**VARIATIONS ON THE LSTM DEEP NEURAL NETWORK ARCHITECTURE FOR CLASSIFICATION OF EEG SIGNALS FOR BCI APPLICATIONS**

***a project report submitted by***

**PRASHANT SRINIVASAN SARKAR**

**(REG. NO: URK20AI1020)**

***in partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

***in***

**PROGRAMME NAME**

***under the supervision of***

**DR E. GRACE MARY KANAGA**

****

**DIVISION OF DATA SCIENCE AND CYBER SECURITY SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY**

**KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES** (Declared as Deemed-to-be-University under Sec-3 of the UGC Act, 1956) **Karunya Nagar, Coimbatore - 641 114, India.**

**APRIL 2024**

**BONAFIDE CERTIFICATE**

Certified that this project report **“VARIATIONS ON THE LSTM DEEP NEURAL NETWORK ARCHITECTURE FOR CLASSIFICATION OF EEG SIGNALS FOR BCI APPLICATIONS”** is the bonafide work of “PRASHANT SRINIVASAN SARKAR (REG. NO: URK20AI1020)**”** who carried out the project work under my supervision.

|  |
| --- |
| **SIGNATURE SIGNATURE (SUPERVISOR)** |
| **Dr. E. Grace Mary Kanaga Dr. E. Grace Mary Kanaga** |
| **Head of the Division Head of the Division/Professor** |
| **Division of Data Science and Cyber Security Division of Data Science and Cyber Security** |

Submitted for the Project Viva Voce held on……………………….

**Examiner**

**Project Completion Certificate**

**ACKNOWLEDGEMENT**

First and foremost, I praise and thank **ALMIGHTY GOD** for giving me the will power and confidence to carry out my project.

I am grateful to our beloved founders Late **Dr. D.G.S. Dhinakaran**, **C.A.I.I.B, Ph.D.,** and **Dr. Paul Dhinakaran**, **M.B.A., Ph.D.,** for their love and always remembering us in their prayers.

I extend my thanks to **Dr. G. Prince Arulraj, Ph.D.,** Vice-chancellor, **Dr. E. J. James, Ph.D.,** and **Dr. Ridling Margaret Waller, Ph.D.,** Pro-Vice-Chancellor (s) and **Dr. R. Elijah Blessing, Ph.D.,** Registrar for giving me the opportunity to carry out this project.

I am thankful to **Dr. Ciza Thomas, Ph.D.,** Dean (School of Computer Science and Technology) for her support and encouragement.

I would like to place my heart-felt thanks and gratitude to my guide **Dr. E. Grace Mary Kanaga, Ph.D.,** HOD, Division of Data Science and Cyber Security for her encouragement and guidance

I also thank all the staff members of the Division for extending their helping hands to make this project a success.

I would also like to thank all my friends and my parents who have prayed and helped me during the project work.

**ABSTRACT**

Deep learning is a branch of artificial intelligence most closely linked with the functioning of an actual human brain. Deep neural networks perform millions of simple computations that, when combined, allow for incredibly complex problem-solving capability. With recent breakthroughs in computational technology, neural networks are now larger than ever, capable of performing complex tasks that were hitherto believed to be science fiction. As modern technology grows closer to being capable of mimicking the human brain, researchers around the globe ponder the question: can this technology understand the brain? This experimental analysis of LSTM Deep Neural Networks proposes multiple variations of the traditional LSTM architecture as an optimized method for the classification of electroencephalogram (EEG) brain signals. It explores 5 variations of the LSTM architecture: Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, Stacked Bidirectional LSTM and LSTM with Attention Mechanism. This experimental analysis proposes a hybrid LSTM with Attention mechanism as a potential solution to create efficient, accurate and lightweight brain computer interfaces. In this experimental analysis, the proposed architecture achieved an F1 score of 94% for the classification of cognitive stimuli-based EEG signal data. These architectures show promise in the realm of brain computer interfacing as potential solutions for the future of human-computer interaction.

**Table of Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chap. No** | **Title** | | | **Page No.** |
|  | **BONAFIDE CERTIFICATE** | | | ii |
|  | **ACKNOWLEDGEMENT** | | | iv |
|  | **ABSTRACT** | | | v |
| **i** | **List of Figures** | | | viii |
| **ii** | **List of Tables** | | | ix |
| **iii** | **List of Symbols and Abbreviations** | | | x |
| **1** | **Introduction** | | | 1 |
|  | 1.1 | Brain Computer Interface | | 1 |
|  | 1.2 | Deep Learning | | 2 |
|  |  | 1.2.1 | Machine Learning | 2 |
|  |  | 1.2.2 | Limitations of Machine Learning | 3 |
|  |  | 1.2.3 | How Deep Learning is Changing Computer Science | 4 |
|  |  | 1.2.4 | Deep Learning for BCI Technology | 5 |
|  | 1.3 | Objectives of the Project | | 6 |
|  | 1.4 | Problem Statement | | 6 |
|  | 1.5 | Chapter-wise Summary | | 7 |
| **2** | **Literature Survey** | | | 9 |
|  | 2.1 | The Status of Deep Learning for BCI Applications | | 9 |
|  | 2.2 | Approaches to BCI Applications with Deep Learning | | 10 |
|  | 2.3 | Justification for LSTM as a Solution for BCI Applications | | 11 |
|  | 2.4 | Performance Metrics for Deep Neural Network Evaluation | | 12 |
|  | 2.5 | Need for BCI Technology | | 14 |
|  | 2.6 | Applications of BCI Technology | | 14 |
|  | 2.7 | Summary of Literature Survey | | 16 |
| **3** | **VARIATIONS ON THE LSTM DEEP NEURAL NETWORK ARCHITECTURE FOR CLASSIFICATION OF EEG SIGNALS FOR BCI APPLICATIONS** | | | 17 |
|  | 3.1 | Data Collection | | 17 |
|  | 3.2 | LSTM | | 19 |
|  | 3.3 | Variations of the LSTM Architecture | | 21 |
|  | 3.4 | Selection of 4-Layer stacked LSTM model | | 23 |
| **4** | **Implementation** | | | 2 |
|  | 4.1 | Algorithm | | 25 |
|  | 4.2 | Implementation Details | | 26 |
| **5** | **Results and Discussion** | | | 32 |
| **6** | **Conclusions** | | | 39 |
|  | 6.1 | Drawbacks in Methodology | | 40 |
|  | 6.2 | Future Scope | | 40 |
| **7** | **References** | | | 42 |
| **8** | **Appendices** | | | 46 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| No. | Title | Page |
| Fig 1. | Cognitive Stimuli Example | Pg. 18 |
| Fig 2. | Positions of EEG electrodes and corresponding labels | Pg. 19 |
| Fig 3. | LSTM Architecture | Pg. 20 |
| Fig 4. | Vanilla LSTM F1-Score vs Number of Parameters | Pg. 32 |
| Fig 5. | Stacked LSTM F1-Score vs Number of Parameters | Pg. 33 |
| Fig 6. | 1-Layer Bidirectional LSTM F1-Score vs Number of Parameters | Pg. 34 |
| Fig 7. | 2-Layer Bidirectional LSTM F1-Score vs Number of Parameters | Pg. 35 |
| Fig 8. | Vanilla LSTM with Attention F1-Score vs Number of Parameters | Pg. 36 |
| Fig 9. | Stacked LSTM with Attention F1-Score vs Number of Parameters | Pg. 37 |
| Fig 10. | F1 Score vs Number of Parameters for Best Architecture of All Models | Pg. 38 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| No. | Title | Page |
| Table 1. | Confusion Matrix | Pg. 12 |
| Table 2. | Performance Metrics for Classification Models | Pg. 13 |
| Table 3. | Sample Dataset for Cognitive Stimuli-based EEG Signal Data using GUSB AMP | Pg. 18 |
| Table 4. | LSTM Layers vs Accuracy and Number of Parameters | Pg. 24 |
| Table 5. | Vanilla LSTM Performance Metrics | Pg. 32 |
| Table 6. | Stacked LSTM Performance Metrics | Pg. 33 |
| Table 7. | 1-Layer Bidirectional LSTM Performance Metrics | Pg. 34 |
| Table 8. | 2-layer Bidirectional LSTM Performance Metrics | Pg. 35 |
| Table 9. | Vanilla LSTM with Attention Mechanism Performance Metrics | Pg. 36 |
| Table 10. | Stacked LSTM with Attention Mechanism Performance Metrics | Pg. 37 |

**LIST OF SYMBOLS AND ABBREVIATIONS**

|  |  |
| --- | --- |
| EEG | Electroencephalogram |
| BCI | Brain-Computer Interface |
| LSTM | Long Short-Term Memory |
| ML | Machine Learning |
| AI | Artificial Intelligence |
| CSV | Comma Separated Values |
| CPU | Central Processing Unit |
| GPU | Graphics Processing Unit |
| TPU | Tensor Processing Unit |
| σ | Sigmoid Function |
| ⊕ | XOR Gate |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Brain Computer Interface**

A brain computer interface (BCI) is a technological apparatus that facilitates communication between a user and a computer device via brain signals. This is most commonly achieved by means of an Electroencephalogram (EEG). EEG signals are electrical impulses produced by brain activity. These are often used by medical practitioners to monitor a patient’s brain activity and function. Since these signals correlate to the person’s cerebral functions, they can be used to interpret thought patterns. The goal of numerous researchers worldwide is to successfully implement a BCI capable of accurately and reliably interpreting thought patterns to facilitate new means of interacting with computer devices.

The primary challenge in implementing a BCI is the uniqueness of each individual human brain. The brain is a highly complex network of cells which communicate through electrical impulses. The unique wiring of a person’s neurons means that any method of interpreting brain signals must be unique to that individual’s brain. This obstacle proved to be a major roadblock that made the implementation of a BCI logistically infeasible for decades.

However, recent developments in deep learning have allowed for the identification of complex patterns and relationships by using deep neural networks. These complex mathematical architectures are capable of identifying and extracting features from data that a human being would be unable to identify. This allows us to leverage large volumes of data to find complex correlations and use them to classify and interpret data at a rapid rate. Deep learning can be applied to EEG data to identify the unique patterns of an individual’s brain and identify their thought patterns in order to facilitate communication between the person and a computer via thought.

* 1. **Deep Learning**

Deep learning is a subset of artificial intelligence which has seen rapid growth and advancement in recent years. It deals with the training of deep neural networks. A deep neural network is a collection of artificial neurons which pass signals to each other by ways of neuron activation. Each neuron performs a series of mathematical operations on the input it receives before passing the result as output if the result exceeds a threshold for neuron activation.

Deep learning evolved from machine learning, a larger subset of artificial intelligence. Machine learning, and therefore deep learning, focuses on training a model to understand data and make its own inferences through a learning process. Data is fed to the model along with the desired output. The model then learns from the data and adjusts itself in a process called ‘fitting’. Once a model is fit to the data, it would ideally be able to produce the desired output when given the respective input to a high degree of accuracy. Assuming the data is of high quality and is a good representation of the real-world scenario from which it was derived, this model would then be capable of generalizing and solving a real-world problem.

**1.2.1 Machine Learning**

Machine learning applies numerous mathematical techniques to create an abstract model to provide a solution to the problem. Techniques like linear regression allow us to create a linear model that reflects the general trend of the data in order to make fairly accurate predictions and forecasts. A classification model such as an SVM (Support Vector Machine) would draw a hyperplane to separate data points on a graph and classify any new data points based on their position relative to the hyperplane. Both these examples function on the basis of a linear model of the data. Linear regression assumes a linear correlation between the data and attempts to fit a line to represent this correlation. An SVM assumes that the data is linearly separable and attempts to fit the hyperplane to separate the data along this axis. This process involves continuous trial and error while making adjustments based on how the model performs in relation to the target data. Over the course of the training process, the model will adjust itself based on the feedback from the target data until it is told to stop. The end result, ideally, will be a model that is capable of producing the desired output from a given input and that is capable of generalizing the training data to unseen data from the same data source.

More complex machine learning models can learn even more complex relationships. Models such as decision trees can take more abstract features and find relationships between the presence of a feature and the output that can be expected. Bayesian networks apply the concept of Baye’s theorem to find a probabilistic relationship between events and their outcome. These techniques within machine learning have proved to be powerful tools that help solve complex problems and provide solutions to generalize large volumes of data.

**1.2.2 Limitations of Machine Learning**

However, in spite of the amazing capabilities of these models, they are still limited in their abilities. The largest limitation to machine learning is that the features of the data must be extracted before the model can analyse them. This becomes incredibly challenging when dealing with massive volumes of data or when we are unaware of what features to look for. In the context of a BCI, it would take thousands of data points for a model to be able to successfully understand the data, and these features are incredibly tricky to extract manually. This is where deep learning becomes a vital tool.

Deep learning allows data to be processed and learned from without manually extracting features. A deep neural network (DNN) can automatically extract features and learn from them. These features may even be features that would make no sense to a human observer, but can very easily be applied by a DNN to successfully produce the desired output. The obvious downside to these models is that because features are extracted automatically, we cannot really understand what knowledge the model has learned and how it arrives at the conclusions it makes. This makes it impossible to identify any bias in the model by solely observing the internal logic. We must wait and see how well it performs in a real-world context before any flaws in its training can be identified. Regardless, deep learning provides computer scientists with a way to understand incredibly complex data and provide tangible solutions to understand trends that are otherwise incomprehensible. With deep learning, data as complex as photographs can be understood by extracting features and identifying objects within the image, human language can be interpreted by a machine and responded to in an organic way that mimics human conversation, and enormous volumes of text can be summarized into a few short paragraphs while retaining all the relevant information.

**1.2.3 How Deep Learning is Changing Computer Science**

The advent of deep learning was brought about by the rapid leaps in technology and computational power at the turn of the century. These advancements have enabled us to be able to train these large models and adjust individual neurons to generate more complex cognitive capabilities in a machine. Applying deep learning to problems that involve complex data has allowed for the development of numerous technological solutions that have brought about massive change both in the field of technology and to the general human population. This technology is now being applied to EEG signal data by a number of researchers around the globe to try to create brain computer interfaces that can have a significant impact on the way we communicate with our computers.

Advancements in deep learning could in turn re-define our professional and everyday lives. As we find new ways to learn from data, and the neural networks we develop become more powerful, we can develop new ways to solve complex problems. As our technology advances, we can use the data we gather to tackle new issues, find patterns, and develop solutions. Deep learning has enabled researchers around the world to do things with computers that were thought impossible (at least in this lifetime) only a few years ago. As we become more and more capable of computing large volumes of data, we become capable of applying our machines in new ways. Computers are now capable of communicating in human language, understanding pictures, and even writing code. As we continue to advance our computational capabilities and the way we process data, technology will become more and more capable of understanding and interpreting the world around it. This, in turn, will launch our technology to a higher plane of capability, enabling us to create solutions to more complex problems. As this technology grows, new solutions will become apparent, but so will new problems. It is the continuous cycle of innovation and setback that fuels the progression of humankind.

**1.2.4 Deep Learning for BCI Technology**

The complexity of EEG data lies in the fact that it is time series data. Information contained within an EEG signal is not only dependent on its current state but also the previous states. One cannot determine a peak in brainwaves without knowing the previous values and states of the signal. Deep learning has a solution for this problem in the recurrent neural network (RNN) architecture. Furthermore, a subset of this architecture, the long short-term memory (LSTM) provides powerful contextual learning based on past states of the data. These architectures have been applied in a variety of areas, but their capabilities are perhaps not fully realized within the field of BCI technology. Many BCI applications incorporate the CNN architecture to interpret images or matrix representations of the signals. While this method has produced very good results, LSTM architectures could potentially eliminate the need for passing 2-dimensional data into a model by simply using the raw data values.

* 1. **Objectives of the Project**
* To build upon past research titled “Exploration and Optimization of Various Deep Neural Networks for Classification of EEG Signals for BCI Applications” (Sarkar et al.) and the positive results it yielded regarding the feasibility of Recurrent Neural Networks in the classification of EEG signals.
* To explore the long short-term memory (LSTM) architecture and its variations in order to build a powerful neural network capable of accurately classifying cognitive stimuli-based EEG signal data.
* To develop a variety of neural networks based on the LSTM architecture capable of classifying EEG signals into 7 target classes. These variations are: Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, Stacked Bidirectional LSTM, Vanilla LSTM with Attention, and Stacked LSTM with Attention.
* To train all of these architectures on a dataset of cognitive stimuli-based EEG signal data to build an accurate classifier.
* To determine the feasibility of LSTM architecture and its variations for classifying EEG signals and assess their feasibility for implementation within a brain computer interface.
  1. **Problem Statement**

Around the world, millions are affected by physical disabilities. These people have fully functioning brains and are limited only by their bodily ailment. BCI technology, facilitated by deep learning, could provide a solution to enable these individuals to overcome their physical limitations. Thid project aims to tackle this problem by developing variations on the LSTM architecture and training them to classify cognitive stimuli-based EEG signals. The end goal of this project is to identify a lightweight and accurate deep neural network architecture which may provide a feasible solution to develop powerful and deployable BCI applications.

* 1. **Chapter-wise Summary**

In Chapter 2: Literature Survey, this report will analyse multiple studies concerning BCI technology and the classification of EEG signals. Several contemporary research papers and scholarly articles will be evaluated to gain further understanding on the topic and identify areas where further research must be conducted. This project will analyse these papers in order to fully understand numerous studies within the same realm of research. This is done to augment the research conducted in this project and to emphasize the need for the research described by this report.

Chapter 3 will outline the research conducted in this project. It will describe the methodology of the research and the metrics used to quantify the results. It will demonstrate the technology applied, additional resources and technologies used to support this research, and will highlight the feasibility and legitimacy of the methodology. This chapter will demonstrate the process and methodology involved in achieving the results demonstrated later on, and will show the successful implementation of a clearly defined research methodology.

In Chapter 4: Implementation, the algorithm and details of implementation will be described. After explaining the prior research in chapter 2, and outlining the premise that this paper is exploring, this section will outline how the methodology described in chapter 3 was carried out. This methodology, when followed correctly, also ensures the reproducibility of this research, further lending to its legitimacy. It will describe how the goals of the project were achieved and what problems were faced during the process. This chapter will also define key checkpoints in the progression of this research and how each step is important in the implementation of this project. The purpose of this chapter is to demonstrate the successful implementation of the concepts and methodologies described in this report. It will serve as evidence of a properly implemented methodology and an adherence to proper guidelines and research practices.

Chapter 5: Results and Discussion will present the results of the research. This section will provide sufficient graphical and quantitative data demonstrating the results yielded by the methodology and algorithm described in the prior chapters. This chapter serves to demonstrate the quantifiable contribution of this research to the field and demonstrate the successful execution of the described methodology. Not only does this chapter demonstrate the successful implementation of the methodology, but also serves as evidence for the claims asserted in this report. It demonstrates the need for the research outlined in this report.

Chapter 6: Conclusion will outline the interpretation of the results and the implications of the results on the area of research. It will further investigate weaknesses in this methodology as well as future scope for this research.

**CHAPTER 2**

**LITERATURE SURVEY**

A brain computer interface is a device that is capable of converting human thought and cognitive activity into machine action. This is most commonly achieved through the interpretation of electroencephalogram (EEG) signals, which are electrical impulses created by the brain due to mental activity. While these signals can be difficult to interpret as they are unique to an individual and extremely complex, deep learning provides a way to rapidly train a bespoke model for a user while being capable of interpreting the complex data that EEG signals produce.

* 1. **The Status of Deep Learning for BCI Applications**

An evaluation of classical and modern methods of brain-computer interface technology delves into the current landscape of BCI research and the applications of artificial intelligence in augmenting BCI methodologies (Hossein et al., 2023). This study explored numerous public research databases such as PubMed, ERIC, JSTOR, IEEE Xplore and Google Scholar. 110 articles and nearly 20 years of research were reviewed to gain a thorough understanding of the timeline, history and progression of BCI research.

This study explored a variety of AI-based approaches to BCI applications including more traditional AI algorithms such as KNN clustering, Bayes Classification, and SVM regression as well as more modern and advanced deep neural networks such as LSTM and CNN. Finally, it highlights potential areas of exploration for future research such as Graph CNNs and Generative Deep Learning.

This study demonstrated the growing popularity and tremendous advancements in BCI technology as well as the impact of AI on its development. Furthermore, it demonstrates a strong correlation between the improvement of AI technology and the enhancement of BCI application accuracy and performance.

* 1. **Approaches to BCI Applications with Deep Learning**

Along the same vein of research, Alzhab et al. experimented with hybrid deep learning (HDL) networks based on Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Deep Belief Network (DBN) architectures to construct more powerful and accurate BCI applications (Alzhab et al., 2021). This research was unable to prove any tangible benefit to applying these hybrid networks as they were not able to out-perform the plain RNN architecture. However, the results themselves were quite promising as the RNN architecture was able to exceed an accuracy of 95%

A critical review of research and publications carried by Mansoor et al. has explored a wide variety of applications of deep neural networks towards EEG classification (Mansoor et al., 2020). This study was carried out to assess the capabilities of these models to classify signals specifically for application in a BCI. The results conclude that deep neural networks have tremendous potential for present and future applications of BCI technology.

An editorial by Ahn et al. analysed two reviews and seven research papers about the applications of deep learning in BCI technology. It took into account the results of the research as well as the authors’ perspectives and opinions on the outcome of the research (Ahn et al., 2022). This editorial not only demonstrates the capability of deep learning for augmenting BCI technology, but also highlights the number of methodologies and approaches towards applying deep learning as well as the number of researches pursuing this field of research.

An evaluation of current literature regarding the field of BCI technology and the applications of deep learning within the field reveals the potential for very promising exploration. It is evident that deep neural networks are capable of classifying very complex data to a high degree of accuracy. This makes them ideal for decoding the intricate patterns of EEG signals and interpreting the unique impulses generated by a subject’s brain.

Research conducted by Ingolfsson et al. led to the development of a temporal convolutional neural network for BCI application. This model achieved an accuracy of 0.84 and is considered a fairly accurate model. This architecture utilizes the popular CNN model that is often used for BCI applications while also analysing the temporal dimension to improve the accuracy of time-series data (Ingolfsson et al., 2020).

Research by Li et al. describes their attempt to develop an accurate EEG signal classification model using an LSTM model with an attention mechanism. This study yielded exceptionally promising results with an accuracy of 90.04%, or 0.90 (Li et al., 2020). This model utilized a CNN submodule in parallel to the LSTM/attention model which was then concatenated with a softmax classification. This model performed very well on binary emotional stimuli-based EEG data, but was a fairly large model as it contained 2 deep neural networks running in parallel.

* 1. **Justification for LSTM as a Solution for BCI Applications**

Prior research on cognitive stimuli-based EEG signals and potential deep neural networks for classification of this data yielded promising results to justify the use of LSTM cells to provide accurate classification. The objective of this paper was to develop and optimize recurrent neural network architectures for use with a brain computer interface. Using EEG data collected from subjects, a variety of neural network models were created to learn from the data. The models that were used were simple recurrent neural networks (RNN), long short-term memory (LSTM), and gated recurrent units (GRU). This paper proposes a novel approach to EEG signal classification, demonstrating the capabilities of recurrent networks which are seldom explored for this purpose (Sarkar et al., 2024).

This study produced promising results for recurrent models, obtaining a 91% accuracy with the 4-layer LSTM architecture. This presents a solid foundation for the argument that LSTM and similar architectures are feasible for BCI applications.

* 1. **Performance Metrics for Deep Neural Network Evaluation**

There are a variety of different approaches to determining the performance of a deep neural network. These are largely dependent on the task that the model performs. In the context of this project, the model will be performing a classification task. There are a variety of different classification metrics that can be applied: Accuracy, Precision, Recall, F1 Score, etc.

In his 2016 publication in Nature Methods, J. Lever describes the confusion matrix as the optimal method for visualizing and evaluating a model’s classification capabilities. The confusion matrix takes into consideration the expected prediction or ground truth as well as the predicted output by the model. The output is then placed on the matrix according to whether it was a correct classification and whether it was negative or positive (in the case of binary classification).

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1. Confusion Matrix** | | | |
|  | | Actual | |
| Positive | Negative |
| Predicted | Positive | TP | FN |
| Negative | FP | TN |

The confusion matrix allows us to identify how well the model classifies based on the class it is classifying into. This provides a more thorough understanding of the classification capabilities of the model and can demonstrate potential weak points in the model when it comes to specific classes (Lever, 2016).

A paper published by Hossin & Sulaimain outlines the various ways to score a model based on the confusion matrix values. These more precise scores provide metrics by which a model can be more accurately measured and compared to another to determine the capability of the classifier (Hossin and Sulaimain, 2015).

|  |  |  |
| --- | --- | --- |
| **Table 2. Performance Metrics for Classification Models** | | |
| **Metric** | **Formula** | **Evaluation Focus** |
| Accuracy (acc) |  | Measures the total number of correct classifications in relation to the total predictions made |
| Error Rate (err) |  | Measures the total number of incorrect classifications in relation to the total predictions made |
| Sensitivity (sn) |  | How many positive data points are correctly classified |
| Specificity (sp) |  | How many positive data points are correctly classified |
| Precision (p) |  | Out of the total positive classifications, how many are correct |
| Recall (r) |  | What fraction of positively classified data are correctly classified |
| F\_Measure or F1 Score (F1) |  | Harmonic mean between precision and recall values |

In the case of EEG signal classification, the Accuracy and F1 Score would be the most important metrics to consider, with the F1 Score taking the highest priority in the case of multiclass classification. This is because it is more important that the positive classifications are predicted correctly. In multiclass classification, especially when dealing with brain signals, we must ensure that we maintain a low tolerance for incorrect classifications.

* 1. **Need for BCI Technology**

The need for BCI technology ranges from healthcare to cybersecurity to entertainment and leisure. Mudgal et al. describe the need for BCI technology in a vast array of disciplines and industries, outlining the multitude of benefits as well as some potential pitfalls of the introduction of BCI technology. According to this explorative analysis, BCI technology can be applied to not only healthcare but domains that one would not typically associate with such technology. For example: gaming, education and marketing (Mudgal et al., 2020).

In a similar 2021 study, researchers from France explored the advancement of BCI technology and its place in the modern technological and professional landscape with respect to Industry 4.0. This study put forth the conjecture that BCI technology has advanced far enough that it will soon be ready to be taken out of the lab and deployed in the field for use in a professional/industrial setting. However, this study also warned against implementing this step prematurely. It highlighted the risks involved in deploying such technology without firmly establishing its reliability. This study established the feasibility and benefits of integrating BCI technology with Industry 4.0, while also stressing the need for caution and proper testing before doing so (Douibi et al., 2021).

* 1. **Applications of BCI Technology**

Douibi et al. further elaborate on their thesis point, outlining a wide variety of potential applications of BCI technology within the framework of Industry 4.0. This paper highlights the growing interest in BCI applications across domains, emphasizing the various universal roles of companies and how a brain-computer interface may help to enhance these roles. For example, one area in which BCI technology may provide necessary professional support to employees is in workplace safety (Douibi et al., 2021).

Research from AIIMS, Rishikesh underscores the advancements in neuroscience in the first quarter of the 21st century, suggesting that BCI technology will likely be the next big advancement in the field. This study goes on to explore the various potential applications of BCI technology across multiple domains. It describes the various types of procedures to implement a sensor apparatus for monitoring brain activity: invasive, semi-invasive, non-invasive. This paper then goes on to describe the clinical uses of BCI technology for the rehabilitation and aid of patients with various physical and mental disabilities.

This study then adopts a broader viewpoint, outlining potential applications of BCIs in various industries. For example, a BCI apparatus can be used to evaluate the effect of advertisements on a potential customer. They can also be used as biometrics for enhanced security in the private or government sector. BCI technology could also be used as a novel way of gaming by passing instruction via brainwaves rather than a traditional controller (Mudgal et al., 2020).

Lastly, this study delves into the pitfalls of BCI technology and what potential hurdles researchers would have to overcome. The most daunting of these is the fact that, due to genetic diversity as well as the unique human experience, all brains are different. It is therefore difficult to make a BCI that would be suitable for general purpose use while also maintaining a high level of accuracy and versatility in its functionality. Conversely, one could implement a bespoke device that is tailor made to the user. However, this would be far more costly and take longer to manufacture.

Other researchers have delved into the applications of BCI technology to help us navigate the web. The internet has revolutionized the way we connect with people and data. With BCI technology, we could further revolutionize the largest communication medium and information superhighway on the planet. With BCI technology, we would be able to introduce new methods of navigation, communication, authentication and recreation over the internet. This would empower more people to access the internet in new ways and share new information with each other (Janapati et al., 2022).

* 1. **Summary of Literature Survey**

A thorough evaluation of contemporary research within the field of BCI technology and deep learning has demonstrated the capability of brain-computer interfaces and has shown the feasibility of deep neural networks to further their development. This literature survey has provided a strong framework of the methodology for applying deep learning to the development of BCI technology. Moreover, this survey has demonstrated a gap in current research where further exploration and experimental analysis can be conducted.

Recent advancements in deep learning technology have seen the introduction of the LSTM network and the Seq2Seq model. This model is capable of encoding long, complex sequences and decoding them into a desired new sequence. This technology can be applied to a BCI application simply by replacing the LSTM decoder with a simple softmax layer to classify rather than produce sequential data. This can be further enhanced by exploring variations of the LSTM architecture such as Bidirectional LSTMs or LSTMs with Attention Mechanisms.

**CHAPTER 3**

**VARIATIONS ON THE LSTM DEEP NEURAL NETWORK ARCHITECTURE FOR CLASSIFICATION OF EEG SIGNALS FOR BCI APPLICATIONS**

This paper explores the LSTM architecture and its capability to classify EEG signals. The purpose for this task is for application within a brain-computer interface. This will be achieved by constructing multiple such models of several sizes, exploring the numerous variations of the LSTM architecture. These variations are: Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, and LSTM with Attention.

**3.1 Data Collection**

This project sought to utilize cognitive stimuli-based EEG signals to train the classification model. These signals were gathered using a G-Nautilus brain cap. Data was gathered by having subjects perform various cognitive challenges while wearing the cap. These challenges were designed to stimulate logic and reasoning as well as image recognition and language. The goal of these challenges was to stimulate the brain across multiple different cognitive tasks to facilitate training on the numerous processes of the brain.

The cognitive stimuli the subjects were exposed to varied in numerous ways, but were largely based on visual stimulation to trigger logical, spatial and verbal centres of the brain.



Penguin

Duck

Fig 1. Cognitive stimuli example

The cognitive stimulus shown above is designed to trigger cognitive dissonance. The stimuli present a visual component that is in direct contradiction to the linguistic component shown in the text box. This contradictory information stimulates the brain and allows for a large amount of data to be collected (Bran et al., 2020).

This data was then cleaned to remove any null values or noise that occur during times during which the patient was not subject to any cognitive stimuli. The resultant collected data contained 16 features and over 200 thousand rows of time series data. This data was passed to the model in the comma separated values (CSV) format, facilitated by the pandas library for Python.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 3. Sample dataset for cognitive stimuli-based EEG signal data using GUSBAmp** | | | | | | | | | | | | | | | | |
| **FP2** | **F4** | **C4** | **P4** | **F8** | **T4** | **T6** | **O2** | **FP1** | **F3** | **C3** | **P3** | **F7** | **T3** | **T5** | **O1** | **Label** |
| 1.21 | 7.41 | -5.47 | -1.07 | -1.43 | -7.3 | 4.09 | 6.81 | -4.09 | 38.49 | -19 | 4.88 | -6.17 | -7.83 | 1.81 | 5.66 | 1 |
| -1.89 | 55.6 | 0.45 | 5.32 | 69.4 | 9.52 | 2.8 | 7.9 | 13.6 | -46.7 | -9.65 | -63.1 | -71 | -14.7 | 27.8 | 79.5 | 3 |
| -1.4 | -2.25 | -16.4 | -6.98 | -1.49 | -1.38 | -3.44 | -47.39 | 1.41 | -5.66 | -6.71 | -4.17 | -8 | -9.65 | -4.82 | -1.06 | 2 |
| 60.81 | -20.14 | -6.87 | 77.84 | 51.84 | -5.53 | -8.37 | -1.47 | -9.29 | 0.66 | -8.16 | -9.82 | -6.86 | -7.32 | -0.36 | -82.41 | 4 |
| -72.6 | -9.15 | -48 | -6.08 | 68.4 | -2.1 | -9.69 | -3.95 | -7.06 | -3.14 | -27.1 | -50.1 | -61 | -0.3 | 5.53 | -87.4 | 1 |
| -8.28 | -37.7 | -3.34 | -56.54 | 1.87 | 30.17 | -7.9 | -1.49 | -2.85 | -6.79 | -2.76 | -5.61 | -3.28 | -9.47 | -7.07 | -6.52 | 7 |
| -2.6 | 9.87 | -0.3 | -5.22 | 11.85 | -8.1 | -23.2 | -1.26 | -1.28 | -67 | -28.8 | -57.8 | -59 | -68.8 | -2.26 | -25 | 5 |
| -8.69 | -21.2 | -0.58 | 13.32 | -2.22 | -2.38 | -8.85 | -9.69 | -3.45 | -7.32 | -2.06 | -7.54 | -7.79 | -9.29 | -61.76 | -7.96 | 2 |
| … | | | | | | | | | | | | | | | | |
| -0.38 | 12.2 | 8.73 | 31.07 | 76.7 | 6.8 | 18.2 | 93.8 | 88 | -68 | -7.28 | -85.3 | -84.1 | -4.3 | 26.5 | 80.6 | 6 |

The table above shows a small sample of the dataset. The first 16 rows represent the electric potentials at each sensor on the EEG sensor array. The column names represent the position of the sensors on the patient’s head. These positions correspond to standard medical notation as seen below.

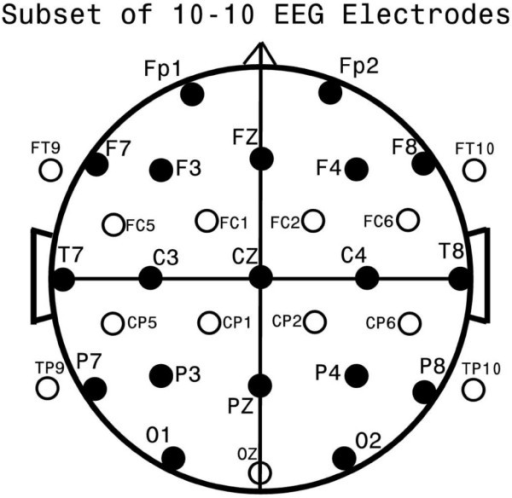


Fig 2. Positions of EEG electrodes and corresponding labels (Duffy et al., 2011).

**3.2 LSTM**

Long Short-Term Memory (LSTM) is a type of artificial neuron based off of the Recurrent Neural Network (RNN) neuron. This subset of neural network takes inputs at regular time intervals and is capable of retaining information from previous time steps. This past information is used to make more informed inferences for data that is dependent on past data. Such data includes audio, weather and signal data. However, one of the larger issues with the RNN architecture is that it struggles to maintain long-term dependencies. After a short amount of time, the RNN is less capable of retaining relationships between data. The LSTM cell was developed to overcome the vanishing and exploding gradient that the RNN architecture faced (Hochreiter et al., 1997).

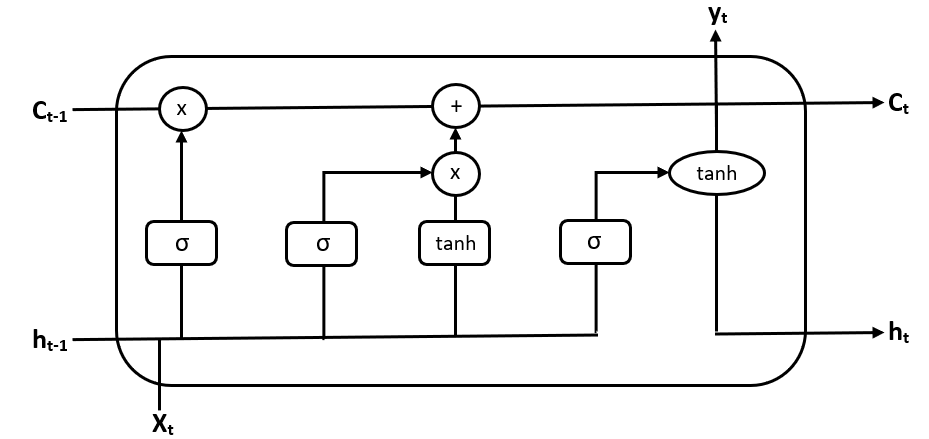


Fig 3. LSTM Architecture

The LSTM architecture is based on 3 ‘gates’, namely the input, output and forget gate. It also has a cell state that is transferred to the next cell along with the cell’s output. The forget gate is the first gate in the cell. The input is passed into this gate along with a weight and bias as a sigmoid function. This is then multiplied with the cell state. Essentially, if the result is 0, the information is forgotten, and if the result is 1, the information is remembered and passed to the next cell state.

The input gate adds any new, useful information from the current timestep to the cell state. First, in a similar way to the forget gate, a sigmoid function is applied to the input to filter the values that are to be remembered. Then, a tanh function is applied to the input to form a vector. This vector is multiplied with the filtered information to generate useful information for the cell state to retain.

The output gate is responsible for extracting useful information from the cell state and presenting it as output. First, a tanh function is applied to the cell state value to create a vector. Then, a sigmoid function is applied to the input to once again generate the values to be remembered. These 2 values are multiplied together to generate output. The output for that timestep finally exits the cell as a processed inference. The output and cell state are also passed to the next cell in the network.

This more complex architecture provides the cell with more precision in terms of what information to retain and what information is unnecessary and might produce noise. This helps the overall network better retain information over longer periods of time while reducing the propagation of error through time due to noisy, unnecessary or inaccurate data.

These advantages to using LSTM networks makes them a powerful tool for language modelling, image processing, speech/handwriting recognition and even music generation. However, there are some drawbacks to using LSTM networks. First of all, due to the increased complexity of the neurons, LSTM networks can take longer to train. The 3-gate structure requires 5 functions each with their own weight and bias, while the standard RNN model contains a single weight and bias for each cell. Moreover, as a result of the increased complexity, they are more memory intensive. It is therefore important, when using LSTM networks, to find an ideal balance between size and accuracy (Hochreiter et al., 1997).

**3.3 Variations of the LSTM Architecture**

There are many variations to the LSTM architecture that improve upon the various capabilities of these networks to further augment the power of LSTM. The most basic LSTM network is called the Vanilla LSTM. This network consists of a single layer of LSTM cells of a variable number. For classification using Vanilla LSTM, a Softmax layer is added to classify into any number of target classes. If multiple such LSTM layers are connected sequentially, this is called a Stacked LSTM. The goal of a Stacked LSTM, as with any multi-layered neural network, is to extract better and more useful features to use for classification. Unlike with a Convolutional Neural Network, these features can also be oriented along the dimension of time. Stacking multiple layers of LSTM cells results in a longer training time, but an overall more accurate model in most cases.

LSTM architectures are designed to predict patterns based on sequential information. The standard LSTM model assumes that the sequence flows in the direction of time and predicts the sequence in a single direction. However, it is possible that more information might be inferred based on the backward flow of information. This is where the Bidirectional LSTM comes in. This architecture has a forward and backward layer connected parallelly to each other. This allows for features to be extracted based on the flow of sequential data in both directions. While this adds even more complexity to the network and a higher number of parameters, it allows for superior feature extraction for data where patterns can be observed in the reverse direction.

While the aforementioned architectures help with feature extraction and provide a more powerful model overall, the trade-off is that the model will be much larger. With EEG signal data, it is not necessarily complex features that make data difficult to classify. Instead, it could be the length of the sequence itself. And when it comes to long and complex sequences, the Attention Mechanism is perhaps the most powerful tool available. The Attention Mechanism is a relatively new neural network design that frees the encoder-decoder architecture from the fixed length internal representation. This means, in essence, that the model can focus or “pay attention” to specific inputs and relate them to the output regardless of the internal fixed length that the network is supposed to analyse. The benefit of applying an Attention Mechanism to LSTM is that the model is able to focus on important data and features without adding more layers. This helps the model scale its accuracy without the number of parameters growing to quickly (Kang et al., 2023).

This methodology applied all of the aforementioned LSTM-based models to attempt to train a classifier for cognitive-stimuli based EEG signals. 5 models of varying sizes were trained for each LSTM variation. These sizes were based on the number of LSTM cells in each layer: 16, 32, 64, 128 and 256.

For the Stacked LSTM architecture, a 4-layer architecture was used. For the Bidirectional LSTM model, a 1-layer and 2-layer model were both tested to determine whether the benefits of a Stacked-LSTM architecture could be carried over to a bidirectional architecture as well. For the LSTM with Attention, a single attention layer was applied to both a Vanilla and Stacked LSTM architecture.

For the stacked architectures, a 4-layer architecture was applied in order to maintain a reasonable number of features for the architecture. The objective of adding more layers is to extract more features. However, the result is a model with far more parameters. The 4-layer architecture contains nearly 7 times more features than the vanilla architecture with the same layer size. Therefore, it was concluded that any larger and this study would become highly impractical and the architecture would be computationally infeasible for BCI application.

The Keras library was utilized to facilitate the model construction while NVIDIA T4 GPUs via Google Colab were utilized to facilitate the training of these models. The data was then consolidated and analysed with respect to accuracy and model size to determine the ideal model for use in a BCI application.

**3.4 Selection of 4-Layer stacked LSTM model**

For the stacked architecture to be implemented in this methodology, there is no fixed requirement to implement a specific number of layers. With the addition of more layers, the model will be able to extract deeper features and be able to classify to a higher degree of accuracy. However, every additional layer greatly increases the number of parameters and the complexity of the model. The various possibilities for layer size were tested to determine an optimal number of layers for a stacked LSTM based on the accuracy and the respective increase in number of parameters. The results of this experiment are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 4. LSTM layers vs accuracy and number of parameters** | | | | |
| **Number of Layers** | **Training Accuracy** | **Validation Accuracy** | **Test Accuracy** | **Number of Parameters** |
| 1 | 61.69% | 62.16% | 62.27% | 17351 |
| 2 | 81.90% | 91.61% | 87.01% | 50375 |
| 3 | 89.89% | 92.85% | 84.91% | 83399 |
| 4 | 88.15% | 89.26% | 90.97% | 116423 |
| 5 | 91.43% | 90.03% | 91.44% | 149447 |
| 6 | 91.27% | 92.31% | 91.86% | 182471 |

From the results seen in table 2, it is evident that the increase in accuracy becomes minimal after the number of layers in the architecture exceeds 4. Meanwhile, the number of parameters continues to increase at a dramatic rate. Therefore, in the context of this study, the best architecture would be the 4-layer Stacked LSTM. This provides a good level of accuracy and a reasonable trade-off between accuracy and number of parameters.

For the reason outlined above, supported by experimental data, the 4-layer architecture was chosen to further explore the capabilities of LSTM architectures for the classification of the specified EEG data.

**CHAPTER 4**

**IMPLEMENTATION**

Over a 1-month period, all the architectures were built and trained. Each model took roughly 1 hour to train using the T4 GPU provided by Google Colab. The training process for each model followed roughly the same algorithm and was facilitated by Python libraries such as Pandas, Numpy and Keras. The code for this implementation can be found in the appendices.

**4.1 Algorithm**

The training process followed the algorithm below:

1. Import the necessary libraries (Pandas, Numpy, Keras, Sklearn, etc.)
2. Using pandas, load the dataset into a dataframe
3. Divide the dataset into X and y based on features and target class columns
4. Split the dataset into train and test using sklearn train\_test\_split with 30% test size
5. Create the desired neural network model using Keras
6. Define the hyperparameters for training
7. Train the model on the dataset for 15 epochs
8. Predict using the test set and evaluate the model
9. Record the Accuracy, Precision, Recall and F1 Score.
10. Repeat the above steps for all architectures
11. Compare the results and determine the best model for BCI application

After applying this algorithm, all the results were consolidated into tables and analysed to determine the ideal architecture for BCI application. The analysis of the collected data revealed interesting results about the behaviour of the LSTM architecture and its capability to understand the EEG signal data.

**4.2 Implementation Details**

The Python libraries used to facilitate the implementation of this project were: Pandas, Scikit-learn TensorFlow, Keras and NumPy. Pandas is a Python library used for working with datasets. It enables a developer to convert structured data into data frames. These data structures make it easy for one to store data for training with deep learning models. Pandas facilitates the conversion from CSV data to Python-readable data that can be used to train the deep neural networks.

NumPy is a library used for working with arrays in Python. NumPy arrays, unlike typical arrays in Python, are optimized for mathematical operation. This is an ideal library for deep learning which requires the performing of millions of array operations every second. NumPy helps accelerate the deep learning process and reduce the computational load.

Scikit-learn is an ML (Machine Learning) library in Python that facilitates the construction of numerous learning models. Moreover, it also facilitates the preprocessing of data and the evaluation of learning models using metrics such as accuracy, precision, recall and f1 score.

TensorFlow is an open-source library for high-performance computation in Python. Its flexible architecture facilitates high-volume mathematical computation on GPUs and TPUs. As the name implies, TensorFlow is a library for operation with tensors. Tensors are n-dimensional arrays of numbers. These numerical fields are vital to deep learning and contain the primary data used for building deep neural networks. TensorFlow facilitates the building and training of complex deep neural networks.

Keras is a library that acts as an interface for the TensorFlow library, further augmenting and facilitating the deep learning process. Keras helps developers build more complex neural network with greater ease as well as build more conventional neural networks faster as well. The Keras library is ideal for this project as it demands the construction of complex neural networks that use architectures such as Attention Mechanisms and bidirectional LSTM cells. Keras facilitates more complex deep learning development at a more rapid pace with greater ease and understandability. With these libraries, the algorithm can be more easily implemented.

**Dataset Details**

The dataset contained over 200,000 rows and 17 columns of data. The data was collected from a GUSBAmp EEG sensor and stored as a CSV (Comma Separated Values) file. This data was loaded into a Pandas dataframe for easy training of the deep learning model. The Pandas library was also used to split the target column (dependent variable) from the rest of the dataset (independent variables) in order to pass both into the learning model.

**Train Test Split**

A train test split is used to separate the dataset into 2 smaller datasets, namely “train” and “test”. The train dataset is used for the training of the learning model. The model will use the “X\_train” set for input to predict the corresponding class in the “y\_train” set. After doing so, the model updates its parameters using backpropagation and its learning rate. In more complex models such as the ones implemented in this project, there are further steps such as cross validation in which a smaller dataset called a validation set is derived from the train set to update the parameters as well.

The test set is used to then evaluate the model on unseen data. If the same train data is used to evaluate the model, one cannot determine whether the model has learned any valuable information from the data, or whether it has simply memorized the corresponding outputs. To gain a better understanding of the model’s output accuracy, i.e. how well trained the model is, one must test the model on a completely unseen dataset. For this reason, the model is tested on the test set.

**Model Creation and Hyperparameters**

A deep neural network is typically constructed with the help of a number of python libraries in order to simplify the process and reduce the amount of code required. For this project, the TensorFlow and Keras libraries were used.

To build a typical deep neural network, the first step is to define the hyperparameters. These are specific rules by which the neural network will learn. Values such as the learning-rate will determine the scaling of the weight updates during training steps. If the learning rate is very high, larger changes will be made to the weights of the neurons, which will facilitate fast learning, but may make the model prone to overshooting the ideal weight. Meanwhile, a very low learning rate may lead the model to not learn anything at all. Another important hyperparameter is the optimizer. This defines what algorithm that will be used to optimize the learning process as the model is trained. The chosen optimizer for this project tis Nadam, which uses the Nesterov lookahead algorithm and an adaptive learning method to adjust its learning rate according to the gradient at which the error reduces.

Once the hyperparameters are set, the next phase is building the neural network itself. The first step to do this is to define the input layer to take the desired data in the correct dimensions to pass to the learning layers. Once the input layer is defined, the learning layers can be constructed. TensorFlow and Keras have numerous pre-defined layer types such as CNN, RNN and LSTM layers. For this project, the LSTM layer from the Keras library was used. Furthermore, the Bidirectional function was also used to pass one LSTM layer to another in a bidirectional structure before sending the result forward. However, Keras does not have a pre-defined Attention Mechanism structure. Instead, the Attention Mechanism had to be constructed as a user-defined class to construct the desired neural networks for this project. The final layer is a Dense layer which is a cluster of artificial neurons which are fully connected to every other neuron in the previous layer. These layers are usually the most useful for finetuning a neural network to a specific dataset. The dense layer then passes output to a SoftMax classification layer to produce the final output class.

Once the neural network is constructed, we can build it and see the final constructed model to get more details on whether all the layers have been built properly. We can also get other data on the model such as the number of parameters in the final constructed neural network.

**Training the Model**

Once the model is constructed, we must train it. Typically, in deep learning, one requires a large volume of data. The more (accurate) data, the better the model will be. For this reason, a dataset of over 200,000 rows was used to train this model. Deep learning uses an iterative approach to training. This means that the model goes through the entire dataset once before evaluating itself. This process is then repeated as many times as needed before the model reaches the most accurate it can be. The unit of training for which the model goes through the entire dataset is called an epoch.

For this project, 15 epochs were used to train each model. This was set as a standard as it allowed the models to reach a good level of accuracy without consuming too much time and computational resources. It also prevents the model from becoming overfit to the dataset and being unable to generalize well.

For this project, 30 deep learning models were trained on 200,000 rows of data each. For this, tremendous computational power was required. In order to speed up this process to complete the research and training in a suitable timeframe, powerful GPUs were utilized. The Google Colab cloud platform was leveraged to facilitate this heavy computational load. This platform provides access to Nvidia T4 GPUs which are capable of performing millions of operations in a fraction of the time it would take a normal CPU. For this reason, all of the models described in this project were trained and tested using the Google Colab platform.

**Evaluating the Models**

To evaluate the models, it is important to select the right metrics. For this project, there were 4 metrics that were the most important to analyse: Accuracy, Precision, Recall and F1 Score. The accuracy score is the simplest metric and it gives a raw evaluation of the model’s performance. This is the most straightforward way to evaluate exactly how ‘correct’ the model’s predictions were according to the test set. Precision and recall are both very useful to measure the model’s capability of predicting the correct class in relation to incorrect predictions. Finally, the F1 Score provides a harmonic mean between the precision and recall, thereby providing the best measure for the quality of output of the model, while accounting for biases within the data.

**Comparing Results**

After all the models were trained and evaluated, the results were collected and compared. To do so, the first step was to put all of the metrics for a single architecture type into one table. For example, for this project, 5 Vanilla LSTM architectures were trained. These were based on different layer sizes: 16, 32, 64, 128, 256. The results for each different model were combined into a single table. These results were then graphed to see the change in F1 Score as the layer size increased.

Once these results were consolidated for each architecture, the best model was identified. This was done by observing the trade-off between model size and F1 Score. Once the best models were identified for each architecture, they were catalogued in a separate table and compared.

A separate graph was then created to compare all of the best models. This graph also took into account the model size, based on the number of parameters. This allowed for a more thorough evaluation of the models by evaluating not only their classification capability, but also the speed at which they could classify. These models were also compared to a reference model that used a hybrid CNN/LSTM architecture with an Attention Mechanism. This was done to verify the capability of the models developed in this project with reference to other work within the same domain.

**CHAPTER 5**

**RESULTS AND DISCUSSION**

As per the methodology presented above, the LSTM models were constructed and trained on 200,000 rows of EEG signal data. Each row consists of 16 input variables and a single target output. The trained models were then evaluated and compared with results from the bidirectional LSTM with self-attention mechanism as developed by Li et al. The benchmark set by this architecture was 0.90 which was considered quite accurate, showing potential for future application in BCI technology.

For the Vanilla LSTM architecture, the results are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 5. Vanilla LSTM Performance Metrics** | | | | | | |
| **LSTM Cells per Layer** | **No. Params** | **Size** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
| 16 | 1271 | 4.96 | 0.45 | 0.49 | 0.54 | 0.45 |
| 32 | 4583 | 17.9 | 0.56 | 0.60 | 0.64 | 0.56 |
| 64 | 17351 | 67.78 | 0.6 | 0.63 | 0.67 | 0.6 |
| 128 | 67463 | 263.53 | 0.61 | 0.64 | 0.68 | 0.61 |
| 256 | 265991 | 1034.24 | 0.63 | 0.65 | 0.68 | 0.63 |

Fig 4. Vanilla LSTM F1-Score vs Number of Parameters

As seen in the table and graph above, the Vanilla LSTM has a moderate classification capability that scales as expected as the number of LSTM cells in the layer (and the number of parameters) increase. From the above trend, it is also evident that this architecture is unlikely to be able to achieve an F1-Score above 0.7. Therefore, while we can observe the capability of the LSTM mechanism, it is clear that the Vanilla architecture is not complex enough for this classification task.

The next model that was tested was the Stacked LSTM. The results for this architecture are given below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 6.** **Stacked LSTM Performance Metrics** | | | | | | |
| **LSTM Cells per Layer** | **No. Params** | **Size** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
| 16 | 7607 | 29.71 | 0.74 | 0.75 | 0.76 | 0.74 |
| 32 | 29543 | 115.4 | 0.79 | 0.79 | 0.8 | 0.79 |
| 64 | 116423 | 454.78 | 0.91 | 0.91 | 0.91 | 0.91 |
| 128 | 462215 | 1802.24 | 0.94 | 0.94 | 0.94 | 0.94 |
| 256 | 1841927 | 7198.72 | 0.91 | 0.91 | 0.91 | 0.91 |

Fig 5. Stacked LSTM F1-Score vs Number of Parameters

As seen in the above table and graph, the F1-Score increases along the same logarithmic trend as for the Vanilla architecture. Moreover, the proportion of the number of parameters also increases at the same rate as in the case of the Vanilla model. However, the accuracy is significantly higher for this architecture as it is capable of extracting more complex features from the data.

The Stacked LSTM architecture achieves a very high accuracy, though it is worth noting that the number of parameters is between 6 and 7 times higher than the Vanilla model. Other variations of the architecture have been explored to attempt to reduce this size constraint.

The Bidirectional LSTM was explored to determine whether more features could be extracted by passing the data in reverse, to supplement fewer layers and LSTM cells.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 7. 1-Layer Bidirectional LSTM Performance Metrics** | | | | | | |
| **LSTM Cells per Layer** | **No. Params** | **Size** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
| 16 | 5895 | 23.03 | 0.73 | 0.74 | 0.76 | 0.73 |
| 32 | 15879 | 62.03 | 0.8 | 0.81 | 0.82 | 0.8 |
| 64 | 48135 | 188.03 | 0.82 | 0.83 | 0.84 | 0.82 |
| 128 | 161799 | 632.03 | 0.9 | 0.90 | 0.9 | 0.9 |
| 256 | 585735 | 2283.52 | 0.9 | 0.90 | 0.91 | 0.9 |

Fig 6. 1-Layer Bidirectional LSTM F1-Score vs Number of Parameters

From the results of the 1-Layer Bidirectional LSTM model training, one can clearly observe the increased classification power of the Bidirectional LSTM architecture. This model is achieving results comparable to the stacked LSTM model with only double the number of parameters as the Vanilla LSTM model. In fact, the 128 model achieved the same score as the 256 model and is smaller than the best performing Vanilla LSTM by 100 thousand parameters.

As the 1-Layer Bidirectional demonstrated such excellent results, the 2-Layer bidirectional model was also explored in the hopes that it would yield the similar results:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 8. 2-layer Bidirectional LSTM Performance Metrics** | | | | | | |
| **LSTM Cells per Layer** | **No. Params** | **Size** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
| 16 | 12167 | 47.53 | 0.78 | 0.78 | 0.78 | 0.78 |
| 32 | 40711 | 156.03 | 0.93 | 0.93 | 0.93 | 0.93 |
| 64 | 146951 | 574.03 | 0.94 | 0.94 | 0.94 | 0.94 |
| 128 | 556039 | 2170.88 | 0.94 | 0.94 | 0.94 | 0.94 |
| 256 | 2160647 | 8437.76 | 0.95 | 0.95 | 0.95 | 0.95 |

Fig 7. 2-Layer Bidirectional LSTM F1-Score vs Number of Parameters

As expected, the 2-Layer Bidirectional model performed better than the 1-Layer variant in the same way that the Stacked LSTM performed better than the Vanilla LSTM. While the highest performance of 0.95 is only a small increment in relation to the much higher number of parameters, the size 64 model was able to achieve an F1-Score of 0.94 with far fewer parameters. The results from the Bidirectional LSTM models shows that a neural network can classify EEG data very effectively when it is passed in both directions.

The last variant of LSTM that was tested in this investigation was the LSTM with Attention. This variation was divided into 2 separate architectures. The first was a Vanilla LSTM with an Attention Mechanism. The second was a Stacked LSTM with an Attention Mechanism. The results for the Vanilla LSTM Attention are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 9. Vanilla LSTM with Attention Mechanism Performance Metrics** | | | | | | |
| **LSTM Cells per Layer** | **No. Params** | **Size** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
| 16 | 1303 | 5.09 | 0.68 | 0.69 | 0.7 | 0.68 |
| 32 | 4631 | 18.09 | 0.68 | 0.69 | 0.71 | 0.68 |
| 64 | 17431 | 68.09 | 0.77 | 0.77 | 0.77 | 0.77 |
| 128 | 67607 | 264.09 | 0.84 | 0.85 | 0.87 | 0.84 |
| 256 | 266263 | 1044.48 | 0.9 | 0.90 | 0.9 | 0.9 |

Fig 8. Vanilla LSTM with Attention F1-Score vs Number of Parameters

As seen in the results above, the Attention Mechanism is able to significantly augment the results without significantly increasing the size of the model. By adding an Attention Mechanism to the Vanilla LSTM network, the F1-Score is increased until it is comparable to the stacked model while maintaining the size of the Vanilla LSTM model.

To leverage the power of the Attention Mechanism on more complex features, an Attention layer was added to the stacked LSTM architecture as well. The results of this architecture are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table 10. Stacked LSTM with Attention Mechanism Performance Metrics** | | | | | | |
| **LSTM Cells per Layer** | **No. Params** | **Size** | **Accuracy** | **F1 Score** | **Precision** | **Recall** |
| 16 | 7639 | 29.84 | 0.76 | 0.77 | 0.78 | 0.76 |
| 32 | 29591 | 115.59 | 0.8 | 0.81 | 0.82 | 0.8 |
| 64 | 116503 | 455.09 | 0.94 | 0.94 | 0.94 | 0.94 |
| 128 | 462359 | 1802.24 | 0.95 | 0.95 | 0.95 | 0.95 |
| 256 | 1842199 | 7198.72 | 0.95 | 0.95 | 0.95 | 0.95 |

Fig 9. Stacked LSTM with Attention F1-Score vs Number of Parameters

As seen in the data above, the Attention Mechanism is able to squeeze out a little more accuracy from the model with very little increase to the number of parameters. This allows for a noticeably more capable model with little to no compromise in terms of model size.

Looking at the data gathered from all the models, one can observe a number of interesting trends. The first is that each model follows roughly the same trend in terms of the increase in F1-Score in relation to number of parameters. A clear logarithmic relationship is present in all of the variations that were tested. From this trend, one can infer that regardless of the number of layers, there is a limit to how accurate the model can be based on the type of architecture used. Therefore, to achieve an effective EEG signal classifier, it is not size that is the primary constraint, but the neural network architecture.

Based on the capability of these various LSTM-based networks in relation to their size, one can plot a graphical representation of F1-Score against the number of parameters for each model:

Fig 10. F1 Score vs Number of Parameters for best architectures of all models

From the consolidated data above, one can see that the LSTM architecture, in general, is very effective at classifying EEG signal data. If the LSTM architecture is further augmented by modifying the way in which features are learned, one can further enhance the classification capability of this architecture to construct a very powerful classification model.

Comparing all the best trained models, and evaluating them based on accuracy and size, it is evident that the Stacked LSTM with Attention provides the architecture most appropriate for use in a BCI.

**CHAPTER 6**

**CONCLUSION**

This experimental analysis sought to build and test multiple variations of the LSTM architecture on cognitive stimuli-based EEG signals. Upon implementation, the results demonstrate without any reasonable doubt that the LSMT is an entirely viable means to interpret and classify EEG signal data.

The results of the best model, the Stacked LSTM with Attention Mechanism proved to provide an accuracy and F1 score of 94% while maintaining a comparatively small number of parameters (116 thousand). This is not only a testament to the capability of the LSTM architecture, but also to the remarkable power of the attention mechanism that has so heavily impacted the domain of deep learning.

The methodology applied provides a solid foundation for the justification of the focus of this research. The testing of numerous different layer sizes allowed for a more comprehensive representation of the capabilities of each of the tested models. The consideration of the F1 score and comparison to the number of parameters is appropriate for the evaluation of each model given the context in which these models would ideally be deployed. However, a more thorough analysis of the distribution of class-based performance and insight into potential biases within the model or data might have provided a more thorough study and analysis of these architectures.

Through the implementation of the methodology described in this paper, the capability of the LSTM architecture has been clearly demonstrated. The LSTM architecture, in all trials, has shown its capability to successfully classify EEG signal data to a high degree of accuracy. By further augmenting the LSTM architecture, an even stronger classifier can be designed which is even more adept at deciphering the complexities of EEG data. The results of this experimental analysis establish the feasibility of the LSTM architecture as a candidate for developing BCI technology for a wide array of use cases.

Based on the results of this experimental analysis, the architecture of best performance utilises a unique attention based seq2seq variation with no decoder as the output does not require a sequential output but rather a single output deciphered using a softmax function. This resultant architecture provides a potential solution for future BCI technology and may hold potential to revolutionize the way we communicate with our technology

**6.1 Drawbacks in Methodology**

While the methodology was carried out successfully and has yielded very promising results, there are some drawbacks. The first and most notable drawback is that this methodology assumes the perfectness of the dataset. No cleaning or augmentation was performed on the dataset as it assumes that the collection is without error. The second drawback is similar to the first. This methodology involves a very complex dataset of EEG signals. It is therefore nearly impossible to identify erroneous data or areas in which data can be augmented without introducing an element of error. Lastly, this project does not outline the potential sources of error in this methodology. While a lot of this can be attributed to the prior 2 drawbacks, this is also a drawback that is detrimental to further development and improvement of the current methodology.

**6.2 Future Scope**

This project proposes a lightweight and accurate classification model for potential deployment in a brain computer interface. There are 2 potential paths of development with this research, both of which must be pursued. The first is to augment this network to achieve even higher accuracy. This would include further tuning the hyperparameters of these models to achieve higher accuracy without increasing the number of parameters, i.e. the size of the model. This would be a valuable augmentation to this research as it would provide a more powerful model that would further lend to the positive impact these models could have.

The second way in which this project can be taken further is to incorporate it into a device for actual BCI application. This model can be trained for specific tasks to aid the differently abled or to be deployed in an industrial setting to aid in complex manufacturing and maintenance roles.

In these ways, this research can contribute to further innovation and advancements in the field of deep learning and BCI technology.

**REFERENCES**

[1] Hossain, K. M., Islam, Md. A., Hossain, S., Nijholt, A., & Ahad, M. A. (2023). Status of deep learning for EEG-based brain–computer interface applications. Frontiers in Computational Neuroscience, 16. https://doi.org/10.3389/fncom.2022.1006763

[2] Jaipriya, D., & Sriharipriya, K. C. (2022). A comparative analysis of masking empirical mode decomposition and a neural network with feed-forward and back propagation along with masking empirical mode decomposition to improve the classification performance for a reliable brain-computer interface. Frontiers in Computational Neuroscience, 16. https://doi.org/10.3389/fncom.2022.1010770

[3] Alzahab, N. A., Apollonio, L., Di Iorio, A., Alshalak, M., Iarlori, S., Ferracuti, F., Monteriù, A., & Porcaro, C. (2021). Hybrid deep learning (hdl)-based brain-computer interface (BCI) systems: A systematic review. Brain Sciences, 11(1), 75. https://doi.org/10.3390/brainsci11010075

[4] Mansoor, A., Usman, M. W., Jamil, N., & Naeem, M. A. (2020). Deep Learning Algorithm for Brain-Computer interface. Scientific Programming, 2020, 1–12. https://doi.org/10.1155/2020/5762149

[5] Ahn, M., Jun, S. C., Yeom, H. G., & Cho, H. (2022). Editorial: Deep learning in Brain-Computer interface. Frontiers in Human Neuroscience, 16. https://doi.org/10.3389/fnhum.2022.927567

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[7] Altaheri, H., Muhammad, G., Alsulaiman, M., Amin, S. U., Altuwaijri, G. A., Abdul, W., Bencherif, M. A., & Faisal, M. (2021). Deep learning techniques for classification of Electroencephalogram (EEG) motor imagery (MI) signals: A Review. Neural Computing and Applications, 35(20), 14681–14722. https://doi.org/10.1007/s00521-021-06352-5

[8] Kang, Q., Chen, E. J., Li, Z.-C., Luo, H.-B., & Liu, Y. (2023). Attention-based LSTM predictive model for the attitude and position of shield machine in tunneling. Underground Space, 13, 335–350. https://doi.org/10.1016/j.undsp.2023.05.006

[9] Ingolfsson, T. M., Hersche, M., Wang, X., Kobayashi, N., Cavigelli, L., & Benini, L. (2020). EEG-TCNet: An accurate temporal convolutional network for embedded motor-imagery brain–machine interfaces. 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). https://doi.org/10.1109/smc42975.2020.9283028

[10] Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of Deep Learning. Neurocomputing, 452, 48–62. https://doi.org/10.1016/j.neucom.2021.03.091

[11] Lieskovská, E., Jakubec, M., Jarina, R., & Chmulík, M. (2021). A review on speech emotion recognition using Deep Learning and attention mechanism. Electronics, 10(10), 1163. https://doi.org/10.3390/electronics10101163

[12] Liu, G., & Guo, J. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification. Neurocomputing, 337, 325–338. https://doi.org/10.1016/j.neucom.2019.01.078\

[13] Lieskovská, E., Jakubec, M., Jarina, R., & Chmulík, M. (2021). A review on speech emotion recognition using Deep Learning and attention mechanism. Electronics, 10(10), 1163. https://doi.org/10.3390/electronics10101163

[14] Li, W., Qi, F., Tang, M., & Yu, Z. (2020). Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification. Neurocomputing, 387, 63–77. https://doi.org/10.1016/j.neucom.2020.01.006

[15] Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., & Muller, K.-R. (2021). Explaining Deep Neural Networks and beyond: A review of methods and applications. Proceedings of the IEEE, 109(3), 247–278. https://doi.org/10.1109/jproc.2021.3060483

[16] Sarkar, P. S., Mathew, J., Stephen, C., Kanaga, E. G. M., & Bhuvaneswari, M. (n.d.). An Experimental Analysis of Various Deep Learning Architectures for the Classification of Cognitive Stimuli based EEG Signals. GRENZE International Journal of Engineering and Technology, 10(1), 250–255

[17] Bran, A., & Vaidis, D. C. (2020). On the characteristics of the cognitive dissonance state: Exploration within the pleasure arousal dominance model. Psychologica Belgica, 60(1), 86–102. https://doi.org/10.5334/pb.517

[18] Duffy, F. H., McAnulty, G. B., McCreary, M. C., Cuchural, G. J., & Komaroff, A. L. (2011, July 1). EEG spectral coherence data distinguish chronic fatigue syndrome patients from healthy controls and depressed patients-A case control study. BMC Neurology, 11(1). <https://doi.org/10.1186/1471-2377-11-82>

[19] Lever, J. (2016). Classification evaluation: it is important to understand both what a classification metric expresses and what it hides. Nature Methods, 13(8), 603+. <https://link.gale.com/apps/doc/A459507798/HRCA?u=anon~40edfc7e&sid=googleScholar&xid=4e2762b0>

[20] Hossin, M., & Sulaiman, M. N. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. International Journal of Data Mining & Knowledge Management Process , 5(2).

[21] Craik, A., He, Y., & Contreras-Vidal, J. L. (2019). Deep learning for electroencephalogram (EEG) classification tasks: A Review. Journal of Neural Engineering, 16(3), 031001. <https://doi.org/10.1088/1741-2552/ab0ab5>

[22] Mudgal, S. K., Sharma, S. K., Chaturvedi, J., & Sharma, A. (2020). Brain Computer Interface Advancement in Neurosciences: Applications and Issues. Interdisciplinary Neurosurgery, 20, 100694. <https://doi.org/10.1016/j.inat.2020.100694>

[23] Douibi, K., Le Bars, S., Lemontey, A., Nag, L., Balp, R., & Breda, G. (2021). Toward EEG-based BCI applications for Industry 4.0: Challenges and possible applications. Frontiers in Human Neuroscience, 15. <https://doi.org/10.3389/fnhum.2021.705064>

[24] Mudgal, S. K., Sharma, S. K., Chaturvedi, J., & Sharma, A. (2020b, June 20). Brain computer interface advancement in neurosciences: Applications and issues. Science Direct. <https://www.sciencedirect.com/science/article/pii/S2214751920300098>

[25] Janapati, R., Dalal, V., Kumar, G. M., Anuradha, P., & Shekar, P. V. (2022). Web interface applications controllers used by autonomous EEG-BCI Technologies. INTERNATIONAL CONFERENCE ON RESEARCH IN SCIENCES, ENGINEERING & TECHNOLOGY. <https://doi.org/10.1063/5.0081780>

[26] Li, Y., Zhu, Z., Kong, D., Han, H., & Zhao, Y. (2019). EA-LSTM: Evolutionary attention-based LSTM for time series prediction. Knowledge-Based Systems, 181, 104785. <https://doi.org/10.1016/j.knosys.2019.05.028>

[27] Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016, November). Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the 2016 conference on empirical methods in natural language processing (pp. 606-615).

[28] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.

**APPENDICES**

**Base Code (Same for all models):**

from tensorflow import keras

import tensorflow as tf

import pandas as pd

import numpy as np

from keras import layers

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

df = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/Final Project/all\_subjects.csv')

df.head()

X = df.iloc[:,:16]

y = df.iloc[:,16]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

*{Build Model Here} # Replace this section with the model to be built*

opt=tf.keras.optimizers.Nadam(

learning\_rate=0.0005,

name="Nadam"

)

model.compile(optimizer=opt, loss = 'sparse\_categorical\_crossentropy', metrics=['accuracy', 'mse'])

model.summary()

model.fit(X\_train, y\_train, epochs = 15)

y\_pred = model.predict(X\_test)

outputs = []

for row in y\_pred:

outputs.append(np.where(row == max(row))[0][0])

print(classification\_report(outputs, y\_test))

**Vanilla LSTM:**

model = keras.Sequential()

model.add(layers.LSTM(128, input\_shape=(X\_train.shape[1],1)))

model.add(layers.Dense(7, activation='softmax'))

**Stacked LSTM:**

model = keras.Sequential()

model.add(layers.LSTM(128, input\_shape=(X\_train.shape[1],1), return\_sequences = True))

model.add(layers.LSTM(128, return\_sequences = True))

model.add(layers.LSTM(128, return\_sequences = True))

model.add(layers.LSTM(128))

model.add(layers.Dense(7, activation='softmax'))

**Bidirectional 1-Layer:**

from tensorflow.keras.layers import Bidirectional

model = keras.Sequential()

layer1 = layers.LSTM(256, return\_sequences=True)

layer2 = layers.LSTM(256,return\_sequences=True,

go\_backwards=True)

model.add(layers.Bidirectional(layer1, backward\_layer=layer2, input\_shape=(X\_train.shape[1],1)))

model.add(layers.Flatten())

model.add(layers.Dense(7, activation='softmax'))

**Bidirectional 2-Layer:**

from tensorflow.keras.layers import Bidirectional

model = keras.Sequential()

layer1 = layers.LSTM(256, return\_sequences=True)

layer2 = layers.LSTM(256,return\_sequences=True,

go\_backwards=True)

layer3 = layers.LSTM(256, return\_sequences=True)

layer4 = layers.LSTM(256,return\_sequences=True,

go\_backwards=True)

model.add(layers.Bidirectional(layer1, backward\_layer=layer2, input\_shape=(X\_train.shape[1],1)))

model.add(layers.Bidirectional(layer3, backward\_layer=layer4))

model.add(layers.Flatten())

model.add(layers.Dense(7, activation='softmax'))

**Vanilla LSTM with Attention:**

import keras.backend as K

class attention(Layer):

def \_\_init\_\_(self,\*\*kwargs):

super(attention,self).\_\_init\_\_(\*\*kwargs)

def build(self,input\_shape):

self.W=self.add\_weight(name='attention\_weight', shape=(input\_shape[-1],1),

initializer='random\_normal', trainable=True)

self.b=self.add\_weight(name='attention\_bias', shape=(input\_shape[1],1),

initializer='zeros', trainable=True)

super(attention, self).build(input\_shape)

def call(self,x):

# Alignment scores. Pass them through tanh function

e = K.tanh(K.dot(x,self.W)+self.b)

# Remove dimension of size 1

e = K.squeeze(e, axis=-1)

# Compute the weights

alpha = K.softmax(e)

# Reshape to tensorFlow format

alpha = K.expand\_dims(alpha, axis=-1)

# Compute the context vector

context = x \* alpha

context = K.sum(context, axis=1)

return context

a = Input(shape=(X\_train.shape[1],1))

lstm= layers.LSTM(256, return\_sequences=True)(a)

attention\_layer = attention()(lstm)

outputs = Dense(7, activation='softmax')(attention\_layer)

model = Model(a, outputs)

**Stacked LSTM with Attention:**

import keras.backend as K

class attention(Layer):

def \_\_init\_\_(self,\*\*kwargs):

super(attention,self).\_\_init\_\_(\*\*kwargs)

def build(self,input\_shape):

self.W=self.add\_weight(name='attention\_weight', shape=(input\_shape[-1],1),

initializer='random\_normal', trainable=True)

self.b=self.add\_weight(name='attention\_bias', shape=(input\_shape[1],1),

initializer='zeros', trainable=True)

super(attention, self).build(input\_shape)

def call(self,x):

# Alignment scores. Pass them through tanh function

e = K.tanh(K.dot(x,self.W)+self.b)

# Remove dimension of size 1

e = K.squeeze(e, axis=-1)

# Compute the weights

alpha = K.softmax(e)

# Reshape to tensorFlow format

alpha = K.expand\_dims(alpha, axis=-1)

# Compute the context vector

context = x \* alpha

context = K.sum(context, axis=1)

return context

a = Input(shape=(X\_train.shape[1],1))

lstm= layers.LSTM(256, return\_sequences=True)(a)

attention\_layer = attention()(lstm)

outputs = Dense(7, activation='softmax')(attention\_layer)

model = Model(a, outputs)

a = Input(shape=(X\_train.shape[1],1))

lstm\_1 = layers.LSTM(256, return\_sequences=True)(a)

lstm\_2 = layers.LSTM(256, return\_sequences=True)(lstm\_1)

lstm\_3 = layers.LSTM(256, return\_sequences=True)(lstm\_2)

lstm\_4 = layers.LSTM(256, return\_sequences=True)(lstm\_3)

attention\_layer = attention()(lstm\_4)

outputs = Dense(7, activation='softmax')(attention\_layer)

model = Model(a, outputs)