# *A modified long short term memory cell*

Article

Accepted Version

Haralabopoulos, G. ORCID: https://orcid.org/0000-0002-2142-4975, Razis, G. and Anagnostopoulos, I. (2023) A modified long short term memory cell. International Journal of Neural Systems, 33 (7). 2350039. ISSN 1793-6462 doi: 10.1142/S0129065723500399 Available at https://centaur.reading.ac.uk/111809/

International Journal of Neural Systems, Vol. 03, No. 03 (2023) 1–13
© World Scientific Publishing Company

# A Modified Long Short Term Memory Cell

GIANNIS HARALABOPOULOS*
*Business Informatics Systems Accounting Department*
*Henley Business School, University of Reading*
*Reading, United Kingdom*
*E-mail: i.haralabopoulos@henley.ac.uk*

GERASIMOS RAZIS and IOANNIS ANAGNOSTOPOULOS
*Department of Computer Science and Biomedical Informatics*
*School of Science, University of Thessaly*
*Lamia, Greece*
*E-mail: {razis, janag}@uth.gr*

Machine Learning (ML), among other things, facilitates Text Classification, the task of assigning classes to textual items. Classification performance in ML has been significantly improved due to recent developments, including the rise of Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU) and Transformer Models. Internal memory states with dynamic temporal behaviour can be found in these kinds of cells. This temporal behaviour in the LSTM cell is stored in two different states: "Current" and "Hidden". In this work, we define a modification layer within the LSTM cell which allows us to perform additional state adjustments for either state, or even simultaneously alter both. We perform 17 state alterations. Out of these 17 single-state alteration experiments, twelve involve the Current state whereas five involve the Hidden one. These alterations are evaluated using seven datasets related to sentiment analysis, document classification, hate speech detection and human-to-robot interaction. Our results showed that the highest performing alteration for Current and Hidden state can achieve an average F1 improvement of 0.5% and 0.3%, respectively. We also compare our modified cell performance to two Transformer models, where our modified LSTM cell is outperformed in classification metrics in 4/6 datasets, but improves upon the simple Transformer model and clearly has a better cost-efficiency than both Transformer models.

*Keywords*: LSTM; Text Classification; Transformer Models; BERT

## 1. Introduction

A variety of computer tasks such as text, image classification,[4] video analysis, and speech recognition can be automated thanks to the advancements of Machine Learning (ML). The utilisation of data for training of predictive algorithms is the foundation of ML. Support Vector Machines (SVM), Decision Trees (DT), Multi-Layer Perceptrons (MLP), and Neural Networks (NN) are some of the best known types of ML models. In the past ten years, these predictive algorithms have significantly improved, primarily thanks to the rise of Deep Learning (DL), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) networks. All of these improved models are based on Neural Network principles. Interconnected

*Reading, United Kingdom

network of neurons/cells that simulate the operation of the human brain, with the aim of uncovering hidden patterns in data.

These ML developments and the use of sophisticated embedding methods have been quite beneficial for Natural Language tasks. In such tasks, text is first evaluated and processed using Natural Language Processing (NLP) techniques, and then is fed as an input to a model, which aims to predict the class of the input. Sentiment analysis,[12, 24, 58] Text Classification,[16, 56] Hate Speech detection,[2] and the creation of Spam filters,[3, 36] are typical Natural Language Classification tasks.

An RNN advances NN by considering temporal relationships. The internal memory state of each Recurrent Neuron enables this dynamic temporal behaviour. John Hopfield firstly introduced RNNs in 1984,[25] and since then, they have evolved into more elaborate networks like LSTMs (1999)[17] and GRUs (2014).[6] Compared to RNNs, LSTMs can obtain a broader range of contextual data and avoid network deterioration or exponential growth.[19] Whereas GRUs are quite similar to LSTMs, with the difference that GRUs lack an output process for each cell.

In multiclass and multilabel text classification tasks, LSTMs exhibit state-of-the-art performance.[22, 23] Multiclass tasks are those where an item can be categorised into one of more than two classes, whereas multilabel tasks are those where each item can be categorised into multiple classes simultaneously. LSTM networks have been employed extensively in numerous NLP tasks as well. Apart from text classification,[10, 33, 34] they have been used in sentence selection,[28] word prediction,[51] and sentence topic prediction.[63]

The cornerstone of an LSTM network are its LSTM cells. In short, each cell remembers values and has three different gates to control the data that passes through the network. These gates are input, output, and forget. Each one of them performs a unique function, based on the data fed to the cell, but ultimately they affect the memory of the cell: its current and Hidden states. These states, and how we can manipulate them efficiently, are the focus of this study.

We investigate whether we can improve the LSTM cell's performance by incorporating additional state calculations, with minimal increase in complexity and training time. To address that, a modification layer is added to the LSTM cell architecture. The modified LSTM cell will be referred to as LSTM-CS (from Custom State). We want to increase classification performance by further influencing the current and Hidden states. These modifications are employed on a state-of-the-art single network with two bi-directional LSTM (bi-LSTM) layers and are evaluated in seven diverse text classification datasets. The evaluation is based on three F1 scoring metrics and accuracy. The model is trained on accuracy, whereas the F1 scores present the classification efficiency of our model. Furthermore, our best performing proposed alteration is compared to two Transformer models, a simple one (SIMPLE) and a pretrained one (BERT).

## 2. Related Work

We will present some studies that have improved performance over a range of tasks. We will follow with a short mention of the most important NLP tasks and how these have been improved by using LSTM networks. The last three paragraphs of the section are focused on various LSTM network and cell modifications that have been proposed and have shown improved performance when compared to a baseline LSTM network.

RNNs and LSTMs have significantly enhanced the performance of automated tasks across a variety of applications. Since the early 2000s, LSTMs have been used to create music,[14] classify phonemes[20] or faces,[32] detect/predict seizures,[35, 54] recognise speech[18] and a range of classification tasks.[9, 40, 40, 62] These applications are a subset of the numerous applications across a wide range of domains where LSTMs have outperformed conventional machine learning approaches and neural networks.

In Natural Language Processing task LSTMs have offered unparalleled performance, some of these are Information Retrieval,[43] Named Entity Recognition,[61] machine translation,[49] topic modelling,[29] text similarity,[59] text summarisation,[44] sentiment analysis,[55] text generation,[44] and fake news or hate speech detection.[7, 31] The majority of these tasks typically involve a classifier predicting the relevant class or classes for the content item in question. For instance, positive and negative sentiment are the classes in polarity/simple sentiment analysis,[48] whereas feelings like joy, sadness, etc. are the classes in emotion sentiment analysis.[27] Likewise, in fake news detection, we usually have a single binary class,[52] fake or not.

The LSTM cell architecture has been under research long before its widespread use, with researchers looking for ways to extract more performance. This has resulted in numerous alterations that have been proposed in a variety of domains. Batch Normalisation (BN) was integrated by Wand et al.[53] into the LSTM cell's update function for all state transitions. In comparison to theLSTM, the BN-LSTM had faster convergence and better performance. Hu et al.[26] developed an LSTM cell that could follow trends in time series data by combining Gradient Descent selection and Particle Swarm Optimisation. The training and testing performance of three environmental forecasting tasks was improved thanks to these adjustments.

A grid-based LSTM network was proposed by Kalchbrenner, Danihelka, and Graves.[30] The network layout causes the LSTM cells to interact inside a three-dimensional structure, but the LSTM cell is not altered in its core. According to the authors' findings, such a grid enhances parity, addition, and memorisation. A Tree-LSTM is presented in Chen et al.,[5] where each cell's input is dependent on the input vector and two hidden vectors. The model employing these modified LSTM cells is evaluated in language inference tasks and the study concludes that sequential inference has unexplored possibilities. A Hierarchical Multimodal LSTM, essentially a Tree-LSTM extension with syntactic awareness, is introduced in Niu et al.[41] The model is evaluated in a dense visual-semantic embedding, while the findings imply that the suggested network can produce phrases with qualitative context. The M-LSTM, which Ye, Li, and Chen[60] proposed, involves feeding the LSTM cell with an extra input signal. The proposed network relies on past data in the iterative update function. The M-LSTM was evaluated in brain scans and achieved state-of-the-art results. Another LSTM cell modification was proposed by Qiu et al.[46] by changing the cell's gate mechanisms. The outcomes of the evaluation of their Bidirectional-LSTM model, in a bearing fault simulation task, indicate a significant advancement over traditional Bi-LSTMs. Shortcut connections that can detour the LSTM cell and a spatiotemporal LSTM cell, were proposed by Dai, Li, and Li.[11] The model was tested and evaluated in dense traffic prediction tasks, where it performed better than both M-LSTMs and LSTMs.

A trainable hypernetwork generating weights for the LSTM network, proposed by Ha, Dai, and Le[21] excelled at image recognition, handwriting production, and language modelling. A feedback LSTM that considers the distinction between prediction and observation was proposed by Kamil Rocki.[50] A prediction error is computed and used when new predictions are made. Feedback LSTMs appear to be more capable of generalisation than regular LSTMs based on their performance in a character prediction task. By computing an extra Hadamard product, Wu et al.[57] proposed a Multiplicative Integration LSTM which alters the way information flows within the LSTM cell. Four tasks—Character Level Modelling, Skip-through Models, Speech Recognition, Reading and Comprehending— were used for testing and evaluation purposes. A non-saturating activation function and a mixture of inputs are the characteristics of LSTM with working memory, abbreviated LSTWM, proposed in Pulver and Lyu.[45] The proposed cell and its network were evaluated in three tasks, namely text recognition, digit recognition, and digit combination. The results indicate that the LSTWM performs better than the LSTM, whereas in some cases it does so while requiring fewer parameters. Finally, Mittal et al.[38] proposed the reset of the LSTM cell's internal memory by using such a mechanism. By relying on a training dataset of approximately 1000 instances, their modified LSTM outperformed LSTM at a sign language recognition task.

Our proposed LSTM modification is not domain specific, nor it requires any major architectural change to an already existing model. By replacing the LSTM layer of our model with our LSTM-CS we could gain up to 0.7% performance improvement.

## 3. Methodology

The original LSTM cell structure along with our suggested alterations are presented in this chapter.

### 3.1. *The LSTM cell*

As shown in Figure 1, the most widely used LSTM cell has an input gate, an output gate, and a forget gate. The input vector, the Hidden state from the previous time step and the current cell state of the previous time step, are all inputs to the cell. These are then put through these gates and operators to create the Current cell state of this time step as well as its Hidden state, which will be fed into the cell for the following time step.

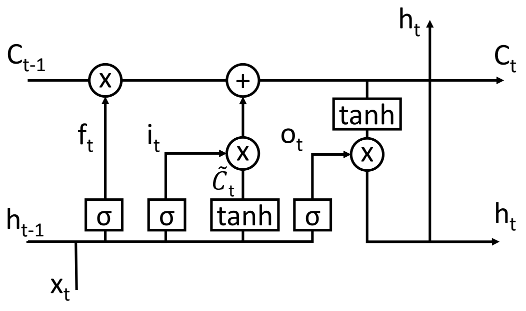

Figure 1. The LSTM Cell

Regarding the internal structure of the LSTM cell, it interacts with five distinct variables, as analysed below. This is how a standard LSTM cell operates, these variables are initialised by the cell and not manually. They are later affected by the input data fed into each cell. At first, the input vector $x_t$ is combined with the prior Hidden state $h_{t-1}$ and biases $b_f$ to generate the activation vector for the forget gate, presented in Equation 1. $W$ stands for the input weights, while $U$ stands for the recurrent connection weights. Both of these initial weights are commonly sampled from distributions with a zero mean and preset variances.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$

Likewise, the activation vector of the input or update gate is determined by considering the input vector $x_t$, the prior Hidden state $h_{t-1}$, as well as the biases $b_i$. As presented in Equation 2, this input has its own relationships and weights.

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$

Equations 3 and 4 determine the activation vectors for the output gate and the cell input, respectively.

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

The four aforementioned activation vectors are then used for determining the state of the Current cell, presented in Equation 5, where $\circ$ stands for the Hadamard product, also known as the element-wise product of two matrices.

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{5}$$

The calculation of the Hidden state vector, defined in Equation 6, which incorporates the state of the most recent cell along with the activation vector of the output gate, is the last step of the process. $\sigma_{c/t}$ represents the hyperbolic tangent and $\sigma_g$ the sigmoid function.

$$h_t = o_t \circ \sigma_h(c_t) \tag{6}$$

## 3.2. *Our modified LSTM cell*

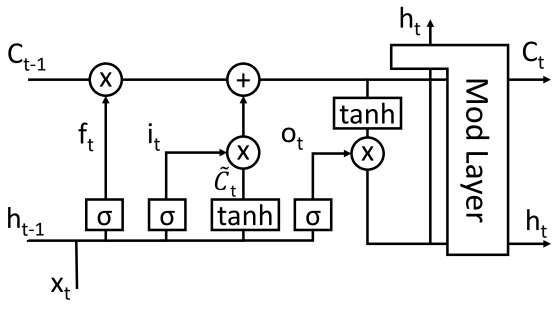

Figure 2. Our modified LSTM Cell

Table 1. Our experiments on the Current and Hidden state alterations

| Number | Current | Hidden |
|---|---|---|
| 1 | $c_t = c_t \circ o_t$ | $h_t = c \circ h_t$ |
| 2 | $c_t = c_t \circ h_t$ | $h_t = h_t \circ \sigma_g(c)$ |
| 3 | $c_t = c_t \circ o_t \circ h_t$ | $h_t = c \circ \sigma_g(h_t)$ |
| 4 | $c_t = c_t + c_t \circ h_t$ | $h_t = h_t \circ \sigma_c(c)$ |
| 5 | $c_t = c_t \circ \sigma_g(o_t)$ | $h_t = c \circ \sigma_c(h_t)$ |
| 6 | $c_t = c_t \circ \sigma_g(c_t)$ | |
| 7 | $c_t = c_t \circ \sigma_g(h_t)$ | |
| 8 | $c_t = h_t \circ \sigma_g(c_t)$ | |
| 9 | $c_t = c_t \circ \sigma_c(o_t)$ | |
| 10 | $c_t = c_t \circ \sigma_c(c_t)$ | |
| 11 | $c_t = c_t \circ \sigma_c(h_t)$ | |
| 12 | $c_t = h_t \circ \sigma_c(c_t)$ | |

Prior to the final Current and Hidden states being fed to the following LSTM cells, we introduce a new layer, Figure 2. This layer allows us to apply any kind of adjustment to both the Current or

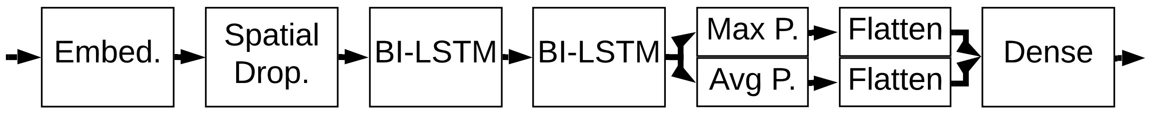

Figure 3. Our Stacked LSTM model

Hidden states of the cell. Even though we are also able to modify both cell states, for our experimentation section we decided to alter one of the Hidden or the Current cell states. This single-state alteration makes sure that we are able to accurately measure not only the performance changes but also the training time effect of that particular alteration. If we were to alter both Current and Hidden state, we would not be able to discern the per state alteration performance and training time effects. Additionally, by only making one alteration, we reduce the increase in training time. Due to the high dimensionality of the mathematical objects involved, even a simple addition would further affect training time.

Our aim is to increase the performance of each LSTM cell. Without any other architectural changes, our altered LSTM executes only a single additional state process inside each cell, with the goal of improving prediction outputs. Our experimentation includes five distinct Hidden state alterations and twelve Current state alterations, presented in Table 1. Although we also experimented with three different alterations to the output gate vector, our major focus is on the product of the final cell state $c_t$ with the Hidden state $h_t$ or vice-versa. Inside each LSTM cell, every alteration simply serves as the equation number seven of the LSTM-CS cell.

### 3.3. *Model*

The suggested changes are evaluated using a top-performing[22, 23] single network stacked Bi-LSTM model, Figure 3. The model relies on Bag-of-Words (BoW) embeddings that simply assign a numerical value to each term. A spatial dropout is applied to the data, which is then fed into a series of two Bi-LSTMs. The second Bi-LSTM receives as input the output of the first. Then, before the final fully connected dense layer, two separate pooling flows are established and later combined, one based on maximum and one on average pooling. Pooling reduces the dimensionality of the feature vectors. By combining both maximum and average pooling vectors into a slightly bigger vector, we aim to retain any pattern that might persist in any of the two. The stacked nature of this model will amplify the performance differences of our proposed LSTM cell. The modified LSTM-CS cell replaces all cells in both Bi-LSTM layers.

It should be noted that the inclusion of additional techniques, including trainable ensembles, pre-trained embeddings or even attention layers, could further enhance this approach. However, such architectural model enhancements are beyond the scope of this work, as the raw performance comparison of the LSTM cell and the custom state LSTM cell is our main research question. Furthermore, any pre-trained embedding implementation (Word2Vec, GloVe) would increase the total training time of our experiment by more than 8 times.

#### 3.3.1. *Hardware and Tensorflow*

The training was performed on a 5950X CPU, with 32GB of 3200hz RAM. All the code was implemented and executed in Python 3.8.8. The LSTM modification layer was coded around the basic LSTM cell on package Tensorflow-cpu from version 2.4+ up to version 2.9. Interested researchers can implement our approach by inserting any of our proposed alterations in line 1377 of the file: *recurrent_v2.py*, located in the directory: $Python\_Version//Lib//site-packages//keras//layers//$.

### 3.4. *Datasets*

Seven diverse text classification datasets are used for evaluating our modification layer. These datasets differ in Document size and classes, Maximum Sentence Length, Number of Unique Tokens and domain of application. The datasets used are the following.

The MLMA dataset[42] contains a sentiment label classifying a tweet against a set of one or more types of hate speech. The HASOC[37] dataset contains text from social media that has been classified as "of-

Table 2. Dataset characteristic, original and pre-processed

| | Original | | | | Pre-Processed | | | |
|---|---|---|---|---|---|---|---|---|
| Dataset | Documents | Classes | MaxLen | Tokens | Documents | Classes | MaxLen | Tokens |
| MLMA | 5647 | 71 | 35 | 14969 | 4138 | 48 | 23 | 12018 |
| SEMEVAL | 6838 | 11 | 33 | 24449 | 6495 | 11 | 22 | 14110 |
| HASOC | 5852 | 4 | 93 | 32168 | 1120 | 4 | 42 | 6078 |
| AG | 120000 | 4 | 122 | 123762 | 88431 | 4 | 22 | 4730 |
| ROBO | 525 | 5 | 29 | 466 | 254 | 5 | 10 | 149 |
| CROWD | 40000 | 11 | 34 | 83297 | 11377 | 11 | 11 | 1458 |
| HATE | 1011 | 3 | 611 | 6356 | 869 | 3 | 342 | 3235 |

fensive", "hate-speech", "profane", or "none". The SEMEVAL dataset[22,39] consists of a collection of Tweets annotated with emotional labels. The HATE dataset[a] is a class-balanced dataset with documents categorised into one of the three available classes of "hate speech", "severe hate speech", and "none". ROBO is a small-scale multi-class dataset with labels around human to robot interaction. Its size alone, less than 530 documents, positions ROBO as a unique dataset in our study. Such small datasets are known not to perform well with LSTMs, nonetheless we want to assess the proposed improvements to an extreme dataset like this one. CROWD is a crowdsourced dataset from Crowdflower with emotional labels. Finally, the AG News Topic Classification Dataset (AG)[64] contains topical categories for news related documents.

Table 2 presents the characteristics of the datasets before and after pre-processing. For each dataset, the term "Documents" denotes the number of sentences, "Classes" the number of classes per item, and "MaxLen" the length of the maximum sentence in the dataset, which directly affects the dimensionality of the training data. Finally, "Tokens" is the number of unique terms in each dataset.

Pre-processing is exactly the same for all datasets. We remove stop-words, non-alphanumeric characters, and extremely uncommon terms. We then convert all text to lowercase and replace the contractions, the shortening and combining of two words using apostrophes, using a custom GloVe method. To improve training time the AG dataset was further reduced by using a Term Frequency - Inverse Document Frequency (TF-IDF) algorithm, with minimal performance hit.[23]

[a] *https://github.com/GiannisHaralabopoulos/HateSpeech*

Table 2 displays the characteristics of the final datasets after pre-processing. In each dataset the number of unique terms and the maximum sentence length have been significantly reduced without affecting the number of documents. Excluding CROWD, where approximately two thirds of the data were deleted because of the brief and informal nature of the textual items, our pre-processing process slightly decreased the number of documents per dataset.

## 4. Evaluation

We conduct 10 separate runs, each with a different random seed, per dataset and state modification. In each run the data is split into train-validation-test splits of 80-10-10, with a 10-fold cross validation based on the random seed. For each dataset, the hyper-parameters are identical, as not to introduce volatility into the experimental process.

For each dataset (and its 10 runs) we will present the average percentage improvement. Given the space constraints, we will only highlight the best three Current and Hidden state alterations based on the combined improvement of Accuracy, Macro-F1, Micro-F1, and Weighted F1. Hidden state alterations are presented in the "H-number" format, whereas Current state alterations in the "C-number" format.

### 4.1. *Results*

Table 3. Improvement (%) for HASOC dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 83.984 | 35.546 | 65.811 | 64.799 |
| C-1 | **-0.064** | **1.192** | **-0.347** | -0.305 |
| C-4 | -0.101 | 0.202 | -0.460 | -0.527 |
| C-7 | -0.284 | 0.814 | -0.496 | **0.057** |
| H-2 | -0.167 | **-0.152** | **-0.266** | **-0.086** |
| H-3 | **-0.128** | -0.316 | -0.394 | -0.367 |
| H-5 | -0.255 | -0.792 | -0.812 | -0.701 |

Table 4. Improvement (%) for ROBO dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 80.198 | 1.562 | 2.476 | 1.965 |
| C-2 | -0.022 | 42.834 | 22.174 | 21.570 |
| C-5 | 0.087 | 48.085 | 24.028 | 16.320 |
| C-9 | **0.127** | **86.825** | **72.293** | **61.932** |
| H-1 | **0.156** | **96.034** | **68.244** | **57.003** |
| H-2 | -0.003 | 36.926 | 19.060 | 11.098 |
| H-5 | 0.047 | 71.101 | 57.587 | 40.362 |

In six of the seven datasets, our custom state LSTM cell improves F1 performance and accuracy. Results from HASOC and ROBO datasets, Tables 3 and 4, are particularly interesting since state alterations in HASOC fail to increase accuracy and micro-F1 score, whereas the same alterations in ROBO significantly increase performance across all metrics. We briefly mentioned the peculiarities of ROBO, very few documents, a low number of classes and tokens, and short sentences. HASOC on the other hand is a dataset with enough documents, a low number of classes, lengthy sentences, a lot of tokens, and most importantly imbalanced (74.3% of documents are labelled with the same class). In short, HASOC is single-class heavy and has a high encoded dimensionality. We believe that these factors strongly affect the performance of our Custom State LSTM, but further testing (ideally on datasets with similar properties) needs to be performed to provide safer conclusions with regards to this performance hit.

Table 5. Improvement (%) for MLMA dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 96.642 | 2.123 | 42.642 | 34.077 |
| C-1 | **0.021** | 1.396 | 1.595 | 1.367 |
| C-4 | **0.021** | 1.497 | 1.744 | 1.507 |
| C-6 | 0.018 | **1.595** | **1.891** | **1.585** |
| H-1 | 0.017 | 1.162 | 1.333 | 1.126 |
| H-3 | 0.014 | **1.461** | **1.802** | **1.466** |
| H-4 | **0.018** | 1.084 | 1.235 | 1.058 |

Table 6. Improvement (%) for AG dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 90.853 | 80.695 | 80.862 | 80.665 |
| C-1 | 0.293 | 1.006 | 0.832 | 1.008 |
| C-3 | 0.241 | 0.869 | 0.700 | 0.870 |
| C-10 | **0.325** | **1.020** | **0.860** | **1.020** |
| H-1 | 0.189 | 0.736 | 0.553 | 0.731 |
| H-3 | **0.262** | **0.908** | **0.731** | **0.900** |
| H-4 | 0.180 | 0.721 | 0.546 | 0.716 |

The modifications to the LSTM cell in MLMA and AG demonstrated the most improved classification results. As presented in Tables 5 and 6, Current state alterations resulted in F1 improvement up to 1.9% in MLMA and 1% in AGNEWS, respectively. Regarding accuracy metrics, they are improved by up to 0.3% in AGNEWS but remain largely unchanged in MLMA. While still improving classification results, alterations of Hidden state are slightly outperformed by Current state ones.

Table 7. Improvement (%) for CROWD dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 89.237 | 13.577 | 26.840 | 24.625 |
| C-5 | -0.045 | **0.770** | 0.249 | 0.398 |
| C-7 | **-0.033** | 0.535 | 0.537 | **0.646** |
| C-11 | -0.041 | 0.668 | **0.608** | 0.591 |
| H-2 | -0.062 | -0.123 | **0.341** | **0.376** |
| H-3 | -0.053 | 0.126 | -0.109 | -0.038 |
| H-5 | **-0.034** | **0.414** | 0.217 | 0.354 |

Table 8. Improvement (%) for HATE dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 77.397 | 65.710 | 66.101 | 66.714 |
| C-2 | **0.470** | **0.871** | **0.825** | **0.858** |
| C-6 | 0.233 | 0.580 | 0.517 | 0.557 |
| C-8 | 0.193 | 0.585 | 0.487 | 0.506 |
| H-1 | -0.056 | -0.099 | -0.138 | -0.149 |
| H-4 | 0.029 | 0.166 | 0.131 | 0.195 |
| H-5 | **0.248** | **0.641** | **0.544** | **0.589** |

Alterations in the CROWD dataset demonstrate a similar performance, Table 7. Two out of five Hidden state alterations result in a consistent classification improvement, whereas Current state alterations increase F1 score by up to 0.65%. Although all F1 metrics have improved, the accuracy metric for Current state alterations has slightly decreased in all three of the top performing ones.

Table 9. Improvement (%) for SEMEVAL dataset

| Metrics | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| Baseline | 83.833 | 44.106 | 57.981 | 56.083 |
| C-3 | 0.003 | 0.128 | 0.444 | 0.230 |
| C-6 | **0.099** | 0.199 | **0.448** | **0.246** |
| C-7 | 0.012 | **0.248** | 0.321 | 0.146 |
| H-2 | -0.042 | 0.259 | 0.218 | **0.189** |
| H-3 | **0.026** | 0.167 | **0.277** | 0.098 |
| H-4 | -0.016 | **0.395** | 0.138 | 0.077 |

Table 10. Average Improvement (%) for Current State Alterations (Excluding ROBO dataset)

| | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| 1 | 0.018 | 0.643 | **0.436** | **0.41** |
| 2 | -0.012 | **0.659** | 0.145 | 0.326 |
| 3 | -0.047 | 0.112 | 0.143 | 0.199 |
| 4 | -0.009 | 0.599 | 0.346 | 0.376 |
| 5 | -0.021 | 0.3 | 0.184 | 0.227 |
| 6 | -0.015 | 0.381 | 0.319 | 0.391 |
| 7 | -0.123 | 0.356 | 0.135 | 0.23 |
| 8 | **0.037** | 0.182 | 0.369 | 0.34 |
| 9 | -0.109 | 0.176 | 0.003 | 0.099 |
| 10 | 0.028 | 0.249 | 0.23 | 0.233 |
| 11 | -0.054 | 0.068 | 0.206 | 0.176 |
| 12 | -0.024 | 0.455 | 0.229 | 0.282 |

Table 11. Average Improvement (%) for Hidden State Alterations (Excluding ROBO dataset)

| | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| 1 | -0.05 | -0.301 | -0.027 | -0.055 |
| 2 | -0.047 | 0.24 | **0.305** | **0.317** |
| 3 | -0.065 | **0.312** | 0.276 | 0.242 |
| 4 | -0.13 | -0.038 | -0.145 | 0.026 |
| 5 | **0.008** | 0.2 | 0.085 | 0.165 |

As presented in Table 8, the Hidden state alterations for the HATE dataset function similarly to those for CROWD, where only two of five experiments result in improved classification results. Current state alterations result in up to 0.87% improved results, with an improvement in Accuracy as well. Table 9 shows that both Current and Hidden state alterations with the SEMEVAL dataset demonstrated a marginal improvement. The best-performing alterations improve classification for Current by 0.45% and 0.28% for Hidden states.

Overall, the majority of Current and Hidden alterations result in significant improvements in each classification task, as presented in Tables 10 and 11. The ROBO dataset results are not included, as they are considered outliers due to their unique characteristics in terms of document and token size.

F1 metrics are improved by an average of 0.5% using the top-performing Current state alteration $c_t = c_t \circ o_t$. Even the least effective ninth Current state modification $c_t = c_t \circ \sigma_c(o_t)$ increases F1 metrics by an average of 0.1%, according to Table 10.

Hidden state modifications performed inconsistently. Only one of our proposed alterations improved across all F1 metrics, while two alterations did not produce any improvement at all. The top-performing alteration $h = h \circ \sigma_g(c)$ raised F1 scores on average by 0.29%, as presented in Table 11. The worst-performing alteration ( $h = c \circ h$, not only does not increase the classification results, but instead reduces all metrics.

## 5. Limitations

As already mentioned, our proposed modification layer enables alterations to both the Current and Hidden states of an LSTM cell. The state functions inside the LSTM cell are the basis for every alteration we propose. However, this does not imply that

Table 12. Accuracy Score for each modification and model (%)

| | Modification | MLMA | HASOC | AGNEWS | ROBO | CROWD | HATE |
|---|---|---|---|---|---|---|---|
| LSTM-CS | No cell mod | 96.642 | 83.984 | 90.853 | 80.198 | 89.237 | 77.397 |
| | Top Current State Mod | 96.662 | 83.931 | 91.120 | **80.266** | **89.293** | 77.304 |
| | Top Hidden State Mod | 96.654 | 83.844 | 91.029 | 80.195 | 89.182 | 77.227 |
| SIMPLE | No cell mod | 96.210 | 84.152 | 92.403 | 77.551 | 87.557 | 76.369 |
| | Top Current State Mod | 96.179 | 84.308 | 92.390 | 78.449 | 87.767 | 75.412 |
| | Top Hidden State Mod | 96.258 | 83.795 | 92.437 | 78.289 | 87.480 | 76.063 |
| BERT | N/A | **96.992** | **89.286** | **92.943** | 76.111 | 87.881 | **81.590** |

only pre-existing functions can be used in the modification layer. Apart from incorporating additional functions like Relu or Softmax, we could also alter both Current and Hidden states at the same time.

Moreover, inside the modification layer an additional single equation is solved. The high dimensional mathematical entities known as tensors, however, are subject to a number of calculations based on that single equation. For the HATE dataset, for instance, 1,222,560 parameters are the input to the second LSTM. Practically speaking, any additional calculations involving objects of a similar size will increase training time. To illustrate the higher computational resource utilisation on our modified cell, we present in Table 13 the increased training time per state alteration, for the MLMA dataset as an example.

An initial observation is that increased training time does not always result in better classification results. The top-performing MLMA alterations for Current state were C-1, C-4, and C-6, while for Hidden state H-1, H-3, and H-4. Both states had alterations that increased the training time but did not improve performance. Since the extra training time does not scale linearly with the dataset size, we avoid including an averaged metric for all datasets.

## 6. Comparison Against Transformer Models

Table 13. Training time per fold, in seconds

| Dataset | LSTM-CS | SIMPLE | BERT |
|---|---|---|---|
| MLMA | 9 | 37 | 383 |
| HASOC | 5 | 21 | 234 |
| AGNEWS | 83 | 345 | 15,265 |
| ROBO | <1 | 6 | 56 |
| CROWD | 54 | 224 | 5,410 |
| HATE | 19 | 79 | 181 |

Since 2018 and after the introduction of BERT,[13] transformer models have been considered the top performing methods for a multitude of NLP tasks, and among them text classification. In most cases, the performance of such models has been better than past Deep Learning Networks, with minor exceptions in small datasets.[8,15] In this chapter, we will compare our proposed LSTM modification (LSTM-CS) performance against a simple transformer model (to be referred to as SIMPLE) and BERT (BERT), the most commonly used pre-trained transformer model.

Table 14. Number of trainable parameters for CROWD dataset

| Model | Trainable Parameters | Avg Training time per Fold |
|---|---|---|
| LSTM-CS | 940,596 | 34s |
| SIMPLE | 1,194,987 | 118.7s |
| BERT | 109,881,611 | 3588.2s |

In the previous section, we noticed increased training time when applying our LSTM cell modification. Similarly, an expected limitation of using a transformer model is increased complexity. By introducing higher complexity to the model, more computational resources are required for training. This impacts the training time per epoch and, subsequently, the time to complete the task as a whole.

Table 15. Macro F1 Score for each modification and model (%)

| | Modification | MLMA | HASOC | AGNEWS | ROBO | CROWD | HATE |
|---|---|---|---|---|---|---|---|
| LSTM-CS | No cell mod | 2.123 | 35.546 | 80.695 | 1.562 | 13.577 | 65.710 |
| | Top Current State Mod | 2.153 | 35.970 | 81.506 | 1.534 | 13.618 | 65.581 |
| | Top Hidden State Mod | 2.146 | 35.492 | 81.259 | 2.138 | 13.561 | 65.517 |
| SIMPLE | No cell mod | 2.940 | 37.138 | 84.538 | 30.941 | 15.479 | 61.260 |
| | Top Current State Mod | **2.970** | 37.107 | 84.471 | 28.273 | 14.856 | 61.474 |
| | Top Hidden State Mod | 2.447 | 38.143 | 84.579 | 27.084 | 15.176 | 61.291 |
| BERT | N/A | 2.759 | **54.430** | **85.910** | **36.028** | **20.872** | **71.726** |

In this chapter, we will evaluate the classification performance of our proposed modified LSTM cell and the two aforementioned transformer models. The first transformer model is not pre-trained and can be efficiently combined with LSTM layers, whereas the BERT model uses pre-trained embeddings and cannot be efficiently used with LSTM layers, mainly due to Input/Output differences in the dimensions of the tensors.

The datasets we will use are the same as before, with the exception of SEMEVAL which is a multi-label dataset. This is due to the fact that Multi-label classification requires a completely different BERT architecture that not only introduces more complexity, but is also fundamentally different from the two models of the comparison, SIMPLE and LSTM-CS.

Once again, we perform 10 runs with different random seeds, each with a 10-fold cross validation. For all three methods we use an 80-10-10 split and exactly the same hyper-parameters, such as epoch and batch numbers. The evaluation numbers are based on the average testing performance of all folds and iterations. In total, more than 6,000 training and testing epochs were required for the purposes of this comparison alone.

The accuracy of each method is depicted in Table 12. BERT outperforms all methods by 0.93% on average when evaluated with accuracy. Similarly, the macro F1 scores have improved by 16.43%. Keep in mind that BERT is pre-trained, which means that the encoding is vastly more complex than the simple encoding of the BoW method we incorporate on LSTM-CS and SIMPLE models. The effect of non pre-trained embeddings is evident by the small improvement (0.668%) of SIMPLE compared to our LSTM-CS. However, these improvements come at the great cost of computational resources. In order to better convey the exponentially increased complexity of a BERT model compared to the LSTM, in Table 14 we include the number of trainable parameters and the average training time per fold for each of the three compared models for the CROWD dataset. In addition, the detailed time per fold training times for each dataset are depicted in Table 13.

Our best performing modification for Current state demonstrates better performance in 3 out of 12 datasets, while it even manages to improve performance when used with a simple transformer model. Although BERT -and transformer models in general- provide overall better results, they do come with a great computational cost. The BERT model required more than a hundred times the training time needed for our best Current state LSTM modification to produce disproportionately improved results.

## 7. Conclusions

In this work, we propose a LSTM modification layer that enables alterations to Current and Hidden states of the LSTM cell. Our experiments involve altering either state, however the layer gives us the option to alter both states at once. Our experimentation focuses on introducing simple equations with the goal of improving the LSTM cell's performance. Out of our 17 single-state alteration experiments, 12 involve the Current state, whereas 5 deal with the Hidden state.

Table 16. Percentage improvement over baseline for best performing state alterations

| Alteration | Accuracy | Macro | Micro | Weighted |
|---|---|---|---|---|
| $c_t = c_t \circ o_t$ | **0.018** | **0.643** | **0.436** | **0.41** |
| $h = h \circ \sigma_g(c)$ | -0.047 | 0.24 | 0.305 | 0.317 |

Using seven diverse datasets, we assess the performance of our modified LSTM and each of the 17 single-state alterations. The datasets relate to sentiment analysis, document classification, hate speech detection, and human-to-robot interaction tasks, whereas their properties, such as document and term sizes, vary. Our findings reveal that both state alterations result in improvements of the classification result.

Table 16 lists the best-performing alterations for each state. Since the ROBO dataset shows a bigger improvement than the other datasets, we believe this performance is an outlier due to the dataset properties, and thus has not been included in the calculations with the other datasets. The best-performing Current state alteration (No1 in Table 1): $c_t = c_t \circ o_t$, improving Macro-F1 by up to 0.643%, whereas the best-performing Hidden state alteration (No2 in Table 1): $h = h \circ \sigma_g(c)$, improving Weighted-F1 by up to 0.317%. With the aim of improving classification results, the modification layer adds an additional training step inside the LSTM cell.

These training enhancements do, however, increase training time. Although there is a less than 10% increase in the required training time, it can nonetheless have an adverse effect on the scalability of similar methods. In the end, even for improving the classification accuracy by 0.5%, in case that the classification process involves sensitive data, such as law enforcement or health issues, then small increases in resource needs can be justified. There is a level of subjectiveness in this threshold. For example, BERT demonstrated a 16% F1 improvement over the LSTM cell, but with 10,000% increased training time. If the dataset size and the initial training time are small enough, we could definitely recommend the use of a more complex pre-trained Language Transformer model.

The single-state alteration facilitates a more accurate measurement of performance per state alteration. We only experimented with single-state alterations despite being able to modify both cell states at once. Our findings reveal that Current state alterations not only result in improved performance but also typically involve shorter training periods. Furthermore, we are currently working to identify the dataset properties that inhibit LSTM-CS performance. Once we identify these, we will be able to test specific alterations or conclude on the efficiency of LSTM-CS for this type of datasets.

In the future, we intend on the one hand to experiment with co-occurring alterations for both Current and Hidden states as well as to apply adjustments to gated outputs, while on the other hand, to identify the most effective alteration(s) to maximise prediction results.[1,47] To achieve this, we will collaborate with colleagues from Statistics and Mathematics to optimise our suggested single-state alteration. We would also like to apply our LSTM-CS to more ML tasks where LSTM outperforms other Deep Learning methods. We believe that the results of our work will encourage other researchers to investigate the possibility of further altering the LSTM gate or state outputs with the goal of extracting more performance from each LSTM cell.

## Bibliography

1. K. M. R. Alam, N. Siddique, and H. Adeli. A dynamic ensemble learning algorithm for neural networks. *Neural Computing and Applications*, 32:8675–8690, 2020.
2. F. Alkomah and X. Ma. A literature review of textual hate speech detection methods and datasets. *Information*, 13(6):273, 2022.
3. Z. Alom, B. Carminati, and E. Ferrari. A deep learning model for twitter spam detection. *Online Social Networks and Media*, 18:100079, 2020.
4. J. E. Arco, A. Ortiz, J. Ramírez, Y.-D. Zhang, and J. M. Górriz. Tiled sparse coding in eigenspaces for image classification. *International Journal of Neural Systems*, 32(03):2250007, 2022.
5. Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.
6. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.
7. S. Chopra, S. Jain, and J. M. Sholar. Towards automatic identification of fake news: Headline-article stance detection with lstm attention models. In *Stanford CS224d Deep Learning for NLP final project*, 2017.
8. W. Cunha, V. Mangaravite, C. Gomes, S. Canuto, E. Resende, C. Nascimento, F. Viegas, C. França, W. S. Martins, J. M. Almeida, et al. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *Information Processing & Management*, 58(3):102481, 2021.
9. O. K. Cura and A. Akan. Classification of epileptic eeg signals using synchrosqueezing transform and machine learning. *International journal of neural*

*systems*, 31(05):2150005, 2021.

10. J. Dai, C. Chen, and Y. Li. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878, 2019.
11. S. Dai, L. Li, and Z. Li. Modeling vehicle interactions via modified lstm models for trajectory prediction. *IEEE Access*, 7:38287–38296, 2019.
12. N. C. Dang, M. N. Moreno-García, and F. De la Prieta. Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3):483, 2020.
13. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
14. D. Eck and J. Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103:48, 2002.
15. A. Ezen-Can. A comparison of lstm and bert for small corpus. *arXiv e-prints*, pages arXiv–2009, 2020.
16. W. Fang, H. Luo, S. Xu, P. E. Love, Z. Lu, and C. Ye. Automated text classification of near-misses from safety reports: An improved deep learning approach. *Advanced Engineering Informatics*, 44:101060, 2020.
17. F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. 1999.
18. A. Graves, D. Eck, N. Beringer, and J. Schmidhuber. Biologically plausible speech recognition with lstm neural nets. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 127–136. Springer, 2004.
19. A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2008.
20. A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5):602–610, 2005.
21. D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.
22. G. Haralabopoulos, I. Anagnostopoulos, and D. McAuley. Ensemble deep learning for multilabel binary classification of user-generated content. *Algorithms*, 13(4):83, 2020.
23. G. Haralabopoulos, M. T. Torres, I. Anagnostopoulos, and D. McAuley. Text data augmentations: Permutation, antonyms and negation. *Expert Systems with Applications*, page 114769, 2021.
24. G. Haralabopoulos, C. Wagner, D. McAuley, and E. Simperl. A multivalued emotion lexicon created and evaluated by the crowd. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 355–362. IEEE, 2018.
25. J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.
26. Y. Hu, X. Sun, X. Nie, Y. Li, and L. Liu. An enhanced lstm for trend following of time series. *IEEE Access*, 7:34020–34030, 2019.
27. F. Huang, X. Li, C. Yuan, S. Zhang, J. Zhang, and S. Qiao. Attention-emotion-enhanced convolutional lstm for sentiment analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
28. Y. Huang, W. Wang, and L. Wang. Instance-aware image and sentence matching with selective multimodal lstm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2310–2318, 2017.
29. P. Jansson and S. Liu. Topic modelling enriched lstm models for the detection of novel and emerging named entities from social media. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4329–4336. IEEE, 2017.
30. N. Kalchbrenner, I. Danihelka, and A. Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.
31. D. P. Kasseropoulos, P. Koukaras, and C. Tjortjis. Exploiting textual information for fake news detection. *International Journal of Neural Systems*, 32(12):2250058, 2022.
32. A. L. Levada, D. C. Correa, D. H. Salvadeo, J. H. Saito, and N. D. Mascarenhas. Novel approaches for face recognition: template-matching using dynamic time warping and lstm neural network supervised classification. In *2008 15th International Conference on Systems, Signals and Image Processing*, pages 241–244. IEEE, 2008.
33. C. Li, G. Zhan, and Z. Li. News text classification based on improved bi-lstm-cnn. In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pages 890–893. IEEE, 2018.
34. G. Liu and J. Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.
35. G. Liu, L. Tian, and W. Zhou. Patient-independent seizure detection based on channel-perturbation convolutional neural network and bidirectional long short-term memory. *International Journal of Neural Systems*, 32(06):2150051, 2022.
36. A. Makkar and N. Kumar. An efficient deep learning-based scheme for web spam detection in iot environment. *Future Generation Computer Systems*, 108:467–487, 2020.
37. T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, and A. Patel. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th forum for information retrieval evaluation*, pages 14–17, 2019.
38. A. Mittal, P. Kumar, P. P. Roy, R. Balasubramanian,

and B. B. Chaudhuri. A modified lstm model for continuous sign language recognition using leap motion. *IEEE Sensors Journal*, 19(16):7056–7063, 2019.

39. S. Mohammad, F. Bravo-Marquez, M. Salameh, and S. Kiritchenko. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17, 2018.
40. A. Nandi, F. Xhafa, L. Subirats, and S. Fort. Reward-penalty weighted ensemble for emotion state classification from multi-modal data streams. *International journal of neural systems*, pages 2250049–2250049, 2022.
41. Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. Hierarchical multimodal lstm for dense visual-semantic embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 1881–1889, 2017.
42. N. Ousidhoum, Z. Lin, H. Zhang, Y. Song, and D.-Y. Yeung. Multilingual and multi-aspect hate speech analysis. In *Proceedings of EMNLP*. Association for Computational Linguistics, 2019.
43. H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707, 2016.
44. D. Pawade, A. Sakhapara, M. Jain, N. Jain, and K. Gada. Story scrambler-automatic text generation using word level rnn-lstm. *International Journal of Information Technology and Computer Science (IJITCS)*, 10(6):44–53, 2018.
45. A. Pulver and S. Lyu. Lstm with working memory. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 845–851. IEEE, 2017.
46. D. Qiu, Z. Liu, Y. Zhou, and J. Shi. Modified bi-directional lstm neural networks for rolling bearing fault diagnosis. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.
47. M. H. Rafiei and H. Adeli. A new neural dynamic classification algorithm. *IEEE transactions on neural networks and learning systems*, 28(12):3074–3083, 2017.
48. G. Rao, W. Huang, Z. Feng, and Q. Cong. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57, 2018.
49. B. Ren. The use of machine translation algorithm based on residual and lstm neural network in translation teaching. *Plos one*, 15(11):e0240663, 2020.
50. K. M. Rocki. Surprisal-driven feedback in recurrent networks. *arXiv preprint arXiv:1608.06027*, 2016.
51. M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.
52. M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. S. Choi, and B.-W. On. Fake news stance detection using deep learning architecture (cnn-lstm). *IEEE Access*, 8:156695–156706, 2020.
53. L.-N. Wang, G. Zhong, S. Yan, J. Dong, and K. Huang. Enhanced lstm with batch normalization. In *International Conference on Neural Information Processing*, pages 746–755. Springer, 2019.
54. X. Wang, G. Zhang, Y. Wang, L. Yang, Z. Liang, and F. Cong. One-dimensional convolutional neural networks combined with channel selection strategy for seizure prediction using long-term intracranial eeg. *International journal of neural systems*, 32(02):2150048, 2022.
55. S. Wen, H. Wei, Y. Yang, Z. Guo, Z. Zeng, T. Huang, and Y. Chen. Memristive lstm network for sentiment analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.
56. S. Wolyn and S. J. Simske. Summarization assessment methodology for multiple corpora using queries and classification for functional evaluation. *Integrated Computer-Aided Engineering*, (Preprint):1–13, 2022.
57. Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, and R. R. Salakhutdinov. On multiplicative integration with recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.
58. A. Yadav and D. K. Vishwakarma. Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, 53(6):4335–4385, 2020.
59. L. Yao, Z. Pan, and H. Ning. Unlabeled short text similarity with lstm encoder. *IEEE Access*, 7:3430–3437, 2018.
60. C. Ye, X. Li, and J. Chen. A deep network for tissue microstructure estimation using modified lstm units. *Medical image analysis*, 55:49–64, 2019.
61. D. Zeng, C. Sun, L. Lin, and B. Liu. Lstm-crf for drug-named entity recognition. *Entropy*, 19(6):283, 2017.
62. G. Zhang, X. Zhang, H. Rong, P. Paul, M. Zhu, F. Neri, and Y.-S. Ong. A layered spiking neural system for classification problems. *International journal of neural systems*, 32(08):2250023, 2022.
63. W. Zhang, Y. Li, and S. Wang. Learning document representation via topic-enhanced lstm model. *Knowledge-Based Systems*, 174:194–204, 2019.
64. X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.