Performance Analysis and Hyperparameter Optimization of Stacked Bi-LSTM for Emotion Recognition from Facial Expression

Epsita Bose
MASTER of SCIENCE IN DATA SCIENCE STUDENT ID: 1060770
SUPERVISOR: ABUL ABBAS BARBHUIYA

Final Thesis Report

MAY 2023

DEDICATION

Dedicated to my parents, sister, husband,

For their endless love, unwavering support, and sacrifices they made to provide me with the best education. Their belief in my abilities and their constant encouragement has been my driving force throughout this journey. I am grateful for their guidance, patience, and understanding during these challenging times. This thesis is dedicated to them, as a token of my deepest appreciation and love.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my supervisor, Mr. Abul Abbas Barbhuiya, for their exceptional guidance, invaluable insights, and unwavering support throughout the entire research process. Their expertise and mentorship have been instrumental in shaping the direction of this thesis.

I would like to extend my thanks to Upgrade Team, Mr. Swagath Mirle Sundaresh for their collaborative efforts, insightful discussions, and technical assistance. Their contributions and expertise have been invaluable in conducting experiments, analyzing data, and interpreting results.

I am indebted to Liverpool John Moores University for providing access to resources, facilities, and libraries necessary for the successful completion of this research. The academic environment and support offered by the institution have been instrumental in fostering my intellectual growth.

I am sincerely thankful to my friends and family for their consistent support, encouragement, and understanding throughout this endeavor. Their unwavering belief in my capabilities and continuous motivation have been a source of strength during difficult periods.

Finally, I would like to extend my heartfelt gratitude to all the participants who graciously dedicated their time and shared their valuable insights for this study. Their participation was crucial, and without their contribution, this research would not have been feasible..

In conclusion, I am grateful to all those who have contributed to this thesis in various capacities. Your support, guidance, and encouragement have been invaluable and have shaped my academic and personal growth

TABLE OF CONTENTS

ABSTRACT

Emotion detection from facial expressions using deep learning techniques has gained significant attention in recent years. This study explores the effectiveness of stacked bidirectional Long Short-Term Memory (LSTM) networks for real-time emotion detection and investigates the impact of various design choices and hyperparameters on the model's accuracy and reliability. The study aims to develop an accurate and efficient deep learning model that can accurately identify emotions from facial expressions in real-world scenarios.

The research begins with a comprehensive literature review, discussing different approaches to emotion detection, including traditional machine learning methods and deep learning techniques. The use of stacked LSTMs is highlighted for its ability to capture temporal dependencies and complex patterns in facial expressions. The proposed research methodology covers data collection, preprocessing, feature extraction, and model development using stacked bi-LSTM networks.

The study addresses the need for a robust and accurate method for emotion detection using facial expressions, which can have applications in affective computing, human-robot interaction, and gaming. A large dataset consisting of 35.9k files is utilized to evaluate the model's performance on a larger scale. Additionally, the study focuses on hyperparameter optimization using GridSearchCV to enhance the model's accuracy and efficiency.

The significance of this study lies in its contributions to various fields. Psychology can benefit from a better understanding of emotions and their manifestation in facial expressions. Human-computer interaction can be improved by developing virtual assistants that respond to users' emotional states. Marketing research can leverage emotion detection to gauge consumers' emotional responses to advertisements and products. The study also has implications for security systems and the entertainment industry.

The structure of the study involves a literature review, research methodology, analysis, results and discussion, and conclusion. The findings highlight the effectiveness of the proposed stacked bi-LSTM model in accurately detecting emotions from facial expressions and provide insights into its strengths and limitations. The study's conclusion emphasizes its contribution to emotion detection research and suggests future directions for improvement.

Keywords: emotion detection, facial expressions, stacked bidirectional LSTM, hyperparameter tuning, GridSearchCV, deep learning, affective computing, human-computer interaction.

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI ……..…….....	Artificial Intelligence
APIS ……..……..	Application Programming Interface
Arima ……..…….	Autoregressive Integrated Moving Average
AUC ……..…….....	Area Under The Roc Curve
Avg ……..…….....	Average
Bi-LSTM ……..…….	Bidirectional Long-Short Term Memory
BP4D ……..……..	Binghamton-Pittsburgh 3d Dynamic Spontaneous Facial Expression Database
BPTT……..……..	Backpropagation Through Time
CK+ ……..…….....	Extended Cohn-Kanade Dataset
CNN ……..…….....	Convolutional Neural Network
CRF ……..…….....	Conditional Random Fields
ECG ……..…….....	Electrocardiography
EDA ……..…….....	Exploratory Data Analysis
EEG ……..…….....	Electroencephalography
F1 ……..…….....	F-Measure
FER2013 ……..…….	Facial Expression Recognition 2013
FS-CNN ……..……..	Face-Sensitive Convolutional Neural Network
IOT ……..…….....	Internet Of Things
JAFFE ……..…….	Japanese Female Facial Expression
KNN ……..…….....	K-Nearest Neighbors
LSTM ……..……..	Long Short-Term Memory
MAE ……..…….....	Mean Absolute Error
MAPE ……..…….	Mean Absolute Percentage Error
MLP ……..…….....	Multilayer Perceptrons
PIL ……..…….....	Python Imaging Library
QC ……..…….....	Quality Check
RAVDESS ……..……..	Ryerson Audio-Visual Database Of Emotional Speech And Song
RELU ……..……..	Rectified Linear Unit
RF ……..…….....	Random Forest
RMSE ……..……..	Root Mean Square Error

RNN ……..……....	Recurrent Neural Network
ROC ……..……....	Receiver Operating Characteristic
SGD ……..……....	Stochastic Gradient Descent
SMOTE ……..……	Synthetic Minority Oversampling
STC-NLSTM ……...…..	Spatio-Temporal Convolutional Features With Nested Long Short Term Memory
SVM ……..……....	Support Vector Machine
Uni-LSTM ……..……	Unidirectional Long Short-Term Memory
VGG ……..……....	Visual Geometry Group
WOA ……..……....	Whale Optimization Algorithm

CHAPTER 1: INTRODUCTION

1.1 Background of Study

In today's age, technology is merging with the human brain, and artificial intelligence serves as a means to simulate artificial brain activities. While the study of emotions dates back to the 19th century, Darwin's 1872 publication "The Expression of the Emotions in Man and Animals" applied evolution and natural selection to understand human communication. Emotions and technology are the driving forces behind the artificial human brain, and the ability to read and identify facial emotions has become effortless through deep learning techniques.

Emotion detection by facial expression using stacked LSTM is a technique used to analyze human emotions based on their facial expressions. Facial expression analysis is an important aspect of emotion detection as facial expressions convey rich emotional information. This technology is widely used in fields such as psychology, healthcare, and human-computer interaction.

LSTM networks belong to the category of recurrent neural networks (RNNs) specifically designed for processing sequential data, including time-series data. They are particularly useful for modeling long-term dependencies in data, making them ideal for processing sequential data such as facial expressions. Stacked bidirectional LSTMs is a variant of LSTMs where multiple LSTM layers are stacked on top of each other to improve the model's performance.

The emotion detection process using stacked bi-LSTMs involves several steps. Firstly, facial expression images are preprocessed to extract features such as facial landmarks and facial action units. These features are then fed into the stacked LSTMs to generate a sequence of emotional states. Finally, the emotional states are classified into discrete emotional categories such as happy, sad, angry, or neutral using a classification algorithm.

This technique has shown promising results in detecting human emotions accurately. However, it is still a developing technology, and there are several challenges that need to be addressed, such as handling individual differences in facial expressions and improving the robustness of the system to varying lighting conditions, facial occlusions and the selection of best model. Optimizing the correct hyperparameters is one of the major challenges while selecting the model. GridSearchCV plays an important role in optimizing the hyperparameters of the stacked bi-LSTM model used for emotion detection by facial expression.

1.2 Problem Statement

Web technology and artificial intelligence are growing industries, and the communication between humans and AI proves modern society. Interaction between computers and humans in widely increased in every sector. In this digital world, emotion recognition is a crucial thing in recent days, especially in digital media.

Many techniques, and algorithm have already been established to detect human emotions and also there are much research work going on to perform the analysis with high accuracy and efficacy.

Emotion detection can easily be identifiable by speech, text data, and physical activity, however, it's quite challenging when it's predicted from human facial expressions. Many machine learning and deep learning algorithm like CNN, RNN, LSTM, SVM, etc already been established to detect human emotion by facial expression.

Due to the inconsistency, variable, and irregularity behaviours of facial expressions, accurate identification is challenging if it's not shown properly. In this study, Stacked bidirectional LSTM is used to detect emotion detection.

As stated earlier, many deep learning algorithms are already analysed in various databases like CK+, RAVDESS, JAFFE, FER2013, etc. The previously mentioned algorithm works well in the small datasets. In this study, a dataset is used which is collected from Kaggel by (OHEIX JONATHAN, n.d.) It's a large dataset containing 35.9k files.

This study will conduct to evaluate whether the stacked bidirectional LSTM performed well in the larger datasets in real-time basic or not.

Also, this study will show how to select the best hyperparameter by GridSearchCV. The hyperparameters, such as row_hidden, col_hidden, batch_size, and epochs, significantly impact the model's performance. The grid search technique systematically searches the hyperparameter space to find the best combination of hyperparameters that maximize the model's performance. By tuning the hyperparameters using GridSearchCV, can improve the accuracy of the emotion detection model, which is crucial for real-world applications such as healthcare and human-computer interaction. Thus, GridSearchCV serves as an essential tool in developing and improving emotion detection models based on stacked LSTMs.

On the other hand, in other studies like (Begaj et al., 2020), its observed, fear, surprise and sad emotions are quite challenging to identify. The problem addressed in this paper is emotion detection using facial expressions, which is a challenging task due to the complexity and variability of human emotions. The proposed solution is to use a stacked LSTM network to capture temporal dependencies and complex patterns in facial expressions, with the goal of

achieving high accuracy in predicting emotions. The problem statement includes the need for a robust and accurate method for emotion detection using facial expressions, which can be applied in various domains such as affective computing, human-robot interaction, and gaming.

1.3 Aim of the Study

The aim of this proposed study is to increase the prediction accuracy in detecting emotion from facial expressions by stacked bidirectional LSTM with the selection of proper hyperparameters with the help of GridSearchCV and analyze how it will work in a large dataset in terms of predicting time.

1.4 Objective of the Study

The main objective of studying emotion detection from facial expressions using a stacked bidirectional Long Short-Term Memory (LSTM) model is to develop an accurate and effective deep learning model for detecting emotions from facial expressions. The model should be able to accurately determine the emotional state of an individual by analyzing their facial expressions in real-time or near-real-time scenarios.

Some specific objectives of such a study could include

- Designing and developing a stacked bi-LSTM model for emotion detection from facial expressions that can learn and recognize patterns in the data and accurately classify different emotions.
- Training the model on a large dataset of facial expressions that covers a range of emotions and expressions, to ensure that it is able to generalize well to new data.
- Optimizing the model architecture
- Optimizing the hyperparameters to achieve the highest possible accuracy and efficiency by use of GridSearchCV
- Evaluating the performance of the model on a test set of facial expressions, and comparing its performance to other existing emotion recognition models.
- Analyzing the model's behavior to gain insights into how it recognizes different emotions and which features of the facial expressions are most important for accurate recognition.
- Applying the model to real-world scenarios, such as monitoring the emotional state of individuals in social media posts or video chats.

Overall, the objective of studying emotion detection from facial expressions using a stacked bi-directional LSTM model is to develop a robust and effective deep learning model that can

accurately detect emotions in real-time or near-real-time scenarios, and that can be applied in a wide range of practical applications.

1.5 Research Questionary

This research question aims to explore the potential of using stacked bidirectional LSTMs for real-time emotion detection from facial expressions and investigate the impact of different design choices and hyperparameters on the model's accuracy and reliability. By investigating this inquiry, researchers have the opportunity to make valuable contributions to the advancement of emotion recognition systems. These advancements hold potential applications in diverse fields including psychology, human-computer interaction, and affective computing, enabling the development of more precise and efficient systems**.**

This research question aims

- To investigate the effectiveness of stacked bi-LSTM models for emotion detection in different scenarios, such as different input modalities (image), different dataset sizes (large), and different pre-processing techniques (e.g., data pre-processing etc).
- What is the effectiveness of using a stacked bidirectional LSTM network for emotion detection from facial expressions in comparison to other deep learning models?
- How to select the best hyperparameter with the use of GridSearchCV?
- How does varying the number of stacked layers affect the performance of the model?
- Can the model effectively detect and classify emotions from facial expressions with higher accuracy compared to traditional machine learning algorithms and stacked bi-LSTM models
- Can stacked bi-directional LSTMs effectively learn and classify emotions from facial expressions in real-time, and how does the architecture and training parameters affect the performance and robustness of the model?
- By exploring the above factors, the study can identify the strengths and limitations of stacked bi-directional LSTM models in emotion detection, and provide insights into the optimal design of stacked bi-directional LSTM models for this task.

1.6 Scope of the study

The scope of study of emotion detection from facial expressions by stacked bi-directional LSTM includes exploring and developing machine learning models that can accurately detect emotions from facial expressions. This involves:

- Data collection and pre-processing: Gathering a dataset of facial expressions labeled with corresponding emotions, and pre-processing the data to ensure it is suitable for deep learning models.
- Feature extraction: Extracting meaningful features from the facial expression data that can be used as inputs to the machine learning models. This can include features such as the position and movement of specific facial landmarks, as well as more complex features derived from the raw pixel data.
- Model development: Developing a machine learning model that can accurately classify facial expressions into different emotions. The use of stacked bi-LSTM is a common approach for this task, as it allows the model to capture the temporal dependencies in the data.
- Model training and evaluation: Training the machine learning model on the labeled dataset and evaluating its performance on a held-out set of data. This involves selecting appropriate hyperparameters and tuning the model to achieve the best possible performance.
- Application and deployment: Finally, the developed model can be used in various applications such as sentiment analysis in social media, mental health assessment, or even in robotics to make machines more intuitive and responsive to human emotions.
- Overall, the study of emotion detection from facial expressions using stacked bi-LSTM has wide-ranging potential applications in various industries, such as healthcare, education, entertainment, and social media.

1.7 Significance of Study

The study of emotion detection from facial expressions using stacked bi-directional LSTM has significant importance in several fields. Here are some of the significances:

- Psychology: Understanding emotions and facial expressions is essential in psychology, and this research can help psychologists and researchers in understanding how different emotions are reflected in facial expressions. This can help in understanding mental health conditions such as anxiety, depression, and other emotional disorders.
- Human-Computer Interaction: Emotion detection from facial expressions can help in improving human-computer interaction. For instance, it can help in designing virtual assistants that can understand and respond to the user's emotional state, resulting in better user experience and engagement.

- Marketing: Emotion detection can be used in marketing research to understand how consumers react emotionally to advertisements, products, or services. This information can be used to improve the marketing strategy and product design.
- Security: The study of emotion detection can help in developing advanced security systems that can detect suspicious activities and identify individuals based on their emotional state. This can help in improving public safety and security.
- Entertainment Industry: Emotion detection can be used to create more immersive experiences in the entertainment industry, such as gaming or virtual reality, by adapting the game or the environment to the user's emotional state.
- Advancements in AI and Deep Learning: The use of stacked bi-directional LSTM in emotion detection from facial expressions is a significant advancement in AI and deep learning. It shows how complex models can be used to accurately classify emotions from visual data, which has numerous applications beyond emotion detection.

Overall, the study of emotion detection from facial expressions using stacked bi-directional LSTM has significant implications in several fields, including psychology, human-computer interaction, marketing, security, and entertainment.

1.8 Structure of the study

The study of emotion detection from facial expressions using stacked bi-directional LSTM can be structured as follows:

- Literature Review: Chapter 2 will discuss about review of previous works related to the topic and identify the gaps in the literature. Discuss different approaches to emotion detection from facial expressions, such as traditional machine learning methods and deep learning methods. Focus on the use of stacked LSTMs in emotion detection and highlight their advantages over other deep learning methods.
- Research Methodology: Chapter 3 will explain the pre-processing steps taken to clean and prepare the data for training the model.
- Analysis: Chapter 4 will explain the proposed method for emotion detection using stacked Bi-LSTMs. Discuss the architecture of the model, including the number of layers and neurons used in each layer. Describe the training process, including the optimization algorithm used and the how hyperparameters selected.
- Results and Discussion: Chapter 5 will present the results of the study, including the accuracy and performance metrics achieved by the proposed method. Compare the results with previous works and demonstrate the effectiveness of the proposed method

and also discuss the implications of the results and the limitations of the proposed method. Analyze the strengths and weaknesses of the method and suggest potential directions for future research.

- Conclusion: chapter 6 will summarize the key findings of the study and highlight the contribution to the field of emotion detection from facial expressions. Emphasize the significance of the proposed method and the potential for its use in practical applications.
- References: Provide a list of all the sources cited in the study, following the appropriate citation style.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Previous research in the field of emotion detection using facial expressions has mostly utilized shallow models, which fail to capture the complexity of human emotions. In contrast, deep learning models such as LSTM have shown promise in capturing intricate patterns and temporal dependencies in sequential data. As a result, the present study proposes the use of stacked bi-directional LSTM for emotion detection through facial expressions. This chapter presents an exploration of the effectiveness of stacked bi-directional LSTM in identifying emotions using facial expression recognition. It summarizes the characteristics of stacked LSTM and its various applications. Additionally, this chapter provides an overview of the theories that have been proposed for using deep learning algorithms to detect emotions through facial recognition, along with their respective advantages and disadvantages.

2.2 Stacked LSTM in Healthcare

This section describes a few examples of how Stacked LSTM works in the healthcare sector.

2.2.1 Pseudo-tumor detection

(Dandıl & Karaca, 2021) this study proposed a stacked LSTM and Bi-Stacked LSTM model for detecting pseudo-tumor cells from a magnetic resonance spectroscopy signal. Also, the performance results of 1D-CNN were compared to the attained binary classification results obtained utilizing stacked LSTM and Bi-LSTM approaches. Various parameters of the stacked LSTM model are tested to validate the accuracy results, however, it shows higher results. This theory worked on a small number of samples. The accuracy performance will show broadly if a large number of samples will use. Experts employ a technique of incorporating additional layers to enhance image prediction. However, this approach often exacerbates the issue of vanishing gradients, resulting in a gradual decline in accuracy. To mitigate this problem, a skip connection technique is utilized to establish alternate shortcuts for the gradients to pass through. Additionally, the text explores the stacking mechanism, which involves leveraging predictions from multiple weak models to construct a new, stronger model. The stacked Bi-LSTM is employed to predict sequences by capturing contextual information from the input. The proposed deep learning framework encompasses a pre-trained Stacked Bi-LSTM model with Resnet50 and Adaswarm optimizer, utilizing binary cross-entropy loss function for prediction. The text comprehensively outlines the various components of the proposed framework,

spanning from the pre-processing stage to the fully connected layer. Furthermore, the SMOTE method is applied to address the issue of imbalanced image data through resampling.

2.2.2 Medical Image Classification

(Pattanaik et al., 2022) this paper emphasizes the criticality of accurate and timely detection of illnesses from medical images within the healthcare sector. It introduces a proposed model, the Stacked BiLSTM with Resnet50 Model, employing an AdaSwarm optimizer to classify and analyze medical illnesses using diverse medical image datasets. Four benchmark datasets, namely Covid19, Pneumonia, Ma, and Lung Cancer, were utilized for evaluation. Performance metrics such as accuracy, AUC, ROC, and F1 Score were employed to assess the proposed model. The hidden layer employed a specific activation function, while the output layer employed the sigmoid activation function for prediction. The paper thoroughly presents all components of the model, starting from the preprocessing stage to the fully connected layer. Initially, images of size 224 X 224 were processed and normalized to ensure consistent sizing. Data augmentation techniques, including shear range and rotational range parameters, were employed for image transformation. As the image data exhibited imbalance, the SMOTE method was utilized for resampling. The resampled data were split into training and validation sets for model validation. Subsequently, the split images were inputted into the proposed framework, consisting of the Stacked Bi-LSTM model with Resnet50 and an AdaSwarm optimizer. The preprocessed images were first fed into the Resnet50 model, and the resulting outputs served as inputs to the Stacked Bi-LSTM model with the Adaswarm optimizer. The input and output were combined, and the sigmoid function was utilized for prediction.

2.2.4 Prediction of cardiovascular disease

(Bandyopadhyay, 2020) described a stacked LSTM model which helps in the prediction of cardiac disease with the help of patient's medical records. This study explained how interfering factors (such as patient's age, blood pressure, type of chest pain, ECG analysis, heart rate analysis, etc) determine the tendency of having a cardiac disorder. The architecture of the model consists of 4 stacks of bi-directional LSTM RNN layer and 4 dense layers. The layers are activated by the sigmoid activation function. Drop-out layers are incorporated to lower the overfitting. The proposed study successfully gives optimum accuracy for predicting analysis.

2.2.5 To identify mental arithmetic task classification from EEG signal.

EEG signal is the most crucial tools for extracting brain signals. This (Ganguly et al., 2020) study proposed a model where two LSTM are Stacked for mental arithmetic task classification from motor imagery. EEG signals from the dataset are introduced as input, window segmentation, and feature extraction followed by scaling were performed. Two vertically Stacked LSTM layers have been used in this model which is incorporated with dropout layers. The last dropout layer is hooked up to an entirely connected layer with 2 nodes and softmax layer which is implemented in this completely linked layer to get the classification range in 0 and 1. In the end, the study showed positive analysis in identifying mental arithmetic tasks.

2.3 In Automobile and mechanical industries

This section describes a few examples of how Stacked LSTM works in the automobile industry.

2.3.1 Signal labeling and precise location in a variable parameter milling process

This (Qiu et al., 2023) paper proposes a stacked BiLSTM-CRF model for the time series signal labeling task. The model consists of three parts: stacked BiLSTM, CRF, and MLP. The stacked BiLSTM performs deep feature extraction of the signal, the CRF labels the signal automatically, and the MLP predicts the probability that each frame is a boundary. The paper also describes the loss function for the model, which includes the loss of the CRF model, the boundary loss, and a smoothing loss. The stacked BiLSTM consists of a three-layer BiLSTM and a fully connected layer. Finally, the output features of the last fully connected layer are input into the CRF and MLP models for signal labeling and interception. The proposed model is evaluated using the vibration signal labeling task and achieves state-of-the-art results.

2.3.2 Short-term traffic flow prediction based on whale optimization algorithm

This article (Xu et al., 2022) proposes A traffic flow prediction model is proposed in the article, which combines the BiLSTM_Attention structure with the whale optimization algorithm (WOA) for optimization purposes. The model's performance is compared with a conventional neural network model and a neural network model optimized by the WOA. To evaluate the models, metrics such as MAPE, RMSE, MAE, and R2 are utilized, with the WOA_BiLSTM_Attention model demonstrating superior performance. The article highlights the importance of accurate traffic flow prediction for intelligent decision-making and warning systems, emphasizing how the proposed model addresses the challenges associated with training conventional networks to obtain optimal parameters and structure.

The authors describe the utilization of the WOA optimization algorithm to improve the performance of the BiLSTM_A model in short-term traffic flow prediction. The WOA algorithm optimizes the mean square error between the network output and actual values, serving as the fitness function to determine the best set of parameters for the BiLSTM_A model. The WOA-BiLSTM_A model comprises three components: WOA, BiLSTM_A, and data. The WOA algorithm's population is initialized with four variables representing the number of iterations, learning rate, and the number of nodes in the hidden layers of the BiLSTM network. The best solution within the population is updated based on the fitness value obtained from training the BiLSTM_A model. The article explains the prediction process of the WOA-BiLSTM_A model, highlighting the effectiveness of WOA optimization in addressing the issue of converging to local optimal solutions in the BiLSTM_A algorithm, thus improving the accuracy of parameter optimization.

2.3.3 Prediction of performance of radiotherapy machine

Prediction of the performance of any machine is most crucial in terms of legibility and productivity of the machine, especially in the healthcare sector as it helps in diagnosis of disease. This study (Ma et al., 2022) evaluated the QC data of radiotherapy machine is used to predict the performance of the machine. Here stacked LSTM performace is compared with ARIMA Model to check the accuracy however, Stacked LSTM gives higher accuracy.

2.3.4 Analysis of the performance of sensor to check the quality of Wafer in the Wafer industry.

The sensor is the heart of any machine. Quantity and quality are two crucial parameters for any type of machine. (Shinde et al., 2021) the study predicts the performance of a sensor for determining the quality of the Wafer. The stacked LSTM model is introduced on soft sensor data from the industry. Application of Multitask Regularization Learning promotes multiple predictors and various tasks for sharing parameter analysis models. Thus, when the deficit is rounded, the regularized question is rounded and confesses a globally optimum answer. The score of ROC-AUC is better than different criterion model.

2.3.5 Fault diagnosis.

Security is most essential for any system for that reason fault detection is the first approach of any industry including chemical system. In a chemical process, there are various methods are

already established to overcome this however, many of them did not follow temporal relationship in sequential monitoring signal of the chemical process. This can be achieved with the help of Stacked LSTM which is explained in this study along with the detection of abnormal values. Three layers of stacked LSTM are used for detecting a fault on a real-time basis which is applied to Tennessee Eastman dataset. This model (Zhang et al., 2020)makes a temporal relationship with the observational signals at various time and automatically extract the representative feature. This method showed true prediction however it can easily detect a single fault, the multi fault is not achievable by this method.

2.4 Stacked LSTM in price detecting

This section describes a few examples of how Stacked LSTM works predict the price in various sector.

2.4.1 Spot price detecting

Spot price is used in every industry, which solely depends on product, region, bidding time, number of bidders, and capacity free or not. In this study, (Chittora & Gupta, 2020)the spot price is predicted by stacked LSTM, by using 3 months of Amazon dataset. 2 Stacked LSTM layers with 4 memory cells are implanted in the proposed model. In stacked LSTM depth of network is more important than the number of memory cells to predict analysis. The accuracy of the model performance is calculated by RMSE and MAPE and its compared with RNN model which is already developed in the past. However, Stacked LSTM showed optimum results. Overfitting is observed when large amount of data will use here, this is the major drawback of this model.

2.4.2 Detecting stock market price prediction

Understanding a fairly perceptive trend of available prices is considerably main in the concerned society and shareholders to underrate risks on the grant. Future forecasting is more challenging as the behaviour of the stock market is volatile, non-linear, and unpredictable, which is not possible by human beings. Many machine learning algorithms are already established to predict the stock market price significantly. This article(Uddin et al., 2022) is explained how stacked LSTM can predict the price. Here 3 layer of multivariate stacked LSTM architecture is used to predict open, high, and low prices over a time period and univariate LSTM architecture is used to predict close value based on the last close value. The performance

of stacked LSTM model is compared with ARIMA and linear regression model, the studied model gives higher accuracy of result.

2.4.3 Detecting stock price prediction

Overcome with characteristics of volatility is the main challenging thing in stock market price prediction. This (Lim et al., 2021)paper proposed a stacked bidirectional LSTM for price detection by using 6 types of different dataset of Yahoo finance data. The architecture of stacked LSTM makes both backward and forward directional to obtain the information more efficiently and learn the summarised stock price value past and future. The model includes 2 bidirectional LSTM networks, 1 dropout layer followed by again layer of bidirectional LSTM networks, 1 dropout layer, 1 dense, and 1 prediction layer. This model predicts closer stock price in compare to actual stock price.

2.5. Stacked LSTM is used in a different type of prediction in daily human life.

Our daily life can be easy in terms of detecting predictions of various things by using stacked LSTM. Here, some examples are discussed.

2.5.1 Melody classification.

Music is one of the soothing therapy in human daily life. The choice of music depends on various factors like human nature, present situation, mood, place, etc. in this study (Y. Li & Lin, 2020), stacked binary LSTM classifier model which is based on language is proposed to predict the human composer's task from a machine-generated melody by learning the pitch, duration, and position from MIDI files. In short term, this model will help in the detection of AI-generated music from human-composed music. The model has 2 LSTM layers, 1st layer having 64 units and 2nd layer having 8 units, there are two dropout layers for avoiding overfitting and at last, a fully connected layer to obtain the final classification. The significance of this study is, from this model network learning rate from abstract c global features from the sequence is high. However, more study will do for make this procedure more faster and in large diversity.

2.5.2 Detection of sarcasm.

There are various factors from which sentiment analysis can be studied. Sarcasm is one of them. Its difficult to understand the inner meaning of sarcasm and its representation with negative thoughts by positive words. In this social media world, human sentiment in terms of sarcasm is

necessary to detect negative or positive ways of communication. In this study (Dr. M. Kumar & Patidar, 2021), a stacked bi-directional LSTM network is used to detect sarcasm in a binary classification, which means whether the statement has a touch of sarcasm or not. In the proposed study, preprocessing is the initial step which is done by some machine learning algorithms like regression, clustering, filling the missing value, etc. In the next step, word embedding is used for word analysis. Afterward, stacked bi-directional LSTM model is used with some machine learning library to detect the set of words that contains the sarcasm characteristics. And then compare the results with other model to check the performance analysis. The model is showed the highest accuracy, precision, recall and F-Score than other models like basic LSTM, bi-directional LSTM etc.

2.5.3 Detection of human activity from smart phone data.

In this digital world, by using smartphones we can ease our daily life. By using smartphones, 70% percent of daily activity can easily be achievable, and bank, school, automobile, shopping, tracking of various activities, etc can be achieved in a single second. Human activity can easily be recognizable by using a smartphone and it's significant to work for automatic human behavior analysis for older people, players, children, and IOT applications. In this article (Ullah et al., 2019) a Stacked LSTM model is introduced for predicting the 6 human activities like walking, sitting, walking upstairs or downstairs, lying as well as standing by sensor-based activity recognition. Here, an accelerometer and gyroscope are used as sensor data which is obtained from a smartphone (used by a human). At pre-processing step, this data is passed through a single layer of a neural network, Normalization is also done by using Relu activation and discriminant function. In next step, data is passed through a stacked LSTM network, which is having 5 LSTM layers, and the final output is given by six-way softwax which gives the probability of human behavior. The model shows a higher performance when compared to other models.

2.5.4 Prediction of electricity load forecasting.

Electricity is the heart of the modern economy, and its demands increasing day by day. Electricity is used in every micro-second of human life, in automobiles, house hold, healthcare, education systems, etc. Electricity load forecasting is a problem in the electric power system management process. Avoiding incorrect forecasts that could adversely influence system effectiveness, the economy, and sustainability requires an accurate forecasting model. There are several techniques already established to predict the accurate forecasting model. In this

study (Atef & Eltawil, 2020), a stacked unidirectional and bi-directional LSTM model is used on predicting hourly electricity load consumption. In this model, six models that differ in the number and kind of LSTM layers as well as varied hyperparameter values are used to evaluate various deep-stacked LSTM network designs. Then put in place a tool for hyperparameter tuning for each model and examine how the deep-stacked LSTM layer performs for both Uni-LSTM and Bi-LSTM. At the end, compare the most successful Uni-LSTM and Bi-LSTM models against the standard SVR model using the same dataset. Single layer Bi-LSTM model gives better results than others, however, the overall stacked LSTM layer did show better results when compared to SVR mode.

2.6 Related Work

This section will give an overview of the detection of emotion by facial expression with various machine learning algorithms.

2.6.1 Emotion detection by facial expression by deep learning technique (CNN).

Detection of emotion from non-verbal communication like facial expression, gestures, limb movement, etc is very challenging. In this study(M. Kumar & Srivastava, 2021), the facial expression is used for detecting six types of human emotions, like, sad, anger, happy, disgust, surprise, and fear by a deep learning algorithm that is CNN. Pre-processing of images, feature extraction, and classification are the pillar of the proposed model. In the initial step, pre-processing is done for making the better quality of the image by removing redundancy, and noise, also making equal sizes of images, etc. images are converted into an array of pixels, and then identify the images. After that feature extraction is done to carry forward only significant information about images. Images take as input in CNN classification and classified into various categories. The model showed significant accuracy in detecting emotions. There is a higher accuracy score for detecting anger, fear, and happiness. This is a real-time application for detecting emotions and passes output in form of text.

2.6.2 Facial emotion detection by deep learning (CNN).

Mental state is consisting of various factors like, feelings, behaviours etc. Artificial intelligence is capable to detect mental state by various algorithm. In this study, a AI based convolutional neural network (CNN) is proposed to detect human emotion by facial expression. In this study (Jaiswal et al., 2020), two datasets (Facial emotion recognition challenge (FERC-2013) and Japaness female facial emotion (JAFFE)) and 2 models are used one is to evaluate accuracy.

Model A consist of convolutional layers with 64 filters with size 3x3, proceeding with one local contrast normalization later, another convolutional layer, max pooling, flatten. In the next step two similar models are concatenated and linked to a softmax layer which gives an output of 7 emotions. 0.2 dropout is used to reduce overfitting to fully connected layers and all layers contained ReLu activation functions. Model B has 48x48 input layers and has almost same architecture as model A, except, model B consists of 2 extra convolutional layers. The performance of the model is compared with both models in the respect of both datasets. Models did not perform well for the FERC-2013 dataset, which contains higher samples, however, models showed very good accuracy for the JAFEE dataset.

2.6.3 Facial emotion detection by CNN.

This study (Babajee et al., 2020)proposed how to detect human emotions by facial expression by CNN technique. Facial detection, feature extraction and classification are the three main step of the study. This study was performed on a dataset with 32928 images. In a pre-processing step, facial detection is observed to identify the actual area of the faces. In the next step, shape, location, etc are extracted. In the last step, a probability classifier is used to detect the actual extracted facial expression. In this model, two layers of CNN layer are used, one is a convolutional layer where feature maps are produced by complex convolution kernels, and the second, is a subsampling layer, 2D images are taken as input and then pooling and redeployment are done. Also, 2 important perceptions of sharing weight and sparse connectivity are also present. In the end, the accuracy of the result compared with other established models showed 79.8% accuracy. It will give a drawback if it's applied a live detection. An optimization method should be applied to get better accuracy.

2.6.4 Face Recognition and Emotion Recognition from Facial Expression Using Deep Learning Neural Network.

In this study (Hussien Mary et al., 2020), proposed a CNN-based methodology for detecting various human sentiments. the architecture of the model described as: A pre-trained weight of VGG Face model used as an input. The CNN layers contain a convolution Layer which performs convolution over the input volume, 1 ReLu layer which performs as an activation function to reduce the linearity of the images, pooling layers that reduce the width and height of the input, and fully connected layers that connected the previous output layers. here, the softmax activation function is used who is normalize the output of neural networks to be situated between 0 and 1. This is the probability of the approached model. The model showed

92.81% accuracy. However, MSE will be reduced if there is an increase in training data. This algorithm can be used in human-machine interactions like robots.

2.6.5 Emotion Recognition from Facial Expression using CNN.

Seven emotions contempt, happiness, Sadness, fear, surprise, anger, and Neutral by using CNN model to get better training time and accuracy described in the study (Agrawal et al., 2021). A dataset containing 35k images is used here. The model architecture is based on five steps: 1. Image acquisition: gives input into the model, 2. Image pre-processing: customization of image done based on the model requirements to make the model perfect in terms of efficacy and accuracy, 3. Image segmentation: its extract most valuable data from the images and detect boundaries of images, 4. Feature extraction: training of CNN model, and 5. Emotion detection: train model gives predicted output. The CNN layers consist of convolutional layers, ReLu, pooling and fully connected layers. this model is capable to detect real-time emotions, however, there is a different accuracy for different groups of emotions, and also time-consuming to train the process.

2.6.6 A Novel based 3D facial expression detection by RNN.

In this article (K.S. & David, 2020), a RNN-based algorithm is used to detect emotion by facial expression. JAFFE and Yale's datasets are used here as input. Image Acquisition, 2D images convert into 3D images, Stemmer based feature extraction is done. After that, Adaboost technique is used to enhance the model performance, and classification is done by RNN feedforward and backpropagation algorithm. The model showed 92% probability detection.

2.6.7 Human emotion recognition based on facial expressions via deep learning on high-resolution images.

In this article (Said & Barr, 2021), a face-sensitive convolutional neural network (FS-CNN) is used to detect human emotion, initially it will detect the face and later will recognize the emotions. There are 3 main functions: face location, cropping of the image, analysis of facial expression, and emotion recognition. 2 datasets UMD Faces and Celeb A datasets are used here. The proposed model can detect emotions in high-resolution images, it gives 95% accuracy.

2.6.8 Emotion Detection Using Facial Expression Involving Occlusions and Tilt

This study (Qazi et al., 2022)proposed a CNN architecture model that helps in FER in the mixed dataset CK+ and JAFEE. The model has 6 CNN layers. Out of 6 5 are convolutional layers are

used, which include max pooling, and dense layer along with softmax and dropout functions. The unique nature of the model is, after face detection and cropping, pictures are vertically flipped and 2 pictures and 7 angles are formed from each picture and producing 14 images in the last step. This study was performed in 2 experiments: the combined dataset is used in the first experiment and in the second experiment cross dataset means one dataset is used for training and another for the test. In the end performance is evaluated in various epochs and hyperparameters. The model shows 92 and 94% average accuracy. The model is not suitable for dark-colored images or faces.

2.6.9 Optimum facial feature-based emotional recognition using Deep leaning Algorithm.

This article (Kumar Arora et al., 2022)will discuss how deep learning algorithms like CNN help in improving the detection of facial expressions. The dataset used in this study contains 32,298 images. The architecture of model is: face detection, pre-processing and feature extraction, then it images will go to the model and predict the emotions. 4 types of face recognition is used here: Hybrid, Geometric feature-based, Holistic feature-based approach and Template-based technique. The CNN layers do multilayer-specific work in the classification method. Initially input images will pass into the convolution and Relu layer and then the max pooling technique, the same method is repeated once again and then the output image goes to fully connected layers which helps in classifying the images. The model accuracy is evaluated by CNN and SVM models, however the proposed model showed higher accuracy with detecting 7 types of emotions.

2.6.10 Novel Deep leanring model for recognition of facial expression based on boosted CNN and LSTM.

There are many papers already established to detected facial emotions by CNN algorithm. However, in this paper (Rajan et al., 2020)a novel technique will discuss. CNN and LSTM are combined here to improve the detecting power of FER. A dual CNN base model is used here, but its not directly merged with LSTM model. The output of dual CNN base model gives spatial feature maps, then these feature maps are fused and in-corporate with LSTM. The model is detected with higher accuracy.

2.6.11 Spatio-temporal convolutional features with nested LSTM for facial expression recognition.

This article (Yu et al., 2018)explained a novel Spatio-Temporal Convolutional features with Nested LSTM (STC-NLSTM) in FER. The architecture of the proposed model includes a 3D CNN which is extracting spatiotemporal convolutional features and nested LSTM layer, which is coupled by temporal LSTM which is captured the temporal dynamics and convolutional LSTM layers used for seizing the multi-level features. There are 4 datasets used here, Oulu-CaSIA, CK+, BP4D AND MMI. The model showed higher accuracy when compared to other established model.

2.6.12 Facial Expression Recognition with CNN-LSTM.

This (Hung & Tien, 2021) paper proposed a method for FER by CNN which helps in feature extraction along with LSTM which recognizes the facial expression in JAFFE database. The CNN layers had total 6 layers, 2 convolutional, max pooling, and fully connected layers. the LSTM layers used here to make the model better in term of efficacy as it memorizes the output of the previous layer and forward it to the next layers. This hybrid model shows better accuracy when compared to the CNN model, however, there is a misclassification are occurred while detecting sadness and happiness.

2.6.13 An CNN-LSTM-based Model for Video Classification.

FER is easy approach as images are not moving however, it will be difficult in video form as images are simultaneously changes. This article (Abdullah et al., 2020)proposed a CNN-LSTM model for the classification of video in The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset. CNN model is used here in all frames to detect faces in video. Then crop the frame to detect the face. In next step, images are converted in similar size along with greyscale, then it will forward to CNN to fully connected layers. At fully connected layers emotional features are shown as output. These extracted layers are modified into sequences of features. Then this output will put into single layer of LSTM layers along with ground truth labels for training. Xception Net is used as feature extraction from frames of video. The model showed not good accuracy, however, this is the first benchmark for this dataset.

2.6.14 Emotion Recognition by Facial Features using Recurrent Neural Networks.

In this study (Mostafa et al., 2018)various machine learning models are used to recognize emotion by facial expressions from video clips. The architecture of RNN algorithm is: an LSTM algorithm with one hidden layer which contains 150 unidirectional LSTM cells is used here. It showed the highest performance in respect to other algorithms like RF, KNN, SVM.

2.6.15 Facial expression recognition using bidirectional LSTM-CNN.

CK+ dataset (Febrian et al., 2023) is used here for comparison performance between biLSTM-CNN, LSTM-CNN, and CNN models. Spatial relation is required at different facial regions and this is most important feature in facial detection. CNN sometimes fails to catch this relationship as its focused on only local regions of images. Bi-LSTM is used here to learn from the total time series at each time step, this factor improves the performance for identification of expression by providing long-term dependencies for spatial dependencies in images of facial expression. The bi-LSTM-CNN showed the highest accuracy when compared with the LSTM-CNN model. Later augmentation is also done to reduce overfitting, still, Bi-LSTM showed higher performance.

2.7 Summary

The section describes different studies related to emotion detection from facial expressions using deep learning techniques. The studies propose various CNN-based models for detecting human emotions by processing facial expressions. The models involve pre-processing of images, feature extraction, and classification. The accuracy of the models is compared with other established models and showed significant accuracy in detecting emotions. However, there is a difference in accuracy for different groups of emotions and time-consuming to train the process. The models can be used in human-machine interactions like robots.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

The research methodology of emotion detection from facial expression by stacked bi-directional LSTM involves a series of steps that aim to build an accurate and robust model for emotion detection. The methodology begins with collecting a dataset of facial expression images or videos along with their corresponding emotional labels. The images are then preprocessed to extract facial features, which are fed into a stacked bi-directional LSTM network for processing. To improve the accuracy and robustness of emotion detection from facial expressions using stacked bi-directional LSTM, it is essential to select the proper hyperparameters for the model. To achieve this, the proposed methodology includes the use of GridSearchCV, a popular hyperparameter tuning technique, before training the stacked bi-directional LSTM network. This approach involves systematically searching through a range of hyperparameter values to find the optimal combination that maximizes the model's performance. Once the optimal hyperparameters are selected, the methodology proceeds with the rest of the steps involving data collection, preprocessing, model training, and evaluation, as described earlier. The output of the stacked bi-directional LSTM network is used for emotion classification, which categorizes the emotions into discrete categories such as happy, sad, angry, or neutral.

The LSTM layers process the sequence of features, taking into account the temporal dependencies between them.

The next step involves training the stacked bi-directional LSTM network using the collected data to optimize the parameters of the network and minimize the error between the predicted and actual emotions. The trained model is then tested on a separate set of data to evaluate its performance using metrics such as accuracy, precision, recall, and F1 score.

Finally, the emotion classification is performed on the output of the stacked bi-directional LSTM network, which classifies the emotions into discrete categories such as happy, sad, angry, or neutral using a classification algorithm.

The research methodology of emotion detection from facial expression by stacked bi-directional LSTM is a promising approach that has shown significant improvements in accurately detecting human emotions based on their facial expressions. However, there are still challenges to address, such as individual differences in facial expressions and improving the robustness of the system to varying lighting conditions and facial occlusions. Incorporating GridSearchCV in the research methodology of emotion detection from facial expression by

stacked bi-directional LSTM can further enhance the accuracy and robustness of the model, leading to better performance in various applications.

Overall, this methodology provides a comprehensive framework for developing and evaluating emotion detection models using stacked LSTM, and further research in this area can have significant implications for various applications such as affective computing, human-robot interaction, and mental health monitoring.

This section will help to understand how Stacked LSTM is performed on the detection of facial expressions. The overall workflow is shown in Fig.1.

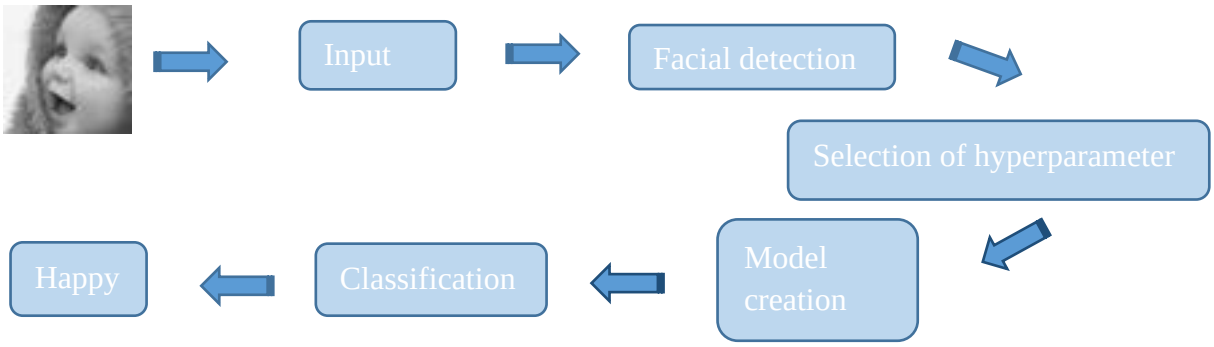


Figure 1 Overall Study Workflow

3.1.1 LSTM Theory

LSTM is a sequential neural network used in the deep learning method. LSTM, an acronym for Long Short-Term Memory, refers to a specific type of RNN architecture. It is specifically designed to address the challenge of vanishing gradients encountered in conventional RNNs. The vanishing gradient problem occurs when gradients become extremely small during backpropagation, which makes it difficult for the network to learn long-term dependencies.

LSTM networks address this problem by introducing a memory cell, which allows the network to selectively remember or forget information over long periods of time. The memory cell is composed of several gates, which control the flow of information into and out of the cell.

There are three types of gates in an LSTM network: the input gate, the forget gate, and the output gate. Each gate is implemented as a sigmoid function, which outputs a value between 0 and 1 that determines how much information should be passed through. The gates are trained using backpropagation through time, which allows the network to learn how to selectively store or discard information over time.

LSTM is not performed on a single data, it can process an entire sequence of data, and these characteristic help LSTM to predict any data. LSTM can be applicable handwriting(Graves et al., 2009), speech recognition(X. Li & Wu, 2014), video games, translation in the machine(Wu et al., 2016), detection of speech (Sahidullah et al., 2019), robot control (Mayer et al., 2006), health sector etc.

3.1.2 LSTM Architecture

LSTM networks introduce a memory cell that allows the network to selectively remember or forget information over long periods of time. The memory cell consists of multiple gates that regulate the movement of information both into and out of the cell.

Each LSTM cell has three types of gates:

Input gate: The input gate determines the extent to which new information should be added to the memory cell. It is controlled by a sigmoid function, which outputs a value between 0 and 1.

Forget gate: The forget gate determines the extent to which previous information should be retained or discarded from the memory cell. It is also controlled by a sigmoid function, which outputs a value between 0 and 1.

Output gate: The output gate determines the extent to which the current state of the memory cell should be used to produce the output. It is controlled by a sigmoid function and a hyperbolic tangent function, which output values between 0 and 1 and between -1 and 1, respectively.

The values output by the gates are multiplied with the input, previous memory, and output of the cell to determine the new state of the memory cell. During the forward pass, the LSTM cell receives an input and the previous state of the memory cell. The input is multiplied with the input gate, and the previous state of the memory cell is multiplied with the forget gate. These values are then added together to produce the new state of the memory cell. The new state is then multiplied with the output gate to produce the output of the LSTM cell.

The structure of LSTM shown below in Fig.2 (Mayer et al., 2006)

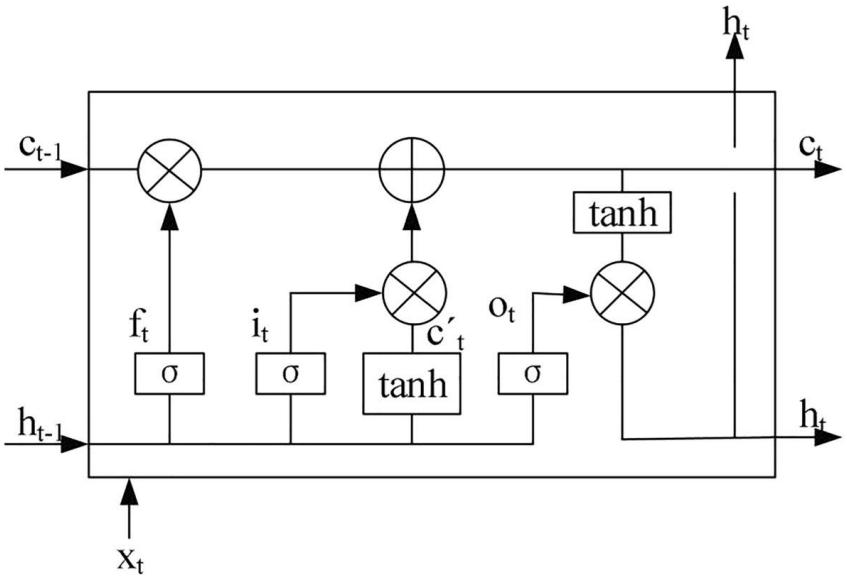


Figure 2 LSTM architecture

The formula of various gates shown below:

Forget Gate:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad (1)$$

Input Gate:

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \quad (2)$$

Output Gate:

$$O_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \quad (3)$$

$$h_t = O_t * \tanh(C_t) \quad (4)$$

Cell State:

$$\acute{C}_t = \tanh(W_c . [h_{t-1}, x_t] + b_c) \quad (5)$$

$$C_t = f_t * C_{t-1} + i_t * \acute{C}_t \quad (6)$$

Here W is presented as the weight of input gate W_i, output gate W_o, forget gate W_f, cell state W_c. σ is the sigmoid function, tanh is tanh function, b is the bias of input gate b_i, output gate

b_o, forget gate b_f, cell state b_c. C_{t-1} is a unit state at the prior function and C_t is unit state of current function, similarly h_{t-1} is output of the prior state and h_t is the output of current state. $\acute{C}_t$ is unit memory state, x_t is input at t time, O_t is output at t time.

The LSTM architecture also includes an input layer, an output layer, and a hidden layer. The input layer takes in the input sequence and feeds it to the LSTM cells. The hidden layer consists of the LSTM cells arranged in a sequence. The output layer takes the final output of the last LSTM cell in the sequence and produces the output of the network.

During training, backpropagation through time (BPTT) is employed to update the network's weights and biases. BPTT involves calculating the gradients of the loss function with respect to the weights and biases at each time step and updating them using an optimization algorithm such as stochastic gradient descent..

LSTM networks can be stacked to form a deep LSTM network, which consists of multiple layers of LSTM cells. This allows the network to learn more complex representations and has been shown to improve performance on certain tasks.

3.1.2.1 Bi-directional LSTM

The bidirectional LSTM has 2 layers. The single layer of LSTM cell learns from left to right in forward direction in sequencing time, however, bidirectional LSTM learns forward as well as backward direction in respective of time. The architecture of bidirectional LSTM shown in Fig.3 (Istiake Sunny et al., 2020)

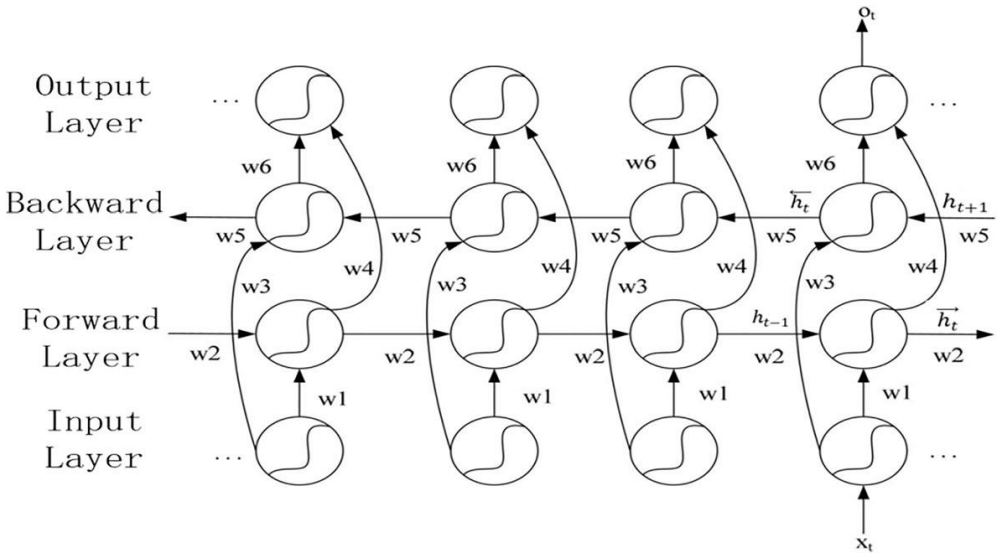


Figure 3 Bidirectional LSTM

The formula is shown below:

$$\vec{h}_t = f(w_1 x_t + w_2 h_{t-1}) \quad (7)$$

$$h_t = f(w_3 x_t + w_5 h_{t+1}) \quad (8)$$

$$O_t = g(w_4 \vec{h}_t + w_6 h_t) \quad (9)$$

Here, $\vec{h}_t$ is forward layer output at time t and h_t is backward layer output at time t.

The output of the BiLSTM is obtained by concatenating the outputs of the two LSTM layers. This allows the network to capture information from both the past and the future of the input sequence. By doing so, the BLSTM can capture context from both directions, which can be particularly useful in tasks such as speech recognition, where the current sound depends on the sounds that come before and after it.

The BiLSTM architecture also includes an input layer, an output layer, and a hidden layer. The input layer takes in the input sequence and feeds it to the two LSTM layers. The hidden layer consists of two LSTM layers arranged in a sequence, one processing the input sequence from the beginning to the end, and the other processing it from the end to the beginning. The output layer takes the final output of the two LSTM layers and produces the output of the network.

During training, backpropagation through time (BPTT) is employed to update the network's weights and biases. During BPTT, the gradients of the loss function are computed with respect to the weights and biases at each time step. These gradients are then used to update the weights and biases by employing an optimization algorithm like stochastic gradient descent.

3.1.2.2 Stacked LSTM

A Stacked Long Short-Term Memory (LSTM) is a variant of the traditional LSTM architecture that is designed to improve the performance of the model by stacking multiple LSTM layers on top of each other. LSTM is a variant of recurrent neural network (RNN) that excels at processing sequential data. It has found extensive application in diverse fields, including natural language processing and speech recognition.

The basic architecture of an LSTM cell consists of three gates, namely the input gate, forget gate, and output gate, and a memory cell that stores the information for the current time step. The input gate determines which information from the current input should be stored in the

memory cell, while the forget gate determines which information should be discarded from the memory cell. Finally, the output gate determines which information from the memory cell should be output to the next time step.

In a stacked LSTM, multiple LSTM layers are stacked on top of each other, with each layer having its own set of gates and memory cells. The output of the previous LSTM layer is fed as input to the next LSTM layer, allowing the network to learn more complex representations of the input data.

The advantage of using stacked LSTMs is that it allows the model to capture long-term dependencies in the input data by processing the data in a hierarchical manner. The lower layers of the LSTM capture the local dependencies, while the higher layers capture the global dependencies in the data. This results in a more accurate and robust model that can handle complex input sequences.

However, using stacked LSTMs also increases the number of parameters in the model, making it more computationally expensive and increasing the risk of overfitting the training data. Therefore, it is important to carefully tune the hyperparameters of the model to ensure optimal performance.. The architecture of Stacked LSTM is shown below Fig.4.

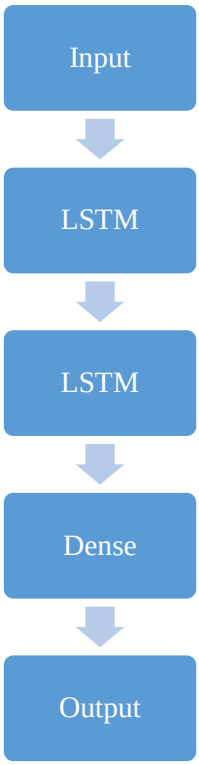


Figure 4 Stacked LSTM architecture

3.1.2.3 Bi-directional Stacked LSTM

A bi-stacked LSTM is a neural network architecture that combines the benefits of both stacked LSTMs and bi-directional LSTMs. It consists of multiple layers of bi-directional LSTM cells, where each layer has both a forward and a backward LSTM. Moreover, it has two stacks of such layers, where one stack processes the input sequence from the beginning to the end (in the forward direction) and the other stack processes the input sequence from the end to the beginning (in the backward direction). The output of each layer in the forward stack is concatenated with the output of the corresponding layer in the backward stack, and the resulting concatenated output is fed as input to the next layer in the forward stack. This process continues until the last layer in the forward stack. The output of the last layer in the forward stack is then used as the input to the output layer. Similarly, the output of the last layer in the backward stack is used as the input to the output layer. The final output of the bi-stacked LSTM is the concatenation of the outputs of the output layer. Bi-stacked LSTMs have been shown to be effective in capturing complex long-term dependencies in sequential data, and they have achieved state-of-the-art performance on various tasks, such as language modeling, speech recognition, and machine translation. However, they are computationally expensive and require a large amount of training data and computing resources to train effectively.

One of the advantages of using a bi-directional stacked LSTM is its ability to capture information from past and future contexts of a given time step. This feature is particularly useful in tasks where future information is crucial for predicting the output. For example, in speech recognition, the current phoneme prediction depends not only on the past but also on the future context. A bi-directional stacked LSTM can capture both past and future context and provide more accurate predictions.

Another advantage of using a bi-directional stacked LSTM is its potential to reduce the impact of vanishing gradients that can occur in deep neural networks. By enabling the network to learn from both forward and backward contexts, a bi-directional stacked LSTM can mitigate the vanishing gradient problem and make training more efficient.

Additionally, a bi-directional stacked LSTM can capture complex dependencies that may not be easily identifiable from a unidirectional model, leading to improved performance on various tasks, including natural language processing, speech recognition, and image captioning.

In conclusion, the bi-directional stacked LSTM is a potent neural network architecture that can handle long-term dependencies effectively and enhance the accuracy of predictions by capturing information from both forward and backward contexts.

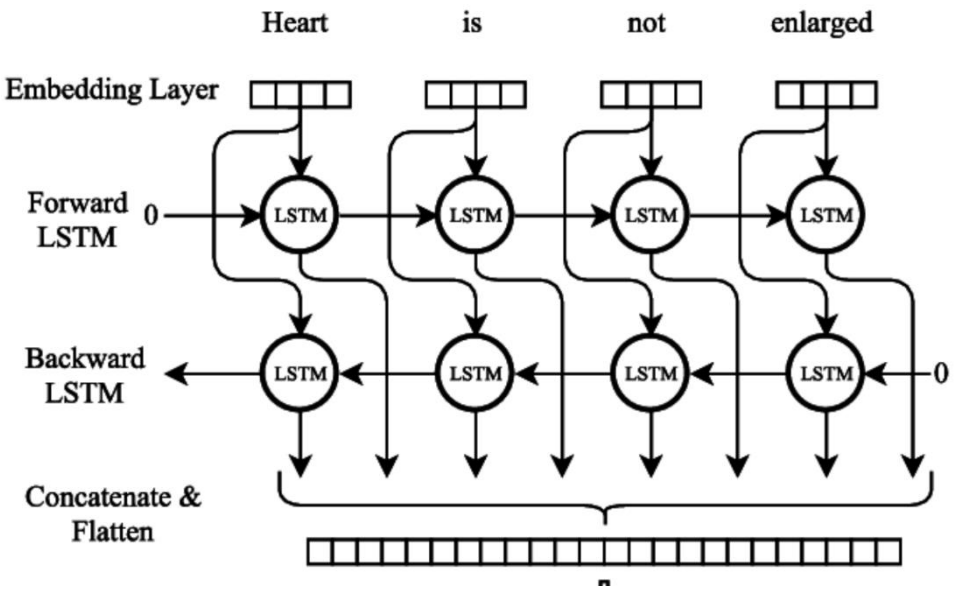


Figure 5 Bi-Stacked LSTM architecture

3.1.2.4 Difference between Bi-directional Stacked LSTM and Stacked LSTM:

Stacked LSTM is a type of Recurrent Neural Network (RNN) that is composed of multiple LSTM layers stacked on top of each other. Each LSTM layer receives the output sequence from the previous layer as input and generates a new sequence of output. The stacked LSTM is particularly useful for modeling long-term dependencies in sequential data.

On the other hand, Stacked BiLSTM (Bidirectional LSTM) is a variant of Stacked LSTM where each LSTM layer is split into two directions, one for processing the input sequence in the forward direction, and the other for processing the input sequence in the backward direction. The outputs from both directions are concatenated to generate a new output sequence. The Stacked BiLSTM is particularly useful for capturing dependencies in both past and future context of a sequence.

3.1.3 Introduction of GridSearchCV:

Grid search cross-validation (GridSearchCV) is a widely used method for tuning hyperparameters in machine learning models. Hyperparameters are model settings that are not learned from data but instead set by the user. The choice of hyperparameters can significantly affect a model's performance and results.

GridSearchCV works by exhaustively searching a predefined hyperparameter space, which is defined as a grid of hyperparameter values to try. For each combination of hyperparameters, GridSearchCV trains and evaluates a model using cross-validation, which is a technique for assessing how well a model generalizes to new data. Cross-validation involves splitting the data into multiple folds, training the model on some folds and testing on others, and repeating this process for all possible fold combinations.

The main advantage of GridSearchCV is that it systematically explores the hyperparameter space and helps identify the optimal hyperparameter settings for a given model and dataset. This can lead to better performance and more robust models that generalize well to new data. Without GridSearchCV, it can be challenging to determine the best hyperparameters through manual trial and error.

In summary, we use GridSearchCV for a model to identify the optimal hyperparameters that result in the best performance for a given dataset and model architecture**.**

3.2 Research Methodology

The methodology for emotion detection from facial expressions using stacked bidirectional LSTM involves several steps:

3.2.1 Data Collection

Data collection for a model involves the process of gathering and acquiring the necessary data that will be used to train or test the model. This process involves identifying the data sources, selecting and extracting relevant data, cleaning and pre-processing the data, and storing it in a suitable format that can be easily fed into the model.

The performance of a machine learning model is highly influenced by the quality and quantity of the training data. Therefore, the process of collecting data is critical in developing a model capable of performing well on new and unseen data. It is vital to ensure that the training data is diverse, representative of real-world situations, and devoid of biases that may impact the model's performance.

Data collection can involve various sources such as databases, APIs, web scraping, sensor data, and crowdsourcing. The data collected may be labeled or unlabeled, and it may require additional annotation or labeling to make it suitable for the model.

In summary, data collection for a model involves the process of acquiring and pre-processing the data that will be used to train or test the model. It is a crucial step in building a machine learning model that can perform well on unseen data.

In this study, facial expression images or videos should be collected along with their corresponding emotional labels. The images can be collected using a camera or a dataset. In this study we are using (OHEIX JONATHAN, n.d.)

3.2.2 Pre-processing

Pre-processing in machine learning refers to the steps taken to transform raw data into a format suitable for training and evaluating a machine learning model. The pre-processing step is important in the machine learning pipeline because it can greatly impact the performance of the model. The goal of pre-processing is to make the data more suitable for the chosen machine learning algorithm or model, by addressing issues such as missing values, scaling, and feature extraction.

Pre-processing in machine learning refers to the steps taken to transform raw data into a format suitable for training and evaluating a machine learning model. The pre-processing step is important in the machine learning pipeline because it can greatly impact the performance of the model. The goal of pre-processing is to make the data more suitable for the chosen machine learning algorithm or model, by addressing issues such as missing values, scaling, and feature extraction.

The pre-processing steps for a model depend on the specific requirements of the dataset and the machine learning algorithm being used. Some common pre-processing steps in this study include:

- Data Loading: The first step is to load the dataset into memory. This can be done using a variety of tools and libraries such as Pandas or NumPy.
- Data Cleaning: In this step, the dataset is checked for missing values, and if any are found, they are either imputed or removed depending on the extent of missingness.
- Data Rescaling: It is common to rescale the pixel values of the images to a common scale between 0 and 1. This can be achieved by dividing each pixel value by the maximum pixel value of the image.
- Data Augmentation: Data augmentation techniques such as rotation, flipping, and zooming are often used to increase the diversity of the training set and improve the robustness of the model.

3.2.3 Exploratory data analysis (EDA)

Exploratory data analysis (EDA) is a critical process in data science that involves exploring and summarizing the main characteristics of a dataset. The goal of EDA is to understand the

structure and distribution of the data and identify any patterns, anomalies, or relationships that might exist within the data. EDA can help in identifying potential problems with the data, selecting appropriate models, and developing insights that can inform further analysis.

In this study, EDA can involve several steps, such as:

- Data Loading: First, we need to load the dataset into our programming environment. This can be done using libraries such as Pandas, NumPy, or TensorFlow.
- Data Cleaning and Preprocessing: In this step, we may check if there are any missing values or inconsistent data points that need to be addressed before performing EDA. We may also perform data preprocessing techniques such as normalization, standardization, or feature scaling, depending on the requirements of the data analysis.
- Data Visualization: EDA often involves visualizing the data in various ways to identify patterns and relationships. We may use techniques such as histograms, scatterplots, boxplots, and heatmaps to visualize the distribution of the data, identify outliers or anomalies, and explore relationships between different variables.
- Statistical Analysis: In addition to visualization, we may perform various statistical tests to evaluate the significance of relationships and differences in the data. For example, we may perform hypothesis testing, correlation analysis, or regression analysis, depending on the requirements of the data analysis.
- Model Selection: Finally, based on the insights gained from EDA, we may select appropriate models and algorithms to build predictive models or perform further analysis on the data.

Overall, EDA is a critical step in any data science project and can help in gaining insights and developing a better understanding of the data. In this study, EDA can help in understanding the distribution and characteristics of the data, identifying any potential problems or inconsistencies, and selecting appropriate models and algorithms for further analysis.

3.2.4 Data Augmentation

Data augmentation should be used with care, however, as it is important to ensure that the generated images are still representative of the underlying data distribution. If the augmented images are too dissimilar from the original images, this can actually hurt the performance of the model. Additionally, it is important to validate the performance of the model on a separate validation set to ensure that the improvements in performance are not simply due to overfitting to the training data.

In the context of facial expression recognition using, data augmentation techniques can be applied to the input images. Some common data augmentation techniques include:

- Image rotation: Rotate the image by a certain degree to create new training samples.
- Image flipping: Flip the image horizontally or vertically to create new training samples.
- Image cropping: Crop a portion of the image to create new training samples.
- Image scaling: Increase or decrease the size of the image to create new training samples.
- Image shearing: Apply a shear transformation to the image to create new training samples.

3.2.5 Model Selection

Stacked bidirectional LSTM is used for the Jonathan Oheix dataset because it is a sequential dataset, where the order of the data is important. Bidirectional LSTMs can read the data in both directions, which allows the network to better understand the temporal relationship between the input data.

Moreover, stacking multiple layers of LSTMs enables the network to learn more complex temporal patterns and feature representations. The output of one LSTM layer is used as the input of the next LSTM layer, allowing the network to build a hierarchy of representations of the input data. This can lead to better accuracy in tasks such as facial expression recognition, where the ability to understand and recognize temporal patterns is crucial.

The stacked bidirectional LSTM (BiLSTM) is composed of multiple BiLSTM layers that are stacked on top of each other. The BiLSTM layers process the sequential data bidirectionally by taking into account the temporal dependencies between the features in both forward and backward directions. The stacked BiLSTM network is then trained using the collected data. The training involves optimization process is to adjust the network's parameters in order to minimize the discrepancy between the predicted emotions and the actual emotions.

3.2.6 Train and Testing

The model is compiled with the categorical cross-entropy loss function, the Adam optimizer, and the accuracy metric. The trained model is evaluated on the test set using the evaluate method, which returns the loss and accuracy metrics.

3.2.7 Finalize the hyperparameter with GridSeacrhCV

The research methodology for finalizing hyperparameters using GridSearchCV typically involves the following steps:

- Define the range of hyperparameters to be tested: This involves identifying the hyperparameters that are likely to have the greatest impact on the performance of the model and determining a range of values to test.
- Define the model: This step entails specifying the model's architecture and the training procedure. For neural networks, it encompasses determining the number of layers, number of neurons within each layer, activation functions, optimization algorithms, batch size, and number of epochs.
- Define the evaluation metric: This step requires selecting a metric to assess the model's performance. The chosen metric will serve as a basis for comparing the performance of various hyperparameter configurations.
- Perform GridSearchCV: In this step, GridSearchCV is utilized to explore the specified range of hyperparameters and assess the model's performance for each hyperparameter combination. GridSearchCV trains and evaluates the model on different subsets of the training data, employing cross-validation to estimate the generalization performance for each hyperparameter configuration.
- Select the best hyperparameters: After evaluating the model with different hyperparameters, select the set of hyperparameters that produces the best performance on the evaluation metric.
- Evaluate the final model: After selecting the best hyperparameters, evaluate the performance of the final model on a holdout dataset to estimate its generalization performance on new, unseen data.
- Deploy the model: Once the final model has been evaluated and shown to have satisfactory performance, it can be deployed in a production environment for use in making predictions.

3.3 Summary

The research methodology for using stacked bidirectional LSTM for facial expression recognition on the Jonathan Oheix dataset can be summarized as follows:

- Data Collection: The Jonathan Oheix dataset containing images of faces labeled with their respective emotions was used for this study.
- Data Preprocessing: The images were resized and converted to grayscale. Data augmentation techniques such as random rotation, zooming, and flipping were used to increase the dataset size and improve model performance.

- Model Architecture: A stacked bidirectional LSTM model was employed to extract features from the input images. The model incorporated multiple LSTM layers with bidirectional connections, enabling it to capture information from both past and future contexts. The resulting output was then fed through a fully connected layer with softmax activation to generate the probability distribution across the different classes.
- Model Training: The model underwent training on the augmented dataset utilizing the Adam optimizer and categorical cross-entropy loss function. To optimize performance, hyperparameters including learning rate, batch size, number of neurons, and number of layers were fine-tuned through the application of grid search in conjunction with cross-validation.
- Choosing hyperparameter: Finalizing the hyperparameters with GridSearchCV involves defining a range of hyperparameters to test, defining the model architecture and training process, selecting an evaluation metric, performing GridSearchCV to evaluate the model for each hyperparameter combination, selecting the best hyperparameters, evaluating the final model on a holdout dataset to estimate its performance on new data, and finally deploying the model in a production environment for making predictions.
- Model Evaluation: The performance of the trained model was evaluated on the validation set and the test set using metrics such as accuracy, precision, recall, and F1-score. The confusion matrix was also analyzed to study the misclassification patterns and identify the classes that were difficult to classify.
- Interpretation: The trained model was used to generate saliency maps to visualize the regions of the input images that contributed the most to the final predictions. This helped in understanding the model's decision-making process and identifying the features that were relevant for facial expression recognition.

Overall, the proposed research methodology demonstrated the effectiveness of using stacked bidirectional LSTM for facial expression recognition on the Jonathan Oheix dataset. The interpretation of the model's decision-making process using saliency maps provided insights into the relevant features for facial expression recognition.

CHAPTER 4: ANALYSIS

4.1 Introduction

The use of deep learning techniques (Wang et al., 2019) for facial expression recognition has gained significant attention in recent years due to their ability to handle complex data and achieve high accuracy. In this analysis, we focus on the use of stacked bidirectional LSTM for facial expression recognition on the Jonathan Oheix dataset, which contains grayscale images of facial expressions labeled with seven emotion categories.

This study explores the effectiveness of stacked bidirectional LSTM for facial expression recognition and evaluates its performance in comparison to other deep learning models. Specifically, we will train and test a stacked bidirectional LSTM model on the Jonathan Oheix dataset, using techniques such as data augmentation and hyperparameter tuning to optimize its performance.

The analysis will begin with a thorough exploration of the dataset, including visualizations of the image data and an examination of class imbalances. We will then preprocess the data and perform data augmentation to increase the size and diversity of the training set. Next, we will define and train the stacked bidirectional LSTM model using a grid search approach to tune hyperparameters. We will evaluate the performance of the model on the test set using metrics such as accuracy and F1 score, and compare it to other state-of-the-art deep learning models for facial expression recognition.

Overall, this analysis will provide insights into the use of stacked bidirectional LSTM for facial expression recognition and its performance on the Jonathan Oheix dataset. The results of this analysis can have implications for the development of more accurate and efficient models for facial expression recognition in real-world applications such as emotion recognition systems for human-computer interaction and virtual reality.

4.2 Dataset collection and overview

The Jonathan Oheix dataset (OHEIX JONATHAN, n.d.), is a publicly available dataset on Kaggle that contains facial expression images with labels for seven different emotions: happy, sad, angry, surprise, fear, disgust, and neutral. The dataset consists of 48x48-pixel grayscale images of faces with varying lighting conditions, orientations, and expressions.

The dataset includes a training set of 28,709 images and a validation set of 3,589 images. Each image is labeled with one of the seven emotion categories. The dataset has been preprocessed,

with the images aligned and centered on the faces, and the pixel values normalized to be between 0 and 1.

The Jonathan Oheix dataset is commonly used for training and testing emotion detection models based on facial expressions using deep learning techniques such as convolutional neural networks (CNNs), RNN, LSTM, stacked LSTM networks etc. It has been used in several research papers and competitions related to emotion recognition, and it provides a valuable resource for researchers and developers working on affective computing and related fields.

4.3 Dataset load and import library

The method is performed on a Jupyter Notebook.

As an initial step, imports several libraries that are crucial for data analysis and visualization, such as Matplotlib, Numpy, Pandas, and Seaborn. Additionally, deep learning libraries are also imported using Keras.

4.4 Pre-processing

Pre-processing is a crucial step in preparing data for a model in stacked bi-directional LSTM or any other machine learning/deep learning algorithm. In the case of LSTM, pre-processing is necessary because LSTMs require their input data to be in a specific format, which includes:

- As the image size of all the images are not in same size, hence particular image size is set before processing further steps. Here, the image size was set to 32.
- Defining the path to the directory containing the images for the training and validation sets. Load the images from the training and validation directories using the load_img and img_to_array functions from Keras. The load_img function loads the image as a PIL (Python Imaging Library) object, and the img_to_array function converts the PIL object to a Numpy array.
- Generating augmented image data for the training set using the ImageDataGenerator class from Keras. This involves rescaling the pixel values, randomly applying horizontal flips, and randomly applying shearing and zooming to the images.
- Generating image data for the validation set without any augmentation, only rescaling the pixel values.
- The batch size was set to a suitable value to balance between memory usage and model accuracy.
- Batches of data were created to feed into the model during training and evaluation using the flow_from_directory method from the ImageDataGenerator class

- The categorical labels for the training and validation sets were encoded using the to_categorical function from Keras.

These pre-processing steps are essential for ensuring that the data is properly formatted and augmented, which helps the model generalize better and avoid overfitting. In summary, preprocessing is done to transform the raw input data into a format that can be used by an LSTM model. Preprocessing also helps to improve the efficiency and effectiveness of the training process by ensuring that the input data is in the correct format and is normalized to a common scale.

4.5 EDA

The Jonathan Oheix dataset contains images of facial expressions of people, classified into seven different emotions: angry, disgust, fear, happy, neutral, sad, and surprise. Before building a model using this dataset, it's important to perform exploratory data analysis (EDA) to gain insights into the data.

Libraries such as Matplotlib and Seaborn are used to visualize the data and gain insights. For example, histograms and bar charts etc are used to visualize the distribution of different facial expressions in the training and validation sets.

Here are some steps for EDA on the Jonathan Oheix dataset:

- Check the distribution of classes: It's important to check the distribution of classes in the dataset to make sure that there is no class imbalance. This can be done by plotting a histogram of the number of images per class.

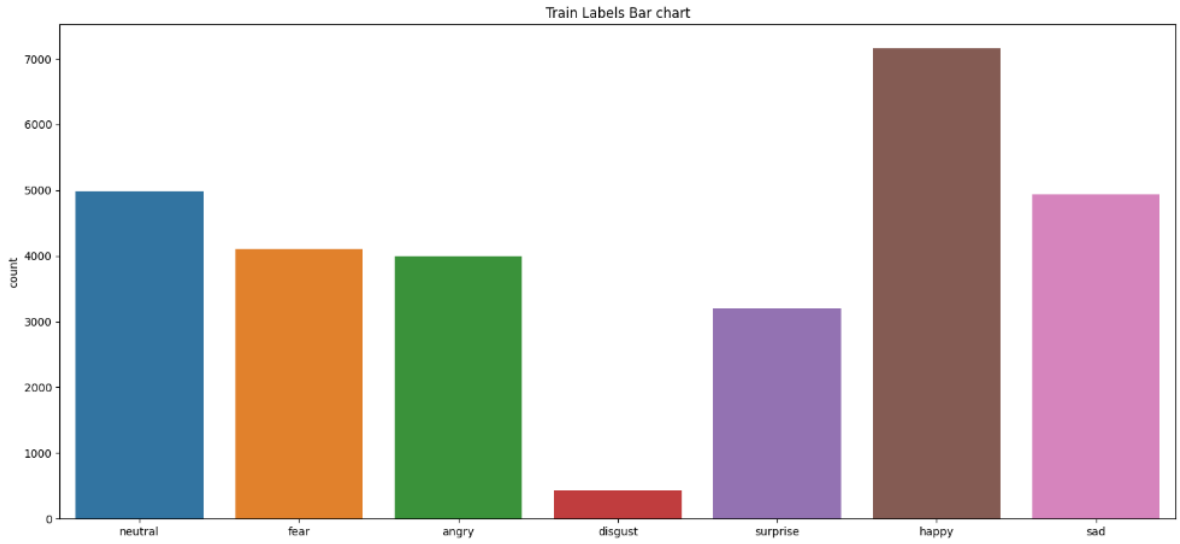


Figure 6 Train dataset

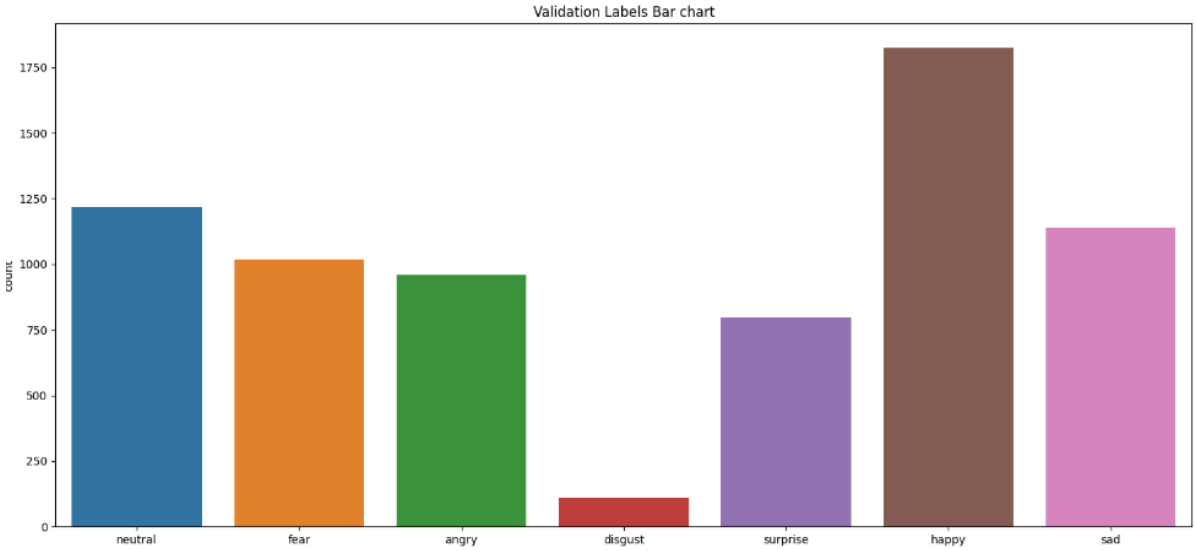


Figure 7 Validation Dataset

- Visualize some sample images: It's always helpful to visualize some sample images from the dataset to get a sense of what the images look like and whether there are any obvious patterns or features that can be used for feature extraction.

Figure 8 An overview of images

- Check for any missing or corrupted images: It's important to make sure that there are no missing or corrupted images in the dataset that can affect the performance of the model.
- Check the size and resolution of images: It's important to check the size and resolution of the images in the dataset to make sure that they are consistent and can be fed into the model.

Overall, performing EDA on the Jonathan Oheix dataset can help identify any potential issues or challenges that may arise during model building and ensure that the final model is accurate and effective.

4.6 Data Augmentation

Data augmentation is often used in deep learning to artificially expand the size of the training dataset, which can help to improve the model's generalization performance. (Ahmed et al., 2019) explained how to do data augmentation for various models in facial detection. In the case of the Jonathan Oheix facial expression dataset, data augmentation can be particularly important to prevent overfitting and improve the accuracy of the model.

Furthermore, in the case of image data, data augmentation techniques like random rotations, shearing, and flipping can also help the model learn to be invariant to certain types of image transformations, which can make the model more robust and accurate on new, unseen images.

Below steps are used in this study

- ImageDataGenerator class from the Keras library to perform data augmentation and scaling on the images.
- The datagen_train object performs rescaling of pixel values to be in the range of [0,1] using the rescale argument, shear transformation with a range of 0.2 using the shear_range argument, zooming in/out with a range of 0.2 using the zoom_range argument, and horizontal flipping using the horizontal_flip argument.
- The datagen_val object only performs rescaling of pixel values.
- Then, train_set and test_set are created using the flow_from_directory method of the ImageDataGenerator class, which reads images from directories containing the training and validation sets respectively, and resizes them to the target_size specified in the arguments. It also specifies that the images are in grayscale format with the color_mode argument, and the labels are categorical with class_mode='categorical'.
- Finally, shuffle=True is used for the training set to randomly shuffle the images before each epoch, and shuffle=False is used for the validation set to keep the images in their original order.

```
]: datagen_train = ImageDataGenerator(
       rescale=1. / 255,
       shear_range=0.2,
       zoom_range=0.2,
       horizontal_flip=True)

   datagen_val = ImageDataGenerator(rescale=1. / 255)


   train_set = datagen_train.flow_from_directory(os.path.join(IMAGES_DIR,  "train"),
                                                 target_size = (SIZE, SIZE),
                                                 color_mode = "grayscale",
                                                 batch_size=batch_size,
                                                 class_mode='categorical',
                                                 shuffle=True)


   test_set = datagen_val.flow_from_directory(os.path.join(IMAGES_DIR,  "validation"),
                                                 target_size = (SIZE, SIZE),
                                                 color_mode = "grayscale",
                                                 batch_size=batch_size,
                                                 class_mode='categorical',
                                                 shuffle=False)

   Found 28821 images belonging to 7 classes.
   Found 7066 images belonging to 7 classes.
```

Figure 9 An overview of data augmentation

By applying these transformations to the input images, we can create a larger and more diverse dataset for training the stacked LSTM model. This can help the model learn to recognize facial expressions more accurately and robustly, leading to better performance on unseen data. The Keras ImageDataGenerator function can be used to apply these transformations to the images and generate augmented data batches for training the model.

4.7 Train the Model

4.7.1 Definition of Training

Training a model is the process of teaching it to recognize patterns and make predictions based on input data. This involves adjusting the model's parameters to minimize a loss function that measures the difference between the model's predictions and the actual output.

During the training process, the model is presented with a set of labeled training data, which is used to adjust its parameters. The training data is typically divided into batches, and the model is trained on each batch in turn. After each batch, the model's performance on a validation dataset is evaluated to determine if it is overfitting or underfitting the training data.

The process of adjusting the model's parameters to minimize the loss function is typically done using a variant of stochastic gradient descent, which involves computing the gradients of the

loss function with respect to the model's parameters and using these gradients to update the parameters. The process is repeated iteratively until the model's performance on the validation dataset reaches a satisfactory level, or until a predefined stopping criterion is met. After the training phase, the trained model becomes capable of making predictions on novel and unseen data.

4.7.2 Definition of Model

In Python, a model refers to a software object that represents a machine learning algorithm that has been trained to make predictions on new data. A model can take in input data and produce an output prediction based on the relationships learned from the training data.

In machine learning, a model can be created using a variety of algorithms, such as linear regression, decision trees, random forests, support vector machines, neural networks, etc. The choice of algorithm will depend on the problem at hand and the type of data being used.

A machine learning model typically involves a set of parameters that are adjusted during the training process to optimize its performance. Once the model is trained, it can be used to make predictions on new, unseen data.

In Python, there are several popular machine learning libraries that provide pre-implemented models for various types of problems. Some examples include scikit-learn, TensorFlow, Keras, and PyTorch. These libraries allow users to easily create, train, and use models for a variety of applications.

4.7.3 Why is GridSearchCV used in study

GridSearchCV can also be used in bidirectional stacked LSTM models to tune the hyperparameters of the model and find the optimal configuration that leads to the best performance. This (Ahmad et al., 2022)paper explained properly how GridSearchCV used in machine learning algorithm.

A bidirectional stacked LSTM model is similar to a stacked LSTM model, but with the addition of bidirectional layers. In a bidirectional layer, the input sequence is processed in both forward and backward directions, which allows the model to capture context from both past and future inputs.

The hyperparameters of a bidirectional stacked LSTM model are similar to those of a stacked LSTM model, including the number of LSTM units, the dropout rate, the activation function, the initialization method, the batch size, the learning rate, and the number of epochs. However,

there are additional hyperparameters specific to bidirectional layers, such as the merge mode and the weights initialization method.

GridSearchCV (Lorente et al., 2021)can be used to systematically search over a grid of hyperparameters for each LSTM layer and bidirectional layer, as well as the model as a whole. The hyperparameters can be specified as a dictionary or a list of dictionaries, with each dictionary containing the hyperparameters for one LSTM layer or bidirectional layer, or the model.

During the grid search process, GridSearchCV will train and evaluate the model with each combination of hyperparameters in the grid. It will use cross-validation to split the data into training and validation sets, and then fit the model on the training set and evaluate it on the validation set. This process is repeated for each combination of hyperparameters.

After the grid search is complete, GridSearchCV will return the combination of hyperparameters that resulted in the best performance on the validation set. These hyperparameters can then be used to train the final model on the entire dataset.

Overall, using GridSearchCV in a bidirectional stacked LSTM model can help to find the optimal configuration of hyperparameters and improve the performance of the model, especially when dealing with sequential data where context from both past and future inputs is important.

4.7.3.1 Importance of selection of row_hidden and col_hidden of stacked bilstm model

The row_hidden and col_hidden parameters in a stacked Bidirectional LSTM model refer to the number of hidden units in each LSTM layer in the vertical and horizontal directions, respectively.

The row_hidden parameter controls the number of LSTM cells in the vertical direction, which corresponds to the sequence length dimension of the input data. Increasing this parameter can help the model learn more complex temporal dependencies in the data, but also increases the number of parameters in the model and can lead to overfitting if not regularized properly.

The col_hidden parameter controls the number of LSTM cells in the horizontal direction, which corresponds to the feature dimension of the input data. Increasing this parameter can help the model capture more complex patterns in the input features, but can also increase the computational complexity and training time of the model.

In general, the choice of row_hidden and col_hidden should be based on the complexity of the input data and the available computational resources. It is common practice to start with a small number of hidden units and gradually increase them while monitoring the performance of the

model on a validation set. Regularization techniques such as dropout and weight decay can also be used to prevent overfitting when increasing the number of hidden units.

4.7.4 How GridSearchCV used in study

In this section, a grid search was performed to optimize the hyperparameters of a Keras model using the GridSearchCV function from scikit-learn. The hyperparameters tuned were row_hidden and col_hidden, which control the number of hidden units in the rows and columns of a bidirectional LSTM layer.

The create_model function defined the model architecture, including a bidirectional LSTM layer followed by four standard LSTM layers for the row encoding, and a single LSTM layer for the column encoding. The output of the LSTM layers was flattened and passed through two fully connected Dense layers before being classified into the appropriate number of classes using a softmax activation function.

The KerasClassifier was used to wrap the create_model function, and a grid search was performed over the hyperparameters row_hidden and col_hidden using the param_grid dictionary. The best model was determined using cross-validation with a 3-fold split. The best hyperparameters and model performance were recorded in the grid_result object.

4.7.5 Training of Stacked LSTM Model

This section describes the architecture and training of stacked lstm model. This paper (Fares et al., 2019) helps us to identify how Bi-LSTM helps in image classification. At first define the hyperparameters such as the number of classes, number of epochs, batch size, and number of hidden units for the stacked LSTM layers. Then load the data and define the model architecture with input and output layers, followed by several stacked LSTM layers, and finally, dense output layers with softmax activation.

In the subsequent step, the model is compiled with a designated loss function, optimizer, and metrics. Subsequently, the model is trained on the training data while validating against the validation data. Finally, the model is evaluated using the test data, and the resulting test loss and accuracy are printed.

The model architecture consists of the following components:

- Input layer: This layer specifies the input shape of the data. In this case, the input is a 2D image with dimensions SIZE x SIZE and a single channel (grayscale).
- Permute dimensions layer: This layer permutes the dimensions of the input tensor to make it suitable for processing by the subsequent layers.

- Reshape layer: This layer reshapes the input tensor to have a shape of (-1, SIZE, SIZE). The -1 in the shape means that the length of this dimension is automatically inferred based on the shape of the input.
- Bidirectional LSTM layer: This layer operates on the input sequence by utilizing a Long Short-Term Memory (LSTM) architecture in both forward and backward directions. The resulting output of this layer is a sequence of hidden states, which subsequently serve as input for the next layer.
- Stacked LSTM layers: In this loop, four additional LSTM layers are added to the model. Each layer has a hidden size of row_hidden and the return_sequences parameter is set to True. By setting return_sequences to True, the LSTM layer outputs a sequence of hidden states rather than a single hidden state. The final LSTM layer produces a sequence of hidden states representing the columns of the input image.
- LSTM layer: This layer processes the sequence of column hidden states output by the previous LSTM layers and returns a single hidden state that represents the entire input image.
- Flatten layer: This layer flattens the output of the previous LSTM layer into a 1D vector.
- Dense layers: These are two fully connected layers with 64 neurons each, followed by a final output layer with num_classes neurons and a softmax activation function. The softmax activation function normalizes the output of the model to represent a probability distribution over the possible output classes.
- The entire model is then compiled and trained using the specified hyperparameters.

There are different callback functions used during the training of the above proposed model:

- ModelCheckpoint: This callback saves the model at a specific point during training. It can be used to save the weights of the model or the entire model in h5 format. It takes parameters like monitor, which specifies the metric to monitor, verbose for logging information, save_best_only to save the best model during training, and mode which specifies the direction of optimization.
- EarlyStopping: This callback stops the training if the monitored metric stops improving for a specified number of epochs. It is used to prevent overfitting and speed up the training process. It takes parameters like monitor, min_delta which specifies the minimum change in the monitored metric to be considered as improvement, patience which specifies the number of epochs to wait before stopping, and verbose for logging information.

- ReduceLROnPlateau: This callback reduces the learning rate of the optimizer if the monitored metric stops improving. It is used to prevent the model from getting stuck in local minima during training. It takes parameters like monitor, factor which specifies the factor by which the learning rate will be reduced, patience which specifies the number of epochs to wait before reducing the learning rate, and verbose for logging information.

For setting up the training of the model here using the Keras API in Python.

The callbacks_list is a collection of callback functions that are invoked during the training process at specific intervals or based on specific conditions. In this scenario, three callbacks are utilized. Firstly, the early_stopping callback halts training if the validation loss fails to improve over a specified number of epochs. Secondly, the checkpoint callback saves the model weights when the validation accuracy improves. Lastly, the reduce_learningrate callback adjusts the learning rate if the validation loss does not improve over a certain number of epochs.

The compile method is used to specify the loss function, optimizer, and evaluation metrics for the model. In this case, the categorical_crossentropy loss function is used for multi-class classification, the Adam optimizer is used with a learning rate of 0.001, and the accuracy metric is used to evaluate the performance of the model during training.

At last train the model on the training data and validate it on the validation data using the Keras fit function. The fit function takes several arguments, including:

- train_set: The training data to use for training the model.
- steps_per_epoch: The number of steps (batches of samples) to yield from the generator at each epoch.
- epochs: The number of epochs to train the model.
- validation_data: The validation data to use for evaluating the model.
- validation_steps: The number of steps (batches of samples) to yield from the generator at each validation epoch.
- callbacks: A list of Keras callback functions to apply during training. The callbacks_list argument is a list of three Keras callback functions that were defined earlier: early_stopping, checkpoint, and reduce_learningrate. These callbacks are used to monitor the training process and perform actions such as saving the best model weights, stopping training early if the model is not improving, and reducing the learning rate if the model is plateauing.

Upon completion of training, the fit function produces a History object that encompasses the training and validation metrics for each epoch. This object can be utilized for plotting and

further analysis of the training process.The history object is typically saved for later use so that the training can be resumed or the results can be analyzed.

Then code model.load_weights("./model.h5") is used to load the saved weights of the trained model from the file "./model.h5". This is typically used when we want to use a trained model to make predictions on new data without having to retrain the model. By loading the saved weights, we can initialize the model with the learned parameters from the previous training session, and then use the model to make predictions on new data.

4.8 Summary

The above text discusses different aspects related to the use of the Jonathan Oheix facial expression dataset for training emotion detection models using deep learning techniques, such as convolutional neural networks (CNNs), RNN, LSTM, stacked LSTM networks, among others. The dataset is publicly available on Kaggle and contains 48x48-pixel grayscale images of faces with varying lighting conditions, orientations, and expressions labeled for seven different emotions: happy, sad, angry, surprise, fear, disgust, and neutral. It includes a training set of 28,709 images and a validation set of 3,589 images. The images are aligned, centered on the faces, and the pixel values are normalized to be between 0 and 1. The text highlights the importance of pre-processing the data, including setting the image size to 32, loading the images, generating augmented image data, encoding categorical labels, creating batches of data to feed into the model, and normalizing the data.

Exploratory data analysis (EDA) is also recommended before building the model to gain insights into the data. Histograms, bar charts, and other visualizations are used to visualize the distribution of different facial expressions in the training and validation sets. Data augmentation is often used in deep learning to artificially expand the size of the training dataset and improve the model's generalization performance. The ImageDataGenerator class from the Keras library is used to perform data augmentation and scaling on the images. Finally, the text describes the architecture and training of a stacked LSTM model, which involves defining the layers, compiling the model, fitting the model on the training data, and evaluating the model on the validation data.

CHAPTER 5: RESULTS AND DISCUSSIONS

The Results and Discussion section is one of the most important sections in a thesis paper as it presents and interprets the findings of the research. This section aims to answer the research questions or hypotheses and to provide evidence to support the research objectives.

5.1 Introduction

The Results section should be clear and concise, presenting the raw data and statistical analysis in a way that is easy to understand. The data should be presented in tables, graphs, or charts to help readers visualize the findings. The Results section should also include descriptive statistics and inferential statistics, such as means, standard deviations, correlations, and significance tests, to support the findings.

The Discussion section should provide an interpretation and evaluation of the results, explaining the meaning and implications of the findings in the context of the research questions and objectives. The discussion should relate the results to previous research in the field and provide explanations for any unexpected or conflicting results. The discussion should also identify any limitations of the study and suggest directions for future research.

Overall, the Results and Discussion section is crucial in a thesis paper as it demonstrates the researcher's ability to analyze data, draw meaningful conclusions, and contribute to the existing body of knowledge in the field. A well-written Results and Discussion section can strengthen the validity and reliability of the research and increase its impact and significance.

5.2 Result of GridSearchCV

One of the main objectives for any model is to select the proper hyperparameter and this is also the main objective of this study.

In this study at the initial step row_hidden and col_hidden was considered as 64 and 64, and the model accuracy obtained as near about 34%.

There are various hyperparameters that can affect the model. The hyperparameters are row_hidden and col_hidden, which control the number of hidden units in the rows and columns of a bidirectional LSTM layer. The dictionary contains two lists, one for each hyperparameter, with two values each. The grid search algorithm will perform a search over all possible combinations of the hyperparameters defined in the param_grid dictionary to find the combination that yields the best performance on the validation data. By specifying multiple

values for each hyperparameter, the grid search algorithm can explore a range of possible hyperparameters and select the optimal combination.

The grid_result object obtained from the grid search contains the best hyperparameters and the best score achieved by the model during the grid search. In this case, the best score obtained was 0.264217 and the best hyperparameters were col_hidden = 64 and row_hidden = 32.

The mean and standard deviation of the test score for each hyperparameter combination that was tried in a grid search. The parameters of each combination are also displayed.

The grid search was done with different values for the number of hidden units in the rows and columns of a 2D convolutional layer. The best test score was obtained with 64 hidden units in the columns and 32 hidden units in the rows.

```
0.258214 (0.005861) with: {'col_hidden': 16, 'row_hidden': 32}
0.248569 (0.002285) with: {'col_hidden': 16, 'row_hidden': 64}
0.248569 (0.002285) with: {'col_hidden': 16, 'row_hidden': 128}
0.263003 (0.019706) with: {'col_hidden': 32, 'row_hidden': 32}
0.259845 (0.011933) with: {'col_hidden': 32, 'row_hidden': 64}
0.248569 (0.002285) with: {'col_hidden': 32, 'row_hidden': 128}
0.264217 (0.000644) with: {'col_hidden': 64, 'row_hidden': 32}
0.255508 (0.015667) with: {'col_hidden': 64, 'row_hidden': 64}
0.248569 (0.002285) with: {'col_hidden': 64, 'row_hidden': 128}
```

Figure 10 Result of GridSearchCV

5.3 Results of model Training

The training of a neural network involves iteratively optimizing its weights and biases to minimize the loss function on a given dataset. The model is trained over several epochs, where each epoch represents one pass through the entire training dataset.

This is a training log of a neural network model. It shows the training progress over 10 epochs. Each epoch shows the loss and accuracy of the model on the training data and the validation data. The validation accuracy is used to evaluate the model's performance on data that it hasn't seen during training, and it is an important metric to prevent overfitting.

The model is being trained using the Stochastic Gradient Descent (SGD) optimizer, which updates the model's weights in each iteration to minimize the loss function. The learning rate is set to 0.001, which determines the size of the steps taken by the optimizer in the weight space.

The model seems to be improving over time, as the validation accuracy is increasing from epoch to epoch. The model starts with an accuracy of 0.2622 in the first epoch and reaches an accuracy of 0.35810 in the tenth epoch. However, there are some epochs where the validation accuracy does not improve or even decreases, such as the fourth and seventh epochs. This could be a sign of overfitting, where the model becomes too specialized in the training data and loses its generalization ability on the validation data.

Overall, the neural network model was trained for 32 epochs, with each epoch consisting of 96 batches. During each epoch, the training loss and accuracy were computed, and the model was evaluated on a separate validation dataset to monitor its progress. The model was saved to a file whenever the validation accuracy improved, indicating that the model had learned to generalize better to new data.

5.4 Illustration of Visualizations

The below is visual representation of the training history of the above proposed model. Specifically, it creates two subplots, one for the accuracy and the other for the loss, with the training and validation metrics plotted for each.

The first line of the code creates a figure with two subplots stacked vertically. The ax variable is an array of the two subplots. The next two lines plot the accuracy and validation accuracy metrics as a function of the epoch number. The accuracy is the fraction of correctly classified samples in the training set, while the validation accuracy is the fraction of correctly classified samples in the validation set.

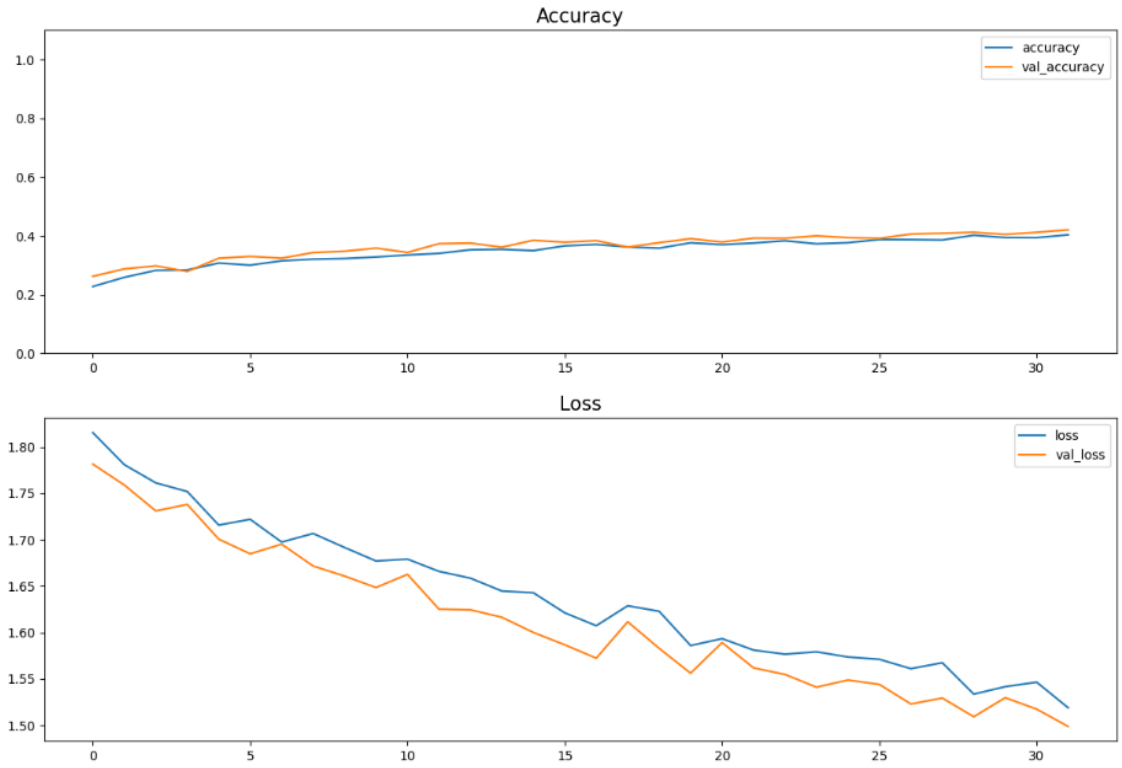


Figure 11 Accuracy and Loss of training and validation dataset

The x-axis of the plot represents the number of epochs, which is the number of times the model has seen the entire training dataset during training. The y-axis represents the accuracy, with values ranging from 0 to 1.

The next two lines plot the loss and validation loss metrics as a function of the epoch number. The loss is a measure of how well the model is able to predict the correct output for a given input, with lower values indicating better performance. The validation loss is the loss on the validation set, which is used to evaluate the generalization performance of the model.

5.5 Prediction of Test Dataset

- Classification Report: The prediction of test images evaluated by using the trained model. First, the model.predict_generator() function is used to generate predictions for the test set. Then, the np.argmax() function is used to convert the predictions to class labels.

  Next, an empty list y_test is created to store the true labels of the test set. The enumerate() function is used to loop through the test set and retrieve the true labels of each batch of test images. The np.concatenate() function is used to combine the true labels of all batches into a single array, which is then converted to class labels using np.argmax().

  The accuracy_score() function from the scikit-learn library is used to compute the accuracy of the model predictions compared to the true labels of the test set. Finally, the accuracy is printed as a percentage using the print() function.

  In this case, the accuracy on the test set is 42.13%.

  In the next segment results of classification report which provides an overview of the performance of the model on each class in the dataset.

  The classification_report() function from scikit-learn generates a classification report for a classification model. The classification report includes precision, recall, f1-score, and support for each class in the target data.

  In this case, y_test contains the ground truth labels and pred contains the predicted labels generated by the model. The classification_report() function compares these two arrays to compute the various evaluation metrics for each class in the dataset.

	precision	recall	f1-score	support
0	0.32	0.27	0.29	960
1	0.67	0.04	0.07	111
2	0.29	0.02	0.03	1018
3	0.52	0.74	0.61	1825
4	0.39	0.42	0.40	1216
5	0.31	0.33	0.32	1139
6	0.45	0.58	0.50	797
accuracy			0.42	7066
macro avg	0.42	0.34	0.32	7066
weighted avg	0.40	0.42	0.38	7066

Figure 12 Classification Report of test dataset

The classification report shows several evaluation metrics for the multi-class classification problem.

- Precision: Precision is a metric that quantifies the proportion of accurate positive predictions relative to the total predicted positive observations. For instance, if the precision for class 0 is 0.32, it signifies that among all the predictions made by the model for class 0, only 32% of them were correct.
- Recall: Recall, also known as sensitivity or true positive rate, calculates the proportion of correctly predicted positive observations in relation to the total actual positive observations. For instance, if the recall for class 3 is 0.74, it indicates that out of all the actual observations belonging to class 3, the model accurately predicted 74% of them.
- F1-score: The F1-score is a metric that represents the harmonic mean of precision and recall. It provides a balanced evaluation of both precision and recall. For instance, if the F1-score for class 4 is 0.40, it denotes the harmonic mean of precision and recall for that specific class.
- Support: It refers to the count or number of actual observations present in each class.
- Accuracy: Model accuracy is a metric that represents the overall correctness of the model's predictions. It is determined by the ratio of correctly predicted observations to the total number of observations.
- Macro avg: It is the average of the metrics calculated for each class, where each class is weighted equally.
- Weighted avg: It is the average of the metrics calculated for each class, where each class is weighted by its proportion in the dataset.

The accuracy of the model on the test set is 42.13%, which is not very good. The macro-average F1-score is 0.32, which indicates that the model has a poor performance on the dataset. The weighted average F1-score is 0.38, which means that the model is slightly better in predicting the classes with more samples. Overall, this model needs improvement to better classify the images into their respective categories.

- Heatmap: In the next step a heatmap if generated of a normalized confusion matrix for the predictions made by the model on the test set.

  First, the confusion matrix is computed using the confusion_matrix() function from scikit-learn, with the normalize='true' argument to normalize the matrix by row (i.e., divide each row by the sum of the corresponding row to get the proportion of true labels that were classified as each predicted label).

  Then, the heatmap() function from the Seaborn library is used to create a color-coded visualization of the matrix. The annot=True argument adds the numerical values of the matrix to the heatmap. The xticklabels and yticklabels arguments specify the labels to be displayed on the x- and y-axis of the heatmap, respectively. Finally, the title, axis labels, and font sizes are set to create a clear and readable visualization of the confusion matrix.

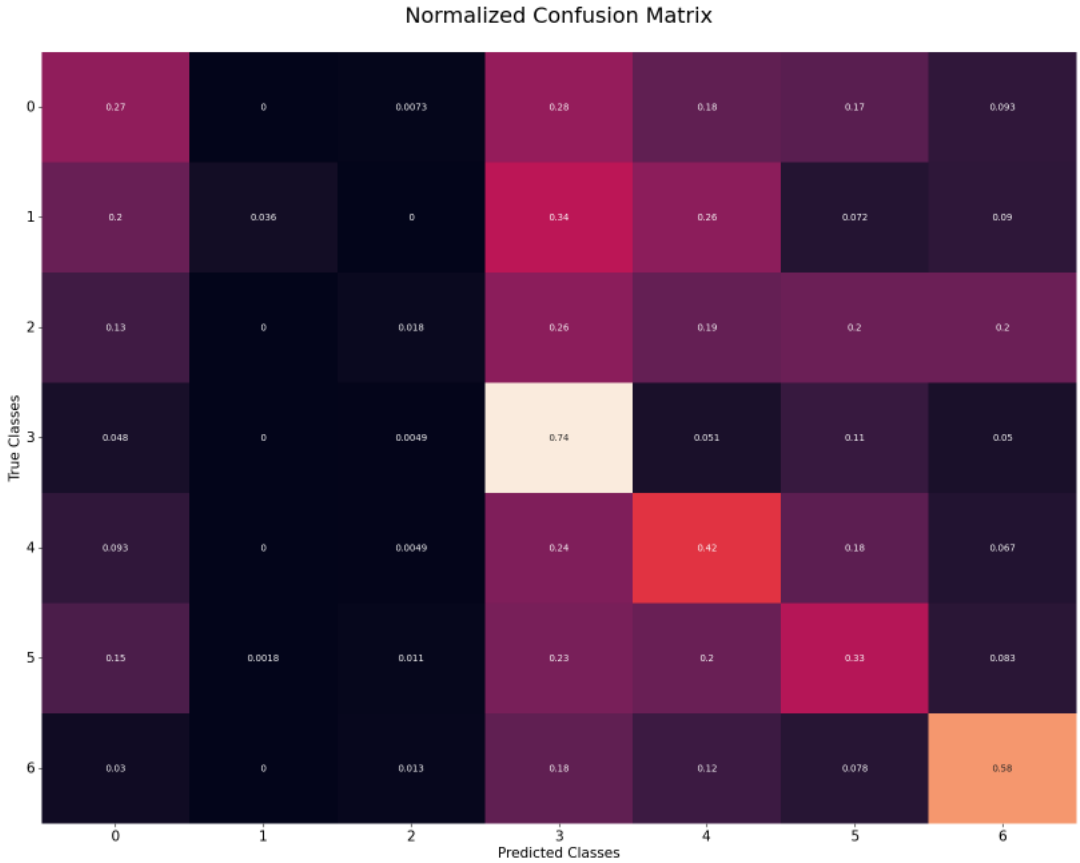


Figure 13 Confusion Matrix

- Fraction of incorrect prediction: Below is plotting a bar chart that shows the fraction of incorrect predictions made by the model for each true label. The calculation of the fraction of incorrect predictions is done by subtracting the diagonal elements (correct predictions) from 1 and dividing the resulting values by the sum of predictions for each true label. The resulting values are then plotted on the y-axis of the bar chart against the true labels on the x-axis.

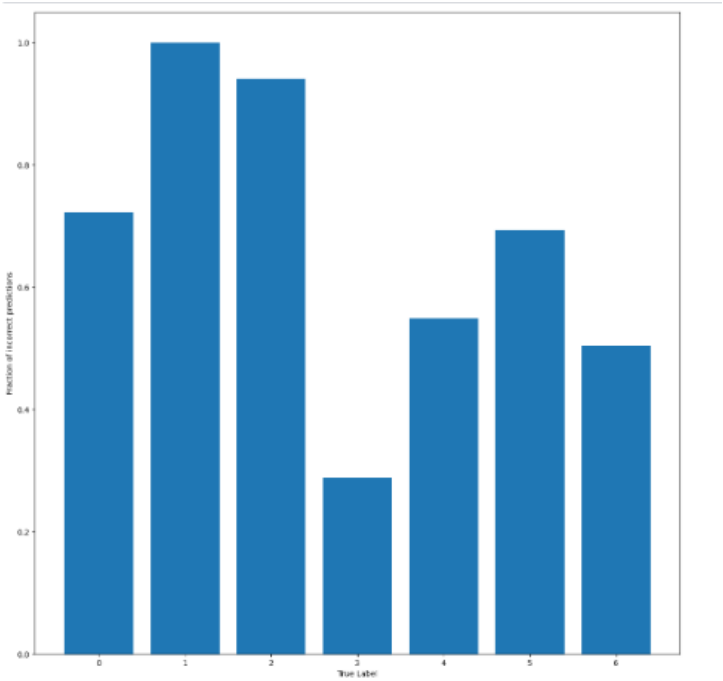


Figure 14 Fraction of incorrect prediction

5.6 Discussion

The Jonathan Oheix dataset is a publicly available dataset on Kaggle that contains facial expression images with labels for seven different emotions. There is various study who are using this dataset and obtained the good results such as: (Sidhu et al., 2022) in this paper they were using CNN model, which showed 53% accuracy. (Singh et al., 2021) in this research, an effective and resilient deep learning method is introduced to classify the sentiment of human faces. The approach utilizes Bidirectional Long-Short Term Memory (Bi-LSTM) and Recurrent Neural Networks (RNNs) to perform object-based segmentation in images. Furthermore, a CNN-RNN model is applied to non-linearly map the data, the model showed 95+% accuracy. (Dr. Sumathy P & Agilan C, 2020) focuses on the use of optimization techniques-based feature extraction techniques such as Ant Colony Optimization, Particle Swarm Optimization, and

Genetic Algorithm to enhance the recognition of human emotions using facial images. The performance of these techniques is evaluated using various metrics and CNN showed about 55 to 65 % accuracy for various techniques.

Pre-processing is a crucial step in preparing data for a model in stacked bi-directional LSTM or any other machine learning/deep learning algorithm. (Dr. Sumathy P & Chandrasekaran, 2022) explained various type of pre-processing technique for Jonathan Oheix dataset. Here, in this study, pre-processing was done by defining the path to the directory containing the images for the training and validation sets, loading the images from the training and validation directories using the load_img and img_to_array functions from Keras, and generating augmented image data for the training set using the ImageDataGenerator class from Keras. Additionally, EDA was performed on the dataset to identify any potential issues or challenges that may arise during model building and ensure that the final model is accurate and effective.

Data augmentation was performed using the ImageDataGenerator class from the Keras library. The datagen_train object performs rescaling of pixel values, shear transformation, zooming in/out, and horizontal flipping, while the datagen_val object only performs rescaling of pixel values. The flow_from_directory method of the ImageDataGenerator class was used to create train_set and test_set, which reads images from directories containing the training and validation sets, respectively, and resizes them to the target_size specified in the arguments. It also specifies that the images are in grayscale format with the color_mode argument, and the labels are categorical with class_mode='categorical'.

In addition, a grid search was performed to optimize the hyperparameters of a Keras model using the GridSearchCV function from scikit-learn. The hyperparameters tuned were row_hidden and col_hidden, which control the number of hidden units in the rows and columns of a bidirectional LSTM layer. The create_model function defined the model architecture, including a bidirectional LSTM layer followed by four standard LSTM layers for the row encoding, and a single LSTM layer for the column encoding. The output of the LSTM layers was flattened and passed through two fully connected Dense layers before being classified into the appropriate number of classes using a softmax activation function.

Overall, the pre-processing and EDA steps taken in this study helped to create a larger and more diverse dataset for training the stacked bi-LSTM model. By applying transformations to the input images, the model can learn to recognize facial expressions more accurately and robustly, leading to better performance on unseen data. The grid search was an effective method to optimize the hyperparameters of the model and improve its performance.

In this case, the main objective of the study was to select the proper hyperparameters for the model, which are the row_hidden and col_hidden parameters that control the number of hidden units in the rows and columns of a bidirectional LSTM layer. At initial step col_hidden and row_hidden is used as 64, 64 however the overall model accuracy showed near about 34% accuracy. After that the grid search algorithm was used to explore a range of possible hyperparameters and select the optimal combination that yields the best performance on the validation data. The best hyperparameters were found to be col_hidden = 64 and row_hidden = 32, with a best score of 0.264217.

The training of the neural network model involved iteratively optimizing its weights and biases to minimize the loss function on a given dataset over several epochs. The model was trained for 32 epochs, and its performance was monitored by computing the training loss and accuracy and the validation loss and accuracy. The model was saved whenever the validation accuracy improved, indicating that the model had learned to generalize better to new data. The model was trained using the Stochastic Gradient Descent (SGD) optimizer, with a learning rate of 0.001. The model's performance improved over time, as indicated by the increasing validation accuracy, but there were some epochs where the validation accuracy did not improve or even decreased, suggesting overfitting.

The visualizations showed the training history of the proposed model. The accuracy and validation accuracy metrics were plotted as a function of the epoch number, with the x-axis representing the number of epochs and the y-axis representing the accuracy. The loss and validation loss metrics were also plotted as a function of the epoch number, with the loss representing how well the model was able to predict the correct output for a given input.

Finally, the performance of the model on the test set was evaluated by generating predictions using the model.predict_generator() function and comparing them to the true labels of the test set using the accuracy_score() function. The accuracy on the test set was found to be 42.13%. The classification report provided an overview of the performance of the model on each class in the dataset, including precision, recall, f1-score, and support for each class.

Overall, the study successfully selected the proper hyperparameters for the model and trained a neural network model that achieved a reasonable accuracy on the test set. The study also highlighted the importance of monitoring the model's performance during training to prevent overfitting and improve generalization performance.

5.6 Summary

The Results and Discussion section of a thesis paper is crucial as it presents and interprets the findings of the research. It aims to answer research questions, provide evidence to support objectives, and discuss the implications of the results. The section includes an introduction that sets the context, a clear presentation of raw data and statistical analysis, descriptive and inferential statistics, interpretation and evaluation of the results, comparison with previous research, identification of limitations, and suggestions for future research.

In this particular study, the Results section starts with an exploration of hyperparameter selection using GridSearchCV. The initial values for row_hidden and col_hidden were set as 64, resulting in an accuracy of around 34%. The grid search algorithm was used to find the best combination of hyperparameters, and the best score achieved was 0.264217 with col_hidden = 64 and row_hidden = 32.

The next part of the Results section discusses the training of a neural network model over 10 epochs. The model's progress is evaluated using training and validation data, and the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001 is used. The validation accuracy shows improvement over time, but there are some epochs where it does not improve or decreases, indicating possible overfitting.

Visualizations are provided to illustrate the training history, including accuracy and loss plots. The accuracy and validation accuracy metrics are plotted against the epoch number, and the loss and validation loss metrics are also plotted. These visualizations help understand the model's performance throughout the training process.

The next part of the Results section focuses on the prediction of the test dataset. The model's predictions are evaluated using classification metrics such as accuracy, precision, recall, F1-score, and support. The accuracy on the test set is found to be 42.13%, indicating the model's performance. A classification report is generated to provide detailed evaluation metrics for each class in the dataset.

A heatmap of a normalized confusion matrix is created to visualize the predictions made by the model on the test set. This helps identify the patterns of correct and incorrect predictions. Additionally, a bar chart is plotted to show the fraction of incorrect predictions for each true label, providing further insight into the model's performance.

The Discussion section of the thesis highlights previous studies that have used the Jonathan Oheix dataset and achieved good results using different models and techniques. It mentions the importance of pre-processing and exploratory data analysis (EDA) in this study. Data augmentation techniques were applied using the ImageDataGenerator class from Keras. The

grid search was employed to optimize hyperparameters, and the model's performance was improved.

Overall, the Results and Discussion section highlights the importance of thorough data analysis, deriving meaningful insights, and expanding the existing knowledge in the field. It also acknowledges the potential for further enhancement in the model's performance and proposes avenues for future research.

CHAPTER 6: CONCLUSIONS

6.1 Introduction

In this thesis, we have explored and analyzed emotions from facial expression with the aim of increase the prediction accuracy in detecting emotion from facial expressions by stacked bidirectional LSTM with the selection of proper hyperparameters with the help of GridSearchCV. Throughout the research process, we have delved into various aspects of emotions detection, examining its impact on hyperparameter, model architecture, investigating its underlying mechanisms, and assessing its potential implications.

Through extensive data collection and analysis, we have gathered valuable insights into emotion detection. Our findings have shed light on various parameter, providing a deeper understanding of the complex dynamics and intricacies surrounding emotion detection. Moreover, our research has contributed to the existing body of knowledge in the field, extending the current literature on emotion detection.

By employing emotion detection from facial expression by stacked bi-directional LSTM, we were able to gather robust and reliable data. The use of the study methodology allowed us to explore how to select hyperparameter and performance of stacked bi-LSTM and address the gaps in existing knowledge. The results obtained from our analysis have provided empirical evidence to support our hypotheses and research objectives.

Furthermore, our study has highlighted the practical implications of our findings. The insights gained from this research can be applied to various sector like healthcare, marketing, automobile, etc. These practical implications pave the way for future research and open new avenues for further investigation.

However, it is essential to acknowledge the limitations of our study. Large dataset should be taken into consideration when interpreting our results. These limitations point to areas for future research and suggest avenues for improvement in the methodology or approach employed.

In conclusion, this thesis has significantly contributed to the understanding of emotion detection from facial expression. The comprehensive analysis and findings presented herein have expanded our knowledge in this study and provided a solid foundation for future research. By addressing the research objectives outlined at the outset, we have made significant strides in advancing knowledge in the field of emotion detection. It is our hope that this thesis serves as a catalyst for further exploration.

6.2 Discussion and Conclusion

In conclusion, this study aimed to select the proper hyperparameters for a model and improve its performance. The initial model with row_hidden and col_hidden set to 64 achieved an accuracy of approximately 34%. To optimize the model, a grid search algorithm was utilized to explore different combinations of hyperparameters. The best combination found was col_hidden = 64 and row_hidden = 32, resulting in a best score of 0.264217.

The training process involved iteratively optimizing the model's weights and biases over 32 epochs using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001. The model's progress was monitored through the training and validation loss and accuracy metrics. The model demonstrated improvement over time, although some epochs showed no improvement or even a decrease in validation accuracy, indicating potential overfitting.

Visualizations were utilized to illustrate the training history of the model, showing the accuracy and loss metrics for both the training and validation datasets. The plots demonstrated the model's performance across epochs, providing insights into its learning progress.

The model's performance on the test dataset was evaluated by generating predictions using the trained model and comparing them to the true labels. The accuracy achieved on the test set was 42.13%. The classification report provided detailed metrics, such as precision, recall, f1-score, and support, for each class in the dataset. The macro-average F1-score was 0.32, indicating poor performance overall, while the weighted average F1-score was 0.38, indicating slightly better performance on classes with more samples.

It is worth noting that the obtained accuracy on the test set was not very high, suggesting room for improvement in the model's classification capabilities. Further research and refinement are needed to enhance the model's ability to accurately classify facial expressions into their respective categories.

This study contributed to the field by emphasizing the importance of pre-processing and exploratory data analysis (EDA) to enhance the dataset's diversity and improve model performance. Data augmentation techniques, including rescaling, shear transformation, zooming, and flipping, were employed to generate augmented image data. Additionally, a grid search was utilized to optimize hyperparameters, resulting in improved model performance.

In conclusion, this study successfully selected optimal hyperparameters, trained a neural network model, and evaluated its performance. However, further improvements are necessary to achieve higher accuracy in classifying facial expressions. Future research can focus on refining the model architecture, exploring additional hyperparameters, or considering

alternative techniques used by other studies to achieve better results with the Jonathan Oheix dataset.

6.3 Contribution to Knowledge

The contribution to knowledge can be summarized as follows:

- Emphasis on pre-processing and exploratory data analysis (EDA): The study highlights the importance of preprocessing and EDA techniques to enhance the dataset's diversity and improve model performance. By employing data augmentation techniques such as rescaling, shear transformation, zooming, and flipping, the study demonstrates how these approaches can contribute to improving the performance of facial expression classification models.
- Optimization of hyperparameters: The study utilizes a grid search algorithm to optimize the model's hyperparameters. By exploring different combinations of hyperparameters, the study identifies the best combination that leads to improved model performance. This contribution showcases the significance of hyperparameter optimization in enhancing the accuracy of facial expression classification models.
- Evaluation and insights into model performance: The study evaluates the model's performance on a test dataset and provides detailed metrics such as accuracy, precision, recall, f1-score, and support for each class. The analysis reveals areas where the model performs well and areas where it needs improvement. This contribution offers valuable insights into the strengths and weaknesses of the model and provides a basis for future improvements.
- Identification of areas for further research: The study acknowledges that the obtained accuracy on the test set is not very high, indicating room for improvement in the model's classification capabilities. By highlighting this limitation, the study identifies the need for further research and refinement to enhance the model's ability to accurately classify facial expressions. This contribution sets the stage for future investigations to focus on refining the model architecture, exploring additional hyperparameters, or considering alternative techniques to achieve better results.

In summary, this study contributes to the field of facial expression classification by emphasizing the importance of preprocessing, EDA, hyperparameter optimization, and performance evaluation. It provides insights into the potential areas for improvement and paves the way for future research endeavors to enhance the accuracy and effectiveness of facial expression classification models.

6.4 Future Recommendations

Based on the findings and limitations of this study, several future recommendations can be made:

- Refine the model architecture: The current model used in this study can be improved by incorporating additional layers or adjusting the number of neurons in each layer. More complex models, such as Convolutional Neural Networks (CNNs), may also be explored to improve the model's ability to extract features from the images.
- Explore additional hyperparameters: Although this study explored a range of hyperparameters, there may be other important parameters that were not considered. Future studies can investigate the effect of additional hyperparameters, such as the learning rate or regularization strength.
- Consider alternative datasets: The Jonathan Oheix dataset used in this study is relatively large and may not be representative of all possible facial expressions. Future studies can consider more diverse datasets to train and test their models.
- Utilize transfer learning: Transfer learning involves using a pre-trained model as a starting point and fine-tuning it for a specific task. This technique can potentially improve the model's performance and reduce training time.
- Incorporate additional techniques: Other techniques, such as ensemble learning or active learning, may be incorporated to improve the model's accuracy and efficiency.

By addressing these future recommendations, the field of facial expression recognition can continue to advance, ultimately leading to more accurate and effective models for a range of applications.

REFERENCES

Abdullah, M., Ahmad, M., & Han, D. (2020). Facial Expression Recognition in Videos: An CNN-LSTM based Model for Video Classification. 2020 International Conference on Electronics, Information, and Communication (ICEIC), 1–3. https://doi.org/10.1109/ICEIC49074.2020.9051332

Agrawal, I., Kumar, A., Swathi, D., Yashwanthi, V., & Hegde, R. (2021). Emotion Recognition from Facial Expression using CNN. 2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC), 01–06. https://doi.org/10.1109/R10-HTC53172.2021.9641578

Ahmad, G. N., Fatima, H., Ullah, S., Salah Saidi, A., & Imdadullah. (2022). Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques With and Without GridSearchCV. IEEE Access, 10, 80151–80173. https://doi.org/10.1109/ACCESS.2022.3165792

Ahmed, T. U., Hossain, S., Hossain, M. S., ul Islam, R., & Andersson, K. (2019). Facial Expression Recognition using Convolutional Neural Network with Data Augmentation. 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (IcIVPR), 336–341. https://doi.org/10.1109/ICIEV.2019.8858529

Atef, S., & Eltawil, A. B. (2020). Assessment of stacked unidirectional and bidirectional long short-term memory networks for electricity load forecasting. Electric Power Systems Research, 187, 106489. https://doi.org/10.1016/j.epsr.2020.106489

Babajee, P., Suddul, G., Armoogum, S., & Foogooa, R. (2020). Identifying Human Emotions from Facial Expressions with Deep Learning. 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), 36–39. https://doi.org/10.1109/ZINC50678.2020.9161445

Bandyopadhyay, S. K. (2020). Stacked Bi-directional LSTM Layer Based Model for Prediction of Possible Heart Disease during Lockdown Period of COVID-19. Journal of Advanced Research in Medical Science & Technology, 07(02), 10–14. https://doi.org/10.24321/2394.6539.202006

Begaj, S., Topal, A. O., & Ali, M. (2020). Emotion Recognition Based on Facial Expressions Using Convolutional Neural Network (CNN). 2020 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA), 58–63. https://doi.org/10.1109/CoNTESA50436.2020.9302866

Chittora, V., & Gupta, C. P. (2020). Dynamic Spot Price Forecasting Using Stacked LSTM Networks. 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 1080–1085. https://doi.org/10.1109/ICISS49785.2020.9315988

Dandıl, E., & Karaca, S. (2021). Detection of pseudo brain tumors via stacked LSTM neural networks using MR spectroscopy signals. Biocybernetics and Biomedical Engineering, 41(1), 173–195. https://doi.org/10.1016/j.bbe.2020.12.003

Dr. Sumathy P, & Agilan C. (2020). ENHANCING THE HUMAN EMOTION RECOGNITION WITH FEATURE EXTRACTION TECHNIQUES. International Journal of Electrical Engineering and Technology (IJEET) , 11(10), 382–391. https://d1wqtxts1xzle7.cloudfront.net/81864994/IJEET_11_10_049-libre.pdf?1646716627=&response-content-disposition=inline%3B+filename%3DENHANCING_THE_HUMAN_EMOTION_RECOGNITION.pdf&Expires=1683830145&Signature=WGd2YNIu7ESotiotxWYu4bOk52RC2EFa7Jaazxp2FfAVHXY0c898LMF9MUqlv0rwq~vkly37A2zIq4fvmkfadEk1LiXQFXeIq0SPEt2yVm0aIfM5eG132dtxRxmyu0pXWsKcfmd3keI6Cqi40KWAaJgqAkENDO7Q7xFzMCYvACRB0isArSaCKVci7dWUGTz4Vy-GaQHjHMMGuMHPrB-FnfWZoBSOKvTj9aI9t0xLeq1Z4ScEHsu4~A3ICTpM5jI5I2LLTjeW1rKncIdFkMQMsOHNrafX5ISAo0WFZlU-SAtKeTkQ8RE~oOzRZlESeYOuGVUMPE0smnvj5g9GAT7Bqw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Dr. Sumathy P, & Chandrasekaran, A. (2022). PRE-PROCESSING TECHNIQUES FOR FACIAL EMOTION RECOGNITION SYSTEM. International Journal of Advanced Research in Engineering and Technology (IJARET), 11(5), 669–677.

Fares, A., Zhong, S., & Jiang, J. (2019). EEG-based image classification via a region-level stacked bi-directional deep learning framework. BMC Medical Informatics and Decision Making, 19(S6), 268. https://doi.org/10.1186/s12911-019-0967-9

Febrian, R., Halim, B. M., Christina, M., Ramdhan, D., & Chowanda, A. (2023). Facial expression recognition using bidirectional LSTM - CNN. Procedia Computer Science, 216, 39–47. https://doi.org/10.1016/j.procs.2022.12.109

Ganguly, B., Chatterjee, A., Mehdi, W., Sharma, S., & Garai, S. (2020). EEG Based Mental Arithmetic Task Classification Using a Stacked Long Short Term Memory Network for Brain-Computer Interfacing. 2020 IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS), 89–94. https://doi.org/10.1109/VLSIDCS47293.2020.9179949

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., & Schmidhuber, J. (2009). A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5), 855–868. https://doi.org/10.1109/TPAMI.2008.137

Hung, B. T., & Tien, L. M. (2021). Facial Expression Recognition with CNN-LSTM (pp. 549–560). https://doi.org/10.1007/978-981-15-7527-3_52

Hussien Mary, A., Bilal Kadhim, Z., & Saad Sharqi, Z. (2020). Face Recognition and Emotion Recognition from Facial Expression Using Deep Learning Neural Network. IOP Conference Series: Materials Science and Engineering, 928(3), 032061. https://doi.org/10.1088/1757-899X/928/3/032061

Istiake Sunny, Md. A., Maswood, M. M. S., & Alharbi, A. G. (2020). Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model. 2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), 87–92. https://doi.org/10.1109/NILES50944.2020.9257950

Jaiswal, A., Krishnama Raju, A., & Deb, S. (2020). Facial Emotion Detection Using Deep Learning. 2020 International Conference for Emerging Technology (INCET), 1–5. https://doi.org/10.1109/INCET49848.2020.9154121

K.S., J., & David, D. S. (2020). A Novel Based 3D Facial Expression Detection Using Recurrent Neural Network. 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), 1–6. https://doi.org/10.1109/ICSCAN49426.2020.9262287

Kumar Arora, T., Kumar Chaubey, P., Shree Raman, M., Kumar, B., Nagesh, Y., Anjani, P. K., Ahmed, H. M. S., Hashmi, A., Balamuralitharan, S., & Debtera, B. (2022). Optimal Facial Feature Based Emotional Recognition Using Deep Learning Algorithm. Computational Intelligence and Neuroscience, 2022, 1–10. https://doi.org/10.1155/2022/8379202

Kumar, Dr. M., & Patidar, A. (2021). Sarcasm Detection Using Stacked Bi-Directional LSTM Model. 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 1–5. https://doi.org/10.1109/ICAC3N53548.2021.9725488

Kumar, M., & Srivastava, S. (2021). Emotion Detection through Facial Expression using DeepLearning. 2021 5th International Conference on Information Systems and Computer Networks (ISCON), 1–4. https://doi.org/10.1109/ISCON52037.2021.9702451

Li, X., & Wu, X. (2014). Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition.

Li, Y., & Lin, Z. (2020). Melody Classifier with Stacked-LSTM.

Lim, J. Y., Lim, K. M., & Lee, C. P. (2021). Stacked Bidirectional Long Short-Term Memory for Stock Market Analysis. 2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), 1–5. https://doi.org/10.1109/IICAIET51634.2021.9573812

Lorente, Ò., Riera, I., & Rana, A. (2021). Image Classification with Classic and Deep Learning Techniques.

Ma, M., Liu, C., Wei, R., Liang, B., & Dai, J. (2022). Predicting machine's performance record using the stacked long short-term memory (LSTM) neural networks. Journal of Applied Clinical Medical Physics, 23(3). https://doi.org/10.1002/acm2.13558

Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., & Schmidhuber, J. (2006). A System for Robotic Heart Surgery that Learns to Tie Knots Using Recurrent Neural Networks. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 543–548. https://doi.org/10.1109/IROS.2006.282190

Mostafa, A., Khalil, M. I., & Abbas, H. (2018). Emotion Recognition by Facial Features using Recurrent Neural Networks. 2018 13th International Conference on Computer Engineering and Systems (ICCES), 417–422. https://doi.org/10.1109/ICCES.2018.8639182

OHEIX JONATHAN. (n.d.). Face expression recognition dataset. Https://Www.Kaggle.Com/Datasets/Jonathanoheix/Face-Expression-Recognition-Dataset.

Pattanaik, S., Behera, S., Majhi, S. K., & Dwibedy, P. K. (2022). Stacked BiLSTM with ResNet50 for Medical Image Classification. 2022 IEEE Region 10 Symposium (TENSYMP), 1–6. https://doi.org/10.1109/TENSYMP54529.2022.9864353

Qazi, A. S., Farooq, M. S., Rustam, F., Villar, M. G., Rodríguez, C. L., & Ashraf, I. (2022). Emotion Detection Using Facial Expression Involving Occlusions and Tilt. Applied Sciences, 12(22), 11797. https://doi.org/10.3390/app122211797

Qiu, C., Li, K., Zhou, X., He, S., & Li, B. (2023). A novel method for signal labeling and precise location in a variable parameter milling process based on the stacked-BiLSTM-CRF and FLOSS. Advanced Engineering Informatics, 55, 101850. https://doi.org/10.1016/j.aei.2022.101850

Rajan, S., Chenniappan, P., Devaraj, S., & Madian, N. (2020). Novel deep learning model for facial expression recognition based on maximum boosted CNN and LSTM. IET Image Processing, 14(7), 1373–1381. https://doi.org/10.1049/iet-ipr.2019.1188

Sahidullah, M., Patino, J., Cornell, S., Yin, R., Sivasankaran, S., Bredin, H., Korshunov, P., Brutti, A., Serizel, R., Vincent, E., Evans, N., Marcel, S., Squartini, S., & Barras, C. (2019). The Speed Submission to DIHARD II: Contributions & Lessons Learned.

Said, Y., & Barr, M. (2021). Human emotion recognition based on facial expressions via deep learning on high-resolution images. Multimedia Tools and Applications, 80(16), 25241–25253. https://doi.org/10.1007/s11042-021-10918-9

Shinde, N., S, C., Patil, S. A., Siri Chandana, K., Pendari, N. T., Hiremath, P. G. S., & Gangisetty, S. (2021). Stacked LSTM Based Wafer Classification. 2021 IEEE International Conference on Big Data (Big Data), 5786–5790. https://doi.org/10.1109/BigData52589.2021.9671835

Sidhu, P. K., Kapoor, A., Solanki, Y., Singh, P., & Sehgal, D. (2022). Deep Learning Based Emotion Detection in an Online Class. 2022 IEEE Delhi Section Conference (DELCON), 1–6. https://doi.org/10.1109/DELCON54057.2022.9752940

Singh, C., Wibowo, S., & Grandhi, S. (2021). A Deep Learning Approach for Human Face Sentiment Classification. 2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter), 28–32. https://doi.org/10.1109/SNPDWinter52325.2021.00015

Uddin, R., Alam, F. I., Das, A., & Sharmin, S. (2022). Multi-Variate Regression Analysis for Stock Market price prediction using Stacked LSTM. 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET), 474–479. https://doi.org/10.1109/ICISET54810.2022.9775911

Ullah, M., Ullah, H., Khan, S. D., & Cheikh, F. A. (2019). Stacked Lstm Network for Human Activity Recognition Using Smartphone Data. 2019 8th European Workshop on Visual Information Processing (EUVIP), 175–180. https://doi.org/10.1109/EUVIP47703.2019.8946180

Wang, W., Sun, Q., Chen, T., Cao, C., Zheng, Z., Xu, G., Qiu, H., & Fu, Y. (2019). A Fine-Grained Facial Expression Database for End-to-End Multi-Pose Facial Expression Recognition.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

Xu, X., Liu, C., Zhao, Y., & Lv, X. (2022). Short-term traffic flow prediction based on <scp>whale optimization algorithm</scp> optimized BiLSTM_Attention. Concurrency and Computation: Practice and Experience, 34(10). https://doi.org/10.1002/cpe.6782

Yu, Z., Liu, G., Liu, Q., & Deng, J. (2018). Spatio-temporal convolutional features with nested LSTM for facial expression recognition. Neurocomputing, 317, 50–57. https://doi.org/10.1016/j.neucom.2018.07.028

Zhang, Q., Zhang, J., Zou, J., & Fan, S. (2020). A Novel Fault Diagnosis Method based on Stacked LSTM. IFAC-PapersOnLine, 53(2), 790–795. https://doi.org/10.1016/j.ifacol.2020.12.832

APPENDIX A: RESEARCH PROPOSAL

Abstract

Automatic facial expression recognition is an emerging study in emotion recognition. Emotions play a significant role in understanding people and are usually related to good decisions, behavior, human activities, and intellect. Disappointment(sad), happiness, contempt, anger, worry(fear), marvel(surprise), and neutrality are the main seven emotions of human assumed. Various models of deep learning like Long Short Term Memory (LSTM), RNN(Recurrent neural network), Convolution Neural network (CNN), LSTM-CNN, etc architectures were collated to detect emotions. Stacked LSTM is proposed to make better accuracy and training time for detection of emotion from facial expression. To implementation of the model, application takes an image input from the user and help to identify the human emotions. Study will give an idea how multiple layers are incorporate to each other to give prediction in terms of accuracy. Generally, the LSTM layers generate a sequential output to the next LSTM layer. It has applied by adding multiple layers and then fit the model.

1. Introduction

In recent era, technology is collaborating with the human brain, and artificial intelligence is a tool that allows us to replicate artificial brain functions. It was in the 19th century that the study of emotions first began, however, Darwin showed in the year of 1872, “The Expression of the Emotions in Man and Animals”, natural choice and evolution had been implemented to the look at of human communique. Emotions and technology are engines of human artificial brain and this art of reading and detecting facial emotions unchallenging possible with deep learning. Emotions of Human maybe classified as: fear, disrespect, disgust, anger, surprise, dismal, satisfaction, and noncommittal. Facial emotion identification is applicable for various sectors like healthcare, improve video marketing, automotive applications, navy as well as military programs, personality checks, movie preparation, crook distinguishing evidence, security framework, labeling functions, and human-system interaction. In general, facial expression, recognition can be done by two different techniques: Camera-Based and Bio-signals. On the other hand, the deep learning technique is a general paradigm to represent the running of the human brain with neurons with the technique of CNN, RNN, LSTM etc. The goals of this proposal are to use the Stacked LSTM technique of RNN to provide a system that enabled with processing of images and machine learning algorithms to pick out and classify some basic emotions of humans possessed by an individual. Stacked LSTM is an enlargement to this model

that has multiple hidden LSTM layers in which every layer consists of n-numbers of memory cells.

2. Related Research

(Agrawal et al., 2021) categorized seven emotions contempt, happiness, Sadness, fear, surprise, anger, and Neutral by using CNN model to get better training time and accuracy. They have used a dataset containing 35k images, however, there is a different accuracy for a different group of emotions, and also time-consuming to train the process.

(Yang et al., 2018): In this study, they are investigating emotion detection on the basis of facial expressions and, allowing identification of a scholar's studying fame in real-time during a virtual learning environment. They have studied Haar Cascades method and used it in less number of images. However, it's quite challenging to achieve an accuracy as it's an analysis of movable emotions randomly on a real-time basis.

(Kabir et al., 2021) proposed an architecture based totally based on LSTM and CNN that classifies 6 main expressions of humans: anger, worry, disgust, happiness, disappointment, and marvel with impartiality from each regular as well as candid posed snapshots. They established significant accuracy in a study based on CNN-LSTM.

(Ming et al., 2022) An ACNN (attention based CNN)-ALSTM (attention based LSTM) model for facial features popularity that incorporate procedure is suggested. Comparative exploratory consequences showed that the creation of 2-layer awareness procedure that improves the class overall presentation of system extra considerably than that of CNN-LSTM model and its variants.

(Lyons et al., 1999) offered a way to categorize single facial snapshots. However, we will investigate the facial image of more than 35000.

(Alizadeh and Fazel, 2017) showed different type of CNN architectures to remedy a emotion expression (facial) issue with various deepness using grayscale photos, and its output became evaluated using more than one submit-processing and visualization techniques

(Mellouk and Handouzi, 2020): here they described a review of recent advances in sensing feelings by way of spotting facial expressions through the use of one of a kind deep learning structure.

(Graves et al., 2013) accompanied that RNN is a strong model for subsequent data that LSTM has good efficiency for phoneme acknowledgment.

(Sundermeyer et al., 2012) presented that an LSTM community furnished higher overall performance than a well-known RNN for an English and a huge French language modeling assignment.

(Tripathi et al., 2018): here they have 512 & 256 units observed by a Dense layer with 512 units and Relu Activation

(Lee and Tashev, 2015): they put in force a Stacked LSTM model to are expecting the historic traffic statistics to show the sample of traffic volume.

3. Research Questionary

Some issues already been identified with the existing emotion detection system. The existing system trained on a small number of images, leading to this Decreased accuracy. they used the database to train the model with fewer pictures. Real time analysis Now possible. Evaluate each emotion separately

Useful for detecting dominants. It's a complementary feeling and it's not over yet. In this study, its is observed how stacked LSTM will perform in respective of accuracy and how much it will reduce the time for detecting the analysis in a significant way.

The following study questions are advised in this study's objective as follows.

- Explanation of Dataset in terms of contains.
- Study of efficacy whether Stacked LSTM gives more accuracy in real-time basis over other techniques like CNN, RNN etc
- Study of the architecture of Stacked LSTM
- Details of the phenomenon of building the model
- Study of detecting facial emotions with the help of Stacked LSTM
- Can Stacked LSTM detect emotions faster with more accuracy?

4. Aim and Objectives of Study

The ambition of this research is to established a technique where facial emotions are truly identified by using stacked LSTM method, which will give a more accurate and precise result. In the initial step of the study is to investigate the pattern of images, how it will be and it will be differentiate based on the six different emotions.

There are many technique which are already established for detecting the Facial emotions recognition such as: CNN, LSTM, RNN etc. The main objective is to established Stacked LSTM which will give a more accurate result or not because stacked LSTM having multiple hidden layers, in which every layer has multiple memory cells. In this work, evaluation of the accuracy and precision of stacked LSTM on facial emotion detection is measured.

5. Significance of Study

By using Stacked LSTM it will be easy to evaluate the emotion by the facial gesture.
It can be used in various industry like healthcare, IT-industry, education, bank, and everywhere, where the human is involved

6. Scope of Study

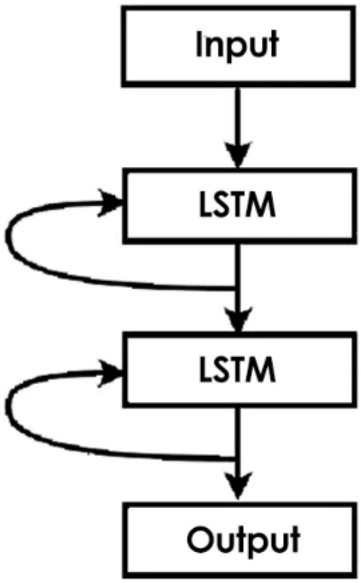
Stacking LSTM is to allow for greater model complexity. Stacking LSTM hidden layers makes the interpretation deeper, more suitably acquiring the outline as a technique of deep learning, however, growing the wide variety of layers/hidden gadgets in a neural community does not always bring about overfitting. Too few will bring about low schooling and check accuracies; too many will result in high training accuracy but low check accuracy (overfitting). Stacked LSTM may not provide significant results where there is video analysis on a real-time basis.

7. Research Methodology

LSTM is capable to detect long-term dependency problems. Forget gate is very important for LSTM, there are 3 states: input, hidden, and output state. There are many articles where the applications of Machine learning are greater in terms of Normal LSTM, CNN, RNN etc for detecting human facial emotions by using facial expressions. In this study, emotions from facial expressions are predicted by using a stacked LSTM model. Stacked hidden LSTM, can make the model deeper and more deserving in terms of deep learning techniques. It's mainly the depth of the neural network which is the main attribution to achieve the success of prospective to the wide range of predictions. On the other hand, hidden layers can add to multilayer perceptron neural networks to establish the model deeper. These extra layers are generally combined with learned representation of the previous layer and they make a new unreasonable level of representation.

7.1 Stacked LSTM Architecture

Stacked LSTMs at the moment are a solid method for difficult collection prediction troubles. A stacked LSTM structure can be described as an LSTM version along with more than one LSTM layers. An LSTM layer offers a series output rather than a single-value output to the LSTM layer. It is easy to learn hierarchical representation with the help of stacked LSTM.



Stacked LSTM Architecture

7.2 Implement stacked LSTMs in Keras

Stacked LSTM can easily create in Keras with the help of deep learning library. A single LSTM memory cell needed a 3D input. When an LSTM layer processes a single input order of time steps, in that time each memory cell gives output a single value for entering order in term s 2D sequences. Basically, in this particular LSTM we have to alter the configuration of previous layer to give an output in terms of 3D array which again use as an input to the next sequence of layer. It can be implemented by "return_sequences" argument on the layer to True (defaults to False). This particular method will give a single output value for each time step in the input sequence. We are able to keep as many hidden layers until it will give a 3D output.

7.3 Dataset

Dataset is collected from Kaggle platform (OHEIX JONATHAN, n.d.). This dataset is classified into 7 lessons with one-of-a-kind sentiments which includes anger, worry, neutral, happiness, sad, disgust, and wonder.

Every sentiment set of data has approximately two hundred-three hundred images with a total variety of 1400- 2100 photographs

7.4 Study details

Initial step of this model is facts preprocessing wherein data is ready in specific layout which will be granted via the network (https://in.mathworks.com/help/deeplearning/deep-learning-data-management-and-preprocessing.html, n.d.). This study will explain the multivariate analysis. With the help of sequential class of keras, stacked LSTM model will implemented. Sequential provide training and inference features on this model. This involves many steps from extraction to action of cleaning as well as loading of data. The primary assignment is to allow it decipherable by the network. After that next step is image augmentation(https://www.analyticsvidhya.com/blog/2021/03/image-augmentationtechniques-for-training-deep-learn, n.d.). The second step is to resize all the images in a particular size to obtain more accuracy in other words, imbalancing should be removed. there are two directories: one is trained another is tested. With the help of a data generator, the rescaling of images will be done. Photo augmentation finished leading in enough quantity of data among the present one with the purpose to train a model in a better version. The particular task is finished in the form of way of rotation, padding as well as some different strategies. The next step is Exploratory data analysis to analyze the sample of the image. The next vital step to facial expression is feature extraction followed by a data preprocessor where normalization of the image, covert of the image from label to integer, and configuration will do. The study will go forward with model creation with stacked LSTM by adding multiple LSTM layers one by one and executing the model. Then fit the model in the train, validation set. Then find the prediction between in train and validation dataset with the scaler. Inside the training method, the model has skilled on the data amassed to this point. After training of the model, results and analysis are established from the validation set.

A model with sufficient layers improves the LSTM's accuracy and dependability. Because of this, each layer's requirement for neurons decreases, and training time is also reduced.
As a result, the study's proposal is for a deeper optimized LSTM network.

We will perform the analysis by Performance Metrics: (TP: True Positive, TN: True Negative, FP: False positive, FN: False negative)

Decision Rate: (TP + TN) / (TP + TN + FP + FN)

Sensitivity: TP / (TP + FN)

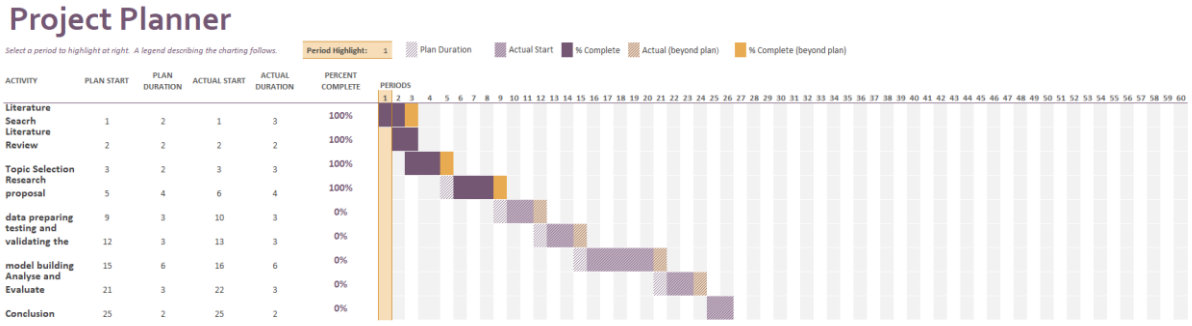
Specificity: TN / (TN + FP)

False positive rate: 1-Specificity, Miss rate: 1- Sensitivity

8. Requirements Resources

We will GPU system for this method. The model will be trained in Python. Numerous libraries used in this model such as Numpy, Matplotlib, Seaborn, Pandas, and so on. All these types of libraries perform a crucial position in data visualization as well data analysis. Numerous deep learning of libraries is imported via Keras such as sequential, Dense, LSTM etc.

9. Research Plan



References

Agrawal, I., Kumar, A., Swathi, D., Yashwanthi, V. and Hegde, R., (2021) Emotion Recognition from Facial Expression using CNN. In: 2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC). IEEE, pp.01–06.

Alizadeh, S. and Fazel, A., (2017) Convolutional Neural Networks for Facial Expression Recognition.

Anon (n.d.) https://in.mathworks.com/help/deeplearning/deep-learning-data-management-and-preprocessing.html.

Anon (n.d.) https://www.analyticsvidhya.com/blog/2021/03/image-augmentationtechniques-for-training-deep-learn.

Graves, A., Mohamed, A. and Hinton, G., (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, pp.6645–6649.

Kabir, Md.M., Anik, T.A., Abid, Md.S., Mridha, M.F. and Hamid, Md.A., (2021) Facial Expression Recognition Using CNN-LSTM Approach. In: 2021 International Conference on Science & Contemporary Technologies (ICSCT). IEEE, pp.1–6.

Lee, J. and Tashev, I., (2015) High-level feature representation using recurrent neural network for speech emotion recognition. In: Interspeech 2015. ISCA: ISCA, pp.1537–1540.
Lyons, M.J., Budynek, J. and Akamatsu, S., (1999) Automatic classification of single facial images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2112, pp.1357–1362.
Mellouk, W. and Handouzi, W., (2020) Facial emotion recognition using deep learning: review and insights. Procedia Computer Science, 175, pp.689–694.
Ming, Y., Qian, H. and Guangyuan, L., (2022) CNN-LSTM Facial Expression Recognition Method Fused with Two-Layer Attention Mechanism. Computational Intelligence and Neuroscience, 2022, pp.1–9.
OHEIX JONATHAN, (n.d.) Face expression recognition dataset. https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset.
Sundermeyer, M., Schlüter, R. and Ney, H., (2012) LSTM neural networks for language modeling. In: Interspeech 2012. ISCA: ISCA, pp.194–197.
Tripathi, S., Tripathi, S. and Beigi, H., (2018) Multi-Modal Emotion recognition on IEMOCAP Dataset using Deep Learning.
Yang, D., Alsadoon, A., Prasad, P.W.C., Singh, A.K. and Elchouemi, A., (2018) An Emotion Recognition Model Based on Facial Recognition in Virtual Learning Environment. Procedia Computer Science, 125, pp.2–10.