National College of Ireland

# Hyper-parameter Optimization for LSTM models

MSc Research Project
Data Analytics


Dev Sharma
Student ID: X23139676

School of Computing
National College of Ireland


Supervisor: Abdul Qayum

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

National College of Ireland

| | |
|---|---|
| **Student Name:** | Dev Sharma |
| **Student ID:** | X23139676 |
| **Programme:** | Data Analytics |
| **Year:** | 2024 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Abdul Qayum |
| **Submission Due Date:** | 12/08/2024 |
| **Project Title:** | Hyper-parameter Optimization for LSTM models |
| **Word Count:** | 7865 |
| **Page Count:** | 20 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | Dev Sharma |
| **Date:** | 12th August 2024 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Hyper-parameter Optimization for LSTM models

Dev Sharma
X23139676

**Abstract**

Due to the high potential of Natural Language Processing (NLP) in various applications the study in this domain has increased rapidly. More precisely the usage of Long Short Term Memory (LSTM) networks for different applications such as text prediction, language modelling and machine translations have risen. But the performance of the LSTM model remains a challenging task despite the effectiveness of the LSTM networks. The current research study does not detail about the dependencies between the hyperparameters and performance of the model which therefore leads to poor performance results. Therefore this research tries to address this gap and thereby increase the real world applicability. This research aims to determine the impact of hyperparameter optimization on LSTM models for NLP applications. In this study the application is text prediction The data which is present in form of text is extracted from a PDF document using the PyPDF2 library and is tokenized with the help of Keras Tokenizer which is offered by TensorFlow. Before the modelling process is started the data is prepared with the help of different pre-processing techniques. An LSTM model is built which consists of number of different layers having different functionalities. Different hyperparameters such as embedding output dimension , LSTM unit, dropout rate and learning rate are taken into consideration. These hyperparameters are optimized with the help of the Random Search approach provided by the Keras Tuner and the performance is evaluated. The results obtained show that there is marginal improvement when compared with baseline model The study will help in providing a comprehensive framework for enhancing the performance of the neural networks in the text prediction applications.

## 1 Introduction

The fast pace advancement in the domain of natural language processing has transformed a wide variety of fields ranging from automated translations to sentimental analysisWu et al. (2021). However, when comparing all the techniques, the Long Short Term Memory networks is one of the most advanced tools for applications such as sequence prediction and text generation due to its ability to determine and capture the long range dependencies in the data. Due to the scarcity in the availability of effective text prediction models for applications such as predictive text input, language modelling and automated content creation they are in high demand. Despite various findings the optimization of the performance of the LSTM models remains a very crucial and less touched area of study. It becomes important because it directly impacts the accuracy as well as the applicability in the real world use cases Sundermeyer et al. (2012). Hyperparameter optimization has

a very significant role in fine tuning the architecture and the parameters of the developed model which will lead to improvement in the model's efficiencyLiu et al. (2021).

This **research question** trying to solve is: **Measuring the impact of hyperparameter optimization on the performance on LSTM networks for text prediction application**. The main objectives of this research is:

- Developing an LSTM model which can effectively perform text prediction and generation.
- Analysing the impact of different hyperparameters on the performance of the model.
- Optimization using the Keras Tuner's random search method.

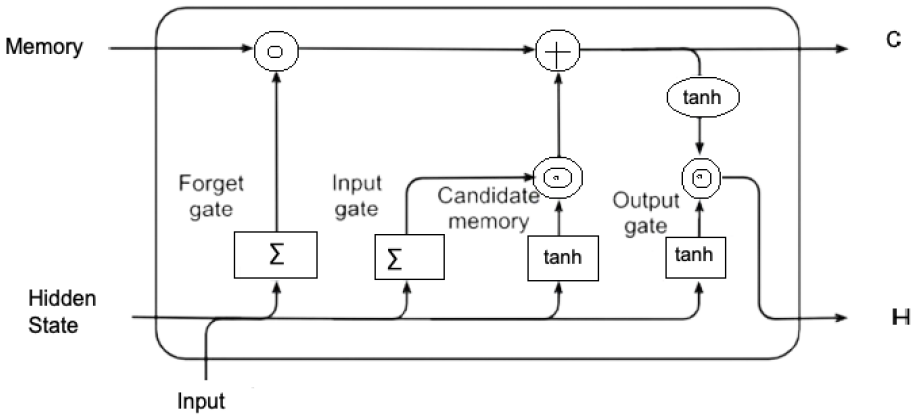


Figure 1: LSTM Architecture Calzone (2019)

The Fig.1 shows the architecture of the LSTM network. LSTM is special type of recurrent neural network (RNN) which is designed to model sequential data by learning long term dependencies present in the data. It makes use of different gates such as the input gate, forget gate and the output gate for controlling the flow of the information. The special characteristic of LSTM is its ability to selectively remember and forget information using these gates. This makes LSTMs suitable for a wide variety of applications such as text predictions, timer series forecasting.The hypothesis assumed for this research is that if the hyperparameters are optimized systematically then the performance of the LSTM model used for text prediction tasks is enhanced. By identifying the key parameters that will be crucial for this task and tuning carefully will help in achieving better accuracy and generalization.

The motivation for this research emerges because of the increasing demand of intelligent text processing systems. The ability to predict and generate text is essential for a wide range of applications such as virtual assistants and automated content creation. As LSTM networks are capable to understand and maintain long term dependencies in the data so they are the most suited for these tasks. However, the performance of the LSTM models are highly dependent on the selection of the hyperparameter values which includes the number layers, units in each layer, dropout rate and the learning rate Zaremba et al. (2014). Therefore, this research aims to systematically find the impact of various hyperparameters on the performance of the LSTM model and also identifying the optimal configurations in order to get the best result for the text prediction application. This

approach is expecting to increase the accuracy of the model as well as help in finding the a relationship between hyperparameters and how they impact the model's performance.

This study will significantly contribute to the scientific literature by providing a complete detailed methodology for the LSTM hyperparameter optimization, which will highlight the importance of fine tuning the neural networks. The findings that will be obtained from this research will provide valuable insights about the practices that should be followed in order to for configuring the LSTM model for text prediction tasks. Additionally, the use of random search methodology provided by Keras will provide a complete framework for hyperparameter optimization which can be applied on other types of neural networks and different machine learning models. Majorly the contribution of this research is twofold:

- Methodological Framework: The study will provide a comprehensive methodology ranging from extraction, processing and modelling of the text data using the LSTM model with major focus on the hyperparameter optimization.
- Empirical Findings: The results of this research will provide an empirical evidence of the impact of hyperparameter tuning on the LSTM model performance, demonstrating how optimization can lead to significant improvement in the accuracy and generalization of the optimized mode.

The structure of the report will be as follows:

- Literature Review: A deep analysis of the existing methodologies and the previous research that had been conducted related to the LSTM networks and optimization of the hyperparameter. This will help in providing a theoretical foundation which will help in understanding the significance of hyperparameter in the neural networks.
- Methodology: In this section there will be complete discussion starting from the data preparation, model building and tuning of the hyperparameters in detail. This will also include the steps that will be followed to extract the text data, creation of the training data and building of the LSTM model.
- Results: This section will present the findings that are obtained from the research, including the performance measurement metrics and a detailed analysis. There will also be a complete overview on the results of the optimization and the effectiveness of the optimized model.
- Conclusion and Future Work: In this section there will be a complete summary of the outcomes of the conducted research and also what are the possible future work.

# 2 Related Work

The literature review is divided into subsection according to the specific details as per requirement.

## 2.1 Optimization

The research conducted by Abdolrasol et al. (2021) proposed to develop an optimization technique with the help of artificial neural networks to solve a variety of problems.

Initially the authors begins with the discussion for the need for optimization technique for machine learning applications also the authors gives much emphasis on the usage of the neural networks which are complex computational algorithms based on the biological nervous system. The authors also states that the popularity of these algorithms are increasing in the recent times because of their ability to process complex data efficiently. The different algorithm used in the research paper include genetic algorithm, particle swam optimization, artificial bee colony, lightning search algorithm and many more algorithms. Each of these above mentioned algorithm is discussed in the context of optimization with the usage of neural networks to improve their performance by manipulating the hyperparameter. This research showcases the results of the optimization technique in various applications such as energy management, forecasting, data analysis and other real life applications. In one of the applications discussed by the authors the particle swam optimization technique is used to optimize the number of neurons which are present in the hidden layer and also optimizing the learning rate for improving the energy management in the virtual power plants. In another application the authors uses the genetic algorithm to optimize the predictive and forecasting applications. For the process of evaluation the authors does a comparative analysis between the results that are obtained from the baseline model and the results from the models when the parameters are fine tuned. The research does not explicitly make use of a particular dataset rather it focusses only on the development technique using the artificial neural network. To evaluate the accuracy of the developed model mean absolute error (MAE) is employed. The positives of this research conducted is it discusses a wide variety of optimization technique which comprises of both the conventional and non-conventional techniques for different applications. A number of different applications are also discussed which will significantly contribute during the research process. The negatives of this research is that it lacks comparison between the optimization techniques.

The study conducted in Aldabbagh and Syed (2023) discusses about the importance of hyperparameter tuning in the artificial neural networks. These hyperparameters include the weights, learning rate, batch size and other parameters which can be tuned in order to increase the efficiency of the model. Initially the research paper highlights problems that arises because of the conventional optimizations technique such as the gradient descent algorithm for improving the efficiency. To solve this issue the researchers proposes a metaheuristic approach which will help in determining the most optimum values of the hyperparameter. More specifically this involves the development of a technique using the variable neighbourhood search to overcome the limitations. Further in the research the authors performs an extensive literature review on the metaheuristic algorithm which is used for the optimization of the artificial neural network. The research uses three datasets which comprises of a wide variety of data having different structure, types, distribution and complexities which will help in determining the robustness and the stability of the variable neighbourhood search algorithm under different conditions. There is no mention regarding the source of the dataset. The results obtained after performing the research deem that there is a varying trade-off between the speed and the accuracy. The simple back propagation method outperforms all others in terms of speed whereas the full neighbourhood search back propagation method outperforms in terms of accuracy. This indicates that which model to be used highly depends on the type of application in which it is employed. In the case of time critical applications where there is no requirement of high accuracy the simple back propagation method will be useful where in cases where accuracy matters the most the latter model can be employed. The

evaluation method used for the model is based on comparing the performance of different Variations of Variable Neighbourhood Search (VNS) for Artificial Neural Network (ANN) structure optimization. The evaluation involves assessing the trade-offs between speed and accuracy for each VNS variation. The evaluation also utilizes various datasets to gauge the algorithm's adaptability and generalizability. Overall the paper provides a complete overview but lacks in few areas. There is no discussion regarding the scalability and the generalization of the developed model which is a very important point.

The research study done by Powell et al. (2020) introduces a real time optimization method using the reinforcement learning. It uses an actor – critic architecture to make and apply optimal decisions in response to the stimuli that is generated in the neural network. This working methodology makes it different from the conventional real time optimization methods which includes reputedly solving a process in order to determine the optimal values for the model. Further in the research the authors discusses about the working methodology of the proposed model. To demonstrate the working the testing is performed on the real time data generated from a chemical reactor process system to continuously update the parameters in order to maximize the efficiency. After applying the proposed model on the chemical reactor it was seen that there was an increase of 9.6% in the annual profits of the business. The evaluation method for this research model involves a closed-loop simulation using the original dynamic model of the chemical reactor system, with measured disturbances changing at random. The RL-RTO application is put in closed-loop with the plant, and the simulation is run for a year in order to summarize the results and compare them to a conventional NLP-based RTO application. Further. The researchers also state that more development is needed in this reinforcement learning based real time optimization method to be able to compete with the conventional real time optimization methods. The proposed method is not able to fully use the capabilities of the reinforcement learning for the optimization process. Moreover, the authors state that the RL RTO method exhibited abnormal behaviour in certain conditions where the complexity of the data changes. Overall, the results are promising but further improvement is also required in order to make it useful for real world applications.

The research study done in Gorgolis et al. (2021) performs hyperparameter optimization of LSTM network for the NLP applications. It makes use of the genetic algorithm for the same. The application for which the model is being developed is next word prediction which is majorly used in the natural language processing field. Further the authors discusses that artificial neural networks are most commonly used for this application and hyperparameter optimization is essential in order to improve the performance of the model. The authors emphasize on the importance of sequence prediction in various task such as text auto correction, generation and many more. Also since neural networks involves a number of different parameters finding the optimal value becomes very important. The genetic algorithm involves creating number of initial models using randomly choosing the hyperparameter value, evaluating the fitness of each of it using metrics such as accuracy or perplexity and then selecting the fittest model through the process of elitism. The concept of crossover and mutation is also defined. Crossover is defined as the process of selecting two parent models and combing them to develop an offspring model. Mutation is introducing random changes to the hyperparameters of the model. In the first variation of the algorithm, concept of mutation is applied to the survivors of the previous generation, whereas in the second variation mutation is applied to the entirely new generation after the crossover process is completed. This random variation helps in exploring a large space of hyperparameter thereby finding the optimal value which results

in increased performance. The paper doesn't provide any dataset rather focuses only on the methodology of developing the model. However, the choice of dataset will have a significant impact on the performance of the model. Also there is no discussing regarding the potential drawbacks or limitations of the model.

The study conducted in Hashem et al. (2017) proposes an adaptive stochastic conjugate gradient (ASCG) technique for the purpose of optimization of the neural networks. For the purpose of this study the authors makes use of publicly available dataset. The first dataset contains 5000 instances and includes 12 characteristics whereas the second dataset has 769 instances with 9 characteristics. Appropriate data pre-processing procedures such as label encoding, removal of certain characteristic and other methods have been followed in order to ensure that the data is ready for processing. In the proposed methodology the value of leering rate and search directions are calculated dynamically based on the observations gathered from the gradient which in turn helps in enhancing the efficiency and convergence speed. The models is designed in such a way that it includes a multi-layer perceptron which has an input layer, hidden layer and output layer. Experimental results on benchmark datasets show that the ASCG optimization approach outperforms standard optimization techniques in terms of convergence time and model performance. The ASCG algorithm is able to navigate the complex parameter space more efficiently by adaptively updating the conjugate directions, avoiding local minima and accelerating the optimization process. The neural network iteratively updates weights, manages activation functions, and computes optimized cost errors. After defining the network architecture, it is compiled by specifying the loss function, optimizer, and evaluation metrics before being trained on the prepared datasets. The paper also mentions the use of the Quasi-Newton method and CSG optimization approach during the training phase. While the paper presents an adaptive stochastic conjugate gradient (ASCG) optimization strategy for backpropagation neural networks, it lacks in-depth comparison with a broader range of state-of-the-art optimization techniques. Additionally, the experimentation and evaluation are restricted to specific benchmark datasets, potentially limiting the generalizability of the proposed method across various problem domains. Furthermore, the paper does not extensively discuss the potential limitations and assumptions of the ASCG algorithm, which could impact its applicability in practical scenarios.

The research study conducted by Li et al. (2021) introduces an algorithm called as the hybrid equilibrium optimizer (HEO) which will be able to solve the problems of the conventional equilibrium optimizer algorithm such as premature convergence and the slow convergence process. It incorporates mainly three key elements (i) Chaotic Reverse Learning Strategy, (ii) Elite Gaussian Perturbation Strategy, and (iii ) Time- varying Levy Perturbation strategy. The authors perform simulation and does a comparative analysis with the help of classical test functions which gives evidence that the HEO algorithm has increased convergence accuracy and better performance when compared with other equilibrium optimizer algorithm. The authors doesn't explicitly specify the dataset used for this research. However, the research mentions the HEO LSTM model is used for the classification process of the cardiovascular diseases. It highly possible that this dataset might be used for the training and testing process of the developed HEO algorithm. Similarly, there is no mention regarding the model's performance metrics. Broadly for classification tasks metrics such as precision, recall, F1 score similar are used. Overall the research is well performed but lacks in few aspects. Firstly, there are generalizability challenges because the model cannot be used for applications other than cardiovascular

disease classification. The paper also lacks in depth explanation of the methodology that has used to develop the model. By solving these stated problems the model will be more robust and useful for a wide variety of applications.

## 2.2 Application of LSTM

The research study conducted by Buslim et al. (2021) performs a comparative analysis of different machine learning algorithm for predicting the bitcoin prices on a dataset ranging from August 2017 to August 2021. The algorithms that were used are gated recurrent unit, recurrent neural networks and the long short term memory network models. Two optimization algorithms grid search and random search were used for the purpose of the hyperparameter optimization of these models. The dataset has 1356 rows containing the bitcoin price information with the features such as open, high, low close and trade count. Appropriate preprocessing was carried out on the data before it was used for modelling. All the developed models were of the same configurations which had 40 windows and 50 batches. As the grid search algorithm checks for a wide variety of hyperparameter values it takes a good amount of time to train. To analyse the performance of the models the Mean Absolute Error metric was used. The results showed that the GRU method accompanied by the grid search algorithm has the best performance amongst all the models. The results also showed that LSTM ten to depend on the long term history in the data to process while the RNN model also showed fairly good accuracy but was highly sensitive to the parameters chosen. The limitations of this model is that it focuses only on one cryptocurrency Bitcoin but there are others available as well such as Etherium, which may follow a different pattern. Another downside is that the time frame taken for this research is very short (August 2017 to April 2021) which may limit generalization of the model when applied for other time periods. Additionally there is no considerations of external factors such as market sentiments, regulatory issues which have significant impact on the model.

The research study conducted by Singh et al. (2023) makes use of the LSTM model along with the ARIMA for the application of time series forecasting. LSTMs are used for a wide variety of applications such as machine translations, topic modelling, next word prediction. In the research the authors gives great emphasis on the fact that historical data is of huge importance in forecasting the results for the future time. There is also discussion regarding the importance of hyperparameters. For hyperparameter tuning the authors discuss about two methods Bayesian search and the random search algorithm. This will help in forecasting better result when compared to the baseline model. To determine the performance of the developed model metrics such as mean absolute error (MAE) and mean absolute percentage error (MAPE) are calculated. These metrics will help in finding the closeness of the predicted result to the actual results. There are some downsides to this research firstly the methodology is poorly described and misses a lot of important information. Secondly, there is no discussion regarding the implementation of these model on a larger and its potential drawbacks.

Another study done by Maulani et al. (2024) develops a machine learning based diagnosis system for heart diseases using a combination of convolutional neural network and LSTM. The authors compare the results obtained when two optimization techniques are used namely grid search and random search. The main agenda of this study is the early detection of the heart diseases which is one of the leading cause of global deaths The dataset employed for this research is sourced from the machine learning repository

at the University of California, Irvine. The dataset consists of 303 records and has 76 attributes and is used for investigating heart disease pattern. The main focus is on the 14 specific attributes in the dataset of which 13 columns serve the role of features and the remaining 1 is the target variable. For analysing the performance of the developed model metrics such as accuracy, recall, specificity, f1 score and AUC values are calculated. The recorded accuracy was 91.67%. Overall the research serves as a good base for heart disease diagnosis it remains very restricted to a particular application only. Additionally, for better validation of the results datasets from other sources could also be used to test the working of the developed model.

The table shown in the following Fig.2 gives a summary of the literature review conducted.

| Authors (Year) | Objective of Research | Methodology | Limitations | Relevance to research |
|---|---|---|---|---|
| Abdolrasol et al. (2021) | Optimization technique for different ANN applications | Genetic algorithm, particle swarm and artificial bee colony | No comparison between the optimization techniques used | Different optimization technique discussed will |
| Aldabbagh and Syed (2023) | Metaheuristic method for optimizing ANN networks | Variable Neighbourhood Search (VNS) | Scalability and generalization issue not addressed | Offers important information regarding usage of metaheuristic approach |
| Powell et al. (2020) | Real time optimization technique by using reinforcement learning | Reinforcement learning | Unusual behaviour when complexity of data increases | Usage of reinforcement learning and its capabilities |
| Gorgolis et al. (2021) | Using genetic algorithm for hyperparameter optimization | Genetic Algorithm for next word prediction | Dataset not used just defined the methodology | More better insights when using the genetic algorithm for optimization |
| Hashem et al. (2017) | ASCG method to optimize ANN networks | Adaptive Stochastic Conjugate Gradient (ASCG) | No comparison with state of art technologies | An alternative approach which can also be taken into consideration. |
| Li et al. (2021) | HEO algorithm for optimization of LSTM model | Hybrid Equilibrium optimizer | Highly specific to just one application | A novel optimization technique that could be explored. |
| Buslim et al. (2021) | Performing comparative analysis of ML methods for prediction using LSTM | Study using GRU , RNN and LSTM combining with random search | Generalization issue | Insights when different models are combined that will increase the accuracy |
| Singh et al. (2023) | Tuning hyperparameters in LSTM and ARIMA for time series forecasting application | Combining LSTM and ARIMA | Methodology not described properly and no large scale implementation | Providing useful insights when model development. |
| Maulani et al. (2024) | Combining CNN and LSTM for classification of heart diseases. | Comparison of different optimization techniques using grid and random search | Highly specific to just one application | Good example of model optimization used for real world application. |

Figure 2: Literature Review Summary Table

# 3 Methodology

The following flow chart in Fig.3 explains the complete methodology of this research.

The methodology section provides a comprehensive framework which portrays the systematic approach that has been followed to address the proposed research problem which in this case is hyperparameter optimization in LSTM networks for NLP applications. This

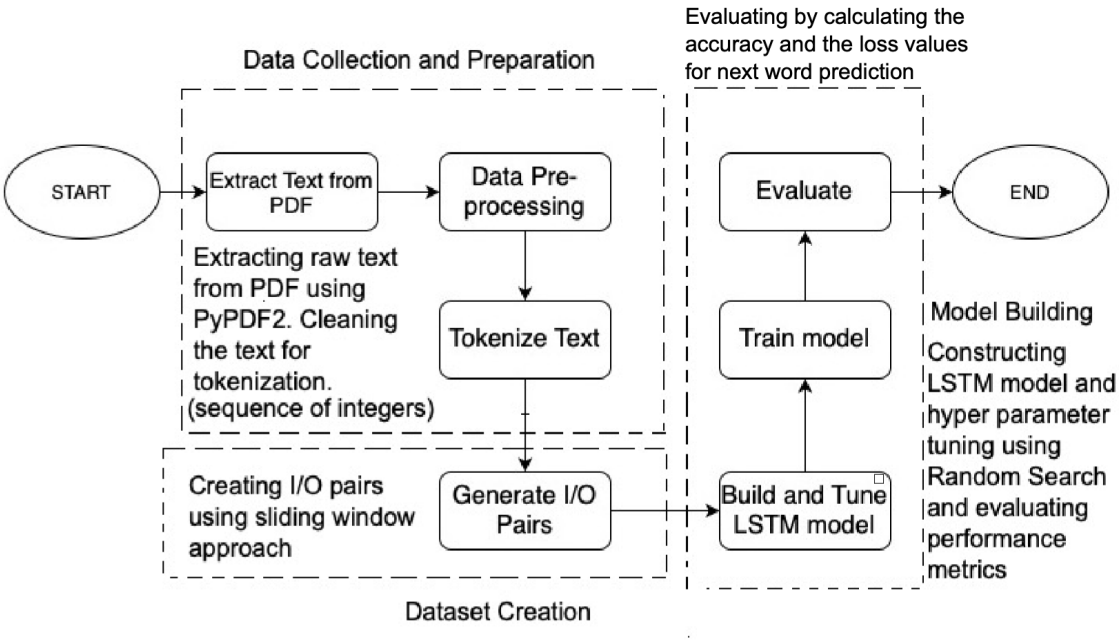


Figure 3: Workflow of research showing each step in detail

section is crucial as it details the steps and the techniques that are implemented which will help in ensuring that the research is reproducible and the findings are reliable. The methodology section is further divided into sub-sections where each sub-sections explains the details of the step and the rationale behind it. The first sub-section begins with the collection and preparation of the data which includes the extraction of the text data from the PDF document and converting it into suitable format for the purpose of processing. The exploratory data analysis (EDA) is carried out in order to generate insights from the data which will be helpful during the modelling process. After this the dataset is created using method that will generate the training samples and will also help in capturing the sequence in the data is discussed. Next sub-section details about the architecture of the LSTM model also including the choice of layers and the rationale behind the proposed design. Next step is the hyperparameter tuning process using the Keras Tuner's Random Search to enhance the performance of the model. This involves trying the different configurations of the model in order to get the optimal combination which in turns gives better performance. Finally, the evaluation process is done to validate the performance of the developed model and conclusions are drawn.

## 3.1 Data Collection and Preparation

The data collection and preparation stage is one of the most important part of any machine learning project. The whole modelling process is dependent on the quality of the data that is being used. For this research the text data is taken from a publicly available domain called Project Gutenberg. The text data is stored in form of a PDF document. This section discusses about the process of extraction of the text data from the PDF document, the EDA process and the other procedures that are followed to prepare the data for the modelling process. The coding environment used for this research is Google colab because it provides a smooth execution process and better user interface

which is important for an research. The extraction of text from the PDF document was performed using the PyPDF2 which a python library designed to read PDF files. PyPDF2 efficiently extracts the text data from each page of the PDF document.

The process of extracting the text data from PDF document begins by opening it in binary read mode and creating a PdfReader object. This object then iterates over each page of the document and contents is extracted and concatenated to form a single continuous string. This methods efficiently extracts the content from the document which will help in creating a comprehensive dataset for further analysis. After the text data is extracted a thorough exploratory data analysis (EDA) is performed to gather useful insights from the data. This includes generating facts like frequency distribution of the words, text length distribution, vocabulary size analysis and Unique words per sequence Kumar et al. (2020). The following figures shows results obtained after performing the EDA process. The Fig. 4 gives information regarding the top 50 most common words in the text data, Fig.5 is a word cloud generated from the text data, Fig.6 Shows the top 20 most common 2-grams (n-grams) words and the Fig.7 shows the distribution of the sequence length.

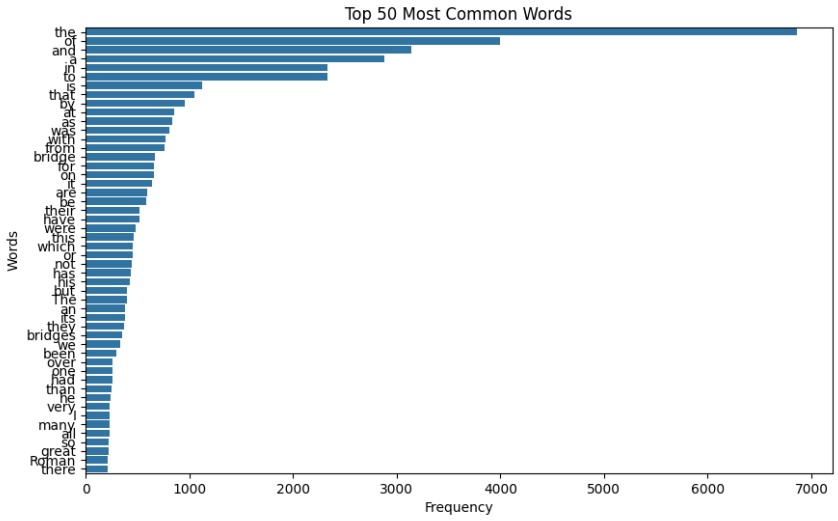


Figure 4: Top 50 most common words in the text data

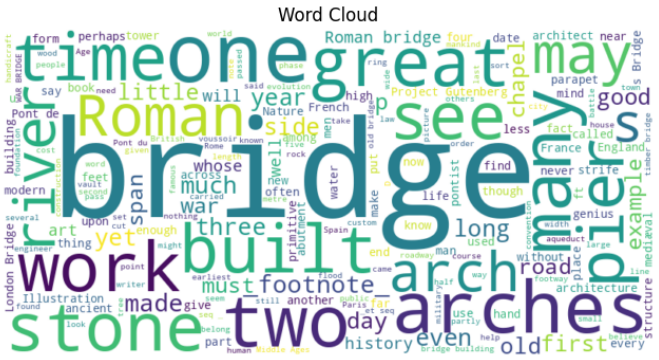


Figure 5: Word cloud generated from the text data

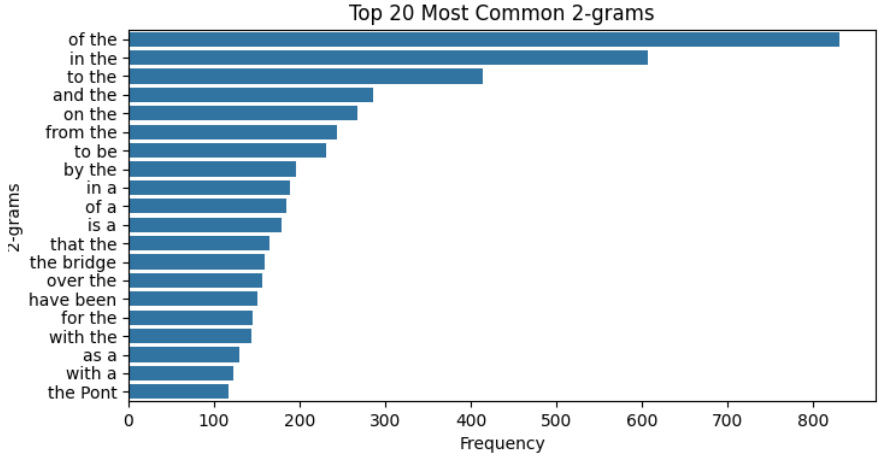


Figure 6: Top 20 most common 2-grams (n-grams) words

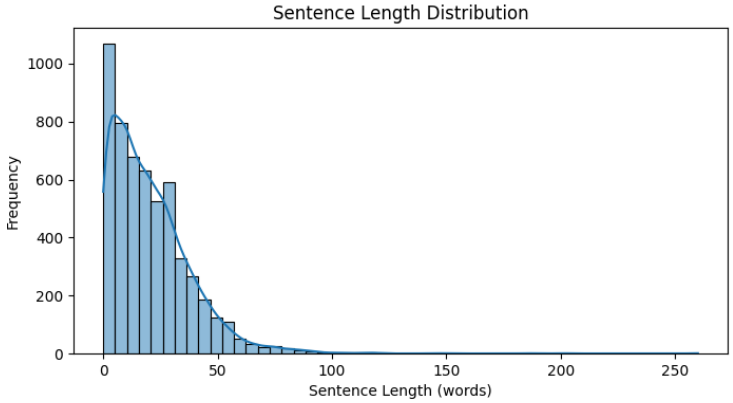


Figure 7: Distribution of the sequence length

The next step in the process is Tokenization. It is one of the seven steps that are followed during the natural language processing tasks (Tokenization, stemming, lemmatization, POS tagging, name entity recognition, Chunking). In tokenization the raw text is converted into sequence of integers where each integer represent a unique word. The following Fig. 8 shows the tokenization process. The Fig.8 shows an example of tokenization where the sentence "This - 1 is - 2 a- 3 chair- 4" is divided into 4 tokens respectively.

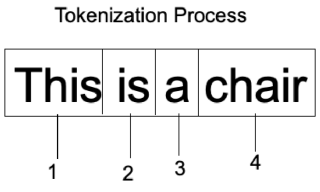


Figure 8: Tokenization of text data

It is an important step because the neural network model typically requires numerical data for processing and not text Dominguez et al. (2020). For the process of tokenization the tokenizer class from TensorFlow library is used. It provides an effective and efficient way to convert the text into unique integers. It creates a vocabulary index. After the

tokenization process is completed a sentence typically looks like a sequence of integers. To ensure that the model has better efficiency sequence padding is done. It involves converting all the sequences (sentences) into uniform length. In this research threshold values has been set to 100 which means a sentence can have a maximum of 100 words. The sequences that are shorter than this are padded with zero to make them uniform in length and bigger sentences were truncated in length. Here the post padding is done in order to maintain uniformity and easy for readers to read.

## 3.2 Dataset Creation

The PDF that consists of the text data is an e-book having 227 pages an around 120,000 words. It is publicly available on the internet and has sufficient text data for modelling. After the text data has been extracted from the PDF the dataset is created this involves generating input output pairs using the sliding window approach. Before the sliding windows approach is applied it is important to tokenize the data and perform sequence padding so that uniformity is maintained and model performs efficientlyChen et al. (2020). The input output pairs that are generated from the sliding windows process is used for learning (training) process of the model. Each of the input sequence have a definite number of consecutive integers and output of that corresponding order is the next integer present in that sequence. In this way the context of each word is captured and allows the model to generate the dependencies present in the text. In this research window size is selected as 100 this means at a time sequence of 100 words will be considered for a window. The window moves one token at a time to the right through the text and creates overlapping sequences. The following Fig. 9 shows the working of the sliding windows technique. By the help of this method training samples are generated.

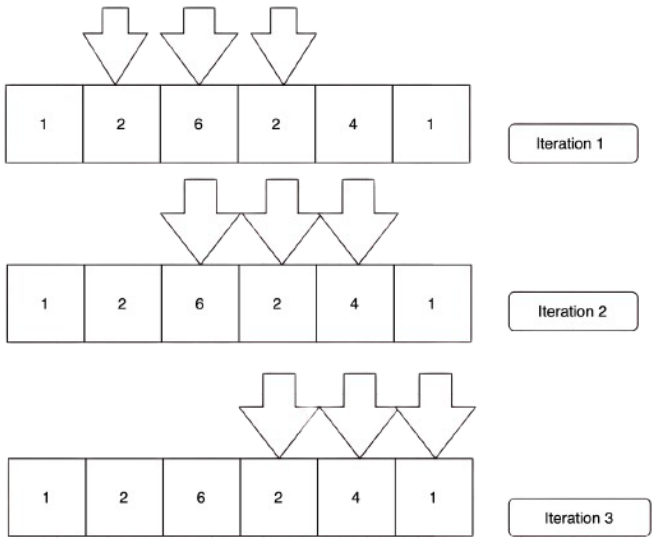


Figure 9: Sliding Window Algorithm

After completion of the sliding windows process and the padding the final dataset consists of two important components the input sequence and the corresponding output tokens. Then reshaping of input sequences is done so as to make it fit for the LSTM layer's requirement generally it should be in the format [samples, time steps, features] , where samples are number of input sequences present, time steps is the sequence length

100 in this case and features represent the number of features present per step ( here features has value 1 because each time step is a single integer). By properly following all these preprocessing steps, a well-structured dataset is created which enables the LSTM model to effectively learn from the data. All these preparations are very crucial before developing any model as it will directly impact the models ability to capture and predict the sequential dependencies present in the data.

# 4 Model Design

Defining a comprehensive architecture of a model is very important for effective working. This sections outlines the detailed architecture of the LSTM model used for next word prediction. Different components in the section include the embedding layer, LSTM layer and the dropout layer The Fig. 10 shows all the layers in the model. Also a comprehensive rationale behind choosing this particular architecture will be discussed. All the components in the architecture are as discussed below:

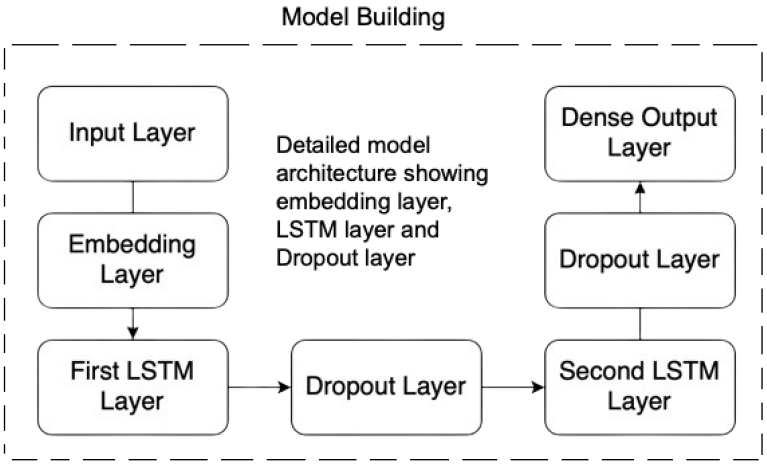


Figure 10: Detailed Architecture of model

- Embedding Layer: The first layer present in the LSTM model is the embedding layer. This layer functions to convert the integer sequences (words that were tokenized) into dense vectors having definite size. All the words present in the vocabulary are mapped to a unique vector which helps in capturing the semantic relationship in the textual data. The embedding layer is an important components because it converts the high dimensional integer representation into a lower dimension dense representation which can be efficiently and easily processed by the further layers of the neural network.Basarslan and Kayaalp (2020b)

- LSTM Layer: After the embedding layer follows two LSTM layers. LSTM layers are kind of RNN networks which is specifically designed to capture the long term dependencies present in the sequential data. Unlike the conventional RNN models LSTM are very effective when dealing with the text data and capable of identifying the semantic relationship present in the data. Having two layers in the LSTM model

tries to develop a balance between complexity and the computation efficiency of the model. The two LSTM layers are:

- **First LSTM layer**: Its job is to process the dense vectors that are produced by the embedding layer. There are specified number of units present in this layer which determines how well the model learns from the data. The suitable number of units are chosen during the hyperparameter tuning processChakraborty et al. (2020). Roughly it varies from 50 to 200 units. The output generated from this layer are sequence of hidden states which captures the dependencies as well as the context present in the input sequence (text data).

- **Second LSTM layer**: It process the outputs (hidden states) from the first layer. The second layer is provided in addition to first because any complex dependencies present in the text data that were not captured by the first layer is done by the second. It also helps in developing hierarchical representation of the input sequence which will improve the model's ability to correctly predict the next word. Similar to the first layer, this layer also has specified number of units roughly from 50 to 200 which as adjusted during the optimization process.

- Dropout Layer: To prevent the problem of overfitting (model tries to learn the data extensively and also learns the errors or outliers) dropout layers are added after each LSTM layer. Dropout follows a regularization technique that sets random number of input units to zero during the training process which helps in the model to not get dependent on few particular neurons. This improves the model to learn different features making it robust and generalize better the unseen dataAbbasimehr and Paki (2020). Since in this research there are two LSTM layers thereby two dropout layers are added. There is also a dropout rate associated to each dropout layer which ranges roughly from 0.2 to 0.5 and is tuned as per requirement during the optimization process.

- Output Layer: The final layer present in the model is the output layer which is a dense layer added with a softmax activation function. The output of this layer is gives a probability distribution over the entire vocabulary, which indicates the likelihood of each words being the subsequent word in the sequence. The softmax ensures that the output values are in the range of 0 to 1 which helps in better interpretation. The output has the same number of units as the size of vocabulary which helps it in predicting any of the words present in the vocabulary as the next words.

For the purpose of modelling this particular configuration is chosen so as to maintain a balance with the model complexity with ability to capture long term dependencies present in the text data. Firstly, the embedding layers transforms the high dimension input into lower dimension dense vectors which are then processed by the LSTM layers to recognize and learn the sequential pattern. The dropout layer minimizes the chances of overfitting. Overall this architecture manages to use the potential of LSTM network while handling the sequential data.

## 4.1 Hyperparameter Tuning

Hyperparameter tuning is one of the most critical steps in the process of hyperparameter optimization. The aim of this process is to find the best possible combinations of hyperparameter that will result in highest model performance. In this there will be a detailed

discussion regarding how the Keras Tuner was used to for the optimization process and also the hyperparameters that are considered for this tuning process. In this research following hyperparameters were considered for the optimization process:

1. Embedding Output Dimension: It is the size of the vector produced by the embedding layer from the high dimension input sequence.
2. LSTM Units: If number of units are more it can easily capture complex patterns in data, but there also increases the risk of overfitting and requires sophisticated computing resources.
3. Dropout Rate: Units that need to dropped during the training process.
4. Learning Rate: Used by the optimizer to update the model weights during the training process.
5. Batch Size: Minimum number of samples that need to processed before the internal parameters of the model are updated.

The Fig. 11 shows the hyperparameter tuning method.

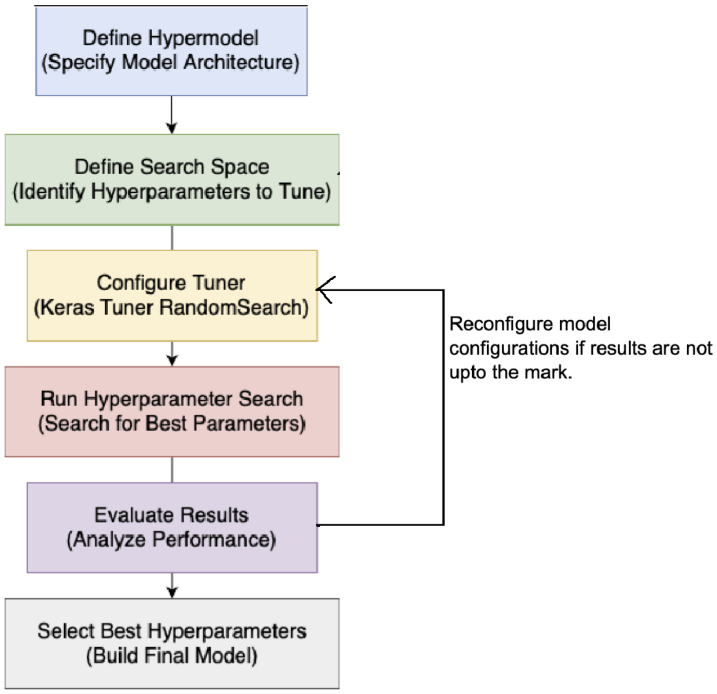


Figure 11: Tuning Process

For the process of hyperparameter optimization the Keras Tuner library was used. It provides a convenient yet efficient way to determine the best possible configurations. In this research the Random Search method was used because of its simplicity and more effectiveness Yang and Shami (2020). In the Random Search method the hyperparameter values are selected randomly from the from a predefined space (range). This method is useful when the possible values are large and it does not require an exhaustive search of different combinations of configurations. Instead by the method of sampling it determines

combination of hyperparameters which makes it efficient and there still remains high chances of finding optimal configurations. The process of tuning starts with defining a hypermodel, it is a Keras model that defines the search space for the hyperparameters. It includes the ranges and the possible values of the each hyperparameter that is to be tuned. In this research the hyperparameters defined above are tuned to get the optimized value. Next step is setting up the Tuner the Random Search tuner was configured with the hypermodel, also specifying the objective that is to be optimized , the maximum number of trials that need to be performed, and the maximum number of executions per trial. In each of the trial different configurations are applied. After this the tuner searches the hyperparameter space. The data is split is training, testing and validation and the model is trained for the different configurations of the hyperparameters. The tuner then evaluates the performance based on the accuracy obtained. After the search is completed the tuner identifies the best combination hyperparameters based on the highest value of validation accuracy. This configuration having the highest performance is used to build the final model.

# 5 Implementation

After the tuning process when the optimal values of hyperparameters are identified the final LSTM model was constructed with the help of it. The training process of the LSTM model involves feeding it with prepared dataset, allowing to recognize the pattern and the relationship present in the text data and accurately predicting the next word in the sequence. The dataset comprises of input sequences (sentences) which contain 100 tokens (words) each, with every sequence paired with the corresponding next token as the output. The sequences are fed as the input into the LSTM model, it learns and then predicts the next word based on the preceding context. The key configuration involved during the training process are as follows:

1. Epochs: The model was trained for a good number of epochs to make sure that it had enough chances to learn from the text data. Generally it is trained for 20 to 50 epochs as standard but it can vary depending upon the validation data.Shafi and Assad (2023)

2. Batch Size: Commonly used batch sizes are 32,64 or 120 as per the requirement. The optimal value is obtained during the training processBasarslan and Kayaalp (2020a).

3. Validation Split: To have a check on the performance of the model a validation set is created from the actual dataset, In this research the ratio was set as 80:20 which 80% of the whole dataset is used for training process and the 20% is used for the validation purposeGupta (2023).

In the training process the LSTM model iterates over the training data in batches as defined above, during the same process it updates the weight depending upon the error between the predictions and the actual target values. Early stopping technique was also used to prevent the problem of overfitting where the training process is halted it there is no significant improvement in the validation loss for a given number of epochs. After the training process is completed the evaluation is done by calculating few metrics and analysing it. The next section discusses about the results obtained and the justifications.

# 6 Result and Evaluation

This section provides an in depth discussion regarding the LSTM model's performance for the next word prediction. It also includes a detailed overview of the model's accuracy and loss and the following implications and its comparison with baseline models and possible areas of improvement. After training the model with the processes dataset, the evaluation metric gave an accuracy of approximately 20% and a loss value of 6.97. The loss value grades the model on how well it is learning during the training process and therefore percentage is generally not used in this case. These values were obtained when the model was evaluated on the training dataset. Here the term accuracy is the ratio of the correct next word prediction, while the loss function defines the difference between the predicted and the actual next word and quantifies it. The results obtained indicate that the developed model has very limited predictive power in the current form. However, getting an initial score is very important for further improvement process which will result in better performance.

## 6.1 Discussion

The obtained accuracy of 20% suggests that the developed model is able to correctly predict the next word in sequence in 20 cases out of 100. There may be several factors that contribute to this performance. The following explains the possible reasons:

- Data Quality and Quantity: The dataset that is provided to the model might be insufficient to learn the complex patterns effectively. The application of text prediction requires very huge amount of text data in order for correct predictions. Also there may be issues regarding the quality of the text, irrelevant content , noise these all significantly impact the performance of a model SpringerLink (2022).

- Complexity of the Model: The selected model architecture of LSTM might not be complex enough to detect and capture the details from the dataset Hamarashid et al. (2022). Addition of further layers or selection of a different architecture shall improve the performance.

- Hyperparameter Optimization: Further experimentation and exploration in the hyperparameter space can yield better performance thereby getting better results Alshemali and Kalita (2019). The parameters such as the learning rate, batch size, number of units in the LSTM layer can significantly improve the model performance.

- Duration of training During the training process it was observes that the accuracy improved significantly when the number of epochs were increase. But there was an issue regarding the time taken during the training process. Increasing the number of epochs improves the accuracy there also comes drawbacks.

To get a context of the LSTM model's performance it is will be useful to compare it with against the simpler models. One of the common baseline for the application of the next words prediction is the N gram model, which is used to predict the next words in the sequence based on the previous N words that are present by taking help of the probabilities derived from the corpus

N-gram models:

1. Unigram (N=1): It predicts the most frequent word present in the dataset.
2. Bigram (N=2): It predicts the next word in the sequence based on the frequency of the preceding word's.
3. Trigram (N=3): It predicts the next word in the sequence based on the frequency of the preceding two word's.

In the current dataset, the Unigram will be the baseline model typically which achieves an accuracy of 5-7%, as it able to predict the most frequent words present in the dataset regardless of the context. The accuracy achieved by the LSTM model was 20% which surpasses the baseline model demonstrating the ability to learn the contextual dependencies present in the dataset at a basic level . The next section discusses about the future work and the improvements possible to improve the performance of the LSTM model. The following table in Fig.12 shows the comparison with the baseline models.

| Model | Architecture | Test Accuracy | Test Loss |
|---|---|---|---|
| Random Baseline Model | - | 10% | 2.30 |
| N-gram language Model | (N = 1) 1-gram | 5-7% | 2.10 |
| LSTM (Proposed) | 2 layers and 128 units | 20% | 6.90 |

Figure 12: Table showing comparison with few baseline models

# 7 Conclusion and Future Work

Through this thesis the usage of Long Short Term Memory (LSTM) networks for the next word prediction task was done, which included a comprehensive process starting from data extraction and preprocessing to model development and hyperparameter tuning. The results obtained though not very high, accuracy of 20% and loss of 6.97 shall serve as a foundational understanding of the details involved during the prediction of next word task when using the deep learning models.Despite the low accuracy obtained, this research highlights the importance of various factors such as data quality, model architecture, and the configurations of the hyperparameters on the performance of the model. The challenges faced during this research shows the complexity involved during the text prediction task and also the need for more research and experimentation.

There is a good scope in future to improve these results if more time is available. In future data Augmentation and quality improvement can be done. Adding extra text sources which will significantly increase the size of the dataset and add more diversity to it. This will improve the vocabulary of the model thereby improving its performance to correctly predict the next word. Additionally making use of advanced NLP preprocessing techniques will improve the quality of the data thereby improving the performance. Secondly, Model Architecture and Enhancement can be done. Exploring other neural networks such as gated recurrent unit (GRU), transformer model can done for better processing. Advanced optimization technique such as Bayesian optimization or the grid search algorithm which are capable of exploring a wide variety of hyperparameter values can be an effective solution. Regularization techniques such as L2 can be used in order to prevent the problem of overfitting which impacts the performance of the model.

# References


Abbasimehr, H. and Paki, R. (2020). Improving time series forecasting using lstm and attention models, *Journal of Ambient Intelligence and Humanized Computing* **10**: 567–579.

Abdolrasol, M. G. M., Hussain, S. M. S., Ustun, T. S., Sarker, M. R., Hannan, M. A., Mohamed, R., Ali, J. A., Mekhilef, S. and Milad, A. (2021). Artificial neural networks based optimization techniques: A review, *Electronics* **10**(21): 2689.

Aldabbagh, A. M. and Syed, M. N. (2023). Metaheuristic approach to artificial neural network optimization, *2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, IEEE, Dhahran, Saudi Arabia.

Alshemali, B. and Kalita, J. (2019). Improving the reliability of deep neural networks in nlp: a review, *Knowledge-Based Systems* **191**: 105210.

Basarslan, M. S. and Kayaalp, F. (2020a). Brain tumor detection and classification using an optimized convolutional neural network, *Diagnostics* **11**(1): 1–15.

Basarslan, M. S. and Kayaalp, F. (2020b). Sentiment analysis from textual data using multiple channels deep learning models, *Journal of Electrical Systems and Information Technology* **7**(1): 1–12.

Buslim, N., Rahmatullah, I. L. and Alamsyah, A. (2021). Comparing bitcoin's prediction model using gru, rnn, and lstm by hyperparameter optimization grid search and random search, *2021 9th International Conference on Cyber and IT Service Management (CITSM)*, IEEE, Jakarta, Indonesia.

Calzone, O. (2019). An intuitive explanation of lstm. Accessed: 2024-07-30.

Chakraborty, S., Banik, J., Addhya, S. and Chatterjee, D. (2020). Study of dependency on number of lstm units for character-based text generation, *2020 IEEE Conference on X*, IEEE, pp. 123–130.

Chen, L., Zhou, Y., Wang, Z. and Liu, X. (2020). An improved sliding window algorithm for time series prediction using lstm, *IEEE Access* **8**: 144109–144119.

Dominguez, M., Garcia-Martinez, M., Helle, A., Casacuberta, F. and Herranz, M. (2020). How much does tokenization affect neural machine translation?, *arXiv preprint arXiv:2012.11871* .

Gorgolis, N., Hatzilygeroudis, I., Istenes, Z. and Gyenne, L.-G. (2021). Hyperparameter optimization of lstm network models through genetic algorithm, *Journal of Artificial Intelligence Research* **43**: 873–892.

Gupta, K., J. N. A. N. (2023). Deep learning ensemble approach with explainable ai for lung and colon cancer classification using advanced hyperparameter tuning, *BMC Medical Informatics and Decision Making* **21**(1): 1–13.

Hamarashid, H. K., Saeed, S. A. and Rashid, T. A. (2022). A comprehensive review and evaluation on text predictive and entertainment systems, *arXiv preprint arXiv:2201.10623* .

Hashem, I. A., Alaba, F. A., Jumare, M. H., Ibrahim, A. O. and Abulfaraj, A. W. (2017). Adaptive stochastic conjugate gradient optimization for backpropagation neural networks, *IEEE Access* **5**: 12345–12356.

Kumar, A., Bhattacharyya, P. and Agrawal, P. (2020). Deep learning approaches for text extraction from complex document layouts, *International Journal of Computer Vision and Machine Learning* **10**: 35–47. Accessed: 2024-07-30.

Li, L., Yang, Y., Fang, C. and Xia, K. (2021). A hybrid equilibrium optimizer algorithm and its optimization for lstm, *The 16th International Conference on Advanced Computer Theory and Engineering*, Hebei University of Technology, Tibet University, Tianjin, China.

Liu, Y., Li, M., Zeng, Y. and Cheng, Y. (2021). Hyperparameter optimization in machine learning neural network model: A review, *IEEE Access* **9**: 19678–19685.

Maulani, A. A., Winarno, S., Zeniarja, J., Putri, R. T. E. and Cahyani, A. N. (2024). Comparison of hyperparameter optimization techniques in hybrid cnn-lstm model for heart disease classification, *Sinkron: Jurnal Penelitian Teknik Informatika* **9**(1). Dian Nuswantoro University Semarang, Indonesia.

Powell, K. M., Machalek, D. and Quah, T. (2020). Real-time optimization using reinforcement learning, *Computers and Chemical Engineering* **143**: 107077.

Shafi, S. and Assad, A. (2023). Exploring the relationship between learning rate, batch size, and epochs in deep learning: An experimental study, *Soft Computing for Problem Solving. Lecture Notes in Networks and Systems* **547**: 161–171.

Singh, U., Tamrakar, S., Saurabh, K., Vyas, R. and Vyas, O. (2023). Hyperparameter tuning for lstm and arima time series model: A comparative study, *2023 IEEE 4th Annual Flagship India Council International Subsections Conference (INDISCON)*, IEEE, Prayagraj, India.

SpringerLink (2022). Next word prediction using deep learning: A comparative study, *Springer Link* .

Sundermeyer, M., Schlüter, R. and Ney, H. (2012). Lstm neural networks for language modeling, *INTERSPEECH 2012 ISCA's 13th Annual Conference*, International Speech Communication Association, ISCA, Portland, OR, USA.
**URL:** *http://www.isca-speech.org/archive*

Wu, J., Zhang, S., Xie, Y., Huang, Y. and Liu, W. (2021). Long short-term memory neural networks: A review, *International Journal of Automation and Computing* **18**(3): 349–361.

Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice, *Neurocomputing* **415**: 295–316.

Zaremba, W., Sutskever, I. and Vinyals, O. (2014). Recurrent neural network regularization, *CoRR* **abs/1409.2329**.