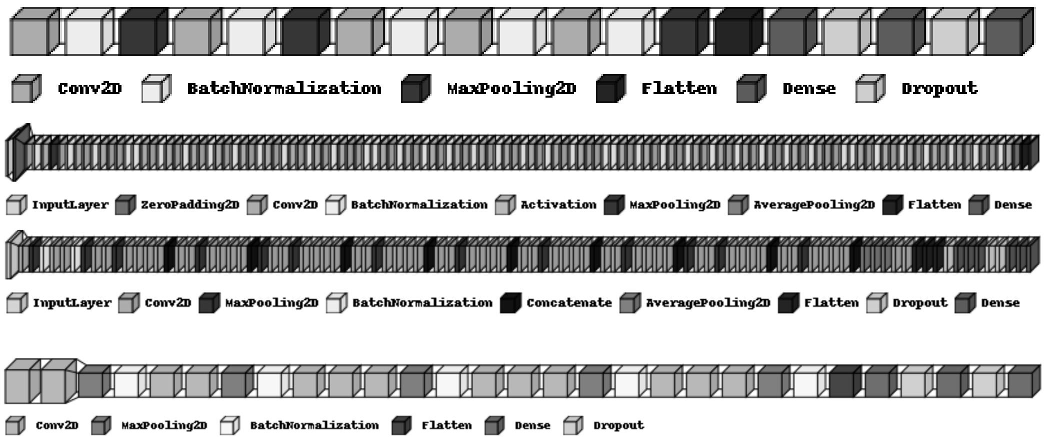


Figure 3.6: Architectural diagrams of the state of the art models: AlexNet (top), ResNet18 (middle-1),GoogleNet (middle-2) and VGG16 (bottom).

CNN architecture being considered, the network depth may present issues when learning features.

ResNet18 was chosen as ResNet architectures offer low training complexity and depth whilst achieving a low error rate. Winning ILSVRC in 2015, ResNet achieved the largest increase in accuracy since AlexNet (89) in 2012. The residual learning concept is the major contribution and advantage of ResNet. Simply put, it improves model accuracy by focusing on learning new features whilst also speeding up the training process.

GoogleNet achieved best accuracy in ILSVRC in 2014. Resource conservation was the main focus of this architecture, a clear advantage when considering the current DL challenges surrounding computational resources. Comprised of 22 layers without pooling, GoogleNet is the deepest of the CNN models assessed. The constant resource usage allowed the analysis of this size of model on a single GPU however, it was the slowest to train.

Runner up to GoogleNet at ILSVRC 2014, VGG16 is a popular architecture for image classification tasks. VGG16 replaces the large kernel sizes in the AlexNet architecture with consecutive 3x3 kernels producing a more discerning decision function. This did have the side effect of slow training and producing large weights resulting in high memory utilisation.

Model Implementation

All models were implemented in the Python programming language, version 3.7, using the Keras (88) package for deep learning. Each architecture implementation follows how the model was presented in the original papers to ensure the state-of-art characteristics that set them apart as high-performing CNNs are present. The DNN architecture followed a simple specification following recommendations for beginners in DL and machine learning so that these groups of practitioners are represented in this work. Furthermore, the simplicity of the DNN resulted in a very quick training time which facilitated the development of the SADL framework. This allowed us to test the framework quickly, apply several iterations of training, compute sensitivity measures and make necessary adjustments whereas the larger models could take hours/days to train, which would have caused serious delays in the framework development process. Utilising the DNN model during framework development overcame the hurdles of training time introduced with larger, deeper architectures. For each dataset used in the experiments the first layers of the architectures had to be adapted to the shape of the input data.

CCN Justification

As reported in the literature, the CNN architecture is the most adopted DL approach (90). The popularity of the CNN can be explained by its applicability to a wide variety of supervised learning problems such as image classification, natural language process and object detection. As a result there has been thorough research into optimising CNN architectures, producing several state of the art models that practitioners and researchers can use in their work such as ResNet, GoogleNet, AlexNet and VGG16. Understanding the influence of hyper-parameters on CNN architectures presents the most potential in terms of useful impact. Presenting a ranking of CNN hyper-parameters that can aid in model optimisation would have a wider-reaching benefit compared to less popular architectures.

3.1.6 Training Data

Data Description

Input data to the CNN models were also a key element of the framework, shown in Fig.3.7. MNIST, MNIST-Fashion and CIFAR-10 are the three image classification datasets used in

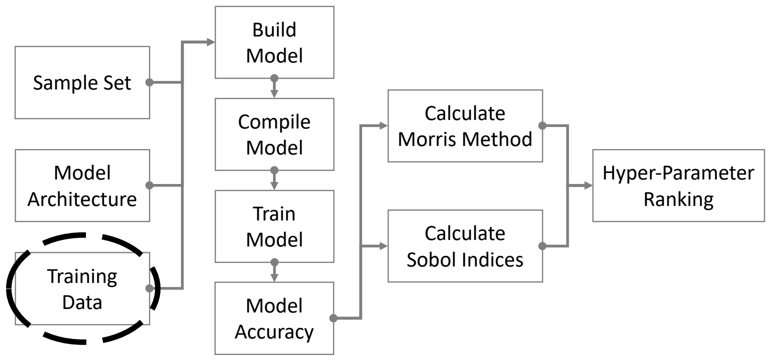


Figure 3.7: Sensitivity Analysis Deep Learning Framework (SADL) datasets.

these experiments, examples of which are shown in Fig. 3.8. One of the most popular benchmark dataset for deep learning is MNIST which consists of 28x28 grey-scale images. Ten classes of handwritten digits present a relatively simple task for DL networks. MNIST is split into 60,000 training and 10,000 test images. To measure dataset complexity a cumulative spectral gradient (CSG) measure gives an indication based on its overall separability. MNIST has low levels of complexity, with a CSG of 0.11 reported in the original paper (91).

The second dataset chosen, MNIST-Fashion (92), shares many features with MNIST including image size, format and test/training split. The images themselves are of fashion products and thus present a new level of complexity as there are additional features that a network must learn in order to accurately classify an image. Following the methodology in the paper by Frederic Branchaud-Charron (91) the CSG of MNIST-Fashion was calculated to be 0.51.

The most complex and challenging of the datasets is CIFAR-10, comprising of larger, coloured images, 32x32, and more varied 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. As a result, this dataset has higher dimensionality which requires more time to train. CIFAR-10 was calculated to have CSG of 3.8, confirming that it is much more complex that MNIST and MNIST-Fashion.

The varying complexity in the chosen datasets presents the opportunity to analyse the effect of dataset complexity on the influence of network hyper-parameters. Experimenting with a range of datasets aims to identify any relationships that may exist between hyper-parameter influence and dataset complexity. Example samples from each dataset are shown in Fig.3.8.

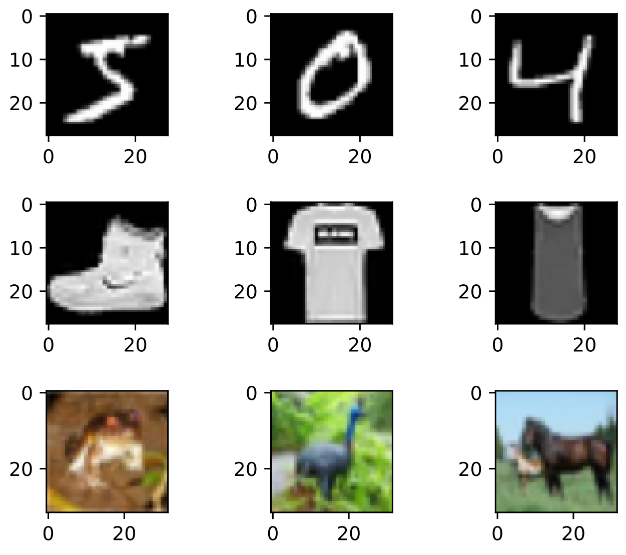


Figure 3.8: Examples from each dataset; MNIST (row 1), MNIST Fashion (row 2) and CIFAR-10 (row 3).

Data Preparation

The preparation of the datasets consisted of three steps:

1. Split data into train and test sets
2. Normalise and re-shape data
3. Encode class data

The first step was separate the training data from the test data that would be used for validation. Both the MNIST and MNIST-Fashion dataset had a split of 60,000 training images and 10,000 test images whereas CIFAR-10 consists of 50,000 training images and 10,000 test images.

The second step in the data preparation was to normalise the images. Rather than take a just-in-time approach, all images were normalised and stored in this format prior to their input to the networks. The aim was to convert the pixels in the images from having a range of 0-255 to 0-1. An additional step in the process includes centring the image, which shifts the pixel distribution so that 0 is in the middle.

The final step was to one-hot-encode the class data. This step is more relevant to the MNIST-Fashion and CIFAR-10 datasets as the class labels are categorical and need to be converted to numerical data for use within the networks. One-hot-encoding works by assigning a numerical value to each unique categorical value which is used to represent that value throughout the training and validation of the network.

Justification for choosing image classification

Image classification is a classic machine learning task where images are assigned pre-defined categories based on learned features and is a fundamental problem within the wider field of computer vision (93). As a result is a popular problem that is tackled in research with various facets to explore. Furthermore, the complexity and size of image data can result in time and resource constraints in the tuning and training process where additional efficiency would be a benefit.

The three image classification datasets were chosen as they are widely used in research which aids the validity and generalisability of this work. Furthermore, they are readily available aiding in the reproducability of this work. Finally, they represent varying task complexity whilst still being small enough that the work could be completed within the proposed time-frame as the development process required re-training models repeatedly the time constraints introduced by larger image datasets would have caused considerable delays. Additionally, these datasets were small enough to be processed and stored with general hardware making them applicable to the general DL practitioners that this work aims to inform and aid.

3.1.7 Model Training

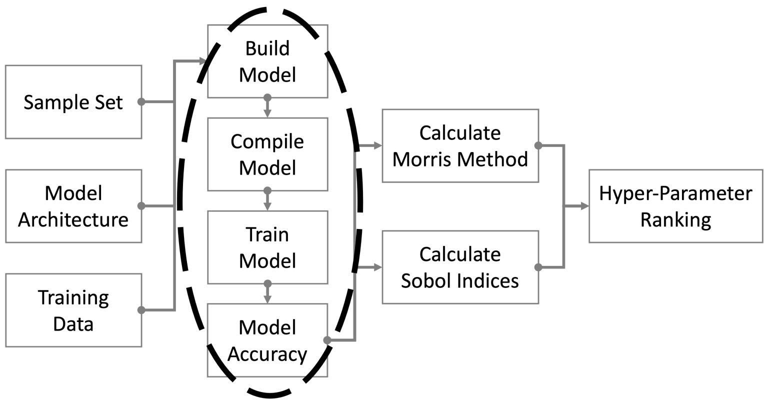


Figure 3.9: Sensitivity Analysis Deep Learning Framework (SADL) training.

For each sample point of hyper-parameter values in the sample set, the chosen model was built with random weight initialisation and compiled with the parameter settings outlined in the sample. Once the model was set-up for that iteration, it was trained. Early stopping was implemented concentrating on the accuracy achieved, with the highest accuracy recorded alongside the value of the varied parameter. This was repeated for each combination of architecture (n=5) and dataset (n=3) 100 times for each hyper-parameter (n=7) resulting in $100 \times 7 \times 5 \times 3 = 10,500$ iterations to feed into the calculation of the sensitivity measures. This process is outlined at a high level in Fig.3.9.

3.1.8 SA Calculation

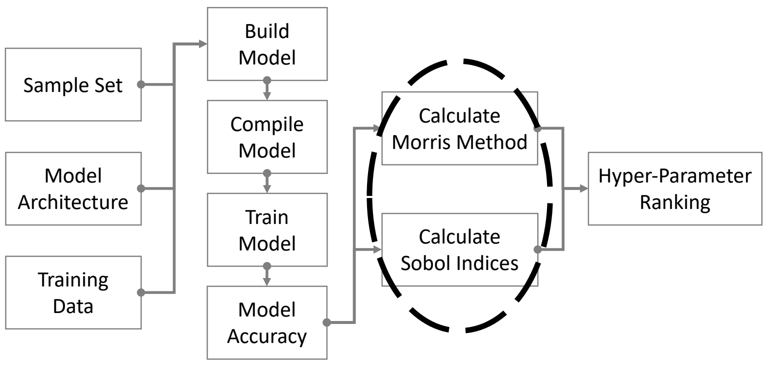


Figure 3.10: Sensitivity Analysis Deep Learning Framework (SADL) sensitivity measure calculations.

One aim of these experiments was to understand and quantify the influence of hyper-parameters on model performance. This could be defined in several ways such as minimal loss, high accuracy or training times to name a few. Within the scope of work, model performance in this instance is defined as the accuracy achieved specifically the test accuracy. This is a key element within the SA measure calculations. Two state of the art SA methods, Sobol and Morris, were implemented as part of SADL which are discussed in depth in Section 3.1.9 as shown in Fig.3.10. Two factors are required to calculate the measures: the model input and the model output. In this case the model input is a sample set of varied hyper-parameters values and the outputs are the corresponding test accuracies. These are used for both methods to produce sensitivity measures which quantify the influence of each hyper-parameter for each combination of model architecture and image classification dataset.

SA has been designed to be implemented on numerical data, however there are some model hyper-parameters that are represented categorically. Both the optimiser and batch size parameters have categorical input values and therefore the framework had to be adapted to conduct SA for categorical data. Previous works had handled this by removing the categorical options and conducting the SA on the absence of a value (14). This approach however, does not allow for further insights into which values of the parameter were most influential. To retain the potential of understanding categorical influence one-hot-encoding was employed. This made it possible to represent the parameter values numerically for the SA calculation whilst also being able to reverse the encoding and understand which parameter values were most influential.

3.1.9 Sensitivity Analysis

The aim of SA is to understand the outputs of a model in terms of its inputs and the most popular methodologies are Sobol Indices and Morris Method. In this context, model inputs are the hyper-parameters and model output is the accuracy achieved. Both methods were calculated separately, as shown in Fig.3.11 and Fig.3.12.

Morris Method or Elementary Effects

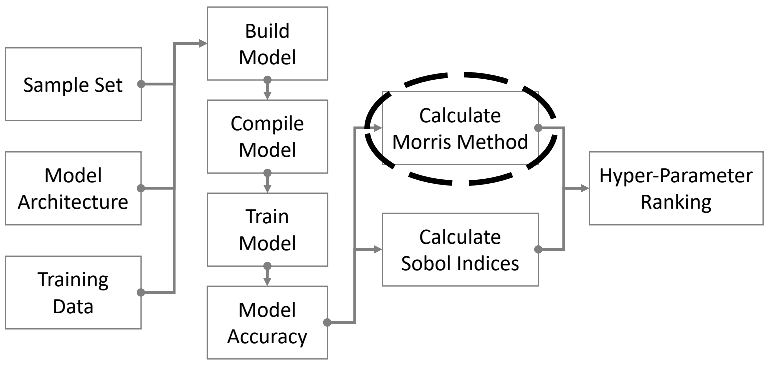


Figure 3.11: Sensitivity Analysis Deep Learning Framework (SADL) Morris Method.

The Morris Method (9) or Elementary Effects (EE) applies local SA across a feature space, to create a global measure and is classed as a screening method. The aim is to determine the effect of input parameters to a model whether they be negligible, linear and additive, nonlinear or involved in interactions with other parameters (21). The measures produced are μ and σ where μ quantifies the overall influence of a parameter and σ represents the independence of the attributed influence. The measures values range between 0 and 1 with a higher value indicating the model is more sensitive to the parameter.

Modified Morris

Campolongo et al proposed a modification to the traditional Morris Method introducing an additional sensitivity measure μ^* (30). Unlike the original measures, μ^* was able to handle groups of parameters, and could be considered a total sensitivity measure, combatting Type 2 errors that the original μ measure was prone to. This method was tested against variance-based measures in Campolongo's paper comparing μ^* against the measure produced from Sobol's Indices for some models. They determined that μ^* was an effective substitute for ST_i. This is promising in terms of efficiency as computing μ^* requires no additional time and can be computed alongside the original EE sensitivity measures with no additional cost.

Sobol's Indices

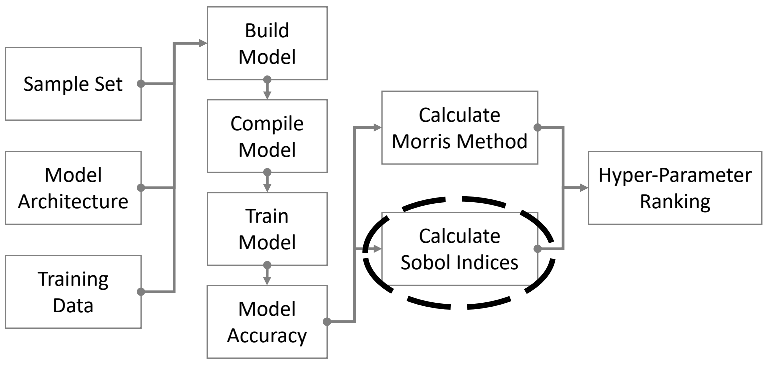


Figure 3.12: Sensitivity Analysis Deep Learning Framework (SADL) Sobol Indices.

Sobol's Indices is a variance-based SA methodology which are considered to be state of the art in the literature. The strengths of variance-based approaches are that they are model independent and consider parameter interactions whilst representing the global search space(21). On the other hand, the computational cost associated with calculating the variance-based sensitivity measures is high and is in addition to the cost of a number of model simulations. Two measures are produced by the Sobol Indices method (8), the first-order effects, S_i, and total-effects, ST_i. These measures can be thought of as a percentage of the variance in model output caused by the input being analysed and, as such, range between 0 and 1. The higher the value of S_i the more the variance in output is attributed to that parameter whereas ST_i attributes the variance to the parameter under consideration and its interactions.

SA Implementation

The SA methods were implemented in the python programming language. There are packages where various SA methods have been implemented such as SALib (94) which is also in python. The choice to code a separate implementation rather than utilise a package for the SA was based on efficiency. SADL includes two SA methods and to reduce computation time both are calculated from the same inputs and outputs of a single sample. This was not possible to do with the package implementation as they are built to create a sample for each run of the method and there are differing default sampling approaches which is dependent on the SA chosen. As a result, code was written to implement both Sobol Indices and Morris Method within the SADL framework. Both sets of produced SA measures were also normalised using min-max normalisation and implementations of these methods can be found in Appendix A for reference.

The Morris method implementation was broken down to four functions:

1. Calculate the elementary effects
2. Get the increased values
3. Get the decreased values
4. Produce sensitivity measures

Sobol Indices was implemented in 3 stages:

1. Calculate S_i
2. Calculate ST_i
3. Produce sensitivity measures

The final stage for both methods was to normalise the produced measures.

3.1.10 Hyper-parameter Rank Generation

The initial measures produced through the SA can be interpreted as rankings for each individual combination of architecture and dataset. Two SA methods were considered as they can emphasise different areas of influence and, as a result, produce differing overall

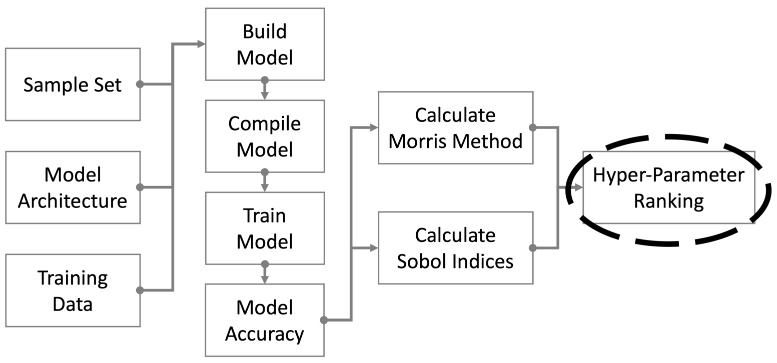


Figure 3.13: Sensitivity Analysis Deep Learning Framework (SADL) hyper-parameter rank generation.

rankings. Therefore, at this point in the process, from the measures produce (n=2), architectures (n=5) and datasets (n=3), there are $2 \times 5 \times 3 = 30$ rankings to take into consideration to produce an overall, generalised picture of hyper-parameter influence.

Each hyper-parameter was ranked based on their influence on model accuracy. Their rank was determined based on the sensitivity measures produced by both SA methods, Morris and Sobol, equally. The sensitivity measures μ^* and ST_i are utilised to understand the influence of each parameter. These were chosen over the other measures produced via SA as they are more robust measures and are considered to be the best in the literature. Both scores have a range of 0–1, the higher the score the more influential the parameter. A high score in both measures indicates that both methods are in agreement, validating the ranking further. When both measures do not agree the parameters scores distance from a perfect score (1,1) is used to distinguish its rank amongst the other hyper-parameters. As both methodologies take different data into consideration to produce the measures they can place emphasis on different parameters which is why the combination of both SA approaches was used to produce a generalised ranking, as shown in Fig. 3.13. This ranking took into consideration both SA methods by computing the Euclidean distance from the highest possible measure value, 1 for μ^* and 1 for ST_i, to the values each parameter achieved. The smaller the distance the higher the parameter was ranked.

3.1.11 Experimental Design

Seven DL hyper-parameters were varied for one DNN and four CNN architectures, expanding on the architectures previously explored (19), on three image classification datasets. Table 3.1 gives an overview these hyper-parameters and their optimal ranges (95). SA experiments were conducted on a V100 GPU and had a sample set of size 100 variations per hyper-parameter. The Keras (88) python library was used for model implementation. The measures were normalised using the min-max method to preserve the rank of hyper-parameters.

The Wilcoxon Signed Rank statistical test was employed to compare the rank score achieved by each hyper-parameter to the others to determine whether a hyper-parameter scored significantly higher than the other parameters. The p-value threshold used was 0.05. The Wilcoxon test was chosen as it is a non-parametric test and is, therefore, more appropriate for for the results than the student-t test.

3.1.12 Limitations

The limitations of this methodology, Section 3.1, is related to numbers: the number of architectures, datasets, hyper-parameters and SA methods. It could be argued that the conclusions drawn from this work would be more robust if more had be been explored however, the major constraints of this work were time and computational resources. The nature of these experiments, repeatedly training DL models, is time consuming and without the computational resources to distribute these processes the scope of work had to be tailored. As a result the DL architectures chosen were all popular in literature and varied in composition. The datasets chosen were small, so as not to introduce more issues in terms of time and resources, but varied in complexity and are three of the most well known image classification datasets. The focus of hyper-parameters was concentrated on the training parameters that are introduced at model compilation as keeping architectural parameters in scope would have increased computation time and required more resources for varying the depth and width of the architectures. The model architectures chosen were stat-of-the-art so that their architectural hyper-parameter settings would already be optimal. In terms of SA methods, more could have been included in the framework to contribute to the rank however usually only one SA method would be used and so by computing the two most popular SA methods and taking into account both in the final ranking this work is taking a more holistic approach to SA. The two SA methods chosen are the accumulation of the methods that came before them and are the most widely adopted in literature.

3.2 Rank Informed Hyper-parameter Tuning

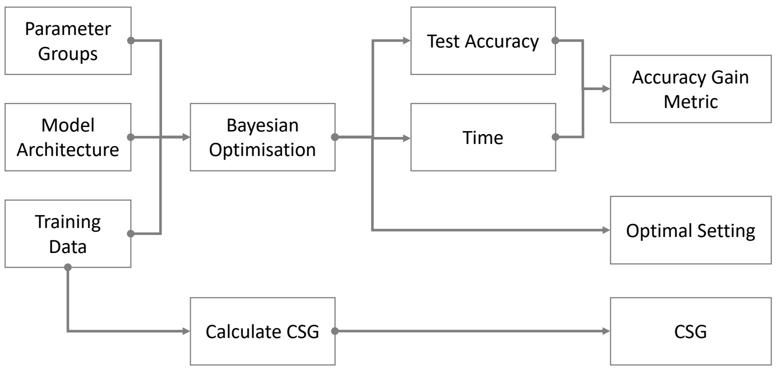


Figure 3.14: Rank Informed Hyper-parameter tuning overview.

The second phase of experimentation, summarised in Fig. 3.14, takes the hyper-parameter ranking produced in phase one with SA and uses it to inform the process of hyper-parameter tuning, HPO. A novel measure, accuracy gain, was created to measure HPO efficiency in terms of tuning time and accuracy achieved and is introduced in more detail in Section 3.2.4. Accuracy gain was considered in relation to dataset complexity, CSG. This facilitates the comparison of tuning efficiency of a variety of parameter groupings to better understand if reducing the HPO search space to influential parameters could reduce time spent on optimisation without compromising accuracy. Furthermore, relationships between tuning efficiency and model architecture or dataset complexity can be identified which could help practitioners make decisions regarding HPO.

3.2.1 Parameter Grouping

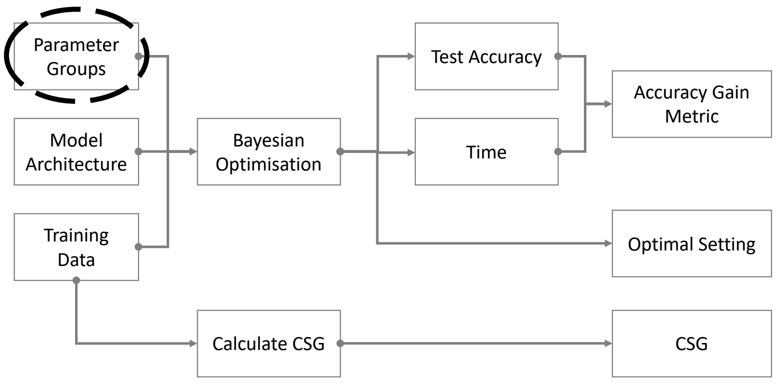


Figure 3.15: Rank Informed Hyper-parameter tuning parameter grouping.

Five groupings of parameters were identified for exploration; all 7 parameters explored by SA (see Table 3.1), the top ranked (batch size), the second ranked (learning rate decay), the top two parameters (batch size and learning rate decay) and the top three parameter (batch size, learning rate decay and learning rate decay step) identified through the SA of CNNs. Defining these groups is the first part of setting the scope of exploration, as shown in Fig. 3.15

The all parameters group represents the traditional approach to hyper-parameter tuning, where the hyper-parameter space is explored in it's entirety. The results of tuning this group will serve as a baseline of performance (time to tune and test accuracy) that the performance of the ranked groups can be compared against. This will then give an indication of whether the performance of the ranked groups is better than tuning all parameters or whether it is faster and less accurate and therefore something practitioners can decide to compromise on for tuning speed.

Four SA influence ranked groups were chosen to explore whether there are optimal groupings of the top ranked parameters that should be considered for tuning. Combinations and subsets of the top three ranked parameters were targeted to tune for performance comparison.

3.2.2 Model Architectures and Training Datatsets

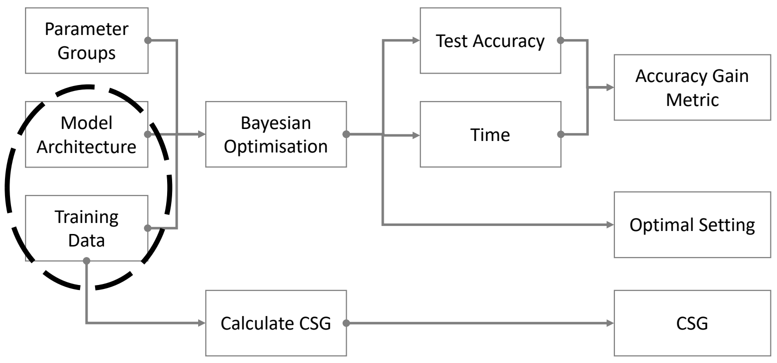


Figure 3.16: Rank Informed Hyper-parameter tuning model architectures and datasets.

Building on the methodology described in Sections 3.1.5 and 3.1.6 the five architectures: DNN, ResNet18, AlexNet, VGG16 and GoogleNet, and the three datasets: MNIST, MNIST Fashion and CIFAR10 continue into the ranked hyper-parameter tuning phase, as shown in Fig. 3.16.

3.2.3 Bayesian Optimisation

Bayesian Optimisation is considered to be the state of the art in hyper-parameter tuning literature and is used to optimise "expensive "black-box" functions" (96) such as neural networks and DL models. This is why this method was chosen, as shown in Fig. 3.17.

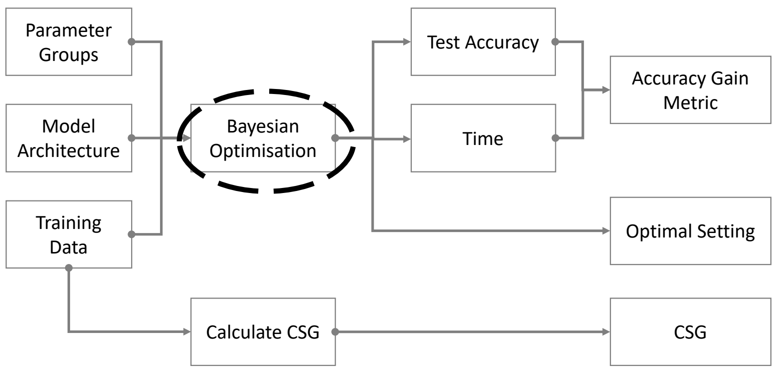


Figure 3.17: Rank Informed Hyper-parameter tuning Bayesian optimisation.

Theory of Bayesian Optimisation

The core principal of Bayesian optimisation is reflected in the following equation which states that the posterior probability $P(y|x_n)$ of a model y, given data x_n is proportional the likelihood $P(x_n|y)$ of observing x_n given model y multiplied by the probability of $P(y)$ (72):

$$P(y|x_n) \propto P(x_n|y)P(y) \tag{3.1}$$

The combination of the distribution of the model $P(y)$ and the data x_n is used to obtain the posterior of the function which can then be used to discover where that function is maximised given some criteria. This criteria, u, is often referred to as the acquisition function and is used to determine the next sample point by aiming to maximise u.

This feeds into the Bayesian optimisation framework which consists of two parts: a probabilistic model and a loss function (97). The probabilistic model reflects the behaviour of the unknown objective function whilst the loss function is the target of the optimisation, either to maximise or minimise depending on its nature.

A key advantage of Bayesian optimisation over other HPO methods is that it is designed to be sample efficient. By employing adaptive sampling strategies Bayesian optimisation can reduce the number of evaluation functions required to find an optimal solution (96).

Bayesian Optimisation Implementation

In this work, the implementation of Bayesian optimisation from the python library hyperopt (98) was used. The approach chosen was the Tree-structure Parzen Estimator (TPE) and was set up to minimise the loss during training. The choices made were to reflect common practice so that the results produced would be useful to a wide-ranging audience.

The hyperopt library provides a framework for implementing hyper-parameter tuning allowing for definition of configuration space and evaluation function, creating a formal approach to the optimisation of DL models. Furthermore, hyperopt was developed with the hyper-parameters of DL models in mind, especially CNNs.

Justification for choosing Bayesian Optimisation

As the state of the art HPO Bayesian Optimisation was chosen as the tuning method as it is considered to be a rigorous approach for the optimisation of DL models (96). Within the application of machine learning and hyper-parameter tuning, Bayesian optimisation has received positive attention in literature over traditional cross-validation methods (97). This approach to HPO suits the scope of this work as it is more successful where there is low to moderate dimensionality (97). By working within a moderate hyper-parameter search space of seven training parameters Bayesian optimisation can perform optimally, though in the context of DL the effect of dimensionality is lessened (97).

3.2.4 Important Measures

To evaluate model performance and understanding the influence of the highest ranked parameters four key measures are considered: dataset complexity (CSG), test accuracy, time and a novel metric accuracy gain (ᛋ).

CSG

Cumulative Spectral Gradient (CSG) is a measure which quantifies the complexity of a dataset based on the overall separability of classes (91). Following the methodology laid out in (91) we calculated 20 CSG values for each dataset to produce a range of complexity for MNIST, MNIST-Fashion and CIFAR10. These ranges facilitate the demonstration of the

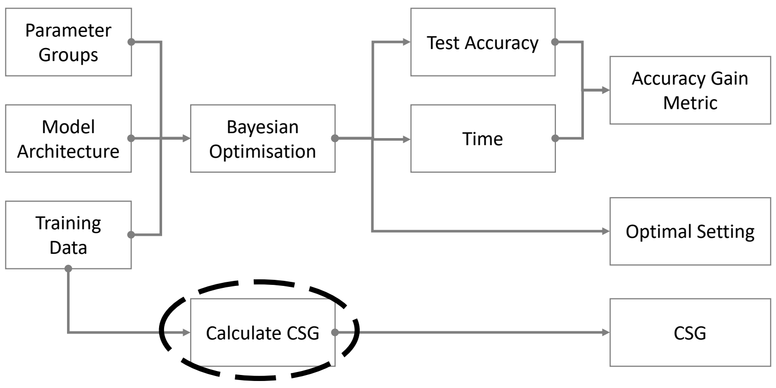


Figure 3.18: Rank Informed Hyper-parameter tuning CSG calculation.

relationship between dataset complexity, CSG, and the accuracy gain, ϟ, of the different parameter groups.

As explained in (91) the ability of a DL model to generalise cannot be based on architecture and parameters alone, the input data also plays a role in this and being able to quantify the complexity can give an early indication of model performance. The CSG measure was tailored for image classification data and CNN architectures where other comparable c-measures which were designed for raw, linearly-separable data. Furthermore, the CSG metric was designed with performance in mind and has a quicker calculation time and as a result is an important measure to be considered as shown in Fig. 3.18.

Test Accuracy

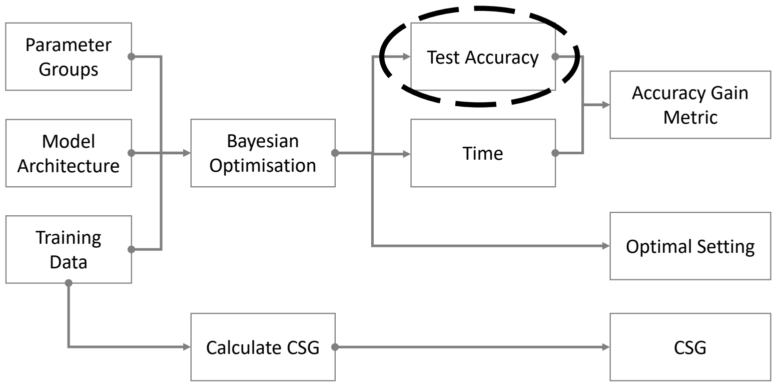


Figure 3.19: Rank Informed Hyper-parameter tuning test accuracy.

When evaluating a DL model, accuracy is a key metric that can indicate the success of training. However, the term accuracy is much broader than it may initially seem and, as a result, may not provide a clear picture of model performance. Exploring accuracy with more granularity reveals differences in training accuracy vs test accuracy. A strength of DL, which contributes to its popularity and adoption, is the ability to generalise to unseen problems and data and a models generalisability is more accurately reflected by test accuracy (99). This occurs because models can adjust their weights to the training data so rigorously that the model achieves very high training accuracy because it has learnt the dataset and therefore cannot perform to the same level on validation data, commonly referred to as over-fitting in the literature. Concentrating on the test accuracy achieved by the model, as shown in Fig. 3.19, gives a better indication on the generalisablability of the model once trained. Ensuring that a model can perform effectively on unseen data is a crucial part of training that needs to be prioritised.

Time

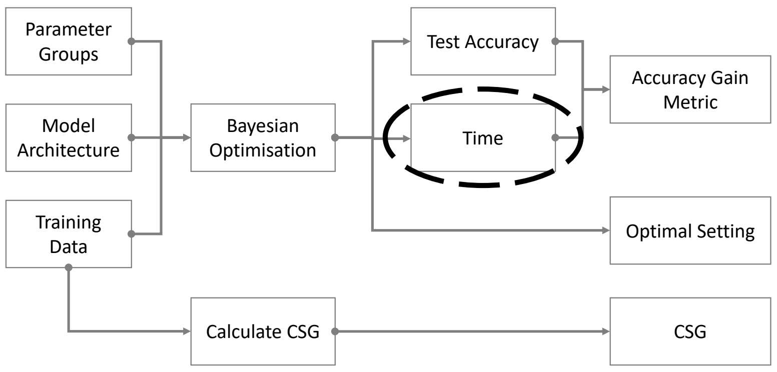


Figure 3.20: Rank Informed Hyper-parameter tuning training time.

Computational cost is an important topic within DL and an element of this cost is the time it takes to train a model. Time constraints can dictate the scope of HPO, training length and as a result the accuracy of the model. Simply, time can be estimated as the real-time implementation of training which, practically, is the most useful to a DL practitioner trying to estimate the cost of work. As model complexity increase, execution time also increases (100) and so understanding the influence of reducing HPO to influential parameters only on run times would be beneficial. So that the time is easily interpretable it will be measured in seconds/minutes/hours, as shown in Fig. 3.20.

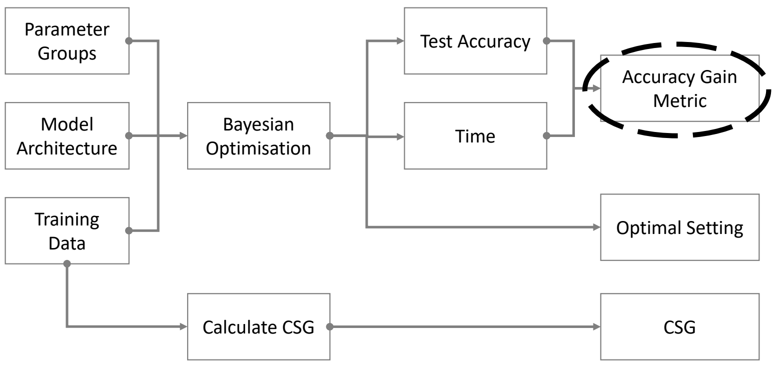


Figure 3.21: Rank Informed Hyper-parameter tuning accuracy gain (ᛃ).

Novel Measure: Accuracy Gain

To quantify the effect a parameters tuning has on model accuracy we developed a measure which conveys the average accuracy gained per unit of time, Accuracy Gain (ᛃ), shown in Fig. 3.21 and defined as;

$$A = a_i - min(a) \tag{3.2}$$

$$T = \frac{\tau_i}{min(\tau)} \tag{3.3}$$

$$A_{T_i} = A \times \frac{1}{T} \tag{3.4}$$

$$\text{ᛃ} = \frac{\sum_{j=1}^{n} \frac{(a_j - min(a))min(\tau)}{\tau_j}}{n} = \frac{\sum_{j=1}^{n} A_{T_j}}{n} \tag{3.5}$$

where a represents a list of accuracies obtained from Bayesian optimisation trials and τ represents a list of timings for those trials. In equation 3.2.4 dividing the top of the fraction by the smallest time value acts as a scaling factor, allowing the measure to be computed and subsequently compared for varying units of time. Doing this allows the comparison of problems which require days vs hours of training so that the resulting measure is not biased to simple/quicker problems. This measure A_T quantifies the efficiency of optimising a set of parameters and can be used to make comparisons between the different groupings of parameters for the different combinations of architectures and datasets. Through this measure it is possible to see whether tuning a specific group of parameters is more time and accuracy efficient than others. Knowing this can help practitioners decide which parameters to tune based on the time and computational constraints they may have.

The rune Jera (101, 102, 103), ᛃ, was chosen to denote Accuracy Gain as it's meaning surrounds the idea of time and cycles, traditionally related to the harvest. Accuracy Gain, ᛃ, as a metric relays the accuracy that can be achieved given time much like what resources a harvest can yield given a set cycle of time and so ᛃ was chosen.

3.2.5 Producing an Optimal Setting

By conducting Bayesian optimisation optimal values for each hyper-parameter value is recommended that should be used for training and the model going forward as shown in Fig. 3.22.

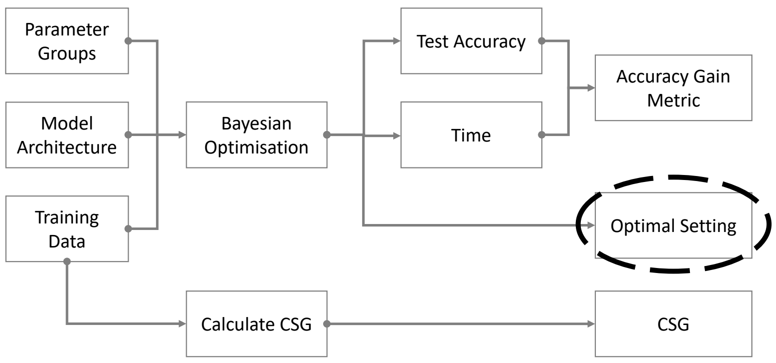


Figure 3.22: Rank Informed Hyper-parameter tuning optimal settings recommendation.

3.2.6 Experimental Design

Five groups of parameters were explored over five DL architectures and three image classification datasets of varying complexities. One hundred trials of Bayesian optimisation were carried out for each combination of parameter group, architecture and dataset and the Bayesian optimisation was implemented using the popular hyperopt python library (98). A combination of a v100 GPU and a distribution of roughly 50 lab machines with Genie VIG830S, Precision T1700 and EU1009695;2110114 were used to run the experiments. The 50 lab machines were split into arbitrary groups of 10 and the 100 Bayesian optimisation trials were distributed across the machines in an arbitrary group for that set of trials. The choice to use a variation of hardware was made to reflect the various set-ups that are realistic for DL practitioners whom we aim to aid through this work. The results showed no significant difference in time between 100 trials run on the v100 or 100 trials distributed across the lab machines.

The Wilcoxon statistical test will be employed here to compare the test accuracies and times achieved by the various parameter groups. This will highlight if a groups performance is significantly better than another. The threshold for the p-value was 0.05.

3.2.7 Limitations

As mentioned above one of the potential limitations was using multiple hardware setups as it could be argued that the results produced are not comparable however there was little to no difference in the GPU results when compared to the distributed set-up approach. The lab machines were not individually compared to the GPU to combat this.

3.3 Case Study

The final phase of experimentation, depicted in Fig. 3.23, takes the outputs from phases one and two and applies it to a real world scenario. The purpose of conducting a case study is to verify the results from the first and second section of methodology and observe whether the results stand and whether SA identified influential parameter groups would improve tuning efficiency outside of the experimental framework created in this thesis. In short, to get an indication of whether the results of this thesis would generalise to other problems.

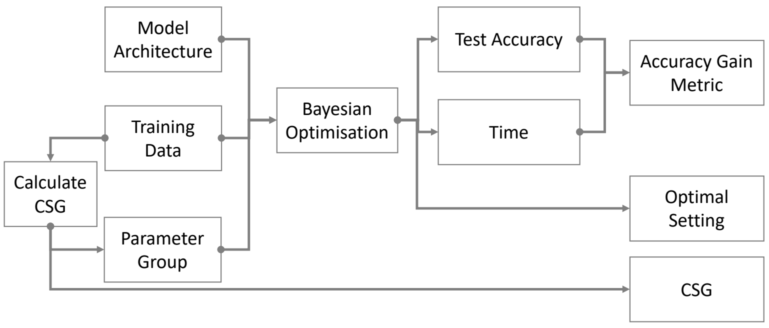


Figure 3.23: Case Study overview.

3.3.1 Application Area

The use of DL in medicine is growing, and medical imaging in particular is exploring the potential of CNNs (104). To better explore how machine learning and DL in particular can be applied and benefit medical imaging, initiatives such as OpenNeuro (105), BioBankUK (106) and The Cancer Imaging Archive (107) have been developed to further research efforts in this area.

Following this vein, the target paper of this case study focuses on the detection of colorectal cancer (108). DL models were trained to classify images as either being benign or malignant and demonstrated reliable, reproducible results that support the benefit of employing DL techniques to medical learning tasks.

3.3.2 Model and Dataset

In the instance of the case study, the model architecture and training data, referred to in Fig.3.24, were dictated by the paper being replicated.

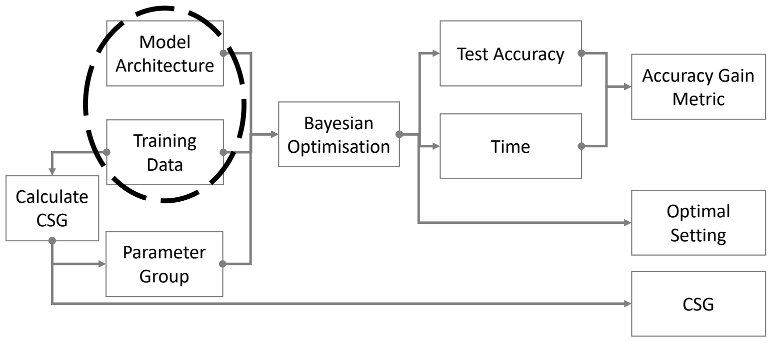


Figure 3.24: Case Study model architecture and dataset.

Model Architecture

The paper (108) focuses on ResNet architectures, specifically ResNet18 and ResNet50. There is overlap here with the work in this thesis where ResNet18 was also explored and so ResNet18 was chosen for the case study as the outputs of SA ranking and the ranked Bayesian optimisation will be directly applicable to this architecture.

Dataset

The dataset used (108) was the Warwick-QU dataset of colorectal cancer images, shown in Fig. 3.25, which was used as part of the Gland Segmentation Challenge Contest (GlaS) (109) in 2015. This dataset consists of 165 images in .bmp format with a 37:48 training split and 37:43 test split (benign:malignant). The images included were collected from University hospitals across Coventry and Warwick in the UK. There is no personal data included, only the images themselves and their classification so there is no need for ethical review of this case study.

Dataset Preparation

The preparation of the dataset followed (108). The images and labels were organised into test and train and converted into grey-scale. Contrast-Limited Adaptive Histogram Equalisation (CLAHE) was applied to improve the contrast in the grey-scale version of the images. Finally, the images were resized to be consistent with one another. Three train test splits were explored (108): 60%:40%, 75%:25% and 80%:20%. The highest accuracy was reported with the 80%:20% train test and so this was chosen for the case study.

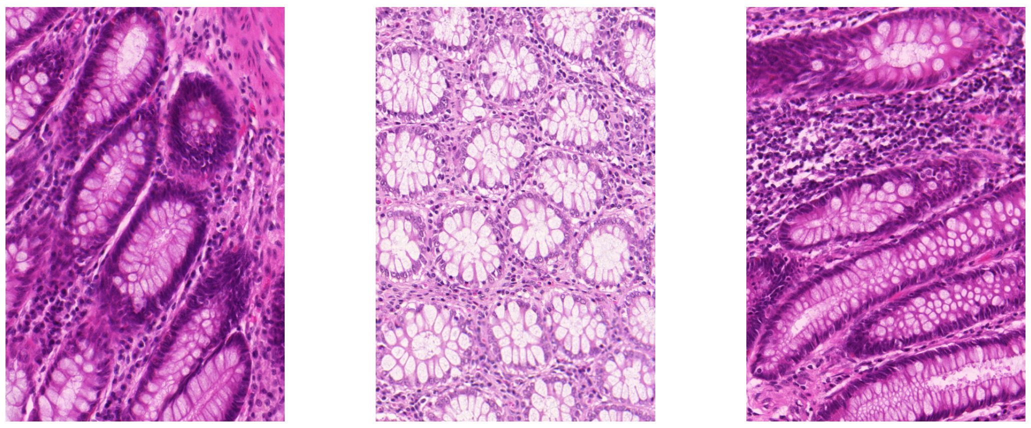


Figure 3.25: Case Study dataset examples.

3.3.3 Parameter Grouping

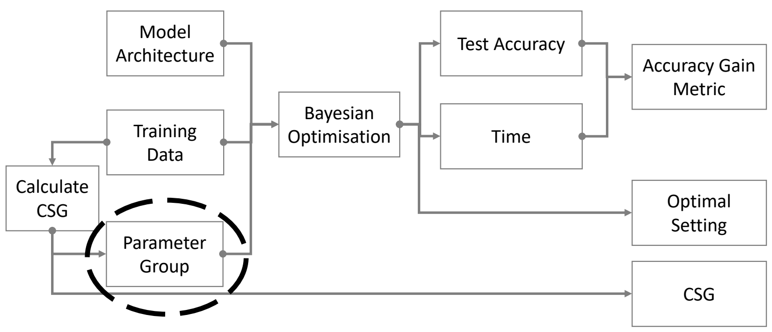


Figure 3.26: Case Study parameter groupings.

The parameter groups need adjusting to reflect those from the paper, as shown in Fig. 3.26 and Table 3.2. The first group of parameters explored were those highlighted in the target work (108), henceforth referred to as the paper parameters. The optimiser used was SGD and the parameters that were tuned were the learning rate and momentum. The loss function used in the paper was binary cross entropy. To keep this as close to the original settings the default optimiser for this group is set to SGD, the loss used was binary cross-entropy. The parameters subjected to tuning are learning rate and momentum.

To determine the top two groups of influential parameters to tune the CSG of the dataset will be calculated and compared against the results of the previous chapters for ResNet18. This highlighted batch size only and top three influential parameters (batch size, learning rate decay and learning rate decay step) as having the best potential for the complexity of the data set and architecture combination.

Table 3.2: Case study parameter group definition summary.

Groups	Parameters
Paper	Learning rate, momentum
Top	Batch size
Top Three	Batch size, learning rate decay, learning rate decay step
All	Batch size, learning rate decay, learning rate decay step, optimiser, momentum, learning rate, epochs

The final group includes all hyper-parameters from the original scope of this work.

3.3.4 Hyper-parameter Tuning

As explained in further detail in Section 3.2.3, the hyperopt library was used to conduct Bayesian optimisation on the above parameter groups. For each parameter group 100 trials of HPO were completed and the test accuracy, and time of each trial was recorded for evaluation, as shown in Fig. 3.27.

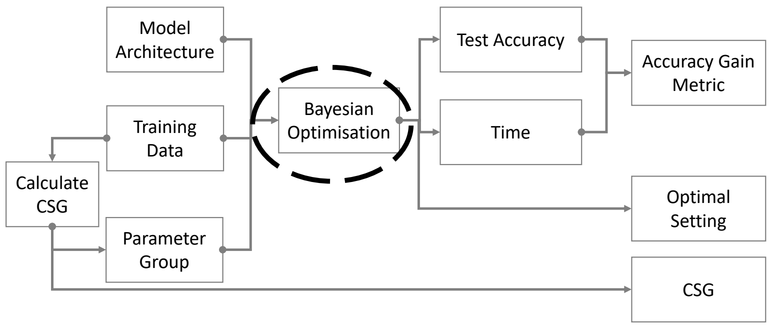


Figure 3.27: Rank Informed Hyper-parameter tuning CSG calculation.

3.3.5 Case Study Evaluation

The evaluation of the case study consisted of producing the test accuracy, recording the time to conduct the tuning and computing the Accuracy Gain ᔕ, as shown in Fig. 3.28.

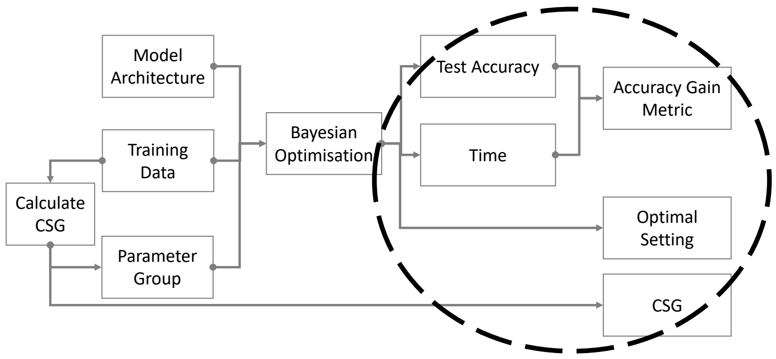


Figure 3.28: Rank Informed Hyper-parameter tuning CSG calculation.

Test Accuracy

The test accuracy was recorded as the main measure of model success and is a commonly used metric (110). Test accuracy indicates model performance on validation data that was not used in the training of the model. This not only shows how accurate the model is but, unlike training accuracy, it also shows that the models performance is less likely to be because it has learned the training examples. As a result, test accuracy as a metric better reflects that the model has learned the problem rather than the training examples.

Precision, Recall and F1-Measure

Classification problem results can be evaluated in the form of a confusion matrix which sorts the predictions made by the model into four categories: True Positive, False Positive, False Negative and True Negative, shown in Fig. 3.29. This can then be used to evaluate the precision and recall of the model which can provide more insight to model performance than test accuracy alone.

Precision, also know as the Positive Predictive Value (PPV), is considered to be a measure of quality. This allows the model to be evaluated in terms of how well it predicts a specific class by representing the percentage of correctly classified positive samples (110).

Recall, also referred to as sensitivity, is considered to be a measure of quantity. This measure evaluates the model in terms of how many times it recognised a specific class.

F1-measure combines both precision and recall and is the harmonic mean of both of these measures, emphasising the importance of both (3). If either precision or recall is low

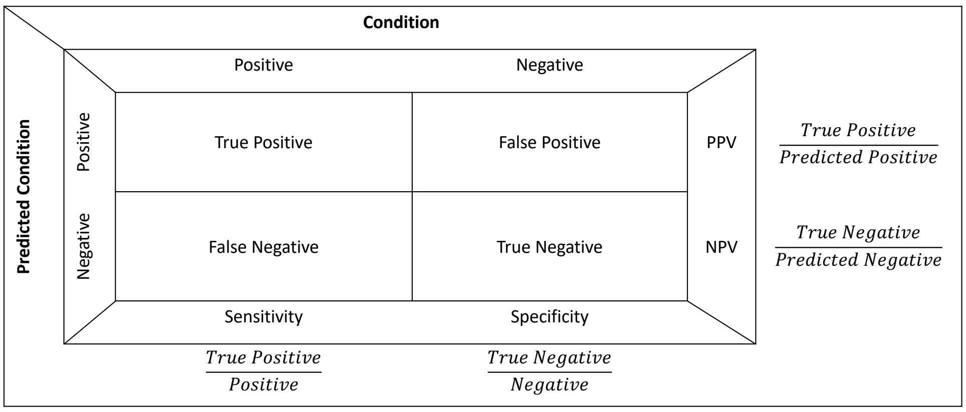

Condition
Positive
Negative
Predicted Condition
Positive
Negative
True Positive
False Positive
False Negative
True Negative
PPV
NPV
$\frac{True\ Positive}{Predicted\ Positive}$
$\frac{True\ Negative}{Predicted\ Negative}$
Sensitivity
Specificity
$\frac{True\ Positive}{Positive}$
$\frac{True\ Negative}{Negative}$

Figure 3.29: Accuracy measures for classification problems. PPV: Positive Predicted Value. NPV: Negative Predicted Value. (After Dinga (3))

then the F1-measure will be and so both need to be high in order for a model to achieve a high F1 score. The higher the F1 score the better the model performance.

Time

As previously, the HPO trials were measured in real-time (seconds/minutes/hours) so as to evaluate the time to complete in a way that DL practitioners would find relatable. Though clock-cycles may facilitate comparisons of the hyper-parameter groups more thoroughly the aim here is to quantify the time saving, if any, to the practitioner and so seconds, minutes and hours of computation time were used for evaluation.

Accuracy Gain

The novel metric, Accuracy Gain ᔓ, was used to evaluate the efficiency of the influential parameter groups against those laid out in the original work. The higher the Accuracy Gain, the more efficient the HPO of the parameter group.

CSG

Dataset complexity, CSG, was calculated to choose the top two parameter groups that should be most efficient in terms of Accuracy Gain ᔓ as early results suggested the complexity of the input data effected which parameter groups were most efficient (effected Accuracy Gain ᔓ). This can also be used to make direct comparison against the results of the case study and the previous works with the various datasets from previous experiments.

3.3.6 Experimental Design

The case study experiments were completed using Google Colaboratory (Google Colab) cloud solution. The associated GPU that was allocated as part of the cloud service was the Tesla P100-PCIE-16GB. For these experiments there was 12GB-16GB of RAM available. This service has been widely adopted allowing for general access to high-performance computing that many ML practitioners will now be able to utilise. Google Colab is also a simpler, more cost-effective option (111) compared to purchasing and setting up the hardware directly. 100 trials of HPO were complete for each parameter group identified: paper, top, top three and all.

The Wilcoxon statistical test will be used to identify parameter groups that performed significantly better than others in terms of test accuracy, time, precision, recall and F1-measure. The threshold observed for the p-value will be 0.05.

3.3.7 Limitations

The case study was limited by and to the information available in the original paper. This study was chosen as it provided details on dataset pre-processing and HPO however it did not share the final settings chosen for the other parameters and so these had to be assumed. Furthermore, there were no specifics regarding image sizes to follow and so this may have differed from the original work. These potential deviations from the original study could explain any differences observed in model performance. To mitigate this limitation the original experiment with the described HPO was replicated and the results produced were used to compare against the influential parameter results.

3.4 Summary

The methodology of this work was split into three distinct sections, the results of which were designed to feed into the next. The initial stage, SA of CNN Hyper-parameters, outlined a novel framework, SADL, to enable SA be conducted for the hyper-parameters of CNN models where the image classification dataset and model architecture were changeable. The aim of this was to produce a general ranking of popular DL training hyper-parameter influence on CNN model accuracy. The second stage, Rank Informed Hyper-parameter Tuning, introduced a novel metric - Accuracy Gain ϟ to evaluate HPO efficiency, and conducted tuning on various parameter groups to compare tuning performance of all parameters against that of the most influential parameters. Finally, the results of the first two stages of work were applied to a real world case study to better understand the significance of SA of CNN hyper-parameters on HPO efficiency. Calculating the Accuracy Gain for the SA identified influential parameters for a new architecture/dataset combination, with the aim to compare the efficiency and final test accuracy against that reported in the chosen paper aims to show whether the results from earlier in this thesis stand and are generalisable.

Chapter 4

Sensitivity Analysis of Convolutional Neural Networks

This chapter begins with the implementation of the SADL framework and any deviations from the methodology set out in Chapter 3, Section 3.1. This is followed by a presentation of this chapters results and a discussion. This is then summarised into a conclusion at the end of the chapter which feeds into the next avenue of exploration.

4.1 Implementation

The implementation of the SADL framework followed the methodology laid out in Chapter 3, Section 3.1.

4.1.1 Software Re-usability

The modular design and implementation of the SADL framework allows for it's re-use to conduct SA for other CNN architectures and image datasets. With minimal modification it could also be adapted to alternative architectures and tasks. This would also allow for the addition of other SA and sampling methods to the framework in the future. SADL was implemented with the Keras python library in mind and therefore can not be used for models created using alternative libraries such as PyTorch.

4.1.2 Sensitivity Analysis

The code for the implementation of the Morris method and Sobol's Indices was influenced by the SAlib python library (94, 112). Where the SAlib library implements specific sampling methods for each SA method, the SADL framework uses one sample and one set of inputs and outputs to calculate both sets SA measures. SADL was implemented this way to reduce computation time by not duplicating the sampling and training trials for the various SA methods.

4.1.3 Resource Constraints

There were elements of running the experiments that did not run as smoothly as anticipated as a result of resource constraints. The SADL framework training loop was programmed to take the hyper-parameter settings for each trial, compile, build and train the model and record the achieved accuracy 100 times for each parameter. The total number of trials for each model architecture and dataset combination was 700 (7 parameters) and these 700 trials were designed to be completed iteratively in one run of the framework. A memory leak in the Keras fit function and the memory intensive nature of the experiments resulted in Out Of Memory (OOM) errors which disrupted experiments. To combat this the code was optimised as much as possible to reduce the use of memory for variables and conserve it for the training loops. Additionally, the input and output of each trial was written to file so that if an OOM occurred the experiment could be continued from the point where it was interrupted with minimal loss of data.

4.1.4 Agile Development

An Agile approach was taken to the development of the SADL framework. The Agile method of software development promotes speed and adaptability (113), repeating the software design life-cycle iteratively and bringing the resulting outputs of these "sprints" of work together into a final product. As SADL was designed to be modular, developing it in this way was complimentary to the nature of the desired output. Each module was designed, implemented and tested in turn: sampling, model architectures, training data pre-processing, SA methods, model fit/train and capturing the outputs. The final iteration brought the modules together and integration testing was completed to ensure that the various parts of the framework were compatible and passing data correctly. An advantage of taking this approach was the ability to adapt to change when necessary. The issues introduced by resource constraints would not have been as easy to mitigate if a waterfall approach had been taken to implement the framework as the testing that uncovered the issues would have taken place much later in the development cycle. By approaching this in an Agile way, when this issue arose, it was relatively simple to adjust aspects of the modules to add mitigating code. Furthermore, it was possible to prioritise the implementation of the fix without too much disruption to the project plan.

4.1.5 Informal Testing

Testing of the framework was completed as it was developed. Results were validated using understanding of the SA methodologies and realistic outputs and results from other implementations of the methods such as SALib. As the framework was implemented in a modular way each module was tested and then the integration of the module was tested. As is common in many solo developed projects many issues were resolved as they arose rather than as a result of strict testing, with bugs and errors being dealt with organically as they became apparent. Several versions were developed during the implementation process improving on flaws to produce a final, robust framework.

4.2 Results

This section presents the results of the SA conducted on the CNN architectures: DNN, ResNet18, AlexNet, GoogleNet and VGG16, on all three datasets: MNIST, MNIST Fashion and CIFAR-10. Fig. 4.1 shows the accuracy of the models for each value of the parameters trialled to show how the variance in the parameter settings resulted in variance in the model accuracy. The rows represent the parameters, whereas the columns represent the datasets and the colours indicate the architecture. The inference from these charts can be considered a simple form of SA on its own, however they cannot consider the influence of a parameter in terms of its interactions which is included in the SA measures. The two categorical parameters, batch size and optimiser, are displayed as point plots where the point is the mean and the lines are the standard deviation. The numerical parameters are displayed as line plots where the central dark line is the mean and shaded area surrounding

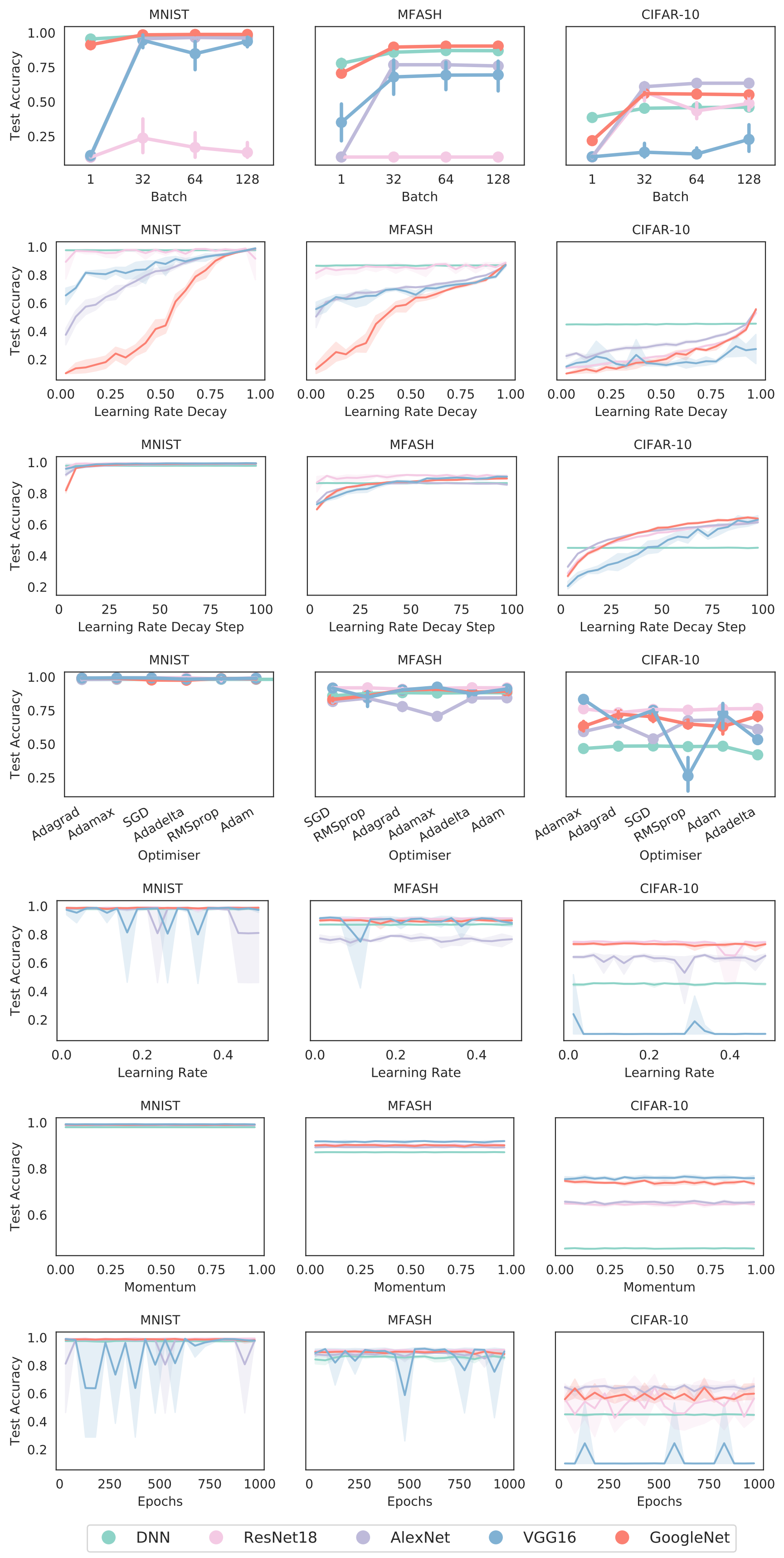


Figure 4.1: Test accuracy mean and standard deviation of hyper-parameter trials at varied values to give early indication of influence. The straighter the line the less influence the parameter will be expected to have. See Table 3.1 for hyper-parameter details.

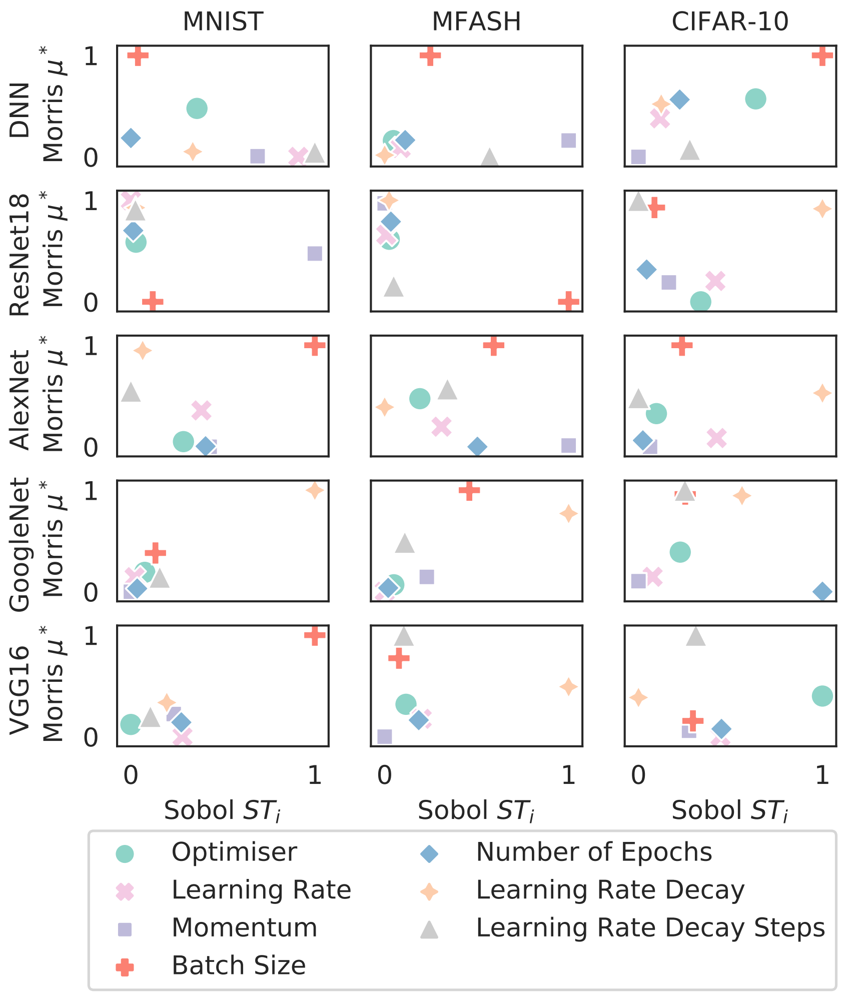


Figure 4.2: ST_I and μ^* measures, column indicates dataset and row indicates architecture. Top-right corner of the plot indicate high rank. See Table 3.1 for hyper-parameter details.

it represents the standard deviation. Fig. 4.2 shows the SA measures that were produced, μ^* and ST_i. Each measure, ranging between 0-1, indicates high influence with a higher score. The columns represent the datasets and the rows represent model architectures. The x-axis is the Sobol measure and y-axis is the Morris measure. Scores in the top-right quadrant of the chart shows that both measures have ranked the parameter as having high influence and if it falls into the bottom-left both measures agree that parameter has low influence. If the parameter falls into either the top-left or bottom-right quadrants this shows that the two methods have not agreed on the ranking of the parameter. This chart is used as an early indicator of importance before the final ranking is produced.

4.2.1 Sensitivity Analysis

The initial results, shown in Fig. 4.1, gives an early indication of the influence of each parameter on the accuracy of a model before the SA measures are calculated. For each architecture, indicated by the coloured lines, on each dataset, represented by the columns, model accuracy is shown for each variation in parameter values. The rows in the chart are the different parameters. The more variation seen in the lines that represent the architectures the more influence the parameter can be said to have. This is observed consistently for batch size, row 1 in Fig. 4.1, and learning rate decay, row 2 in Fig. 4.1, across architectures and datasets. Conversely, there is much less variation observed for learning rate, row 5 Fig. 4.1, particularly for the GoogleNet, DNN and ResNet18 architectures. There is some variation observed for VGG16 and AlexNet, moreso for the MNIST dataset. Momentum, row 6 Fig. 4.1, has the least variation of all parameters across all architectures and datasets indicating low influence.

DNN SA results are shown in Fig. 4.2 row one. Batch size was clearly ranked as most influential on the CIFAR-10 dataset as it is located in the extreme top-right of the chart. It is also ranked highly influential on the MNIST and MNIST Fashion datasets by the Morris measure. Optimiser is also influential for the DNN on the MNIST and CIFAR datasets. The lower ranked parameters consist of number of epochs, which is consistently in the bottom left quadrant of the charts and learning rate is also ranked low for the MNIST Fashion and CIFAR-10 datasets contradicting the importance placed on tuning the learning rate in DL and hyper-parameter tuning literature.

As seen in the DNN results, CIFAR-10 has a clear most influential hyper-parameter for the ResNet18 architecture: Learning Rate Decay. Fig. 4.2, row two, shows that the learning rate itself was ranked amongst the least influential parameters. This result highlights the possibility that the parameters effecting learning rate are more influential than the initial learning rate itself despite the emphasis placed on learning rate in the literature. Momentum is also generally ranked highly for ResNet18 supporting the notion that factors affecting learning rate are more influential.

The results of AlexNet are shown in Fig. 4.2, row three. Batch size is ranked most influential on the MNIST and MNIST Fashion datasets and is ranked second for CIFAR-10 suggesting a generally high influence on AlexNet's test accuracy. Batch size extends to other architectures, ranking most influential for multiple datasets for the DNN architecture. This suggests a potential correlation between optimal batch size and good test accuracy, independent of CNN architecture or dataset. Conversely to observations of the previous architectures, learning rate decay is ranked lowest for MNIST Fashion despite it's influence on AlexNet for the other datasets and in previous experiments. This presents the possibility that the complexity of the dataset can effect the influence of parameter. Once again, learning rate's influence is low, contrary to expectations set by the literature. Number of epochs is also consistently ranked as having low influence.

The GoogleNet results share a common most influential hyper-parameter, learning rate decay, across all datasets. There is also agreement on this by both Morris and Sobol, indicated by its position in the top-rightmost quadrant in Fig. 4.2, row four. Learning rate decay has also been ranked as influential in previous experiments which suggests a potentially generally influential parameter. As observed consistently, learning rate is amongst the lowest ranked hyper-parameters.

VGG16 on the MNIST datatset is most influenced by batch size. It is the most influential parameter by far as shown in Fig. 4.2, row five, as all other parameters are grouped in the bottom left corner of the chart. The second most influential parameter for the MNIST dataset is learning rate decay which is the most influential parameter for the MNIST Fashion dataset. The most influential parameter on VGG16 for the CIFAR dataset was the optimiser. Learning rate is amongst the lowest ranked hyper-parameters once again for all experiments.

These results were analysed to identify patterns of influence of the hyper-parameters. Firstly, a generalised score was calculated for each parameter which determined the final ranking, shown in Table 4.1.

4.2.2 Ranking

Table 4.1 summarises the final rank of hyper-parameter influence taking into consideration the results of both SA methodologies. Results highlighted in bold represent the most influential parameter for that combination of architecture and dataset. The final column takes an average of parameter scores and represents the final ranking produced. Batch size was ranked as most influential overall, having the lowest average distance from perfect SA scores, followed closely by learning rate decay. The Wilcoxon Signed Ranks test confirmed that the ranking of batch size was significantly higher than all other parame-

Table 4.1: SA Euclidean distance from best score, (1,1), to actual score, (μ^*, ST_i). High influence indicated by smaller distance.

	DNN			ResNet18			AlexNet			VGG16			GOOGLENET			
Parameter	M	MF	C	M	MF	C	M	MF	C	M	MF	C	M	MF	C	Average
Batch Size	0.90	**0.75**	**0.17**	1.33	1.00	0.92	**0.00**	**0.41**	0.76	**0.00**	0.95	1.10	1.06	0.54	0.75	**0.71**
Learning Rate Decay	1.14	1.40	0.46	0.98	**0.98**	**0.08**	0.94	1.17	**0.47**	1.04	**0.51**	1.17	**0.00**	**0.23**	**0.44**	0.73
Learning Rate Decay Steps	0.96	1.09	0.69	0.98	1.27	1.00	1.10	0.79	1.12	1.20	0.90	0.69	1.20	1.03	0.75	0.98
Optimiser	**0.84**	1.28	0.41	1.06	1.05	1.20	1.19	0.96	1.13	1.33	1.12	**0.60**	1.23	1.33	0.98	1.05
Learning Rate	1.00	1.29	0.53	1.00	1.05	0.98	0.89	1.06	1.08	1.23	1.15	1.14	1.30	1.41	1.26	1.09
Momentum	1.04	0.85	0.74	**0.52**	1.00	1.16	1.15	0.99	1.37	1.09	1.41	1.19	1.41	1.15	1.34	1.10
Epochs	1.25	1.23	0.43	1.03	0.99	1.17	1.16	1.12	1.35	1.12	1.17	1.07	1.37	1.37	1.00	1.12

Note: Dataset names abbreviated in above table as M for MNIST, MF for MNIST Fashion and C for CIFAR-10.

ters apart from learning rate decay, which was ranked second, with a p-value threshold of 0.05, as shown in Table 4.2. Batch size was ranked highest on the shallower networks, DNN and AlexNet, whereas learning rate decay was ranked highest on deeper networks, ResNet18 and GoogleNet. In the middle, VGG16 was highly influenced by both top ranking parameters depending on the dataset.

As observed in the SA results, learning rate was ranked amongst the least influential parameters, and is third-least influential overall. Momentum is also ranked as having little influence, which is expected due to the lack of variation shown in Fig. 4.1, however does rank highest in one instance which could be attributed to its interactions which are not reflected in Fig. 4.1 and redeems it from being the least influential parameter overall. Number of epochs is the least influential parameter, ranking last overall.

4.3 CNN Sensitivity and Patterns of Influence

The most influential parameter across all architectures and datasets was batch size as shown in Table 4.1. As batch size effects stochastic gradient descent learning algorithms that are widely used in DL (95) this could explain the sensitivity CNNs showed to this parameter. For example, the impact of optimal batch size was demonstrated when ResNet50

Table 4.2: Results of the Wilcoxon Signed Ranks test: p-values for each pair of parameters to demonstrate whether the Euclidean distance rank (and thus influence) of one hyper-parameter is significantly different from that of another hyper-parameter across the datasets and model architectures explore. P-value threshold is 0.05.

	Batch Size	Learning Rate Decay	Learning Rate Decay Step	Optimiser	Learning Rate	Momentum	Epochs
Batch Size	-	0.88866	0.03	0.0198	0.00318	0.01108	0.00452
Learning Rate Decay	0.88866	-	0.11642	0.06876	0.0536	0.0536	0.02144
Learning Rate Decay Step	0.03	0.11642	-	0.17384	0.11184	0.17384	0.03572
Optimiser	0.0198	0.06876	0.17384	-	0.37886	0.56868	0.267
Learning Rate	0.00318	0.0536	0.11184	0.37886	-	0.71138	0.6672
Momentum	0.01108	0.0536	0.17384	0.56868	0.71138	-	0.77948
Epochs	0.00452	0.02144	0.03572	0.267	0.6672	0.77948	-

was trained on ImageNet with 76% accuracy by only increasing the batch size (114). It was also observed that batch size plays a more important role than CNN depth and it was concluded that batch size tuning should be prioritised over network architecture parameters (77). The importance of batch size to CNN performance reported in the literature is supported by the SA conducted, where it was ranked either most or second most influential in the majority of experiments and was considered to be the most generally influential on CNN accuracy. The results of comparing the achieved distances by each parameter to best score using the Wilcoxon Signed Ranks statistical test, reported in Table 4.2, showed that batch size was significantly close than all other parameters but learning rate decay, confirming that the ranking had correctly determined the most influential parameter. The p-values ranged from 0.00453 – 0.03, meeting the 0.05 threshold to reject the hypothesis that the scores achieved were the same. This suggests that batch size should be prioritised when conducting hyper-parameter tuning on CNN architectures.

Learning rate decay ranked second most influential overall, as shown in Table 4.1. Learning rate decay is believed to aid in learning complex patterns (115) and it was observed that it's influence was greater the more complex the architecture of the CNN and the more complex the dataset. Learning rate decay is employed to aid models in avoiding local minima when training (115), which enables them to achieve greater accuracy which was observed in the SA results. The CNNs were sensitive to learning rate decay as certain values would allow them to perform much better than others, accounting for its overall rank. It's position in second place can be explained by the varying levels of influence it had depending on the level of complexity present. It was generally less influential on the simpler DNN whereas it was highly influential on the largest architecture GoogleNet. Furthermore, the additional complexity presented by the CIFAR-10 dataset saw learning rate decay have additional influence on ResNet-18, which was not as noticeable for the simpler MNIST and MNIST-Fashion datasets.

Learning rate was third least influential, contrary to it's importance reported in the literature. The low levels of influence of learning rate was a pattern across datasets and architectures suggesting CNNs are not particularly sensitive to the initial learning rate. This would suggest a more effective approach would be to start with a larger learning rate and adjust it with an optimal learning rate decay. Following this advice has the potential to yield better CNN test accuracy and reduces the number of parameters to subject to hyper-parameter tuning.

Batch size was ranked higher on the shallower models explored, DNN and AlexNet, whereas the influence of learning rate decay was greater on deeper models, ResNet18 and GoogleNet. The pattern that emerges in relation to this suggests deeper models are more susceptible to convergence speeds whereas shallower models are more susceptible to the stochasticity of the learning process.

The margin of separation between the most and the least influential parameters was greater for more complex datasets. Experiments conducted on CIFAR-10, the most complex dataset, showed that the range between values of SA measures for the most and least influential parameters was larger compared to the range of SA measures obtained for MNIST and MNIST Fashion. This result would suggest that for more complex datasets tuning the most influential parameter alone will improve performance and can preserve resources.

These results can also be applied more generally and can be used to make recommendations for future hyper-parameter tuning attempts for CNN architectures. The results of the SA suggest that tuning the top ranked parameters would allow practitioners to reduce the parameter space under consideration without compromising on model performance.

4.4 Conclusions and Next Steps

We applied the SADL framework to state of the art CNN architecture's training parameters to better understand their influence on model accuracy. The results highlighted batch size and learning rate decays as being highly influential across datasets and architectures. Contrary to expectations, the initial learning rate was not considered to be influential and practitioners would benefit from tuning learning rate related parameters, such as learning rate decay, rather than the learning rate itself. Parameter influence was also found to be linked to complexity, that is the more complex and deeper architectures were more sensitive to convergence speeds and the shallower, simpler models were more sensitive to the stochasticity in the learning process. Additionally, the dataset complexity affected the margin of separation between sensitivity measures of the most and least influential parameters suggesting that tuning the most influential parameter alone will benefit performance more whilst preserving resources. The parameters were ranked in the following order based on the SA conducted:

1. Batch Size
2. Learning Rate Decay
3. Learning Rate Decay Steps
4. Optimiser
5. Learning Rate
6. Momentum
7. Epochs

The next steps of this work is to apply these results and to use the ranking outlined above to conduct hyper-parameter tuning and thus observe the theoretical cost savings in practice, and understand what ramifications there are, if any, for the model accuracy. This would then be extended to possible case studies to make direct comparisons against real-world scenarios to emphasise to practitioners the benefit of adapting the tuning process to the most influential parameter set.

4.5 Summary

The key findings of the analysis conducted on CNN hyper-parameters would be that the most influential parameter depends on architecture complexity and the optimal group of parameters to tune depends on dataset complexity. Regardless of this, batch size and learning rate decay are both highly influential parameters that should be prioritised when conducting hyper-parameter tuning.

Chapter 5

Rank Informed Bayesian Optimisation

Following on from Chapter 4, the ranking of CNN parameters produced via SA will be used to inform the approach to HPO. Specifically, this chapter explores the efficiency of Bayesian Optimisation for CNN HPO of various parameter groups of differing influence on model performance. The aim of this chapter is to understand whether tuning the most influential parameters is a viable option to reduce computation time without compromising model accuracy.

5.1 Implementation

5.1.1 Resource Constraints

The resource constraints in this instance also needed mitigation by saving the hyperopt trial data as a pickle which could be read in to continue the Bayesian optimisation in the instance an OOM occurred. This did not occur as often where the task was distributed across several lab machines.

5.1.2 Novel Measure: Accuracy Gain

The aim of this chapter was to gauge the efficiency of the HPO of CNN hyper-parameters and whether reducing the parameter search space to the parameters with the most influence on model accuracy would be more efficient. Efficiency in this case would be reducing the time required to conduct Bayesian optimisation without compromising model performance. To answer this question effectively and provide direct comparisons between the chosen parameter groups a quantifiable measure of tuning efficiency was required. This need prompted the development of the Accuracy Gain ϟ measure.

5.2 Results

5.2.1 SA informed Bayesian Optimisation

Bayesian optimisation was applied to five groups of parameters based on their SADL ranking (19); all parameters, top ranked (batch size), second ranked (learning rate decay), top two parameters (batch size and learning rate decay) and top three parameter (batch size, learning rate decay and learning rate decay step) for all architectures and datasets previously explored. Fig. 5.1 shows the results of the Bayesian optimisation trials conducted. The rows relate to the model architectures and columns represent the datasets whereas the colours each reflect a different grouping of parameters. Each data point shows the accuracy the individual trial achieved and the time that trial took in seconds.

Generally, there is a clear difference in the accuracy achieved depending on the dataset the architecture is being trained on. The CIFAR-10 dataset has much lower accuracy for all architectures and MNIST-Fashion achieves lower accuracy overall than MNIST.

The first row of Fig. 5.1 shows the results of the Bayesian optimisation on the DNN architecture. For both MNIST and MNIST-Fasion datasets the trials for batch size only, the top ranked parameter, generally take the longest but also achieve the best accuracies. The second best accuracies are achieved by optimising the top three ranked parameters; batch size, learning rate decay and learning rate decay step. Despite the individual trials for the "all parameters" group being quick, they also produce trials with the lowest accuracy. CIFAR-10 also achieves high accuracies from tuning the batch size only and top three parameters. It differs in that there are fewer batch size trials that achieve the best accuracy compared to the points observed for MNIST and MNIST-Fashion.

There is less distinction in the ResNet18 results, however it is possible to see that the batch size only and top three groups still achieve good accuracy across all datasets. Additionally, even though there are trials from all parameters that achieve higher accuracy

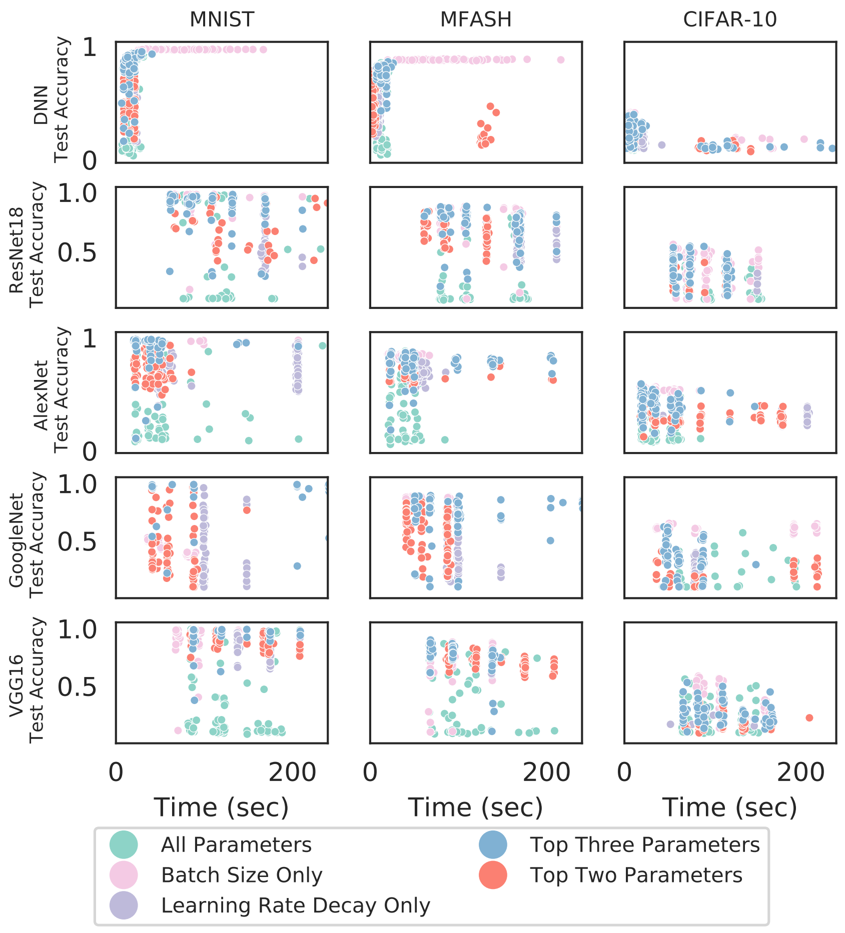


Figure 5.1: The results of 100 trials of Bayesian optimisation on 5 groups of hyper-parameters. The columns are organised by dataset and the rows represent the architectures explored. Each colour represent a different group of parameters. Each data point reports the accuracy and time taken in seconds for each trial of Bayesian optimisation.

there are much more that achieve lower accuracy by comparison to the smaller groupings of parameters. The time taken for individual trials is more varied across the datasets and parameter groupings for the ResNet18 architecture. Across all datasets there is a gap before the trials start showing that even the quickest trials were longer than the majority of the DNN trials.

The trial speeds for AlexNet are generally quicker than those observed for ResNet18 and a little slower than the DNN. Across all three datasets there is a clearer distinction in performance between the group all parameters and the other groupings as there are clusters of all parameter trials in the lower half of the charts indicating low accuracy for many trials. Batch size alone and the top three parameter group are the top performing

Table 5.1: Best Accuracy Achieved in 100 trials of Bayesian Optimisation, highest accuracy is highlighted in bold.

	DNN			ResNet18			AlexNet			VGG16			GOOGLENET			
Parameter Group	M	MF	C	M	MF	C	M	MF	C	M	MF	C	M	MF	C	Average
All Parameters	0.95	0.85	0.32	0.98	0.84	**0.63**	**0.99**	**0.88**	0.55	**0.99**	0.88	0.56	**0.99**	0.87	**0.72**	**0.80**
Batch Size Only	**0.98**	**0.89**	0.42	**0.99**	**0.88**	0.56	0.98	0.87	0.57	**0.99**	0.88	**0.59**	0.53	**0.89**	0.65	0.78
Learning Rate Decay Only	0.87	0.76	0.28	0.94	0.80	0.38	0.95	0.80	0.39	0.97	0.78	0.27	0.96	0.80	0.40	0.69
Top Two	0.87	0.78	**0.47**	0.96	0.84	0.41	0.97	0.82	0.47	0.98	0.85	0.38	0.98	0.84	0.41	0.72
Top Three	0.95	0.86	0.40	0.98	**0.88**	0.55	**0.99**	**0.88**	**0.59**	**0.99**	**0.90**	0.53	**0.99**	**0.89**	0.62	**0.80**

Note: Dataset names abbreviated in above table as M for MNIST, MF for MNIST Fashion and C for CIFAR-10.

groups across the three datasets. Optimising the learning rate decay alone also achieves good accuracy for MNIST and MFASH and is more prominent than previously observed for that grouping.

The GoogleNet optimisation, shown in row 4 of Fig. 5.1, follows patterns previously observed in that the top three parameter group achieves high accuracy. The line of trials that is clear for learning rate decay only on MNIST shows that the trial times are consistent for optimising that parameter. The experiments for all parameters took so long that they are out of scope of the charts which were limited to 240 seconds for comparisons across the architectures.

The final architecture, VGG16, shows clusters of all parameter trials in the lower half of the chart as they achieved lower accuracies in general. Batch Size only and top three parameters achieve high accuracy across all three datasets. There is an obvious decline in accuracy where the trial takes more time.

Table 5.1 and Table 5.2 summarise the results in terms of accuracy achieved in each experiment and the time taken in minutes to perform the 100 trials of Bayesian optimisation in each instance, respectively. The highest average accuracy was achieved by tuning all parameters or the group of top three parameters. It is worth noting that in most instances where the highest accuracy was achieved by tuning all parameters it was matched by tuning a subset, either batch size only or the top three parameters. Furthermore, tuning batch size alone achieved the second highest average accuracy and produced the best results in roughly half of the experiments. The group of parameters that reported the worst

Table 5.2: Time Taken in Minutes to perform 100 trials of Bayesian Optimisation, shortest time taken highlighted in bold.

	DNN			ResNet18			AlexNet			VGG16			GOOGLENET			
Parameter Group	M	MF	C	M	MF	C	M	MF	C	M	MF	C	M	MF	C	Average
All Parameters	170	159	109	5886	3440	3536	839	494	537	1250	1436	**1158**	49456	52163	20739	9424
Batch Size Only	464	574	48	2738	2255	654	366	552	497	1189	1674	1421	703	1189	1514	1056
Learning Rate Decay Only	**35**	**14**	**29**	**563**	**283**	**195**	**298**	**88**	**330**	**482**	**1053**	1810	**254**	**189**	**136**	**384**
Top Two	189	37	49	5312	2058	1522	432	861	691	1480	1717	1311	1112	1228	1276	1285
Top Three	170	163	67	2800	1238	1115	720	614	625	2969	1192	1388	1262	1379	912	1108

Note: Dataset names abbreviated in above table as M for MNIST, MF for MNIST Fashion and C for CIFAR-10.

Table 5.3: Results of the Wilcoxon Signed Ranks test: p-values for each pair of hyper-parameter groups to demonstrate whether the time required to tune a model with one group is significantly different from that required to tune a model with another hyper-parameter group across the datasets and model architectures. P-value threshold is 0.05.

	All Parameters	Batch Size Only	Learning Rate Decay Only	Top Two	Top Three
All Parameters	-	0.06876	0.00318	0.0784	0.06432
Batch Size Only	0.06876	-	0.00148	0.33204	0.8181
Learning Rate Decay Only	0.00318	0.00148	-	0.00214	0.00236
Top Two	0.0784	0.33204	0.00214	-	0.42372
Top Three	0.06432	0.8181	0.00236	0.42372	-

average accuracy was the learning rate decay on its own. Despite achieving the lowest accuracy, learning rate decay was the quickest of all the groups in completing 100 trials of Bayesian optimisation with the Wilcoxon Signed Ranks test, reported in Table 5.3, also showing that it was significantly quicker than all other parameter groups. Table 5.4 shows the average accuracy and times of the architectures across the three datasets, allowing for a direct comparison of the two metrics. Despite having the joint highest accuracy, tuning all of the parameters was the slowest process by far. Tuning the top three ranked parameters achieved the same accuracy as all parameters but executed 8.5 times faster and was the third quickest group overall. Batch size only resulted in the second highest accuracy and the second quickest tuning time.

5.2.2 Accuracy Gain and Dataset Complexity

To understand the efficiency of optimising each group of parameters the measure $\lightning$ was computed to quantify average increase in accuracy gained per time unit whilst optimising each set of parameters, as shown in Fig. 5.2. The first column of Fig. 5.2 considers $\lightning$ in terms of the datasets studied in the experiments whereas column two considers $\lightning$ in relation to dataset complexity and gives an indication of which parameter group to optimise

Table 5.4: Best accuracy and time taken in minutes for 100 trials of Bayesian Optimisation for each dataset and group of hyper-parameters, averaged across the five architectures. The best accuracy and shortest times are highlighted in bold.

	MNIST		MNIST-Fashion		CIFAR10		Average	
Parameter Group	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
All Parameters	**0.98**	11520	0.86	11538	**0.56**	5214	**0.80**	9424
Batch Size Only	0.90	1092	**0.88**	1249	**0.56**	827	0.78	1056
Learning Rate Decay Only	0.94	**327**	0.79	**325**	0.34	**500**	0.69	**384**
Top Two	0.95	1705	0.83	1180	0.39	970	0.72	1285
Top Three	**0.98**	1584	**0.88**	917	0.54	821	**0.80**	1108

given a CSG value.

The larger the value of ϟ the bigger the average increase in accuracy per unit of standardised time. Generally ϟ for all parameter optimisation is low compared to the other groupings across architectures and datasets. Additionally, the efficiency of optimising all parameters is lower on the more complex CIFAR-10 dataset compared to MNIST and MNIST-Fashion and is generally lower than the other groups of parameters. By contrast, optimising Batch size only, the top ranked parameter, has generally high values of ϟ across the state of the art architectures. Batch size achieves a low ϟ for the DNN architecture however when taking into consideration its performance shown in Fig. 5.1 the batch size trials accuracy stays consistently high and so there was little accuracy to be gained. The ϟ achieved by learning rate decay was higher on the simpler datasets, MNIST and MNIST-Fashion, compared to CIFAR10 and it was generally higher for the GoogleNet architecture.

Comparing the ϟ against the CSG value, as shown in the second column of Fig. 5.2, highlights which groupings of parameters were most effective per architecture for varying data complexity. The top three grouping has high ϟ across several architectures for CSG's in the MNIST-Fashion range and it generally has the second largest ϟ for CSG's within the CIFAR10 range. Batch size only performs best within the CIFAR10 range of data complexity and generally outperforms the other groupings on the more complex dataset. As previously observed, tuning all parameters is the least efficient across most architectures and CSG ranges which is supported by having the longest time to tune.

5.3 Discussion

Bayesian optimisation was conducted on five groupings of parameters based on SA rank, produced with the SADL framework (19), across all architectures and datasets. The results showed that tuning subsets of influential parameters would reduce the time taken to conduct Bayesian optimisation without compromising the accuracy achieved by the CNN. Furthermore, a connection was made between the efficiency of the Bayesian optimisation and complexity of the dataset which could aid practitioners in deciding which parameter group to tune based on the data they are using. When prioritising accuracy and speed tuning the batch size alone or the top three SA ranked parameters: batch size, learning rate decay and learning rate decay step, will result in the best accuracy in the shortest

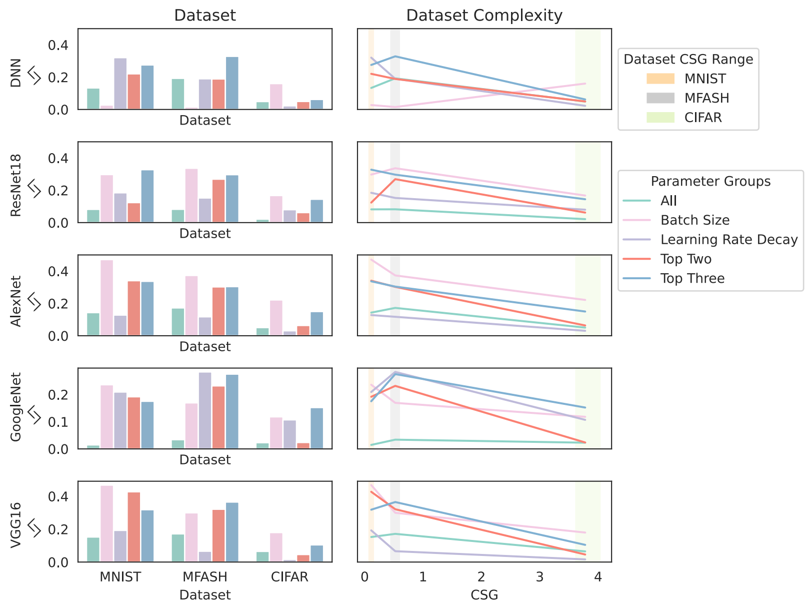


Figure 5.2: The left hand side shows the accuracy gain, ϟ, of tuning the different groups of parameters, grouped by dataset where the rows are the different architectures explored. The higher the ϟ the more efficient the tuning of that group was. The right hand charts compare ϟ to dataset complexity, CSG. The highlight portions are the CSG ranges for the datasets explored. This shows the change in ϟ based on CSG value.

times. If speeding the tuning process is the priority and there is room to compromise on the overall accuracy achieved, then tuning learning rate decay alone was significantly the quickest, completing 100 trials of Bayesian optimisation 24.5 times faster than tuning all the parameters. However, this did result in the lowest average accuracy, 0.69. As learning rate decay affects the convergence speed of a neural network it makes sense that tuning this parameter would increase the speed of the tuning process.

It has been demonstrated previously that increasing the batch size has more impact on model performance than other hyper-parameters (114). This is reflected in the SA influenced Bayesian optimisation where batch size achieved second highest accuracy and the second quickest tuning time, suggesting that concentrating tuning efforts on batch size alone would be an effective and efficient approach. Furthermore, the comparison of accuracy gain, ϟ, against dataset complexity, CSG, showed that tuning the batch size alone worked well on the complex datasets, making this approach well-suited to current research trends in DL. When working with the simpler datasets the combination of the top three SA ranked hyper-parameters showed good accuracy gain and also achieved joint highest accuracy. Tuning this group of parameters was only slightly slower than tuning the batch size alone and would be also be a good option to improve HPO in practise.

The second column in Fig. 5.2 was created to demonstrate the relationship between ϛ and CSG values. Additionally, practitioners could use this chart to inform them on which group of parameters would be most efficient to tune based on the complexity of the dataset they are using. By calculating the CSG value for their dataset they could then, on the chart related to the architecture most similar to what they are using, see which group had the highest ϛ.

The main aim of this work was to aid practitioners and so all practical decisions made aimed to mimic the general practices and, therefore, produce results that could be of benefit to the DL community. The architectures and datasets chosen are state of the art and are commonly used. The hyper-parameters explored are all generally thought to be influential and candidates for tuning. Bayesian optimisation was chosen as it is considered to be the most popular and best method for HPO. These decisions are all strengths of this work. To improve in future, additional datasets and architectures could be explored to further generalise the results and further demonstrate the relationship between hyper-parameter tuning efficiency and dataset complexity.

5.4 Conclusions

As DL grows in popularity and model architectures and datasets grow in size and complexity the cost of HPO is also increasing, forcing practitioners to make compromises that could potentially affect model performance. SA methodology was utilised to identify the most influential CNN training hyper-parameters which we used to inform or implementation of Bayesian optimisation. We compared the accuracy achieved and the time taken to conduct Bayesian optimisation of five varying groups of hyper-parameters: all parameters, top ranked (batch size), second ranked (learning rate decay), top two (batch size and learning rate decay) and top three (batch size, learning rate decay and learning rate decay step), on 5 DL architectures and 3 state of the art image classification datasets. This allowed exploration of whether the efficiency of HPO could be increased by reducing the parameter search space to the SA identified influential parameters without compromising on model accuracy.

Tuning the batch size alone or the top three parameters often matched or beat the accuracy achieved by tuning all parameters in the individual experiments with tuning the top three having the highest average accuracy alongside tuning all parameters. This showed that tuning a subset of most influential parameters could achieve the same performance as tuning all the training parameters. Furthermore, tuning the subsets of parameters was much quicker in every instance than tuning all parameters supporting our hypothesis that SA informed Bayesian optimisation is more efficient than conventional approaches and can still achieve the same results in terms of model performance.

In addition, a connection was observed between dataset complexity and the most efficient hyper-parameter group to tune. Batch size alone was effective on the more complex datasets, especially for the shallower architectures, whereas the top three group performed well on the simpler architectures. Fig. 5.2 was produced to allow practitioners to determine the best group to tune for their own work based on the CSG complexity of their dataset. The right hand side of the chart can be used to see which group has the best accuracy gain ϟ for a given CSG between 0 and 4 for commonly used CNN architectures.

The next steps will be to apply the knowledge from this chapter to a case study to see how this advice performs in a real-world scenario.

5.5 Summary

The key take-aways from this chapter are (1) that SA informed Bayesian optimisation can reduce HPO time without compromising model performance, (2) tuning the batch size alone or tuning the batch size, learning rate and learning rate decay will produce a well performing model in less time than tuning all parameters and (3) that the complexity of the dataset influences which parameter group would result in the most efficient Bayesian optimisation implementation.

Chapter 6

Case Study: Classification of Colo-rectal Cancer

This Chapter applies the findings from both Chapter 4 and 5 to a real world problem. Image classification tasks and CNN architectures are being applied more and more to medical imaging problems and it could be argued that efficiency and accuracy are key elements in the medical sector making this a natural choice for a case study.

6.1 Case Study Justification

The case study was modelled on a paper where ResNet18 was used to identify whether colo-rectal scans showed benign or malignant cancerous cells (108). This specific case was chosen as the architecture used was also explored in this thesis and therefore a direct comparison could be drawn between the results of the case study and that of previous chapters. The dataset used in this paper, Warwick-QU, was openly available and, despite being of a medical nature, required no ethical approval for use as there is no personal or private data attributed to the images. Finally, there was specific mentions of the HPO that had been conducted as part of the work that could be replicated in the case study to make a direct comparison on tuning efficiency against the influential parameter groups.

6.2 Implementation

The implementation of the case study had to be amended slightly to what had been done in previous chapters due to resource constraints and conducting the experiments on the Google Colab platform. Furthermore, some initial issues in the early implementation of the case study led to delays in seeing results.

6.2.1 Data Pre-processing

The initial result of case study showed poor accuracy with the influential groups of parameters not showing any improvement over the other various groups. Investigations into these results showed that the models were choosing the same class every time despite an almost equal test training split in the dataset. This was traced back to the pre-processing of the images where the resizing was obscuring the learn-able features. The resizing was changed to the original image size which allowed for more successful training and the results were more inline with the expectations set by previous chapter results.

6.2.2 Resource Constraints

Having to increase the image size so that the model could learn the features resulted in OOM with larger batch sizes. To compensate for this the default batch size and batch size range were adjusted. Whereas, in previous experiments, the default batch size was 32 and the range included [1,16,32,64,128] these had to be reduced for the case study. As shown below, in Table 6.1, the batch size range was adjusted to [1,2,4,8,16] and the default batch size was changed to 4. The parameter groups are repeated below in Table 6.2 for readers ease.

Table 6.1: Deep Learning Hyper-parameters descriptions, symbols, ranges and default values used for SA experimentation.

Parameter	Description	Range	Default Value
Optimiser	List of gradient descent (GD) algorithms.	Category*	Adam
Learning rate (α)	Initial GD step controller.	[1x10^{-7}, 0.5]	0.001
Momentum (β)	Acceleration factor for GD.	[0, 0.99]	0.6
Learning rate decay (α_{decay})	Reduction rate of (α).	[0, 1]	0.9
Learning rate decay step (α_{d-step})	Number of epochs between Learning Rate Decay.	[1,100]	10
Batch size	Size of training subset for GD update.	Category*	4
Epochs	Number of training cycles.	[5, 1000]	100

Note: Category* indicate that there are two hyper-parameters with categorical ranges (88): (i) optimiser, Adam, SGD, RMSprop, ADAdelta, ADAgrad and ADAmax; and (ii) batch size, 1, 2, 4, 8 and 16.

Table 6.2: Case study parameter group definition summary.

Groups	Parameters
Paper	Learning rate, momentum
Top	Batch size
Top Three	Batch size, learning rate decay, learning rate decay step
All	Batch size, learning rate decay, learning rate decay step, optimiser, momentum, learning rate, epochs

6.2.3 Google Colab Platform

To use the code that had been implemented on the V100 GPU and distributed across the lab machines in previous chapters some adjustment had to be made to run the code on the Google Colab platform. Setting up the environment was slightly different, rather than using a virtual environment to capture the packages and libraries they were installed directly to the colab notebook environment. Additionally, the install of these packages was not always as easy inside the colab environment.

The code that handled file writing to store the experimental results had to be adapted to interface with google drive rather than a native file system. As part of this the drive had to be mounted in the notebook environment at the start of each session to ensure the files could be read in and written to.

The final adaptation required was to the early stopping conditions. It was observed in the initial runs that the models were only being trained for 2 epochs in each iteration. Where the same code on the V100 GPU and lab machines ran for varied numbers of epochs depending on the accuracy for the given early stopping condition which was not the case in the colab environment. To combat this the early stopping patience was increased to 10 to force additional training epochs and this resulted in a more normal, expected, varied number of training epochs for each iteration of training.

6.2.4 Data Formatting

In this chapter, numbers are reported at a more granular level of decimal place compared to previous chapters as there are less numbers to report on and so the formatting and space allows for this. In previous chapters, the number of data points to report meant that they were restricted to two decimal places for the tables to be legible.

Table 6.3: Case study test accuracy statistics for each parameter group over 100 trials of Bayesian optimisation. Best scores highlighted in bold. See Table 6.2 for parameter group definitions.

Parameters	Max	Min	Mean	STD	SKEW
Paper	0.727273	0.333333	0.517576	0.080023	-0.102857
Top	0.787879	0.333333	**0.567273**	0.089081	-0.192065
Top Three	**0.848485**	0.272727	0.555455	0.090611	0.160176
All	0.666667	**0.363636**	0.521212	**0.05437**7	-0.415363

6.3 Results

This section introduces the results of the case study. The tables below show the evaluation metrics, as outlined in Chapter 3 Section 3.3.5. The Wilcoxon Signed Ranks statistical test was employed to infer statistical significance of the results presented.

Table 6.3 shows the test accuracy achieved by the model after the completion of HPO on the various parameter groups. The larger the values of test-accuracy the more successful the HPO. The best test accuracy, 0.85, was achieved by tuning the top three most influential parameters followed by tuning the top parameter, 0.79. The best mean of test accuracy, 0.57, was achieved by the tuning the top most influential parameter, followed by tuning the top three parameters, 0.56. The most successful parameter groups in terms of test accuracy were the two groups of influential parameters, both of which were significant improvements on the parameters used in the paper (p-value of 0.00012 for the top group and 0.00634 for top three) and all parameters (p-value of 0.00012 for the top group and 0.00714 for top three). The worst max accuracy, 0.67, was achieved by tuning all of the parameters despite having the highest minimum accuracy. The accuracy difference between the most successful tuning group, top three, and least successful tuning group, all parameters, was 0.18. Furthermore, both influential groups of parameters out performed

Table 6.4: Case study time (minutes) statistics for each parameter group over 100 trials of Bayesian optimisation. Best scores highlighted in bold. See Table 6.2 for parameter group definitions.

Parameters	Total	Max	Min	Mean	STD	SKEW
Paper	494.875390	12.464958	1.367984	4.948754	2.242040	0.916645
Top	287.714088	**4.720817**	**0.899762**	2.877141	**0.705626**	0.138161
Top Three	253.805812	5.280454	1.225763	2.538058	0.725696	0.838812
All	**208.164265**	5.461838	0.977971	**2.081643**	0.772509	1.292786

Table 6.5: Case study precision statistics for each parameter group over 100 trials of Bayesian optimisation. Best scores highlighted in bold. See Table 6.2 for parameter group definitions.

Parameters	Max	Min	Mean	STD	SKEW
Paper	0.741935	0.209677	0.477897	**0.130789**	-0.241649
Top	0.810345	**0.227273**	**0.570077**	0.134622	-0.213328
Top Three	**0.891304**	0.166667	0.528708	0.180086	-0.103043
All	0.781250	**0.227273**	0.411869	0.165940	0.567739

the parameters tuned in the original paper. The top most influential parameter out performed the paper parameters by 0.06 and the top three most influential parameters out performed the paper parameters by 0.12. The second key performance metric for evaluating the HPO is time, as shown in Table 6.4. The less time taken on tuning the better. The group that completed tuning in the least amount of time was the all parameters group, which also achieved the worst accuracy. The second quickest group to tune was the top three parameters, which also achieved the highest accuracy and was significantly quicker than paper parameters which was the slowest parameter group to tune.

The precision metric indicates the percentage of correctly classified benign samples. Similar to the test accuracy results, the top three parameter group achieved the highest precision, 0.89, and the top parameter group achieved the best mean precision, 0.57. The worst max precision was achieved by the paper parameter group, 0.74, and the lowest mean precision was achieved by the all parameter group, 0.41. The precision results are reported in Table 6.5. The Wilcoxon test confirmed that both groups of influential parameters had significantly higher precision than tuning the all parameters group, as shown in Table 6.8.

The best maximum recall, the ratio of correctly classified benign samples to total be-

Table 6.6: Case study recall statistics for each parameter group over 100 trials of Bayesian optimisation. Best scores highlighted in bold. See Table 6.2 for parameter group definitions.

Parameters	Max	Min	Mean	STD	SKEW
Paper	0.722222	**0.338889**	0.507556	0.079176	0.060406
Top	0.777778	**0.338889**	**0.552333**	0.090069	0.021346
Top Three	**0.833333**	0.300000	0.533833	0.089602	0.606803
All	0.650000	0.350000	0.504556	**0.048543**	-0.118792

Table 6.7: Case study F1-Measure statistics for each parameter group over 100 trials of Bayesian optimisation. See Table 6.2 for parameter group definitions.

Parameters	Max	Min	Mean	STD	SKEW
Paper	0.723206	0.282609	0.465634	0.103585	0.251877
Top	0.780627	**0.312500**	**0.522224**	0.107313	0.123240
Top Three	**0.839024**	0.214286	0.480197	0.119663	0.486358
All	0.645854	**0.312500**	0.405662	**0.081312**	1.005450

nign samples identified, was achieved by the top three parameter group, 0.83, and the best mean recall was achieved by the top parameter group, 0.55. Unlike the precision metric, the lowest max recall was achieved by the all parameter group,0.65, however it did also achieved the lowest mean average, 0.50, as it did with the precision metric. The recall results are shown in Table 6.6. Similarly to precision, the Wilcoxon test confirmed that both groups of influential parameters had significantly higher recall than tuning the all parameters group, as shown in Table 6.8.

The F1-measure represents the harmonic mean between precision and recall and therefore it is unsurprising that the best maximum F1 score was achieved by the top three group, 0.84, and the best mean F1 score was achieved by the top parameter group, 0.52, both signifanctly higher than the all parameter group. As shown in Table 6.7 the all parameters group produced the worst maximum F1 score and the worst mean F1 score, 0.65 and 0.41 respectively.

Understanding the relationship between trial time and accuracy achieved whilst tuning a specific parameter group can give an early indication of HPO efficiency for that group. Fig. 6.1 shows the time taken and accuracy achieved for every HPO trial for each parameter group. This chart clearly shows the parameter group with the worst accuracy overall is the all parameters group, however all trials were also completed in a relatively short time. The paper parameter group has the widest time distribution with the more trials taking a longer time than any other group. The accuracies achieved by the paper parameters group is higher than all parameters but not more than either the top parameter group or top three parameter group. Both the top parameter group and top three parameter group have very similar relationships between time and test accuracy with short trial times a large accuracy range. As shown previously in Table 6.3 the highest accuracy overall is achieved by the top three parameter group.

Table 6.8: Results of the Wilcoxon Signed Ranks test: p-values for each pair of the evaluation metrics explored in the case study for each hyper-parameter group to demonstrate whether one metric value is significantly different from that of another metric value to signify whether there is a significant difference in the performance of a hyper-parameter group. P-value threshold is 0.05.

	Paper Parameters					Top					Top Three					All Parameters				
	Accuracy	Time	Precision	Recall	F1	Accuracy	Time	Precision	Recall	F1	Accuracy	Time	Precision	Recall	F1	Accuracy	Time	Precision	Recall	F1
Paper	-	-	-	-	-	0.00012	0.00001	0.00001	0.00052	0.00054	0.00634	0.00001	0.06148	0.05876	0.47152	0.5287	0.00001	0.00252	0.95216	0.00008
Top	0.00012	0.00001	0.00001	0.00052	0.00054	-	-	-	-	-	0.28462	0.00018	0.03572	0.11876	0.00988	0.00012	0.00001	0.00001	0.00001	0.00001
Top Three	0.00634	0.00001	0.06148	0.05876	0.47152	0.28462	0.00018	0.03572	0.11876	0.00988	-	-	-	-	-	0.00714	0.00008	0.00001	0.01314	0.00001
All	0.5287	0.00001	0.00252	0.95216	0.00008	0.00012	0.00001	0.00001	0.00001	0.00001	0.00714	0.00008	0.00001	0.01314	0.00001	-	-	-	-	-

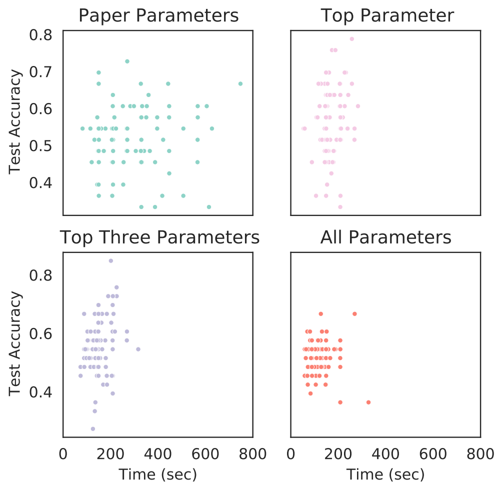


Figure 6.1: Test accuracy achieved and time taken to complete each hyper-parameter tuning trial. See Table 6.1 for hyper-parameter details.

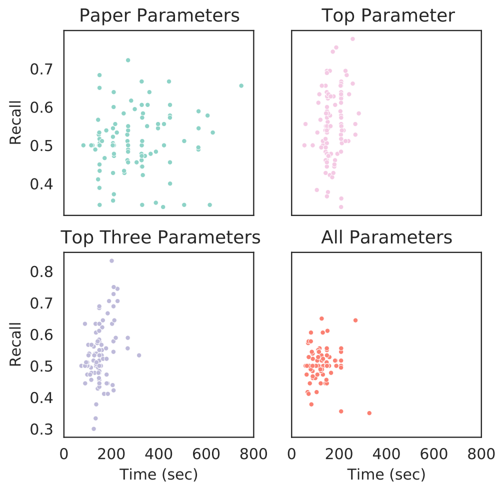


Figure 6.2: Recall achieved and time taken to complete each hyper-parameter tuning trial. See Table 6.1 for hyper-parameter details.

Similarly, Fig. 6.2 shows the relationship between time and recall for the HPO trials. There is no clear pattern for the paper parameter results with a wide breadth of time and recalls. The all parameters group data-points cluster tightly with short trial times and relatively low recall. As with the test accuracy, both the top parameter group and top three

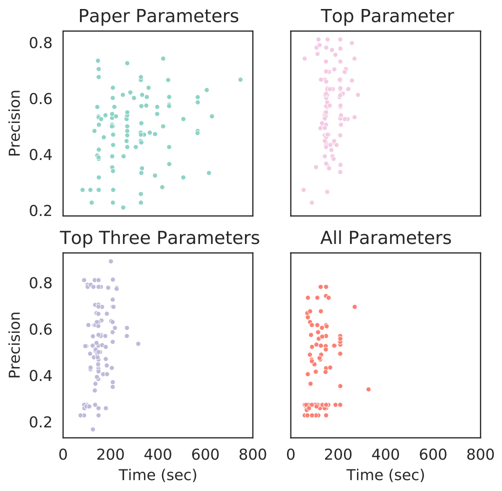


Figure 6.3: Precision achieved and time taken to complete each hyper-parameter tuning trial. See Table 6.1 for hyper-parameter details.

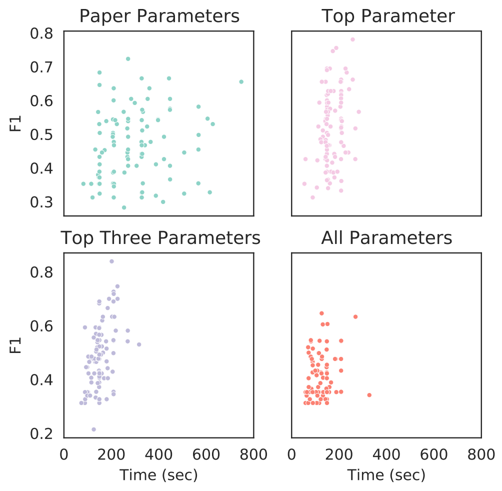


Figure 6.4: F1 achieved and time taken to complete each hyper-parameter tuning trial. See Table 6.1 for hyper-parameter details.

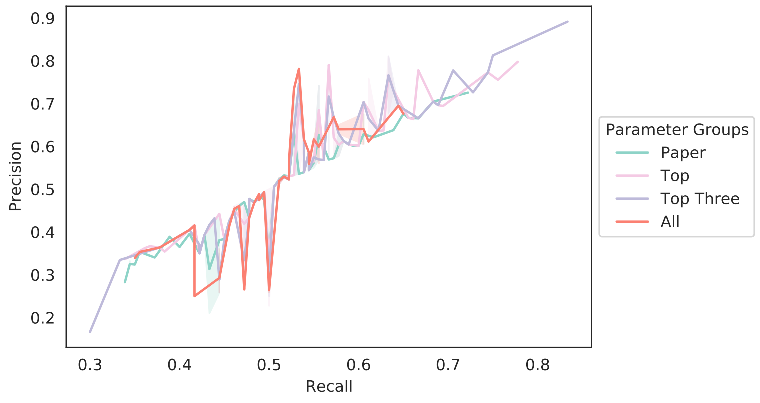


Figure 6.5: Precision and Recall of HPO trials.

parameter group complete trials quickly and achieve the highest recall.

The precision scores are shown alongside trials times in Fig. 6.3. The patterns observed with test accuracy, precision and recall continue where the top parameter group and top three parameter group achieve the highest precision and the paper parameters have broad results in terms of time and precision. However, with all parameters the precision is much more distributed than previously observed.

The F1-measure also follows the patterns seen with the previous measures, as shown in Fig. 6.4. The paper parameters resulted in the largest range in time per trial and the all parameters group had the smallest range in F1. Both the top parameter group and top three parameter achieve the highest F1 measures whilst still having low times per trial.

Precision is plotted against recall in Fig. 6.5. The larger the Area Under the Curve (AUC) the higher the precision and recall for the given parameter group. The AUC values for each parameter group were calculated using both the Trapezoidal and Simpson method and reported in Table 6.9. The top parameter group had the largest AUC value in both instances, 282.19 and 282.24, followed by the top three parameter group, 262.25 and 262.29. The all parameter group had the smallest AUC values, 203.35 and 203.23.

The accuracy gain of each parameter group is shown in Fig. 6.6. The top three parameter group had the highest accuracy gain, followed by all parameters, the top parameter group and finally the paper parameter group. Fig. 6.7 shows the accuracy gain of a subset of the parameter groups from the case study dataset against the results from the previous datasets. Like the other three datasets, one of the influential parameter groups had the highest accuracy gain, in this case it was the top three parameter group. Unlike the other

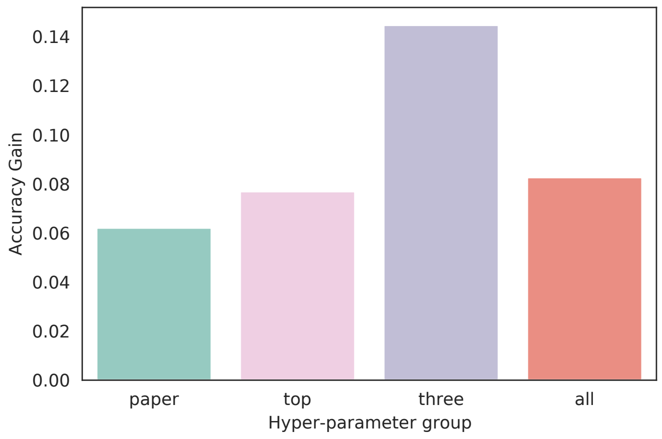


Figure 6.6: Accuracy gain for the parameter groups explored. See Table 6.1 for hyper-parameter details.

datasets, the all parameter had slightly higher accuracy gain than the top parameter group. The addition of the case study dataset, Warwick, to the dataset complexity half of the chart showed that for CSG values close to one the top three parameter group achieves the best accuracy gain.

Table 6.9: Area Under Curve (AUC) for the Precision-Recall curves in Fig. 6.5. See Table 6.2 for parameter group definitions.

Parameters	AUC (Trapezoidal)	AUC (Simpson)
Paper	236.5048769669257	236.46321337166304
Top	**282.196266328103**	**282.23803690655836**
Top Three	262.25444953444287	262.288812827399976
All	203.35484054697164	203.22761734967608

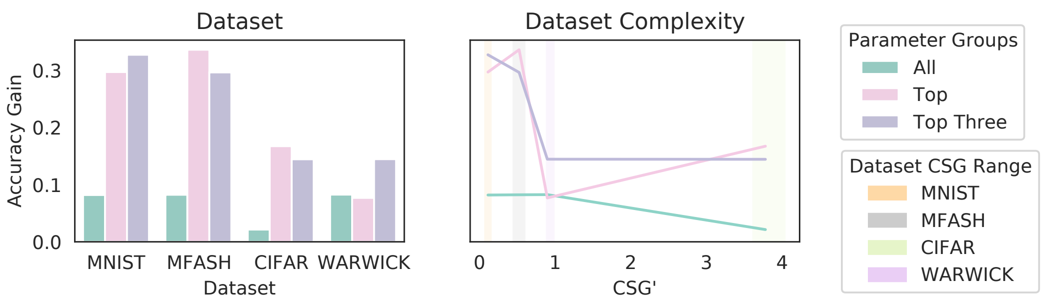


Figure 6.7: Accuracy gain for the parameter groups explored. See Table 6.1 for hyper-parameter details.

6.4 Discussion

The aim of the case study was to apply the theories from the previous two chapters of work to a real world scenario and see if the findings are supported in practice. The key finding of the results showed that applying HPO to a group of influential parameters is more efficient than tuning all parameters.

The top two highest test accuracies achieved were by the two influential parameter groups, Top three and top only, which were significantly higher than that achieved by the paper parameters. The difference between the lowest accuracy achieved by the all parameter group, 0.67, and best accuracy achieved by the top three parameter group, 0.85, shows that applying the tuning effort to the more influential parameters can produce a more accurate model. This supports the hypothesis that the SA identified parameters have more of an effect on model accuracy. In terms of time, the all parameters HPO did complete in less time, 208 minutes, compare to the top three groups, 254 minutes. An additional 46 minutes of tuning for the top three parameter group vastly improved the maximum test accuracy of the model, improving even on the accuracy achieved by the paper parameters in this case study, 0.73, and matched that reported in the original paper (108).

The precision, recall and F1 measures also supported that the model performance was improved when the HPO concentrated on the influential parameters identified by SA, with all three values being significant improvements on the all parameters group. The AUC of the precision recall curves showed that both the influential parameter groups had better precision and recall, 282 for the top parameter group and 262 for the top three parameter group compared to 236 for the paper parameters and 203 for the all parameter group. This suggests that even outside of their influence on the test accuracy these influential parameters contribute to an overall successful model. This continues the pattern that the tuning of influential parameter groups improves model performance.

The accuracy gain metric showed that tuning the top three parameter group was the most efficient. An unexpected result was that the all parameter group was slightly more efficient that the top parameter group. Batch size massively influences training time, additionally the nature of the dataset with medical being quite complex can play a contributing factor (116). The slower training time of the top parameter group may have meant it was a less efficient parameter group to tune however, due to the increased accuracy achieved compared to the all parameter group, it could be argued that the trade off in the additional time for the better accuracy would be worth it. When comparing the accuracy gain of the parameter groups on the case study dataset against the datasets from the previous chapters this is the only dataset where the all parameter does not have the lowest accuracy gain. Despite this, it is clear that performance of a model where the influential parameters are tuned is more successful than the all parameter group.

The results of case study support the conclusions of the previous chapters: that tuning the SA ranked influential parameters is more efficient than tuning all training parameters. The practical implications of this case study is that it confirmed that theoretical results of this work are applicable in real world scenarios.

6.5 Conclusions

The evidence is clear: the SA rank informed Bayesian Optimisation can reduce HPO time without compromising model performance when applied to real world scenarios. For this case study the top three parameter group was the most efficient and achieved significantly higher test accuracy than the non-influential parameter groups whilst being within an hour of the fastest tuning time, and was significantly faster than the parameters tuned in the original paper.

This study did support the hypothesis that tuning a subset of the most influential parameters would benefit practitioners by producing successful models in a more efficient way. However, it did not show that one could accurately determine the parameter group to tune based on dataset complexity as was expected from the results of Chapter 5. This aspect will need additional work to be carried out on more datasets to better understand the part dataset complexity has to play.

The implication of this case study is that there is potential to make hyper-tuning more efficient through the use of SA in real world scenarios where tuning time and model accuracy are key performance indicators for practitioners.

6.6 Summary

To summarise, the case study results showed that the SA ranked hyper-parameters can be tuned more efficiently, achieving good model performance and saving tuning time compared to larger parameter groups. It also serves to show that the framework explored and developed in this work is applicable in real world scenarios and should be considered in future works.

Chapter 7

Discussion and Analysis

This chapter aims to collate the findings from the previous three chapters work where they can be discussed and analysed in tandem, to produce fully rounded interpretations of the results and the significance of the findings for this work as a whole. Furthermore, any limitations can be discussed here completely and recommendations can be put forward. To better facilitate the discussion, this chapter will begin by restating the problem targeted and the aims and objectives.

7.1 Restating the Problem

This work targeted an area very popular in research: hyper-parameter tuning of DL models (4, 70, 73, 74, 75). Rather than explore the tuning algorithms themselves, this work approached this problem from a different vantage point, the hyper-parameter themselves. The question posed: is it possible to identify which parameters are most influential to model performance? and, thereby, increase tuning efficiency by concentrating efforts on that subset of influential hyper-parameters?

This question was very broad and due to time constraints associated with the completion of a PhD, the scope of work was concentrated to the training parameters of CNNs specifically. This allowed for the demonstration of the validity of this research direction, potentially strong initial results to support/challenge the hypothesis and several areas for future work.

7.1.1 Restating the Aims & Objectives

The aim of this work was to produce a general ranking of CNN training hyper-parameters via SA that could inform HPO search spaces, and improve tuning efficiency. The Research Objectives (RO) identified to achieve that aim were:

- RO1: Create a framework to facilitate the calculation of sensitivity measures for the hyper-parameters of DL models which can be used for future work.
- RO2: Rank the influence of a hyper-parameter on model performance, taking into consideration two state of the art sensitivity measures.
- RO3: Discover relationships between parameter influence and network architecture or input data.
- RO4: Demonstrate any potential of reducing the hyper-parameter search space to influential parameters for HPO.
- RO5: Produce measure of hyper-parameter tuning efficiency that considers computation time and model accuracy to facilitate comparisons.
- RO6: Measure hyper-parameter tuning performance of influential parameter groups versus other parameter groups.
- RO7: Apply findings in case study to demonstrate real world scenarios and validate results.
- RO8: Provide guidance and a robust methodology to machine learning and deep learning practitioners in choosing what hyper-parameters to tune and the application of SA to DL.

7.1.2 Achievement of Aims & Objectives

Overall, the aim and objectives set out were completed allowing the problem question to be answered.

RO1

The SADL framework (19) was created to calculate the Sobol Indices and Elementary Effects of CNN hyper-parameters. The creation of the framework in itself is a successful contribution that will facilitate the future directions of research in this area as a robust methodology for the application of SA to DL.

RO2

The sensitivity measures from both SA methods implemented as part of the SADL framework were used to produce a final measure of influence, contributing to a general ranking where both measures were considered.

RO3

From the results it was not possible to conclude whether general patterns existed and so, to meet objective 3, additional work would be required to observe whether some of the emerging relationships and patterns hold true.

RO4

The potential of reducing the hyper-parameter search space to SA identified influential parameters was proven in Chapter 9 and 6. The SA informed Bayesian Optimisation conducted showed that tuning parameter groups consisting of influential parameters was more efficient than tuning all parameters, reducing tuning time without compromising model test accuracy. This was also confirmed and validated in the case study where tuning the influential parameters for an additional 40 minutes increased the test accuracy achieved by 18%.

RO5

The Accuracy Gain ↯ metric was created to quantify tuning efficiency and was used as a measure when comparing the tuning of hyper-parameter groups.

RO6

Accuracy Gain was used to measure tuning efficiency, facilitating the comparison of groups of various parameters.

RO7

Applying the results of the framework and SA informed Bayesian Optimisation to a case study further validated the observations made, demonstrated the potential of reducing the hyper-parameter search space and facilitated recommendations that can provide guidance to DL practitioners.

RO8

The final results showed that three parameters: batch size, learning rate decay and learning rate decay step were most influential on CNN test accuracy and tuning this combination was most efficient. This can guide practitioners in how they approach tuning in the future. Furthermore, the SADL framework developed presents a robust methodology that can be replicated and adapted to continue the exploration and experimentation of SA applied to DL. This could be exploring further DL architectures, different hyper-parameter types or utilising the framework to inform specific attempts at hyper-parameter tuning. The SADL framework represents the major contribution of this work and the results it facilitated presented in this thesis show the potential to be had in researching the application of SA to DL.

7.2 Interpretations

To interpret the results across the three chapters, all key findings will be stated, the hypothesis discussed and any patterns explored.

7.2.1 Key findings

This sections presents the key findings of this work in chronological order.

Robust Framework

The methodology presented in this thesis is, in itself, a key finding and contribution. Outlined in Chapter 3, the SADL framework can be followed to conduct future work exploring the hyper-parameters of DL models using SA. The elements required to conduct experimentation of this nature is explained in detail, presenting a robust framework that can be copied exactly or adjusted to other DL architectures. The modular design of the framework was designed with future adjustments in mind allowing the various elements to be implemented to best suit the architecture and input dataset or to specifcally explore the architectural parameters rather than the training. By nature, it is also buildable, where additional modules for further SA methodology would be able to be added in future to facilitate further work. A replicable framework, such as SADL, presents opportunity for future work in this area to be produced more quickly.

Parameter Influence

The first major finding came in the form of the influence ranking where batch size, learning rate decay and learning rate decay were identified as the top three most influential training parameters on CNN architectures, in that order. Over five CNN architectures and three image classification datasets, these hyper-parameters were identified as being influential on model performance with batch size scoring significantly higher than all other parameters but learning rate decay.

Some studies have emphasised the importance of batch size in the past, where an optimal batch size was linked to increase in model accuracy (114). In the past smaller batch sizes were used due to the memory limitations of computational hardware, however due to newer processors, memory capacities and parallelisation larger batch sizes are being used. However, larger batch sizes have been associated with poorer generalisation (117, 118) which could also explain the significant influence of batch size on test accuracy. If a larger batch size results in a model with a poorer ability to generalise, the performance on unseen, test data would be worse than if that batch size was smaller and the models generalisability was better. The SA rank was computed specifically on a hyper-parameters influence on test accuracy and so it is expected that batch size is ranked highly in this. Shallue et al. (119) took this a step further, concluding that some models can adapt better to larger batch sizes than others and that it is the model architecture rather than the dataset that influences the optimal batch size. In subsequent work (120), they also showed that standard optimisers work across batch sizes and can match the performance of newer optimsiers, suggesting that the optimiser itself may not play as instrumental a role. This is all supported in the SA ranking where optimiser has little influence and batch size is ranked significantly most influential overall. Shallue's conclusion that network architecture is an important factor in determining optimal batch size was also found to be true via SA where shallower models were more influenced by the batch size.

Learning rate decay was ranked second most influential via SA. This parameter has been observed to improve the generalisation and optimisation of a model (121). This can be attributed to the qualities of this parameter helping models avoid local minima but You (121) suggests it is more than that, they pose that a large initial learning rate minimises the models ability to learn noisy data and decaying it assists with learning more complex patterns (121). This idea is supported by the SA as learning rate decay was ranked higher on the more complex models and for the more complex datasets.

Following the importance of learning rate decay, the learning rate decay step was ranked third most influential. As this sets the rate at which the learning rate decays it was expected that an optimal step would be required to see the full benefit of an optimal decay rate.

As discussed in Chapter 2, Section 2.2.1, the learning rate has a reputation for being an important hyper-parameter, where some go so far as to suggest the success of the model hinge on an optimal learning rate setting (18). However, the SA concluded that the learning rate itself did not show high influence on model performance, placing more emphasis on related setting such as learning rate decay and learning rate decay step. Reasons as to learning rates lack of quantifiable influence is discussed later in Section 7.2.3, dedicated to unexpected results.

Tuning Efficiency

When looking at the tuning efficiency of different parameter groups it was proven that tuning the influential parameter groups was more efficient than tuning all training parameters, this was also found to be true when applied in a case study. This means that tuning this group could reduce training time without compromising accuracy or achieve higher accuracy without requiring significantly more training time.

The initial experiments found the top parameter, batch size, to be most efficient followed by the top three parameters, batch size, learning rate decay and learning rate decay step. In the real world scenario, the top three parameter group was the most efficient group overall.

The top three parameter group performance could be explained by breaking down it's components. Batch size and learning rate decay's qualities, when optimised, can in theory cancel out the disadvantages of the other creating a very complimentary group of parameters to optimise. A large batch size makes it harder for a model to generalise whereas learning rate decay improves a models ability to learn complex patterns and generalise. Learning rate decay increases convergence speeds, when tuning it alone it was significantly quicker than the other groups but couldn't match their accuracy, whereas a smaller batch size increase training times. To optimise the learning rate decay completely, the learning rate decay step also needs to be optimal. So, by combining all three of these highly influential parameters into a group and concentrating tuning efforts on them, as shown, the result is reduced tuning times with high model performance. This also explains why the top three group outperforms batch size alone in some instances as the differing batch sizes can negatively affect training time, increasing the tuning time and therefore reducing the overall tuning efficiency measure by our novel metric, Accuracy Gain ᛃ.

Patterns & Relationships

The key pattern observed surrounded parameter influence and complexity, be it in architecture or dataset. Where the level of complexity increased, generally, learning rate decay was ranked as having more influence, whereas batch size was more influential where there was less complexity. Complexity can be present in architectures where they are deeper/wider with additional layers and nodes that need to be adjusted when learning. In datasets, the complexity was quantified using the CSG metric which looked at class separability in the dataset features. As discussed above, a feature of learning rate decay is that it aids models in learning more complex patterns whilst reducing the impact of noise in the dataset which explains this observation in influence and ranking.

7.2.2 Hypothesis

The hypothesis put forward in this work was that tuning the influential parameters could increase the tuning efficiency by improving model performance, reducing tuning time or a combination of both of these things.

The key findings from the three chapters of work support this hypothesis as the influential parameter groups were more efficient to tune than all parameters, in most cases achieving better accuracy and requiring less tuning time. In the case study, the top three influential parameter group achieved the same accuracy reported in the original paper, 85%, and was within an hour tuning time of the all parameter group which was 18% less accurate, a justifiable trade off. Tuning the top three parameter group was almost twice as fast as tuning the parameters reported in the original paper suggesting that if the authors had tuned the batch size, learning rate decay and learning rate decay step rather than the momentum and the learning rate the same model accuracy could have been achieved in half the time. This is a clear demonstration that the results support the hypothesis.

7.2.3 Unexpected Results

Despite clear prominence in the literature, the learning rate was ranked as having relatively low influence on model performance. This result was completely unexpected due to the reputation and precedents set in other works where the learning rate is amongst the, if not the first, hyper-parameters most commonly chosen for tuning efforts. As this study concentrated on testing accuracy predominantly, and the SA itself was calculated based on a parameters influence on a models test accuracy there is a possibility that learning rate has more impact on other performance measures such as time, precision, recall or the training rather than test accuracy. Furthermore, the architectures chosen in this study were tried and tested, known for their award winning performance and potentially learning rate has more scope for influence where the architectures are sub-optimal. Another aspect that could have hindered the influence of the learning rate was it's relationship with batch size. As the batch size increases, the range of optimal learning rates decrease (117). The default batch size was set to 32 which is not very big but maybe the range of usable learning rates would have increased if a default of 16 had been chosen, giving learning rate more opportunity to prove it's influence. Conversely, 32 is a commonly used batch size, hence being chosen as the default in this case, and so there would still be an expectation for learning rate to be as influential as reported elsewhere.

Another unexpected, though explainable result, was that in some instances the most influential parameter, batch size, was less efficient to tune than the all parameter group. Specifically, in the case study batch size achieved a ᔕ of roughly 0.08 whereas the all parameter group achieved a ᔕ of roughly 0.09. As mentioned above, some values of batch size can drastically affect training times and as a result, for the case study, the tuning process was longer for batch size by roughly an hour and a half, which for a 12% increase in accuracy seems reasonable. It was this 12% increase in accuracy for batch size compared to the all parameter group that created the expectation that the batch size parameter would prove to be more tuning efficient, however the time difference meant that the all parameter group was scored as being slightly more efficient. In cases like this, it could be argued that the increase in accuracy is worth the trade off in tuning time.

Finally, the initial results of the rank informed tuning suggested that dataset complexity and accuracy gain were linked with the potential that the CSG value of the dataset could aid practitioners in deciding which influential parameter group to tune. However, the case study did not follow this pattern showing that more data is needed to fully understand this relationship.

7.3 Implications & Significance of the findings

7.3.1 Contextualising Results

Chapter 2, Section 2.3 presented a review of works from the last two decades that explored SA in relation to DL. Where more simplistic measures such as PaD or chart interpretations were adopted to understand the influence in other works the formal, quantification of parameter influence presented in this thesis sets this work apart. The main perceived advantage of approaching the SA of DL hyper-parameters more formally, compared to some reports in the literature, was to be able to analyse trends and patterns more thoroughly and to produce results that had more scope for generalisation and reproduceability than those more open to interpretation used in the literature. The potential added benefit of approaching the SA in this way was proven where the SA measures were compared across architectures and datasets to produce a generalised rank, where the topmost influential parameters were proved to positively impact tuning efficiency.

Furthermore, several works in the literature where SA and DL were combined were application specific, raising the question whether the findings could be applied in other scenarios or whether general conclusions could be drawn. Designing the experiments with generic, popular image datasets and several state of the art architectures the SA ranking and subsequent rank informed tuning made it possible to draw wider conclusions from the results that were then applied successfully in the case study.

Rather than produce a novel metric for SA, as was done in some works (80), the choice to use known measures was purposeful so that the methods were tried, tested and reproducible adding an additional layer of validity to the results and conclusions drawn from them.

A gap identified in the literature by other researchers surrounded the lack of guidance for ML and DL practitioners in hyper-parameter tuning. SA was identified in those instances as having the potential to provide some insights into what elements of the DL process are influential, be it architecture, input data or hyper-parameters (16). The key findings of this work directly address this gap in knowledge, cutting through the confusion surrounding parameter influence and importance and clearly quantifying the influence of various training parameters on CNN test accuracy that practitioners can take into consideration when concentrating tuning efforts in future work. Additionally, a key observation was the lack of literature exploring the application of SA to DL hyper-parameters. Providing a framework, SADL, that can facilitate future work in this area will help grow this body of knowledge.

7.3.2 Confirming and Challenging Theories

These results challenge the theories surrounding the importance of learning rate in DL model performance. These findings may differ to the expectations set in the literature as this study set out to quantify influence whereas previous interpretations of learning rate were instinctual and may have placed the emphasis on learning rate when in actual fact the learning rate decay or, more likely, the batch size was contributing more and being overlooked.

The SADL framework, which is the backbone of this work, shows that SA can identify influential CNN hyper-parameters, and these parameters can be tuned more efficiently has been confirmed via experimentation and in a case study. The ranking produced also supported the literature that placed the emphasis on batch size importance.

To some extent, these results also challenged the idea that tuning hyper-parameters is unique to the combination of architecture and task, as the most influential parameters were shown to be generally influential. The optimal settings of those parameters may differ based on the architecture task pairing but concentrating tuning efforts on an influential subset of parameters was shown to yield higher tuning efficiency, producing competitive test accuracy alongside a time saving.

The No Free Lunch theory (68) was also partially challenged by these results, as they showed that HPO could be generally improved by reducing the parameter search space to influential parameters. Though this may not make the lunch completely "free", it does contribute a hefty discount.

7.3.3 Impact of Findings

The concept that CNNs can be trained efficiently by concentrating efforts on batch size, learning rate decay and learning rate decay step could have a significant impact for DL and ML practitioners, especially those with less experience to draw on. There is no guarantee that with unlimited time and resources a more optimal solution could be found exploring a wider set of parameters. However, where there are time and resource constraints concentrating on these three influential parameters gives a modeller a greater guarantee of success than if they were to tune the less influential parameters or even all of the training parameters.

This work has also contributed a measure of efficiency, ϟ, that can be applied to future tuning efforts which could be calculated for a smaller set of trials to indicate which direction could yield the best accuracy gain over time.

Ultimately, this work has shown the applicability of SA to understanding the influence of a CNN tuning hyper-parameters on the model's test accuracy. This provides some insight into what is influencing the learning process that occurs within the neural network black box and could influence directions of future work. Furthermore, the importance of these influential parameters was highlighted in their impact on tuning efficiency, validated in a case study, and confirming that this avenue of exploration has even more potential. Finally, the SADL framework that was created and presented in this thesis facilitates the future work in this area, an area this thesis has shown to have many avenues of potential.

7.3.4 Proposed Solutions

These results could create scope for change in DL practitioners behaviours and practices that could lead to gains in accuracy and time saving.

Behavioural Changes

The biggest behavioural shift that could come of these results is the attitudes of DL practitioners to learning rate. Rather than assuming the learning rate is influential and concentrating optimising efforts in that direction, these results clearly suggest other hyper-parameters should be prioritised such as batch size.

Another change would be to consider the complexity level of the network architecture and dataset when choosing which hyper-parameters to include in the optimisation search space. Using a measure such as CSG or looking at the numbers of nodes and layers to understand which parameter may be more influential in this instance could benefit practitioners in the long run during HPO.

A behaviour or mindset that dominates in the SA literature is the concept that SA cannot provide insights without clear scope of what is being analysed. This is also true in the design of neural networks and the preparation of datasets. A shift in behaviour would be to also apply this to making decisions around tuning efficiency needs could help practitioners plan and use SA and ϟ to decide how to proceed with tuning. Questions that could be asked to help with this include whether this is a problem where test accuracy could be compromised to increase the speed of the tuning and training process? whether there are additional resource constraints that may impact the tuning efficiency that need to be mitigated by adjusting the parameter space? etc.

Conventional Practices

From this, a proposal for a future conventional DL practice would be to tune the batch size as priority when working with CNNs and image datasets. Batch size was ranked most influential, and though in some instances it was not the most efficient the increased test accuracy was worth the additional tuning time. Where resource and time constraints are present tuning the batch size, learning rate decay and learning rate decay step should be prioritised to benefit from the improved efficiency.

Another suggestion would be to take early measures of accuracy gain when tuning to narrow the pool of parameter groups to explore. This could save the practitioners time in the long run by pursuing the more efficient groups of parameters to tune.

A concept popular in the SA literature was the widespread adoption and application of SA across industries. As this work has proven, SA is very much applicable in DL and

ML, whilst also highlighting it's potential to the wider areas encompassed in computational sciences. The hope is that this work encourages practitioners to adopt some form of SA into conventional DL and ML practices and apply it, where possible, to some of the fields most burning questions such as getting insights into the black box that are neural networks. Furthermore, this thesis provides a framework to make this possible. Practitioners could use the SADL model presented in this work to facilitate the adoption of SA into common practice when evaluating DL model hyper-parameters.

7.4 Limitations

These results can not show with certainty that these findings can be generalised across different architecture types popular in DL such as RNNs, GANs or transformers. Though the evidence is clear for CNN architectures, additional experiments would have to be designed to replicate this work for other network architectures to explore parameter influence and their impact on tuning.

A conclusion made in Chapter 5 explored the possibility that calculating the CSG of a dataset could help practitioners choose the most efficient parameter group for tuning. This was drawn from three datasets which, in itself, is a limitation dictated by resource and time constraints. Though it showed merit, and still shows promise, the case study dataset did not follow the pattern as expected. To improve on this aspect, completing the rank informed HPO on additional datasets would strengthen or disprove this. Currently, there is insufficient data to confirm or deny this theory following the case study.

As touched on above and explored in Chapter 3, the number of architectures and datasets was limited by resource and time constraints. The nature of the experiments made them resource intensive and time consuming and within the time period allocated to complete this work there had to be an arbitrary cut off point to meet this deadline. To mitigate the impact of this the architectures and datasets chosen were representative of those popular in the literature and were varied enough to draw generalised conclusions across CNN architectures and image classification datasets. To further strengthen the results additional CNN architectures and image datasets could be included in a future phase of work.

Similarly, another improvement would be to apply these findings to CNN architectures performing an alternative task such as object detection or natural language processing.

This would add further evidence to whether the parameter influence is related more to network architecture or the task it is being applied to.

Finally, the scope of parameters explored was limited to the training parameters which is insightful for those adopting tried and tested architectures. An analysis of these parameters alongside architectural parameters where the architecture is sub-optimal could highlight these parameters in an alternate light.

7.5 Summary

The key findings presented in this work were as follows:

- The SADL framework is a robust methodology that can be followed and adapted for future work applying SA to DL.
- It is possible to provide empirically based guidance on which parameters should be optimised.
- Batch size, learning rate decay and learning rate decays step are highly influential on CNN architecture performance, witch batch sizes ranking being significantly higher than other parameters.
- Learning rate did not achieve the influence expected.
- HPO can be made more efficient by tuning a group of influential parameters.
- The design of this work builds on that explored in the literature whilst also combatting some of the limitations highlighted in those works.
- Though time and resource constraints limited this work in some aspects, the results are robust and can be built upon in future work.

Chapter 8

Conclusions and Future Work

This chapter will conclude the thesis by summarising the key findings, relating them to the aims and objectives of this work whilst discussing their value. A summary of the limitations will also be included and directions of future work will be recommended.

8.1 Conclusions

The aim of this investigation was to understand, and quantify through SA, the influence of training hyper-parameters on CNN architectures. Furthermore, how this ranking could contribute to increasing the efficiency of HPO for DL practitioners. The results showed that applying tuning efforts to influential parameters could increase HPO efficiency, saving time without compromising model performance. Batch size, learning rate decay and learning rate decay step were highlighted as the most influential parameters where batch size was measured as being significantly more influential than all other parameters apart from learning rate decay. Unexpectedly, learning rate did not score highly suggesting low influence in direct contradiction with the expectations set in the literature. Batch size and the combination of tuning all three influential parameters were highly efficient during tuning, outperforming all parameters and in the case study specifically, tuning the influential parameter groups also outperformed the parameters chosen for tuning in the original work.

Whilst working on this research a conference paper was published in the IEEE International Conference of Tools of Artificial Intelligence. In this paper the initial findings of applying SA to CNN hyper-parameters was presented (19). This paper formed the basis of Chapter 3.1 in this thesis and is available in its entirety in Appendix A.

This work was able to address the research problem by developing and making the contribution of SADL, a novel framework for conducting SA of CNN hyper-parameters, producing a generalised ranking and using this to inform the implementation of Bayesian Optimisation. This robust methodology can be followed when conducting future work on this topic. The findings were also confirmed in their application to a case study, validating the results further. A novel metric, Accuracy Gain, was also developed to better quantify HPO efficiency so that the parameter groups could be more easily compared and contrasted in terms of their affect on tuning efficiency. Taking a modular approach to solving this research problem allowed the results to be built upon in a way that validated the previous step. The first area of work applied SA to hyper-parameters and produced a ranking, the second took that ranking and confirmed that tuning influential parameters was more efficient than tuning non-influential parameters and this was all confirmed in the third area of work - the case study.

The design of this study addressed a clear gap in the literature of SA applied to DL. Specifically using a form of SA that quantifies influence in a way that is reproducible, less open to interpretation of the researcher and can be compared across works makes the results of this study more robust. Furthermore, combining this into a framework allows others to also use this methodology going forward. Where less formal metrics such as PaD or interpreting charts were popular in related works, the use of Sobol Indices and the Morris method addresses some of the limitations of those studies.

The results of this work challenge the theory that the learning rate is the most important hyper-parameter, proving that finding optimal values for batch size, learning rate decay and learning rate decay step can improve model performance. Conversely, this work supports the theory that reducing the parameter search space will reduce tuning time. Combining that theory with reducing the search space to the influential parameters allows for tuning time to be reduced without compromising model performance.

These findings can be directly applied in real world scenarios, as proven by the case study, by DL practitioners. The first step would be to consider tuning the batch size, learning rate decay and learning rate decay step only before exploring other parameters. An alternative option, would be to tune the top three influential parameters identified here, batch size alone and another group of parameters for a shorter amount of time and calculate the accuracy gain of each group. This can allow practitioners to decide to tune the group with the best accuracy gain for additional trials to find optimal settings.

The main limitations of this study were related to time and resource constraints. The fixed time-frame of PhD meant choosing an informed, arbitrary cut-off point for the number of models, datasets and hyper-parameters explored so that it could be completed in time. Also contributing to this was the lack of computational resources available withing the department and the nature of this work being resource intensive meant that some larger, more complicated datasets and architectures could not be included in the scope of this work. Though this work has clearly shown the merit of the application of SA to CNN hyper-parameters and how it can be applied to HPO to increase efficiency, the influence ranking is specific to model test accuracy. To generalise these findings across additional metrics additional experiments would need to be conducted where other parameters such as learning rate may have more influence than was observed in this study. This does not take away from the results presented here as the scope of this work concentrated specifically on test accuracy and showed how test accuracy improvements can be made by optimising the influential parameters. Having said this, to further generalise these findings to other types of data, DL architectures or performance metric future work will have to be carried out. On this note, an additional contribution of this work is a clear methodology that can be followed in the exploration of the future directions highlighted below.

8.2 Future work

Some areas of future work are directly linked to the limitations highlighted above, addressing additional questions raised or applying SA to DL models from an alternative angle. Extending the scope of architectures, datasets and hyper-parameters would be the natural continuation of this work. Firstly, extending the CNN architectures and image datasets, then adding alternative tasks such as object detection and natural language processing and seeing whether the conclusions for CNN hyper-parameters is true for all. Additionally, exploring the architectural parameters of CNNs and their impact for the full range of tasks to see the impact they have on general parameter influence. The second phase would explore a greater range of network architectures. The combination of the results from these future directions could provide a definitive ranking of parameter influence, identifying relationships between architectures, datasets and influence that can aid future DL practitioners.

As explored in Chapter 2, Section 2.4 French proposed eight contexts for the use of SA(24):

1. To build and explore models
2. To explore science and models relationships
3. To determine influential inputs
4. To develop efficient algorithms
5. To design experiments
6. To guide decision making
7. To build consensus
8. To gain understanding

This work and the natural progression of it detailed above specifically target number 1, build and explore models, and 3, determining influential inputs. To some extent the results also apply to number 6 where they have the potential to guide future decisions surrounding HPO. The SADL framework and methodology can be used to conduct the future work.

The application of SA to network architectures in DL, as mentioned above, would be a future work that targeted number 4, developing efficient algorithms. Broadening the scope of future work, considering the behavioural changes suggested above and targeting the concept of developing efficient algorithms, SA could also be applied to other learning algorithms that are being employed in the field of DL and ML such as randomised learning or evolutionary algorithms.

A key area of future work surrounds the final point, gain understanding. The inner workings of DL models, dubbed black-boxes, is a "hot topic" in research where SA could play an important role. Already through this work, and with the future directions highlighted above, SA has quantified hyper-parameter influence, showing which parameters within a network contribute most to the final model output. The granularity of this exploration could be increased, conducting the SA on the hyper-parameter values themselves, almost replacing the need for separate tuning, where specific hyper-parameter values could be highlighted as more influential than others.

8.3 Summary

A robust framework to facilitate the exploration of SA and DL was created and SA was successfully applied to the hyper-parameters of CNN DL models, resulting in a ranking that informed HPO parameter space to clearly show increased efficiency in tuning. These findings were all confirmed and validated in their application in a case study. Batch size, learning rate decay and learning rate decay step were the top three most influential parameters on CNN architecture identified via SA. Tuning batch size alone or the top three parameters together was proven most efficient to tune with a novel metric, accuracy gain. Time and resource constraints meant that the scope of this work was limited to CNN architectures and image classification datasets as training parameters were analysed based on their affect on model test accuracy. Future work, utilising the novel framework developed in this thesis, would look to expand this scope, exploring various architecture types, DL tasks, parameters and performance metrics with the aim of generalising the results further and exploring any relationships and patterns that may exist.

Chapter 9

Reflection

Please note: this reflection will be written in first person as it expresses the thoughts, feelings and experiences of the author.

This experience has been unlike any other, it has also been unlike anything I expected. There have been many unforeseeable challenges to overcome, such as changes in supervision, lack of computational resources available and completing this work during the COVID-19 pandemic. COVID-19 impacted so many experiences that usually come part and parcel with completing a PhD, I had no access to a lab for the majority of my degree, no in person meetings with other PhD students or my supervisors for the majority of my degree and no opportunities to present my work in person internally or at conferences. This naturally led to additional challenges, overcoming feelings of loneliness, feeling without support, struggling to prioritise PhD work whilst supporting vulnerable family members and whilst the world seemingly came to a halt. This was a major hurdle that I had to overcome in order to produce this thesis.

As mentioned briefly above, there were additional challenges related to my supervision that also impacted my progress at times. My initial supervisor left the department, and then my second supervisor left the department. My replacement supervisor had too much work to support me and so a year in I made the difficult decision to change supervisor, who ultimately decided to take their work in a different direction 6 months before I aimed to submit, resulting in my fourth and final supervisor. This particular hurdle had me questioning whether completing this work was possible at times and I am proud that I was able to persevere in the face of so much adversity.

Over the course of the last three years I have developed so many skills and acquired so much knowledge. My abilities to design experiments, write critically and present findings are all skills I have learned during my research degree. More specifically, I have improved my coding skills and my understanding of neural networks, hyper-parameters and machine learning.

If I could do this project again, I would have changed supervisors sooner to minimise the impact on me and my work. I would also try to deal with my feelings of impostor syndrome earlier on, as I feel I would have presented my work for conferences earlier if I had more belief in myself and the contributions I was capable of.

Specifically with the design of my work, I would have adopted the google collab platform from the beginning as the GPU in the department was retired before I could complete my experiments. I hadn't originally planned to complete any work on this platform but I had to adjust the case study.

When I planned my work, I originally thought that I would repeat my methodology for an object detection task, natural language processing task and Recurrent Neural Network (RNN) model architecture. It became apparent, however that this scope of work was unattainable in the time available and so I concentrated on CNNs specifically and the image classification datasets as I knew those other avenues of experimentation could be clear directions for future work.

In conclusion, if I was starting this area of study again I would have requested a change of supervision earlier in the process in order to gain the support and focus I needed. As this directly affected my self-confidence and well-being which resulted in me unnecessarily challenging my capabilities and performance. However, this experience enabled me to develop my communication skills, critical thinking etc and it also highlighted my strengths and weaknesses. I worked with a number of diverse supervisors with contrasting styles, character and expectations which was extremely challenging at the time but has enabled me to develop my resilience and capabilities in communication and research. A highlight of my studies was presenting my work at an international conference, ICTAI, and gaining recognition of my work was highly motivational. I have learned that research is a complex area which expands and develops and so my initial objectives were not realistic and perhaps over ambitious due to time and resources. This learning experience has enable me to plan future work with more clarity and efficiency which results in positive outcomes for myself and my colleagues. I have matured and developed skills in teaching, presenting and writing during the course of my studies and despite the challenges I have faced I am grateful for this experience and the resiliency I have developed as a result.

References

[1] S. Razavi, A. Jakeman, A. Saltelli, C. Prieur, B. Iooss, E. Borgonovo, E. Plischke, S. Lo Piano, T. Iwanaga, W. Becker, S. Tarantola, J. H. Guillaume, J. Jakeman, H. Gupta, N. Melillo, G. Rabitti, V. Chabridon, Q. Duan, X. Sun, S. Smith, R. Sheikholeslami, N. Hosseini, M. Asadzadeh, A. Puy, S. Kucherenko, and H. R. Maier, "The future of sensitivity analysis: An essential discipline for systems modeling and policy support," *Environmental Modelling Software*, vol. 137, p. 104954, 2021.

[2] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener, "Sensitivity analysis of environmental models: A systematic review with practical workflow," *Environmental Modelling & Software*, vol. 79, pp. 214–232, 2016.

[3] R. Dinga, B. W. Penninx, D. J. Veltman, L. Schmaal, and A. F. Marquand, "Beyond accuracy: measures for assessing machine learning models, pitfalls and guidelines," *BioRxiv*, p. 743138, 2019.

[4] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[5] Y.-C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of optimization theory and applications*, vol. 115, no. 3, pp. 549–570, 2002.

[6] A. Saltelli and P. Annoni, "How to avoid a perfunctory sensitivity analysis," *Environmental Modelling Software*, vol. 25, no. 12, pp. 1508–1517, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364815210001180

[7] F. Ferretti, A. Saltelli, and S. Tarantola, "Trends in sensitivity analysis practice in the last decade," *Science of the total environment*, vol. 568, pp. 666–670, 2016.

[8] I. M. Sobol, "Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates," *Math Comput Simul*, vol. 55, no. 1-3, pp. 271–280, 2001.

[9] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.

[10] Y. Li, Y. Shen, H. Jiang, W. Zhang, J. Li, J. Liu, C. Zhang, and B. Cui, "Hyper-tune: Towards efficient hyper-parameter tuning at scale," *arXiv preprint arXiv:2201.06834*, 2022.

[11] D. Güllmar, N. Jacobsen, A. Deistung, D. Timmann, S. Ropele, and J. R. Reichenbach, "Investigation of biases in convolutional neural networks for semantic segmentation using performance sensitivity analysis," *Zeitschrift für Medizinische Physik*, 2022.

[12] G. S. Moussa and M. Owais, "Modeling hot-mix asphalt dynamic modulus using deep residual neural networks: Parametric and sensitivity analysis study," *Construction and Building Materials*, vol. 294, p. 123589, 2021.

[13] J. Pizarroso, J. Portela, and A. Muñoz, "Neuralsens: Sensitivity analysis of neural networks," *CoRR*, vol. abs/2002.11423, 2020. [Online]. Available: https://arxiv.org/abs/2002.11423

[14] A. Hunter, L. Kennedy, J. Henry, and R. I. Ferguson, "Application of neural networks and sensitivity analysis to improved prediction of trauma survival," *Comput Methods Programs Biomed*, vol. 62, no. 1, pp. 11–19, 2000.

[15] Y. Zhang and B. C. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," in *IJCNLP*, 2017, pp. 253–263.

[16] V. K. Ithapu, S. N. Ravi, and V. Singh, "On architectural choices in deep learning: From network structure to gradient convergence and parameter estimation," 2017.

[17] G. Dudek, "Sensitivity analysis of the neural networks randomized learning," in *Artificial Intelligence and Soft Computing*, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, Eds. Springer International Publishing, 2019, pp. 51–61.

[18] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[19] R. Taylor, V. Ojha, I. Martino, and G. Nicosia, "Sensitivity analysis for deep learning: ranking hyper-parameter influence," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI).* IEEE, 2021, pp. 512–516.

[20] A. Saltelli, A. Jakeman, S. Razavi, and Q. Wu, "Sensitivity analysis: A discipline coming of age," *Environmental Modelling Software*, vol. 146, p. 105226, 2021.

[21] A. Saltelli *et al.*, *Global sensitivity analysis: the primer.* John Wiley & Sons, 2008.

[22] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: a new paradigm to machine learning," *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, 2020.

[23] A. Saltelli, S. Tarantola, and K. P. Chan, "A quantitative model-independent method for global sensitivity analysis of model output," *Technometrics*, vol. 41, no. 1, pp. 39–56, 1999.

[24] S. French, "Modelling, making inferences and making decisions: The roles of sensitivity analysis," *Top*, vol. 11, no. 2, pp. 229–251, 2003.

[25] A. Saltelli, T. H. Andres, and T. Homma, "Sensitivity analysis of model output. An investigation of new techniques," *Comput Stat Data Anal*, vol. 15, no. 2, pp. 211–238, 1993.

[26] K. Feng, Z. Lu, and C. Yang, "Enhanced Morris method for global sensitivity analysis: good proxy of Sobol' index," *Structural and Multidisciplinary Optimization*, vol. 59, no. 2, pp. 373–387, 2019.

[27] S. Razavi and H. Gupta, "What do we mean by sensitivity analysis? The need forcomprehensive characterization of "global" sensitivity in Earthand Environmental systems models," *Water Resources Research Progress*, pp. 1–433, 2005.

[28] R. I. Cukier, H. B. Levine, and K. E. Shuler, "Nonlinear sensitivity analysis of multiparameter model systems," *J Comput Phys*, vol. 26, no. 1, pp. 1–42, 1978.

[29] A. Patanè, A. Santoro, V. Romano, A. L. Magna, and G. Nicosia, "Enhancing quantum efficiency of thin-film silicon solar cells by Pareto optimality," *Journal of Global Optimization*, vol. 72, no. 3, pp. 491–515, 2018.

[30] F. Campolongo, J. Cariboni, and A. Saltelli, "An effective screening design for sensitivity analysis of large models," *Environ Model Softw*, vol. 22, no. 10, pp. 1509–1518, 2007.

[31] P. C. Young, G. M. Hornberger, and R. C. Spear, "Modeling badly defined systems: some further thoughts," in *Proceedings SIMSIG Conference, Canberra*, 1978, pp. 24–32.

[32] H. Christopher Frey and S. R. Patil, "Identification and review of sensitivity analysis methods," *Risk analysis*, vol. 22, no. 3, pp. 553–578, 2002.

[33] R. L. Iman and J. C. Helton, "An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models," *Risk Anal*, vol. 8, no. 1, pp. 71–90, 1988.

[34] A. Saltelli and P. Annoni, "How to avoid a perfunctory sensitivity analysis," *Environmental Modelling & Software*, vol. 25, no. 12, pp. 1508–1517, 2010.

[35] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, G. Geraci, R. Hooper, P. Hough, K. Hu *et al.*, "Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: version 6.11 user's manual," *Sandia National Laboratories, Albuquerque, NM*, 2019.

[36] G. Qian and A. Mahdi, "Sensitivity analysis methods in the biomedical sciences," *Mathematical Biosciences*, vol. 323, p. 108306, 2020.

[37] Z. Pang, Z. O'Neill, Y. Li, and F. Niu, "The role of sensitivity analysis in the building performance analysis: A critical review," *Energy and Buildings*, vol. 209, p. 109659, 2020.

[38] D. Douglas-Smith, T. Iwanaga, B. F. Croke, and A. J. Jakeman, "Certain trends in uncertainty and sensitivity analysis: An overview of software tools and techniques," *Environmental Modelling & Software*, vol. 124, p. 104588, 2020.

[39] A. Mathew, A. Arul, and S. Sivakumari, *Deep Learning Techniques: An Overview*, 01 2021, pp. 599–608.

[40] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on

deep learning theory and architectures," *Electronics*, vol. 8, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/3/292

[41] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997.

[42] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[43] J. Lo and C. Floyd, "Application of artificial neural networks for diagnosis of breast cancer," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, 1999, pp. 1755–1759 Vol. 3.

[44] P. Rajeswari and G. S. Reena, "Human liver cancer classification using microarray gene expression data," *International Journal of Computer Applications*, vol. 34, no. 6, pp. 25–37, 2011.

[45] Y. Qia, Z. Zhaob, L. Zhangc, H. Liud, and K. Leie, "A classification diagnosis of cervical cancer medical data based on various artificial neural networks," 2018.

[46] S. Wang, R. V. Rao, P. Chen, Y. Zhang, A. Liu, and L. Wei, "Abnormal breast detection in mammogram images by feed-forward neural network trained by jaya algorithm," *Fundamenta Informaticae*, vol. 151, no. 1-4, pp. 191–211, 2017.

[47] H. G. Hosseini, D. Luo, and K. Reynolds, "The comparison of different feed forward neural network architectures for ecg signal diagnosis," *Medical Engineering Physics*, vol. 28, no. 4, pp. 372–378, 2006.

[48] S. Çimen and B. Bolat, "Diagnosis of parkinson's disease by using ann," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. IEEE, 2016, pp. 119–121.

[49] M. G. Feshki and O. S. Shijani, "Improving the heart disease diagnosis by evolutionary algorithm of pso and feed forward neural network," in *2016 Artificial Intelligence and Robotics (IRANOPEN)*, 2016, pp. 48–53.

[50] C. Pravin and V. Ojha, "A novel ecg signal denoising filter selection algorithm based on conventional neural networks," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 1094–1100.

[51] S. Mukherjee, K. Ashish, N. Baran Hui, and S. Chattopadhyay, "Modeling depression data: feed forward neural network vs. radial basis function neural network," *Am. J. Biomed. Sci*, vol. 6, no. 3, pp. 166–174, 2014.

[52] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[53] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, vol. 25, 2012.

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[57] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. V. Essen, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *ArXiv*, vol. abs/1803.01164, 2018.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *ICVPR*, 2015.

[59] A. Khan, A. Sohail, and A. Ali, "A new channel boosted convolutional neural network using transfer learning," *arXiv preprint arXiv:1804.08528*, 2018.

[60] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International journal of information management*, vol. 35, no. 2, pp. 137–144, 2015.

[61] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[62] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[63] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv preprint arXiv:1409.1259*, 2014.

[64] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[65] S. Sharma and S. Sharma, “Activation functions in neural networks,” *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[66] B. Karlik and A. V. Olgac, “Performance analysis of various activation functions in generalized mlp architectures of neural networks,” *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[67] D. R. Wilson and T. R. Martinez, “The need for small learning rates on large problems,” in *IJCNN’01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, vol. 1. IEEE, 2001, pp. 115–119.

[68] F. He, T. Liu, and D. Tao, “Control batch size and learning rate to generalize well: Theoretical and empirical evidence,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 1143–1152, 2019.

[69] A. Sharma, J. N. van Rijn, F. Hutter, and A. Müller, “Hyperparameter importance for image classification by residual neural networks,” in *International Conference on Discovery Science*. Springer, 2019, pp. 112–126.

[70] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, “Optimizing deep learning hyper-parameters through an evolutionary algorithm,” in *Proceedings of the workshop on machine learning in high-performance computing environments*, 2015, pp. 1–5.

[71] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[72] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[73] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1437–1446.

[74] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *International conference on machine learning*. PMLR, 2014, pp. 754–762.

[75] J. N. Van Rijn and F. Hutter, "Hyperparameter importance across datasets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2367–2376.

[76] J. Q. Davis, A. Gu, K. Choromanski, T. Dao, C. Re, C. Finn, and P. Liang, "Catformer: Designing stable transformers via sensitivity analysis," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2489–2499.

[77] T. Nagasato, K. Ishida, K. Yokoo, M. Kiyama, and M. Amagasaki, "Sensitivity Analysis of the Hyperparameters of CNN for Precipitation Downscaling," in *EGU General Assembly Conference Abstracts*, 2021.

[78] P. Novello, G. Poëtte, D. Lugato, and P. M. Congedo, "Explainable Hyperparameters Optimization using Hilbert-Schmidt Independence Criterion," Feb. 2021, working paper or preprint.

[79] H. Shu and H. Zhu, "Sensitivity analysis of deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4943–4950.

[80] L. Zhang, X. Sun, Y. Li, and Z. Zhang, "A noise-sensitivity-analysis-based test prioritization technique for deep neural networks," *arXiv preprint arXiv:1901.00054*, 2019.

[81] W. Samek, T. Wiegand, and K.-R. Müller, “Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models,” *arXiv preprint arXiv:1708.08296*, 2017.

[82] M. Gevrey, I. Dimopoulos, and S. Lek, “Two-way interaction of input variables in the sensitivity analysis of neural network models,” *Ecol Modell*, vol. 195, no. 1-2, pp. 43–50, 2006.

[83] V. Nourani and M. Sayyah Fard, “Sensitivity analysis of the artificial neural network outputs in simulation of the evaporation process at different climatologic regimes,” *Adv. Eng. Softw.*, vol. 47, no. 1, pp. 127–146, May 2012.

[84] J. Fürbringer, “Sensitivity analysis for modellers,” *Air Infiltration Review*, vol. 17, no. 4, pp. 8–10, 1996.

[85] R. Iman, “Uncertainty and sensitivity analysis for computer modeling applications,” *ASME Aerospace Division*, vol. 28, pp. 153–168, 01 1992.

[86] J. Helton, F. Davis, and J. Johnson, “A comparison of uncertainty and sensitivity analysis results obtained with random and latin hypercube sampling,” *Reliab Eng Syst Saf*, vol. 89, no. 3, pp. 305–330, 2005.

[87] J. C. Helton and F. J. Davis, “Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems,” *Reliab Eng Syst Saf*, vol. 81, no. 1, pp. 23–69, 2003.

[88] F. Chollet *et al.*, “Keras,” https://github.com/fchollet/keras, 2015.

[89] “A review of the evolution of deep learning architectures and comparison of their performances for histopathologic cancer detection,” *Procedia Manufacturing*, vol. 46, pp. 683–689, 2020, iNTER-ENG 2019.

[90] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, “Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions,” *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[91] F. Branchaud-Charron, A. Achkar, and P.-M. Jodoin, “Spectral metric for dataset complexity assessment,” in *CVPR*, June 2019.

[92] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017, arXiv:1708.07747.

[93] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[94] J. Herman and W. Usher, "SALib: An open-source python library for sensitivity analysis," *The Journal of Open Source Software*, vol. 2, no. 9, jan 2017. [Online]. Available: https://doi.org/10.21105/joss.00097

[95] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, vol. 7700. Springer, 2012.

[96] S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, "Bayesian optimization for adaptive experimental design: a review," *IEEE access*, vol. 8, pp. 13 937–13 948, 2020.

[97] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[98] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 115–123.

[99] V. Feldman and C. Zhang, "What neural networks memorize and why: Discovering the long tail via influence estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2881–2891, 2020.

[100] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the computational cost of deep learning models," in *2018 IEEE international conference on big data (Big Data)*. IEEE, 2018, pp. 3873–3882.

[101] T. Houk, "Knowledge from the sacred tree runes: Images and shapes of energy."

[102] K. C. Tauring, *The Runes: A Deeper Journey.* Lulu Press, Inc, 2017.

[103] M. Naylor, “Math runes,” in *Proceedings of Bridges 2013: Mathematics, Music, Art, Architecture, Culture*, 2013, pp. 191–198.

[104] A. S. Lundervold and A. Lundervold, “An overview of deep learning in medical imaging focusing on mri,” *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019.

[105] C. J. Markiewicz, K. J. Gorgolewski, F. Feingold, R. Blair, Y. O. Halchenko, E. Miller, N. Hardcastle, J. Wexler, O. Esteban, M. Goncalves *et al.*, “Openneuro: An open resource for sharing of neuroimaging data,” *BioRxiv*, 2021.

[106] C. Sudlow, J. Gallacher, N. Allen, V. Beral, P. Burton, J. Danesh, P. Downey, P. Elliott, J. Green, M. Landray *et al.*, “Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age,” *PLoS medicine*, vol. 12, no. 3, p. e1001779, 2015.

[107] K. Clark, B. Vendt, K. Smith, J. Freymann, J. Kirby, P. Koppel, S. Moore, S. Phillips, D. Maffitt, M. Pringle *et al.*, “The cancer imaging archive (tcia): maintaining and operating a public information repository,” *Journal of digital imaging*, vol. 26, no. 6, pp. 1045–1057, 2013.

[108] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, “Deep learning in image classification using residual network (resnet) variants for detection of colorectal cancer,” *Procedia Computer Science*, vol. 179, pp. 423–431, 2021, 5th International Conference on Computer Science and Computational Intelligence 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921000284

[109] K. Sirinukunwattana, J. P. Pluim, H. Chen, X. Qi, P.-A. Heng, Y. B. Guo, L. Y. Wang, B. J. Matuszewski, E. Bruni, U. Sanchez *et al.*, “Gland segmentation in colon histology images: The glas challenge contest,” *Medical image analysis*, vol. 35, pp. 489–502, 2017.

[110] A. Tharwat, “Classification assessment methods,” *Applied Computing and Informatics*, 2020.

[111] B. Prashanth, M. Mendu, and R. Thallapalli, “Cloud based machine learning with advanced predictive analytics using google colaboratory,” *Materials Today: Proceedings*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221478532100897X

[112] T. Iwanaga, W. Usher, and J. Herman, “Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses,” *Socio-Environmental Systems Modelling*, vol. 4, p. 18155, May 2022. [Online]. Available: https://sesmo.org/article/view/18155

[113] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, “A decade of agile methodologies: Towards explaining agile software development,” pp. 1213–1221, 2012.

[114] S. L. Smith, P. Kindermans, and Q. V. Le, “Don’t decay the learning rate, increase the batch size,” *ICLR*, 2017.

[115] K. You, M. Long, M. I. Jordan, and J. Wang, “Learning stages: Phenomenon, root cause, mechanism hypothesis, and implications,” *ArXiv*, 2019.

[116] I. Kandel and M. Castelli, “The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,” *ICT express*, vol. 6, no. 4, pp. 312–315, 2020.

[117] D. Masters and C. Luschi, “Revisiting small batch training for deep neural networks,” *arXiv preprint arXiv:1804.07612*, 2018.

[118] A. Sekhari, K. Sridharan, and S. Kale, “Sgd: The role of implicit regularization, batch-size and multiple-epochs,” *Advances In Neural Information Processing Systems*, vol. 34, pp. 27 422–27 433, 2021.

[119] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, “Measuring the effects of data parallelism on neural network training,” *arXiv preprint arXiv:1811.03600*, 2018.

[120] Z. Nado, J. M. Gilmer, C. J. Shallue, R. Anil, and G. E. Dahl, “A large batch optimizer reality check: Traditional, generic optimizers suffice across batch sizes,” *arXiv preprint arXiv:2102.06356*, 2021.

[121] K. You, M. Long, J. Wang, and M. I. Jordan, “How does learning rate decay help modern neural networks?” *arXiv preprint arXiv:1908.01878*, 2019.

Appendix A

Code Implementation

```
def get_increased_values(y, up, lo):

    #adding false values in positon to make it back to the right shape after
    caculating the difference (trj,ss,param)
    up = np.pad(up, ((0, 0), (1, 0), (0, 0)), 'constant') #first ss position
    lo = np.pad(lo, ((0, 0), (0, 1), (0, 0)), 'constant') #last ss position

    #matrix operation, y has shape ik, up+lo has shape ikj, output result with
     shape ij
    #product of y and (up+low) + sum of all axis - haddamard product sum of
    elementwise multiplication
    result = np.einsum('ik,ikj->ij', y, up + lo)

    return result.T

def get_decreased_values(y, up, lo):

    up = np.pad(up, ((0, 0), (0, 1), (0, 0)), 'constant')#last ss position
    lo = np.pad(lo, ((0, 0), (1, 0), (0, 0)), 'constant')#first position

    result = np.einsum('ik,ikj->ij', y, up + lo)

    return result.T

#Elementary Effects
def EE(x, y , param, trajectory,problemtype,ss): #sample, output, trj,
    categorical/numerical, samplesize used aka levels
```

```python

    y = y #list of accuracies of each run
    EE = []

    if problemtype == 'categorical':
        #need something here to onehot encode x for use in the calculations
        labelencoder = LabelEncoder()
        x = labelencoder.fit_transform(x)

    if problemtype == 'numerical':
        x=x

    p = ss
    d = p/(2*(p-1))

  #EE calculation

    x = np.reshape(x,[trajectory, ss,1]) #1 is the number of parameters and I'
    m doing it OAT
    x_dif = np.subtract(x[:, 1:, :], x[:, 0:-1, :]) #everything after the fist
     sample - everything but the last sample

    #Binary of whether the difference is postive of negative
    up = (x_dif > 0)
    lo = (x_dif < 0)

    y = np.reshape(y,[trajectory, ss])#Output Vector of shape levels and
    trajectories

    result_up = get_increased_values(y, up, lo)

    result_lo = get_decreased_values(y, up, lo)

    ee = np.subtract(result_up, result_lo)
    np.divide(ee, d, out=ee)

    return(ee)

def morrisSA(ee):
    mu = np.mean(ee)
    mustar = np.mean(np.absolute(ee))
    sigma = np.std(ee)

    return mu, mustar, sigma
```

Listing A.1: Morris Method implementation

```
def Si(Yxi, Y):
    splitYxi = []
    expectedY = []
    nSplits = 3
  #should be the average of the variance of the output for each fixed value?
  #split Yxi into chunks
    for i in range(0, len(Yxi)):
        splitYxi.append(Yxi[i:i + nSplits])

  #calculate expected value of y given xi
    for i in splitYxi:
        Eyofxi = sum(i)/len(i)
        expectedY.append(Eyofxi)

  #calculate variance in expected value of Y from the different splits
    varE = np.var(expectedY, dtype=np.float64)
  #print(varE)

  #Calculate variance of y
    varY = np.var(Y, dtype=np.float64)
  #print(varY)

  #Calculate and return Si
    Si = varE/varY

    return Si

 def STi(Yxi, Y):
    splitYnotxi = []
    expectedY = []
    nSplits = 3

  #all output except of Xi
    Ynotxi = [x for x in Y if x not in Yxi]

```

```python
 #split Y~xi into chunks
  for i in range(0, len(Ynotxi)):
    splitYnotxi.append(Ynotxi[i:i + nSplits])

 #calculate expected value of y given not xi
  for i in splitYnotxi:
    Eyofxi = sum(i)/len(i)
    expectedY.append(Eyofxi)

 #calculate variance in expected value of Y from the different splits
  varE = np.var(expectedY, dtype=np.float64)
 #print(varE)

 #Calculate variance of y
  varY = np.var(Y, dtype=np.float64)
 #print(varY)

 #Calculate and return Si
  STi = 1 - varE/varY

  return STi

#Method to return Sobol Indices
def SobolSA(modelinput,modeloutput,outputchoice): #input = list of parameters, output = results dataframe
  Y = modeloutput[outputchoice].tolist()
  Y = [item for sublist in Y for item in sublist] #flatten list from list of lists
  sobolresults = []

  for param in modelinput:
    YXI = modeloutput.loc[modeloutput['Parameter'] == param[1], outputchoice]
    YXI = [item for sublist in YXI for item in sublist]
    sobolresults.append((param[1], Si(YXI,Y), STi(YXI,Y)))
  return sobolresults
```

Listing A.2: Sobol Indices implementation

```python
#minmax normalising
MNIST_morris['Mu*_Norm'] = ((MNIST_morris['Mu*']-MNIST_morris['Mu*'].min())/(
    MNIST_morris['Mu*'].max()-MNIST_morris['Mu*'].min()))
MNIST_morris['Mu_Norm'] = ((MNIST_morris['Mu']-MNIST_morris['Mu'].min())/(
    MNIST_morris['Mu'].max()-MNIST_morris['Mu'].min()))
MNIST_morris['Sigma_Norm'] = ((MNIST_morris['Sigma']-MNIST_morris['Sigma'].min
    ())/(MNIST_morris['Sigma'].max()-MNIST_morris['Sigma'].min()))
MNIST_SA['Si_Norm'] = ((MNIST_SA['Si']-MNIST_SA['Si'].min())/(MNIST_SA['Si'].
    max()-MNIST_SA['Si'].min()))
MNIST_SA['STi_Norm'] = ((MNIST_SA['STi']-MNIST_SA['STi'].min())/(MNIST_SA['STi
    '].max()-MNIST_SA['STi'].min()))
```

Listing A.3: Normalising Sensitivity Measures

```
\usepackage{allrunes}

{\Large \textarc{j}}
```

Listing A.4: Using the Jera symbol for Accuracy Gain in LaTeX

Appendix B

Published Work

My paper Sensitivity Analysis for Deep Learning: Ranking Hyper-parameter Influence (19) was published in 2021 and includes early results that contributed to Chapter 4. I produced the paper in it's entirety and completed the experiments. My co-authors provided feedback on results and early drafts of the paper and offered suggestions for improvements.

R. Taylor, V. Ojha, I. Martino and G. Nicosia, "Sensitivity Analysis for Deep Learning: Ranking Hyper-parameter Influence," 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), Washington, DC, USA, 2021, pp. 512-516, doi: 10.1109/ICTAI52525.2021.00083.