

FACIAL EXPRESSION RECOGNITION (FER) USING DEEP LEARNING NETWORK



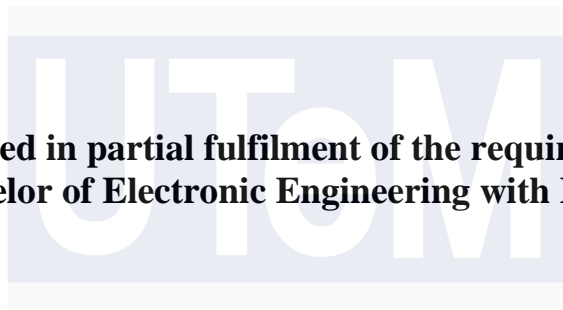
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACIAL EXPRESSION RECOGNITION (FER) USING DEEP LEARNING NETWORK

MUHAMMAD IRFAN BIN ZAIDI



**This report is submitted in partial fulfilment of the requirements for
the degree of Bachelor of Electronic Engineering with Honours**



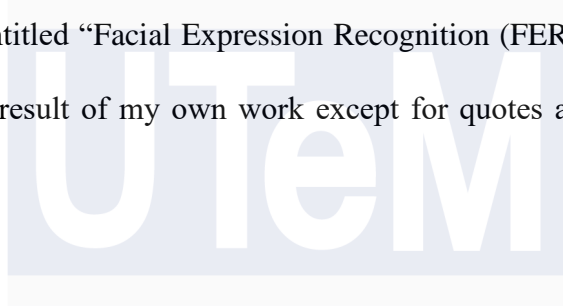
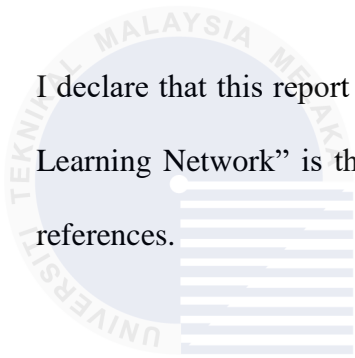
اونيورسيتي تيكنيكل مليسيا ملاك

**Faculty of Electronics and Computer Technology and
Engineering
Universiti Teknikal Malaysia Melaka**

2024

DECLARATION

I declare that this report entitled “Facial Expression Recognition (FER) Using Deep Learning Network” is the result of my own work except for quotes as cited in the references.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

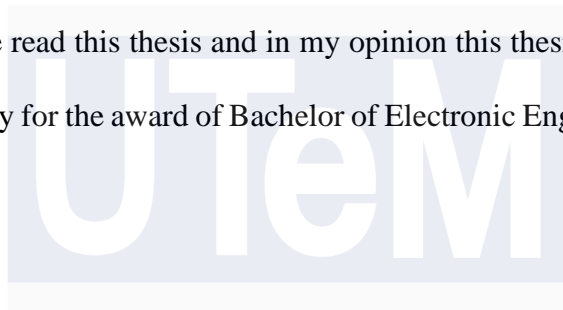
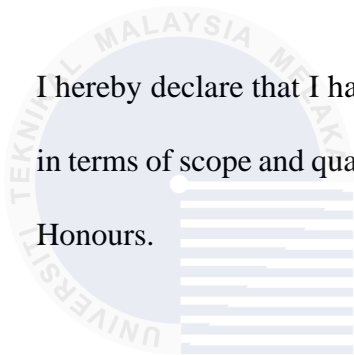
Signature :

Author : Muhammad Irfan Bin Zaidi
.....

Date : 22 January 2024
.....

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Signature : 

Supervisor Name : Prof Madya Dr Abd Majid Darsono

Date : 22 January 2024

DEDICATION

I express my gratitude to the Almighty, Allah, who is most gracious and merciful, for granting me the strength and patience to successfully complete my final year project.

Additionally, I am deeply thankful to my parents for supporting my educational pursuits. I would also like to extend my heartfelt appreciation to Dr. Abd. Majid Bin

Darsono for his invaluable guidance throughout this project. Lastly, I am grateful to all my friends for their assistance during completing this project.

ABSTRACT

Emotion recognition plays a significant role in measuring the emotions of a person. Since our faces are the most expressive parts of our bodies and are frequently used as indicators of our mental states. This project aims to develop facial expression recognition using deep learning technique for recognize seven different emotions such as angry, disgust, fear, happy, neutral, sad and surprise. However, developing a FER system by using FER2013 dataset based on machine learning have limitation in handling complex datasets. Therefore, this project involves deep learning technique by using Convolution Neural Network (CNN) model with data augmentation to handle the complex data and be able to extract facial features from the input images with reduce overfitting. As a result, our proposed model achieved the highest accuracy with 65.27% compared to the pretrained models where VGG16 with 42.11%, AlexNet with 24.71% and MobileNet with 28.35% on the FER2013 test dataset. The FER system also can help in the healthcare and education sector. Thus, this project can be achieved the SDG goal which is SDG 3 for Good Health and Well Being and SDG 4 for Quality Education.

ABSTRAK

Pengecaman emosi memainkan peranan penting dalam mengukur emosi seseorang. Memandangkan wajah kita adalah bahagian tubuh yang paling ekspresif dan sering digunakan sebagai penunjuk keadaan mental kita. Projek ini bertujuan untuk membangunkan pengecaman ekspresi muka menggunakan teknik pembelajaran mendalam untuk mengenali tujuh emosi berbeza seperti marah, jijik, takut, gembira, neutral, sedih dan terkejut. Walau bagaimanapun, membangunkan sistem FER dengan menggunakan set data FER2013 berdasarkan pembelajaran mesin mempunyai had dalam mengendalikan set data kompleks. Oleh itu, projek ini melibatkan teknik pembelajaran mendalam dengan menggunakan model Convolution Neural Network (CNN) dengan penambahan data untuk mengendalikan data yang kompleks dan dapat mengekstrak ciri muka daripada imej input tanpa overfitting. Hasilnya, model cadangan kami mencapai ketepatan tertinggi dengan 65.27% berbanding model pralatihan di mana VGG16 dengan 42.11%, AlexNet dengan 24.71% dan MobileNet dengan 28.35% pada set data ujian FER2013. Sistem FER juga boleh membantu dalam sektor penjagaan kesihatan dan pendidikan. Oleh itu, projek ini dapat mencapai matlamat SDG iaitu SDG 3 untuk Kesihatan dan Kesejahteraan yang Baik dan SDG 4 untuk Pendidikan Berkualiti.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest gratitude to who has given me support and guidance throughout the completion of the final year project. Your encouragement and help are invaluable, and I really appreciate it.

First, I express my deepest appreciation to my supervisor, Dr. Abd Majid Bin Darsono, for his unwavering support, expert guidance, and immense patience throughout the duration of this project. His deep knowledge in the field of machine learning has played an important role in shaping the direction of my work. Dr. Abd Majid always challenges me to think critically and strive for excellence. I am truly blessed to have he is my supervisor, and his guidance has been a source of inspiration for me.

Finally, I would like to express my appreciation to my friends for their motivation, and intellectual discussions. Your support has created something positive that helped me in completing this project.

TABLE OF CONTENTS

Declaration	
Approval	
Dedication	
Abstract	i
Abstrak	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	ix
List of Tables	xi
List of Symbols and Abbreviations	xii
List of Appendices	xv
CHAPTER 1 INTRODUCTION	1
1.1 Background of Project	2
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Project Impact	5

1.5	Scope of Project	5
1.5.1	Simulation	6
1.5.2	Restriction	7
1.6	Significant of Project	8
1.7	Chapter Outline	9

CHAPTER 2 BACKGROUND STUDY **12**

2.1	The Types of Emotion Recognition	13
2.1.1	Overview of Expression Recognition from Speech Analysis	13
2.1.1.1	Speech Expression Databases	14
2.1.1.2	Speech Expression Recognition using Deep Learning Technique	14
2.1.2	Overview of Expression Recognition from Body Gestures Analysis	16
2.1.2.1	Body Gesture Expression Databases	17
2.1.2.2	Body Gestures Expression Recognition using Deep learning Technique.	17
2.1.3	Overview of Expression Recognition from Facial Analysis	18
2.1.3.1	Facial Expression Databases	19
2.1.3.2	Facial Expression Recognition using Machine Learning Technique	25
2.1.3.3	Facial Expression Recognition using Deep Learning Technique (CNN)	32
2.2	Summary	39

CHAPTER 3 METHODOLOGY	40
3.1 Modern Tools	41
3.1.1 Jupyter Notebook	41
3.1.2 Google Colab	42
3.1.3 Open-Source Library	42
3.1.3.1 TensorFlow	43
3.1.3.2 KERAS	43
3.1.3.3 OpenCV	44
3.1.3.4 Matplotlib	44
3.1.3.5 NumPy	45
3.2 Proposed FER Algorithm using CNN.	45
3.2.1 Preparing and Processing Images in Database	47
3.2.2 Design the CNN Custom Model	49
3.2.2.1 Input Shape	50
3.2.2.2 Convolutional 2D Layer	51
3.2.2.3 Pooling Layer	52
3.2.2.4 Activation Function	54
3.2.2.5 Fully Connected Layer	55
3.2.3 Training and Observe the Validation Accuracy	56
3.2.3.1 Tuning Hyperparameter Batch Size	57

3.2.3.2	Tuning Hyperparameter Learning Rate	58
3.2.3.3	Tuning Hyperparameter Epochs	59
3.2.4	Testing the Model	59
3.2.5	Analyzing the Model Performance	60
3.2.6	Integrate the FER system in Real Time	62
3.3	Summary	63
CHAPTER 4 RESULTS AND DISCUSSION		65
4.1	Data Augmentation Analysis for Custom CNN Model	66
4.2	Hyperparameter Tuning for Custom CNN Model	68
4.2.1	Tuning Learning Rate and Batch size	68
4.2.2	Tuning Epochs	71
4.3	Analyzing the Performance of the Proposed Custom CNN Model	73
4.3.1	Model Accuracy and Loss	73
4.3.2	Confusion Matrix	74
4.4	Pretrained Models	78
4.4.1	Model Accuracy and Loss for VGG16 Model	79
4.4.2	Model Accuracy and Loss for AlexNet Model	80
4.4.3	Model Accuracy and Loss for MobileNet Model	81
4.4.4	Comparison Between a Proposed Model and All Pretrained Models	83
4.5	Real Time Monitoring using Webcam.	86

4.6	Summary	88
-----	---------	----

CHAPTER 5 CONCLUSION AND FUTURE WORKS **90**

5.1	Conclusion	91
-----	------------	----

5.2	Future Work	92
-----	-------------	----

5.2.1	Applying feature extraction techniques before training models	92
-------	---	----

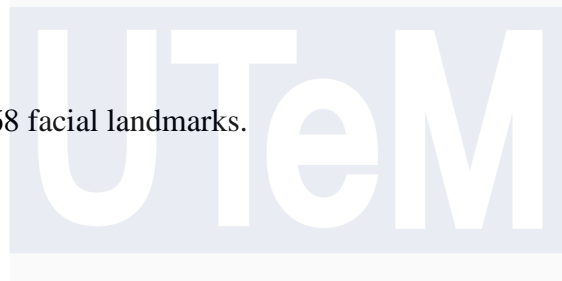
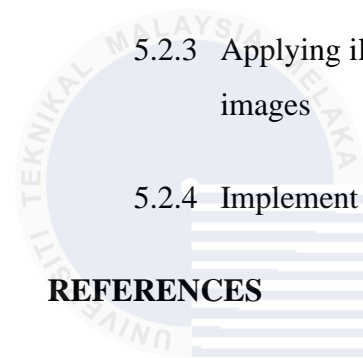
5.2.2	Address class imbalance in the FER2013 dataset	93
-------	--	----

5.2.3	Applying illumination correction techniques during pre-processes images	94
-------	--	----

5.2.4	Implement 468 facial landmarks.	94
-------	---------------------------------	----

REFERENCES		95
-------------------	--	-----------

APPENDICES		100
-------------------	--	------------



LIST OF FIGURES

Figure 2.1: The spectrogram samples of various speech emotions [5]	14
Figure 2.2: The CNN architecture for speech emotion recognition [6]	15
Figure 2.3: The RGB images of body gestures [7]	17
Figure 2.4: The overall process for emotion recognition based on body gestures by using TSN and ST-GCN model [7].	18
Figure 2.5: The basic step for facial expression recognition [9]	19
Figure 2.6: The various types of datasets for FER [10]	21
Figure 2.7: The JAFFE Dataset [12]	22
Figure 2.8: The sample images from FER-2013 dataset [17]	25
Figure 2.9: The schematic concept of the SVM model [20]	27
Figure 2.10: The architecture of SVM [20]	28
Figure 2.11: The FER process by using SVM classifier [21]	30
Figure 2.12: The basic architecture of Random Forest [22]	31
Figure 2.13: Flowchart of MVPCRF FER method [22]	32
Figure 2.14: The illustration of arousal and valence domain emotional labeling [24]	33
Figure 2.15: The basic CNN architecture	34
Figure 2.16: Hidden layer nodes in neural network [25]	35
Figure 2.17: The CNN architecture for FER algorithm [12]	36

Figure 2.18: Illustration CNN+BP and CNN+ELM of FER model [12]	37
Figure 2.19: Illustration of CNN architecture [14]	38
Figure 2.20: The confusion matrix on the FER2013 test dataset [14].	38
Figure 3.1: The flowchart diagram in FER project	47
Figure 3.2: The augmented images	48
Figure 3.3: Designed the CNN custom model.	50
Figure 3.4: The convolutional process	51
Figure 3.5: The RELU activation function	54
Figure 3.6 The training processes captured from 30 to 50 epochs.	57
Figure 3.7: The images of predicted emotion from the test dataset	60
Figure 3.8: The confusion matrix and classification report.	62
Figure 4.1: The model accuracy and loss without and with data augmentation	67
Figure 4.2: The plot graph illustrating the model accuracy and loss for the proposed model.	74
Figure 4.3 The plot graph illustrating the model accuracy and loss for CNN model in paper [26]	74
Figure 4.4: The Confusion Matrix for Proposed CNN Model	76
Figure 4.5: The Confusion Matrix for CNN architecture in paper [14]	76
Figure 4.6: The plot graph illustrating the model accuracy and loss for the VGG16 model.	80
Figure 4.7: The plot graph illustrating the model accuracy and loss for AlexNet model.	81
Figure 4.8: The plot graph illustrating the model accuracy and loss for the MobileNet model.	83

LIST OF TABLES

Table 2.1: The available databases that contain basic expression.	20
Table 2.2: The classification of facial expression in FER2013 dataset	23
Table 4.1: Analyzing the accuracy of custom CNN model by tuning the hyperparameter learning rate and batch size.	69
Table 4.2: Analyzing the custom CNN model by tuning the hyperparameter epochs.	72
Table 4.3: Performance Analysis for Proposed CNN Model	77
Table 4.4: The Comparison Between a Proposed Custom CNN Model and Three Pretrained Models	84
Table 4.5: Real time monitoring using webcam.	86

LIST OF SYMBOLS AND ABBREVIATIONS

FER : Facial Expression Recognition

AI : Artificial Intelligent

CNN : Convolution Neural Network

SDG : Sustainable Development Goals

GPU : Graphic Processing Unit

SER : Speech Emotion Recognition

RELU : Rectified Linear Unit

VGG16 : Visual Geometry Group 16

FABO : French Audiovisual Broadcast

GEMEP : German Multimodal Emotion Perception

LIRIS- : Lyon Institute of Research in Computer Science and Application

ACCEDE : Affective Video Dataset for Content and Emotion Analysis

HMM : Hidden Markov Models

SVM : Support Vector Machine

DBN : Deep Belief Network

CREMA- : Crowd-Sourced Emotional Multimodal Actor Dataset

D

Emo-DB : Emotional Database

RAVDESS	:	Ryerson Audio-Visual Database of Emotional Speech and Song
SAVEE	:	Surrey Audio-Visual Expressed Emotion
RGB	:	Red, Green, Blue
TCN	:	Temporal Segment Network
STGCN	:	Spatial Temporal Graph Convolution Network
CK+	:	Cohn-Kanade +
MMI	:	Multimodal Multiple Instance
JAFFE	:	Japanese Female Facial Expression
SFEW	:	Static Facial Expression in the Wild
KDEF	:	Karolinska Directed Emotional Faces
ICML	:	International Conference on Machine Learning
PCRF	:	Pairwise Conditional Random Forest
BU-4DFE	:	Binghamton University 4D Facial Expression Database
BP4D	:	Biometric Point Distribution Database

UNIVERSITI TEKNIK MALAYSIA MELAKA

FG-NET : Face and Gesture Recognition Network Database

FEED : Face and Expression Emotion Database

MVPCRF : Multi-View Pairwise Conditional Random Forest

RNN : Recurrent Neural Network

LSTM : Long Short-Term Memory

ELM : Extreme Learning Machine

BP : Back Propagation

API : Application Programming Interface

SGD : Stochastic Gradient Descent

TP : True Positive

TN : True Negative

FP : False Positive
FN : False Negative
GAN : Generative Adversarial Network



LIST OF APPENDICES

Appendix A: The simulation code for image processing	100
Appendix B: The simulation code for designing CNN model	101
Appendix C: The simulation code for plotting model accuracy and generating confusion matrix	102
Appendix D: The simulation code for generating classification report	103
Appendix E: The simulation code for testing the trained model with a sample image	104
Appendix F: The simulation code for integrating our proposed model using a webcam	105

CHAPTER 1



INTRODUCTION

UTeM

اونيورسيتي تيكنيكل مليسيا ملاك

In this chapter, we investigate the importance of recognizing emotions through various cues such as facial expressions, tone of voice and body language, emphasizing the role of facial expressions in communicating emotional states. The objective of the project is to accurately recognize seven human emotions such as angry, disgust, fear, happy, neutral, sad and surprise. We use computer vision and deep learning techniques, specifically Convolutional Neural Networks (CNN), to analyze and interpret facial expressions in images and real time monitoring. This chapter outlines the problem statement, highlighting the limitations of existing machine learning methods in handling complex datasets for facial expression recognition. We emphasize data augmentation techniques to address issues such as overfitting and improve accuracy. The project's scope, constraints and assumptions are discussed, along with its potential impact on sustainable development.

1.1 Background of Project

Artificial intelligence (AI) and psychological human emotion recognition are two distinct but interrelated fields of research in automatic emotion recognition. Emotional recognition is the ability to identify and interpret a person's emotional state based on various cues, such as facial expressions, vocal tone, body language, and physiological responses. Emotions are complex mental states related to various physical and psychological changes in a person. These changes can be used as an indicator of a person's emotional state. Most studies are particularly interested in this modality because changes in the face during communication are the first indicators that transmit emotional states [1]. Facial expressions are important to understand human emotions because the face has body parts such as eyes, nose, mouth, and others that make it easier for humans to recognize human emotions. Because of that, this project chooses human faces to recognize seven human expressions, namely angry, disgust, fear, happy, neutral, sad, and surprise, using AI technology. The purpose of the facial expression recognition project is so that it can be applied in the field of psychology because facial expression recognition technology helps researchers and psychologists recognize individual emotions accurately. With that, psychologists can better understand individual mental health issues. In addition, it can also be applied in the fields of mental health, education, driver monitoring, and so on.

This project uses the concept of computer vision to recognize human expressions through their faces because computer vision plays an important role in recognizing a person's expression. It uses AI technology to analyze and interpret human expressions in images and real time monitoring using webcam. This project uses AI technology because it is a machine programmed to mimic how humans think

and behave. AI models can accurately and instantly analyze and categorize facial expressions using deep learning algorithms and neural networks [2]. This project uses deep learning techniques, which consist of multiple layers of interconnected nodes, to learn and extract complex patterns and features from the input data to classify the output. Deep learning has dramatically improved the precision and effectiveness of face expression recognition systems with its capacity to handle complicated and high-dimensional data.

This project only involves a simulation program that is implemented using Google Colab and Jupyter Notebook as well as TensorFlow, OpenCV, Matplotlib, and Numpy as open-source libraries, and the programming language used is Python. The program consists of various tasks where it starts with data collection from Kaggle and then preprocesses the images to improve image quality and give diversity images. Then, the image is extracted for facial features for the training process. When the training model is complete with good validation accuracy, the next step is testing the model to predict and classify the image output from the testing dataset image input. In terms of equipment, this project only uses webcam to classify human expressions in real time.

1.2 Problem Statement

One of the most significant computer vision tasks is Facial Expression Recognition (FER), which is done using deep learning. FER has a wide range of applications in areas including psychology, health, security, and others. However, developing a system for FER2013 dataset based on machine learning have limitation in handling complex datasets for facial expression recognition, highlighting the need

for more advanced or adaptable methodologies. This issue holds significant importance as facial expression recognition plays a crucial role in areas like human-computer interaction and emotion analysis, where accurate interpretation of expressions is essential. Therefore, a proposed solution could involve the development of more sophisticated algorithms, such as deep learning models, which are known for their efficiency in managing complex and high-dimensional data [3]. Besides that, developing a system for FER based on a deep neural network is primarily plagued by overfitting, which is caused by insufficient training data and can result in a model that has very high accuracy on the training set but low accuracy on the test set. The significance of addressing this issue lies in enhancing the generalizability and reliability of the models in practical. Therefore, to address the overfitting problem, solutions could include methods like applying data augmentation technique during pre-process images to increase the size and diversity of the training dataset to ensure the model's robustness across different data samples [4].

1.3 Objectives

The objectives of this project are:

1. To investigate the deep learning technique for facial expression recognition.
2. To develop facial expression recognition using deep learning technique.
3. To evaluate the proposed model performance in term of precision, recall, F1-score and model accuracy.

1.4 Project Impact

Facial expression recognition using deep learning networks has a significant impact on various fields such as healthcare, education, security systems, and human-computer interaction. Accurate interpretation of facial expressions is essential in these fields, and deep learning models have shown promising results in recognizing facial expressions. By improving the accuracy and robustness of facial expression recognition systems, deep learning models can help doctors respond to patients' expressions accordingly, and a positive expression can help treat patients' conditions. In addition, facial expression recognition can be used in human-computer interaction systems with good intelligence and interaction performance. In education, it can be used to monitor students' engagement and attention levels. The impact of facial expression recognition using deep learning networks is significant, and it has the potential to revolutionize various fields by providing accurate and efficient recognition of facial expressions. Therefore, this project can be achieved the SDG goal which is

SDG 3 for Good Health and Well Being and SDG 4 for Quality Education.

1.5 Scope of Project

This section discusses the scope of the project, which consists of constraints and assumptions. It consists of two parts, namely simulation and restriction of this project.

1.5.1 Simulation

The scope of this project involves the development of a FER system using Google Colab as the simulation platform. This project uses Google Colab because it is a cloud based Jupyter notebook environment that allows users to collaborate and access powerful hardware resources such as GPUs for machine learning and data analysis tasks. It offers a computer environment that is interactive and supports several programming languages, including Python, which will be used for this project because of its adaptability and compatibility with other programming languages, making it an ideal option for developing complex systems that must function with a variety of tools and platforms. Overall, Python is a powerful and flexible language that provides a wide range of tools for deep learning applications. In addition, this project uses TensorFlow, OpenCV, Matplotlib, and NumPy as open-source libraries.

TensorFlow is a powerful deep learning framework that provides a large range of tools and packages for building complex neural networks. Therefore, this project will

use it as an open-source software library to build a FER system. OpenCV is an open-source computer vision framework that is used to recognize faces in images and to add text to the image that will describe the emotions that were found. Better results, understanding, and analysis will be possible for the user. The seven elementary emotions will correspond to the emotions that are labelled. As for NumPy, it is an open-source Python library, and it offers support for big, multi-dimensional arrays and matrices as well as a selection of mathematical operations on these arrays. As for Matplotlib, it is an open-source Python graphing library, and it offers a full set of tools for using Python to build different kinds of static, animated, and interactive visualizations.

The simulation program for this project involves several tasks, namely collecting the dataset from Kaggle, preprocessing the images, designing a model, training and testing a model, and measuring the model performance in terms of model accuracy, precision, recall and F1-score. Collecting datasets is important for deep learning model training. For the preprocessing task, it is important to improve image quality and reduce noise. For the task of designing a model, it is important to determine the parameter in deep learning model and the ability of the system to identify and classify facial expressions accurately and reliably. For the model training task, it takes a long time, and it depends on how many epochs are set for training. The more epochs that are set, the longer it takes to train the model. For the feature extraction task, it is important to track facial features during the training process so that the deep learning model can classify seven expressions based on the image facial features that have been extracted through the training process. Finally, for the final task, we measured the model performance. It is important to know and measure the accuracy of the system, whether it can recognize expressions more effectively or less effectively.

1.5.2 Restriction

There are several restrictions that need to be considered in the scope of facial expression recognition projects. Among them, the system must be able to analyze and classify facial expressions quickly to perform in real time, which is the first important criterion for this project. In addition, this project requires high computer processing power to ensure that the model can work properly, and this involves considering the use of memory and computing power. Finally, there may be

restrictions on the number of labelled facial expression datasets that can be accessed. Thus, to overcome the overfitting, techniques such as data augmentation should be investigated. By doing so, the model will be able to generalize well and work accurately. This project can create a FER system that achieves real-time performance, adapts to limited computer resources.

1.6 Significant of Project

Projects involving a FER system are significant in various fields, including psychology, social science, marketing, crime detection, and so on. This is because this project involves a program or system that can understand facial emotions immediately. This project also has an impact on society and the environment and has the potential to contribute to sustainable development. This project is important in the field of psychology because FER technology helps researchers and psychologists recognize individual emotions accurately. With that, psychologists can better understand individual mental health issues. This contributes to promoting psychological well-being and improving the quality of life. In terms of sustainable sustainability, this project can achieve one of the SDG goals, which is SDG 3, Good Health and Well-Being. In addition, this project also helps in the fields of social science and business markets, where it can be used to understand customer reactions to various stimuli such as products, advertisements, and others. This can develop an effective marketing strategy and improve product design, which can also contribute to sustainable economic development. In addition, this project can also help in the field of education, where facial recognition technology can be used to monitor students' emotions in class so that teachers can identify students who are not focusing

and advise them to focus on class. With that, this project can contribute to increasing well-being and economic development while minimizing the negative impact on the environment.

This project is important to ensure that engineering solutions for FER projects are built with sustainability and the impact on the environment and society in mind. It consists of energy efficiency, where algorithmic design optimizes facial expression recognition performance to minimize environmental impact. In addition, privacy and ethical considerations in this project ensure that facial recognition systems respect individual privacy rights. Additionally, it addresses potential biases to avoid inaccurate interpretations. Ensure technology works effectively across diverse populations, including different age, gender, and ethnic groups.

1.7 Chapter Outline

This thesis contains five chapters, and details of the project are explained in each chapter and outlined below:

Chapter 1: In this chapter, we emphasize the significance of AI in recognizing human emotions, particularly through facial expressions. Our project aims to accurately detect seven emotions using computer vision and CNN model. We address the challenge of handling complex datasets and highlight data augmentation to enhance accuracy. The chapter outlines the project's scope and potential impact on sustainable development while providing an overview of the thesis structure.

Chapter 2: In this chapter, we discussed the importance of Speech Emotion Recognition (SER) and its relevance in human-computer interaction. We explored machine learning techniques, including CNN, and their application in recognizing emotions from speech and body language. We also emphasized the significance of databases in FER, with a focus on FER2013 database. Additionally, we highlighted the role of AI and deep learning techniques, such as activation functions Rectified Linear Unit (RELU) and model assessment using confusion matrix, in the context of FER.

Chapter 3: In this chapter, we discuss on how to implement FER using CNN. Besides that, we cover the use of Jupyter Notebook and Google Colab, along with key open-source libraries like TensorFlow, KERAS, OpenCV, Matplotlib, and NumPy. In addition, we discuss in detail the FER algorithm with CNN, including images preprocessing, model design, training, and testing. Finally, we explore integrating the FER system with a webcam for monitoring facial expression in real-time.

Chapter 4: In this chapter, we discuss analysis of data augmentation and optimization of model correctness via the exploration of hyperparameter tuning, including learning rate, batch size, and epochs. Together with a confusion matrix for performance analysis, the suggested custom CNN model with the proper hyperparameters. By using the same hyperparameters, the chapter also examines the accuracy, precision, recall, and F1-score of proposed models in comparison to pretrained models VGG16, AlexNet, and MobileNet.

Chapter 5: In this chapter, we successfully designed a CNN model for real-time recognition of seven emotions, achieving a high accuracy of 65.27%. In future work includes enhancing the model through feature extraction, addressing dataset imbalances, improving lighting conditions, and implementing precise facial landmarks for better face detection and recognition.



CHAPTER 2



BACKGROUND STUDY

In this chapter, we discussed SER and its significance in enabling realistic human-computer interaction. We explored the utilization of machine learning techniques, particularly CNN, pattern recognition, and statistical models, for training algorithms on extensive datasets containing labeled emotive speech samples. Additionally, we delved into the recognition of emotions through body language, pointing out the availability of datasets such as French Audiovisual Broadcast (FABO), German Multimodal Emotion Perception (GEMEP), and Lyon Institute of Research in Computer Science and Application Affective Video Dataset for Content and Emotion Analysis (LIRIS-ACCEDE) for this purpose. Furthermore, we highlighted the importance of databases in FER analysis, with a focus on the widely used FER2013 database. The chapter also touched upon the role of AI and machine

learning in developing algorithms and models for automatic emotion recognition, particularly through deep learning techniques like CNN. Also, the use of RELU as an activation function, max-pooling for efficiency, and the significance of the confusion matrix for assessing model performance were discussed in the context of FER.

2.1 The Types of Emotion Recognition

The study and application of emotion identification technology aims to detect and interpret human emotions from a variety of inputs, including voice, text, facial expressions, and physiological signs. There are numerous types of recognizing emotions methods, including speech emotion recognition, body gesture emotion recognition, and facial emotion recognition.

2.1.1 Overview of Expression Recognition from Speech Analysis

SER is important for many applications in natural human-computer interaction. It is a process of identifying and classifying emotions through speech signals. It also involves analyzing the acoustic features found in spoken language, such as spectrum content, rhythm, and others, to determine several emotions such as happy, sad, disgusted, angry, surprised, fearful, and neutral. In the SER system, the main role is to extract features from speech, which are then categorized to predict different emotional classes in it. Sultana, J., and Naznin, M. [5] state that Deep Neural Network-based features and Hand-Crafted features can both be extracted from speech. Classification of emotion recognition based on speech can be carried out in two ways, using traditional classifiers such as Hidden Markov Models (HMM), Support Vector Machines (SVM), and others, or using deep learning algorithms such as Deep Belief Network (DBN), CNN, and others.

2.1.1.1 Speech Expression Databases

In the paper by [5], there are several audio datasets that can be used for speech emotion recognition, including the Crowd-Sourced Emotional Multimodal Actors Dataset (CREMA-D), which contains 7442 audio clips sampled at 16 kHz; the Emotional Database (Emo-DB), which contains 535 utterances of 10 professional actors; the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), which has both audio and visual files of 24 North American actors and Surrey Audio-Visual Expressed Emotion (SAVEE), which contains 480 utterances from 4 British male speakers with seven different expressions [5]. The audio datasets are shown in spectrogram forms, such as shown in Figure 2.1 where indicate a signal's quality over time at different frequencies that are included in the waveform.

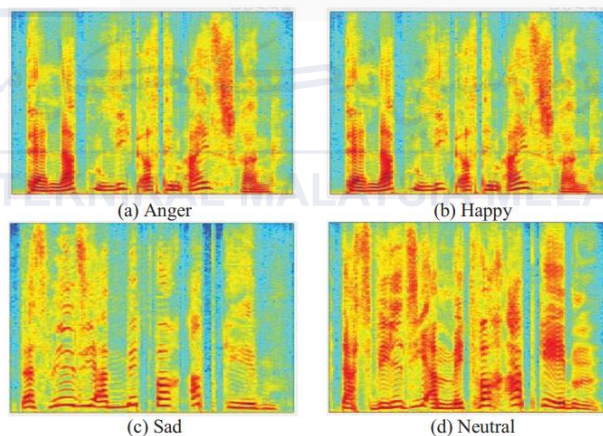


Figure 2.1: The spectrogram samples of various speech emotions [5]

2.1.1.2 Speech Expression Recognition using Deep Learning Technique

Researchers in the field of SER use a variety of machine learning techniques, including deep learning, pattern recognition, and statistical models, to train algorithms on large datasets containing labelled emotional speech samples. Based on Figure 2.2,

shows one of the techniques presented in the paper [6] that can be used in identifying emotions from speech. The technique used is a deep learning technique called CNN. The CNN model consists of 3 convolution layers, which are Conv1 with 64 kernels of size (9x9), Conv2 with 120 kernels of size (7x7), and Conv3 with 200 kernels of size (3x3). The top pooling layer receives the created features. The maximum pooling size is the same for all three convolutional blocks (20 x 32). Two fully connected layers, FC1 and FC2, of 256 and 512 neurons, come after the top pooling layer. All convolutional layers are followed by the RELU activation function. As a result of its greater convergence rates, this activation is employed. Finally, the SoftMax unit does the classification work. Wani [6] concluded that this model gives an overall accuracy of 65.5% after training for 500 epochs.

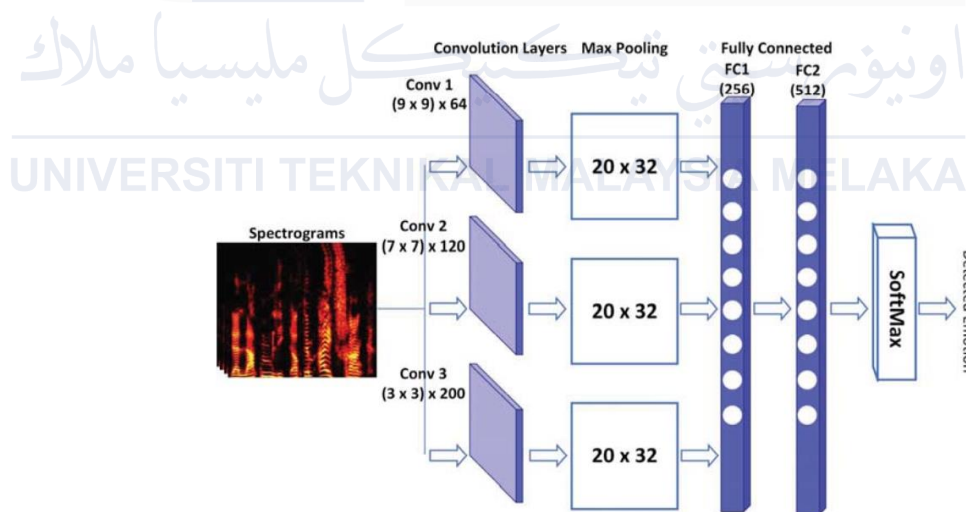


Figure 2.2: The CNN architecture for speech emotion recognition [6]

2.1.2 Overview of Expression Recognition from Body Gestures Analysis

Humans naturally express their emotions not only through their faces but also through their body gestures. In the paper by [7], psychological research shows that body language can convey non-verbal emotional signals that the face and voice cannot. However, less research has been done to identify emotions through body language. This is because body gestures do not have clear emotional characteristics because various people use different body movements to convey the same emotion. Although the same person makes different body movements, they differ depending on the situation. Emotion recognition through body movements is a process of identifying and interpreting emotions conveyed through body movements, which involves analyzing posture and physical movements exhibited by individuals to identify their emotional state. It aims to take advantage of non-verbal communication to accurately understand and classify their emotions. Algorithms can recognize emotions such as happiness, sadness, surprise, fear, disgust, anger, and neutrality by analyzing spatial position, temporal dynamics, and body movement patterns. Techniques that can be used to recognize emotions through body movements are machine learning and deep learning.

2.1.2.1 Body Gesture Expression Databases

Shen, Z. [7] also presented several types of datasets for emotion recognition through body gestures. Among them is FABO, which has 206 samples and 10 emotions involving the face and body. In addition, GEMEP has more than 7000 samples and 18 emotions involving the face and body. Not only that, LIRIS-ACCEDE also has a dataset with six emotions involving the face and upper body. Examples of samples of RGB images for body gestures according to their emotions are shown in the Figure 2.3.



Figure 2.3: The RGB images of body gestures [7]

2.1.2.2 Body Gestures Expression Recognition using Deep learning Technique.

In the paper by [8], the acceleration state for the hand is the basic feature of the ensemble tree classifier, and the model shows the best performance in their study. Based on Figure 2.4, shows one of the deep learning techniques used in the paper [7] to recognize emotions through body gestures. It uses RGB video as an input dataset, and then the video is extracted by using the Temporal Segment Network (TCN) model to extract RGB features and Spatial-Temporal Graph Convolution Networks (ST-

GCN) to extract skeleton features. The output from them is a vector of different lengths. The two feature vectors are combined to create a new fusion feature, which is then tailored by a fully linked residual network into the appropriate category of emotional body gestures. Finally, to classify the emotional body gestures, a residual full-connected network that uses the same architecture as the residual feature encoder Shen [8] concludes that the overall accuracies for the TSN model are 72.09%, while those for the ST-GCN model are 72.00%.

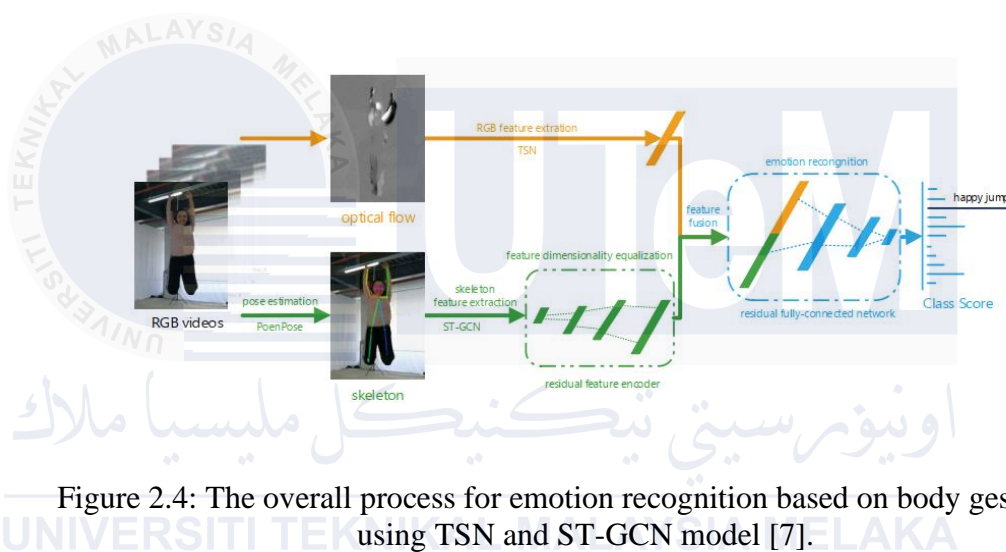


Figure 2.4: The overall process for emotion recognition based on body gestures by using TSN and ST-GCN model [7].

2.1.3 Overview of Expression Recognition from Facial Analysis

FER is a field of study that focuses on detecting, analyzing, and interpreting emotions and facial expressions shown by individuals. One of the most effective, universal, and natural ways for people to express their emotions and intentions is through their faces. Facial expression is important because it is one aspect of communication in daily life. Facial Emotion Recognition technology uses computer vision and Artificial Intelligence techniques to analyze and interpret human emotions from facial expressions. However, based on the paper [9], the author stated that in the field of computer vision, the difficult task of automatically identifying facial

expressions from facial images has a variety of potential applications, including driving safety, human-computer interaction, health care, behavioral research, video conferencing, cognitive science, and others. With that, various algorithms, machine learning, and deep learning models can be used to detect and classify facial expressions accurately. FER analysis consists of face detection, facial expression detection, and expression classification according to emotional states, as shown in the Figure 2.5. Emotion detection is based on the position of facial landmarks such as the eyes, nose, eyebrows, mouth, and others. With that, the algorithm can classify emotions using a person's face, such as sadness, happiness, fear, surprise, and the others, accurately. The accuracy and efficiency of classifying emotions can be increased by using suitable techniques and databases.

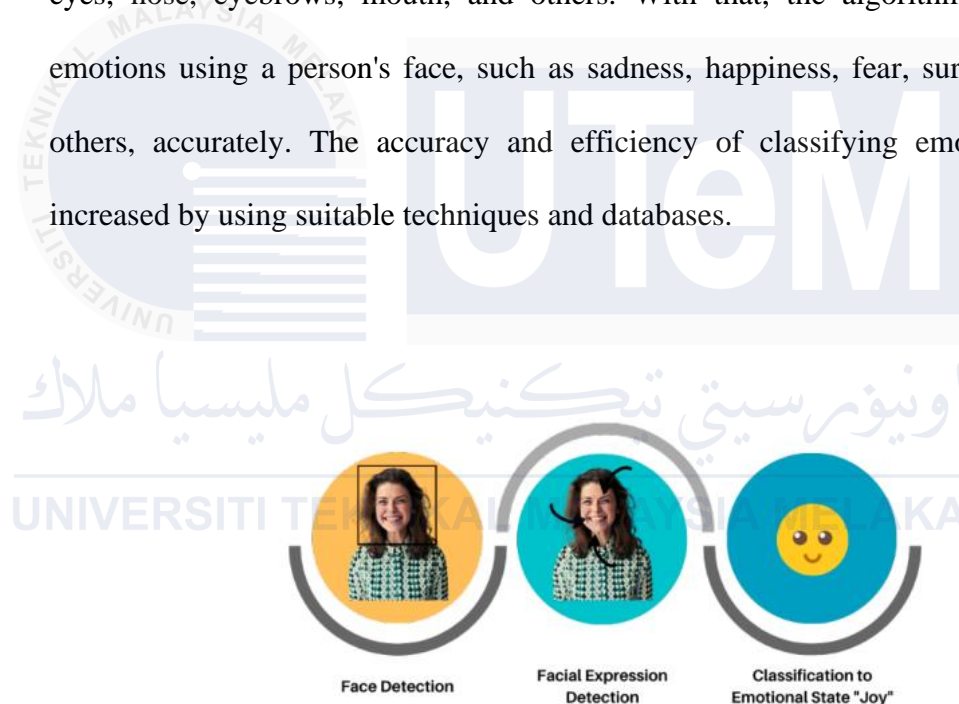


Figure 2.5: The basic step for facial expression recognition [9]

2.1.3.1 Facial Expression Databases

The database plays an important role in FER. The database is a collection of images of different people's faces expressing different emotions. It is used for training, evaluation, and development of FER models. By training on diverse and representative data, machine learning algorithms can learn the underlying patterns and features that

distinguish different emotions, allowing them to make accurate predictions on new, unseen data.

Table 2.1: The available databases that contain basic expression.

Databases	Samples	Collection Condition	Expression Distribution
CK+	593 images	Lab	Seven basic expressions plus contempt
MMI	740 images and 2900 videos	Lab	Seven basic expressions
JAFFE	213 images	Lab	Seven basic expressions
SFEW	1766 images	Movie	Seven basic expressions
KDEF	4900 images	Lab	Seven basic expressions
FER2013	35887 images	Web	Seven basic expressions

Based on the paper [10], the author states that there are several databases available for facial emotion recognition, as shown in the Table 2.1 and Figure 2.6. Among them, the Cohn-Kanade (CK+) database is the most extensively used laboratory-controlled database for evaluating FER systems. Besides, the MMI database is also laboratory controlled. Not only that, The Japanese Female Facial Expression (JAFFE), Static Facial Expressions in the Wild (SFEW), Karolinska Directed Emotional Faces (KDEF), and FER2013 are also FER databases that have seven basic emotion expressions.



Figure 2.6: The various types of datasets for FER [10]

The JAFFE dataset stands for Japanese Female Facial Expression, which is used to conduct experiments and is provided by the psychology department at Kyushu University. According to the paper [11], it consists of 213 grayscale images of size 256x256 pixels depicting 60 Japanese Females. The format of the images is TIFF. This dataset is divided by 2, with 171 images for training purposes and 42 for testing purposes. Based on this paper, the overall accuracy results on the JAFFE dataset are 94.23% for the CNN model. Not only that, according to the paper [12], the author states that the results for training accuracy on the JAFFE dataset are 93.84%, while the testing accuracy is 85.91% for the CNN+ELM model.

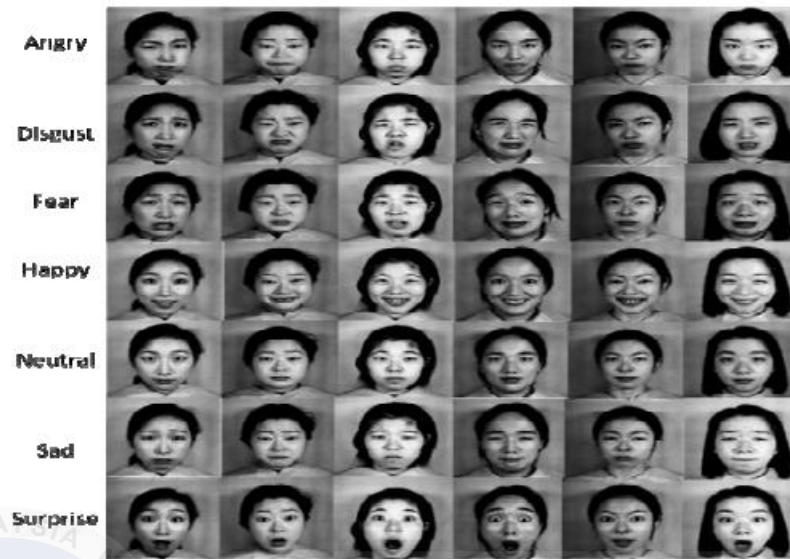





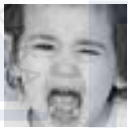



Figure 2.7: The JAFFE Dataset [12]

The FER2013 dataset, also known as the 2013 Facial Expression Recognition dataset, was presented at the International Conference on Machine Learning (ICML) in 2013 [13]. It was created by Pierre-Luc Carrier and Aaron Courville and is available on Kaggle as a popular platform for data science and dataset competitions. Based on articles reviewed that have been conducted for researching the FER database, the author uses the FER2013 dataset as the data for training because, according to the paper [14], the author achieved a test accuracy of 73.73% on the FER2013 dataset. Not only that, according to the paper [15], effective facial expression detection was achieved using the FER2013 dataset, and the author achieved a FER2013 test accuracy of 75.2%. In general, FER2013 is a grayscale image dataset consisting of 28709 images for the training dataset and 7178 images for the test dataset, where each image in this dataset has the same image size of 48 x 48 pixels. It is one of the more difficult datasets, with the best reported results only achieving a test accuracy of 75.2% and a human-level accuracy of 65.5%, and the dataset is available on Kaggle [16].

Table 2.2: The classification of facial expression in FER2013 dataset

No	Expression	Image	Description
0	Angry		Face shows such closed lips, arched eyebrows, and large eyes indicate angry.
1	Disgust		Face shows such elevated upper lips and relaxed eyebrows and eyelids frequently indicate disgust.
2	Fear		Face shows where the eyes are strained, their inner eyebrows curve forward, and their eyebrows are elevated and pinched.
3	Happy		Face shows such relaxed with open lips, raised corners, and relaxed eyebrows.

4	Neutral		Face indicates where there are no obvious signs of tension or contraction in the facial muscles. There don't seem to be any obvious creases or furrows in the forehead.
5	Sad		Face shows where mouths are often relaxed, their eyes are somewhat closed, and their inner eyebrows bend upward.
6	Surprise		Face shows where the mouth is gaping, their upper eyelids are thrown back, and their eyebrows are turned upward to express surprise.

Based on the Table 2.2, showing the classification of facial expression in the FER2013 dataset with expression description and labels such as (0 = angry, 1=disgust, 2=fear, 3=happy, 4 = neutral, 5=sad, and 6=surprise), Since there is an imbalance of data, such as the emotion of disgust, which has an image of only 500–600, To address this issue, the author [17] proposed to use some data augmentation on the data using

the KERAS Images data generator, which generates a batch of images in different poses, and scales.



Figure 2.8: The sample images from FER-2013 dataset [17]

2.1.3.2 Facial Expression Recognition using Machine Learning Technique

AI has a field called "Machine Learning" that focuses on creating algorithms and models that allow computers to learn from data and make predictions or decisions without being explicitly programmed. It includes the use of computational algorithms and statistical approaches to give computers the ability to recognize patterns in data, draw conclusions from them, and carry out specified tasks. To automatically detect patterns, correlations, and trends in machine learning, models are trained on labelled or unlabeled data. By giving models access to unexpected new data, the objective is to enable them to make accurate predictions or decisions. Machine learning can be used to recognize facial expressions. The steps involved in real-time facial expression recognition using machine learning are collecting and preparing labelled facial expression data, extracting relevant features, training models, evaluating their

performance, and using them. Applications in various domains, including human-computer interaction, affective computing, and psychological research, are possible through the automatic identification and classification of facial expressions using this approach.

An overview of machine learning algorithms is given in the research article "Machine Learning Algorithm: A Review". The study of algorithms and statistical models that computer systems employ to carry out a certain task without being explicitly programmed is known as machine learning. Batta Mahesh [18] states that one of the most popular machine learning methods now in use is the SVM. SVMs are supervised learning models with associated learning algorithms that examine data used for regression and classification analysis in machine learning. In 2017 research published in the International Journal of Computer Applications [19], the accuracy of using SVM for identifying six basic facial expressions, which are anger, disgust, fear, anxiety, sadness, and surprise, on the CK+ dataset was reported to be 93.1%.

SVM is an effective machine learning algorithm used for classification and regression problems. In a high-dimensional space, they find an ideal hyperplane that maximally separates classes or regressors. The support vector, the data point closest to the hyperplane, is used by the SVM to determine the decision boundary, as shown in Figure 2.9. In the paper by [20], the author stated that the objective of SVM classification is to identify the best hyperplane that divides two classes of data. Based on the experiment results from this paper as well, the author stated that the SVM algorithm test result resulted in an accuracy classification value of 87%.

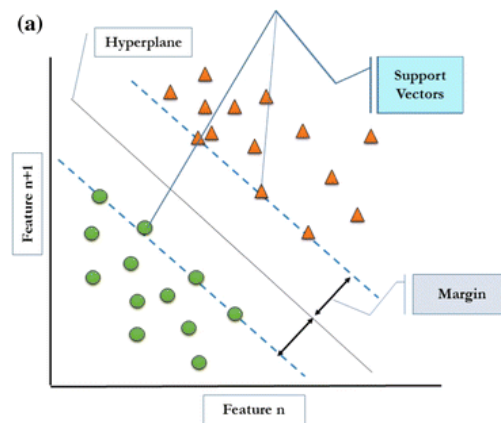


Figure 2.9: The schematic concept of the SVM model [20]

Based on Figure 2.10, the SVM architecture consists of input data, support vector, evaluation kernel, and output classification. As for the data input, the SVM input is pre-labelled training data, with each data point represented by a set of features or attributes and having a class or category label attached. The SVM model is trained using the input data. For the support vector, the closest data point to the hyperplane, known as the support vector, is very important in determining the location and direction of the hyperplane. These factors affect the overall SVM model and serve to define decision boundaries. SVM is memory efficient because it only needs support vectors for classification or regression. As for evaluating the kernel, by using the kernel function, SVM can handle data that cannot be separated linearly. To improve class separation, kernel functions translate the input data into a higher-dimensional feature space. Linear, polynomial, and sigmoid Radial Basis Functions (RBFs) are common kernel functions. The challenges faced and the nature of the data determine which kernel should be used. Finally, for the classification output, the SVM model can be used to predict the results from a brand-new data set. The input data points are passed to the decision function, which uses a kernel function to map them into the

feature space and calculate their location on the hyperplane. Based on this location, the expected class, or regression value, is calculated.

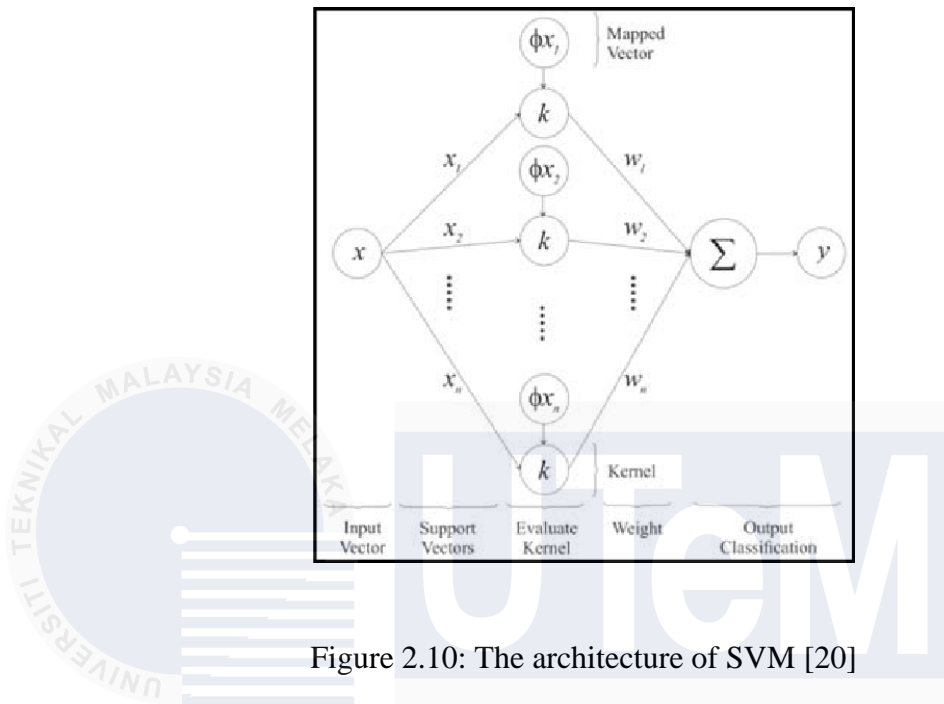


Figure 2.10: The architecture of SVM [20]

In the paper by [21], the author states that SVM is a classifier that classifies data using a hyper-plane. Both linear and non-linear decision boundaries can be created by it. Using Lagrange multipliers, SVM finds the ideal hyperplane. Based on a small sample of training vectors known as support Vectors, the best hyperplane was selected. Basically, a support vector is a template that sits on the edge of any class. In this paper [21], the author also stated that nonlinear SVM is typically used for FER tasks. In nonlinear SVM, the basic decision function of linear SVM is modified by using kernels such as:

$$f(x) = \sum_{i=1}^{N_s} \alpha_i y_i k(x_i, x) + b \quad (2.1)$$

Where:

- $f(x)$: Decision function
- N_s : The number of support vectors in the model
- α_i : Lagrange multipliers associated with each support vector.
- y_i : The class label of the i -support vector
- $k(x_i, x)$: The kernel functions.
- b : The bias term

Based on Figure 2.11, this shows the FER process by using the SVM model to classify the output. The dataset generated the training images for SVM learning. All the images have been cropped to 92x92 proportions to include all facial details. 1500 images of each positive and negative expression are used in the training. Positive expression is not preferred as much as negative expression. All the images have undergone the previously described pre-processing procedures. Utilizing feature vectors created from LH and HL pictures, training is carried out.

The FER technique presented can recognize facial emotions in images that are in grayscale. Before SVM testing, the illumination of the input image is corrected. The image is rotated over the three sizes of windows 92x92, 46x46, and 23x23 with the appropriate window shift. Utilizing the wavelet decomposition method, the features of each window are retrieved. Window size corresponds to the size of feature vectors. All feature vectors are extracted, and then they are provided to a trained SVM for classification. In grayscale images, the FER approach described here may detect face expressions. The input image is illuminated before running SVM testing. The window shifts for the three sizes of windows 92x92, 46x46, and 23x23 are acceptable when the image is rotated. The wavelet decomposition technique is used to extract the

features of each window. Feature vector size correlates with window size. After extracting each feature vector, the feature vectors are sent to trained SVMs for classification. The overall accuracy for FER using SVM classification is 94.1%.

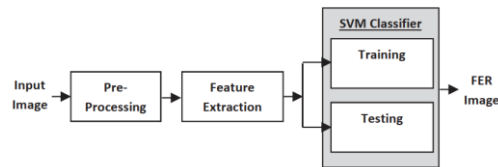


Figure 2.11: The FER process by using SVM classifier [21]

Random Forest is one of the classification techniques in machine learning. It can be used to classify different emotions from facial images based on features that have been extracted. The algorithm works by building an ensemble of decision trees, as shown in Figure 2.12, where each tree is trained on a random subset of the training data and a random subset of the features. During the training phase, the Random Forest algorithm grows multiple decision trees, each of which makes predictions based on different combinations of features. After that, a vote or averaging procedure is used to decide the final prediction, with each tree guess contributing to the result. However, the issue of overfitting affects the decision tree. Nevertheless, by continuously adding nodes to the tree, which increases the depth of the tree and makes it more complex, overfitting only involves making the tree more specific within itself to reach a certain conclusion.

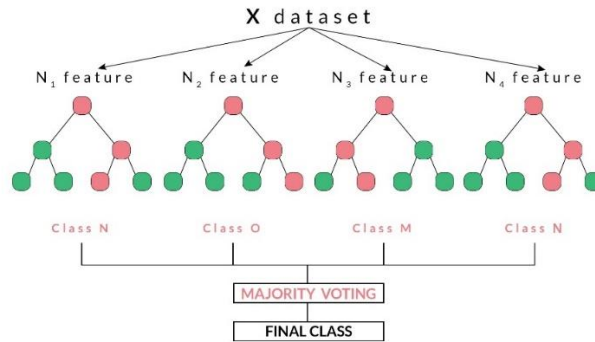


Figure 2.12: The basic architecture of Random Forest [22]

Based on the paper [22], the author used Conditional Random Forest to capture low-level emotional transition patterns. To generate a tree using Pairwise Conditional Random Forest (PCRF), predictions for each previous frame are used to create pairs between the current and previous frames during testing. PCRF pairwise outputs are averaged over time to provide a reliable estimate. For multi-view dynamic FER, the PCRF collection can also be conditioned on head pose estimation. Thus, the method seems to be a logical extension of Random Forest to learn spatial-temporal patterns, perhaps from several angles. The author uses pairs of images that represent patterns of expression transitions to train a Random Forest. To help reduce variability, the forest was conditioned on the expression labels of the first frame. For databases, the author used four databases, namely CK+, Binghamton University 4D Facial Expression Database (BU-4DFE), Biometric Point Distribution Database (BP4D), and Face and Gesture Recognition Network Database (FG-NET).

This paper presents a flowchart of the Multi-View Pairwise Conditional Random Forest (MVPCRF) FER method as shown in Figure 2.13, MVPCRF technique, which is a new approach to training trees using static and dynamic features

in the Random Forest framework. In the field of face alignment and human pose estimation, it uses Conditional Random Forest by generating a collection of specific trees, quantizing the values of global variables of head posture and torso orientation, and then using predictions on these global variables to draw specific trees, resulting in more accurate predictions. To limit the uncertainty of continuous expression transitions from the first frame of a pair to the next, as shown in Figure 2.13, the author proposes to condition the pairwise tree on a specific expression label by conditioning the pairwise tree on the estimated head pose to improve resilience to fluctuations in head pose. Each previous frame in the sequence is connected to the current frame to form a pair when analyzing the video frame. Finally, as a result of this paper, the overall accuracy of the MVPCRF method is 72.1% on the BU-4DFE database.

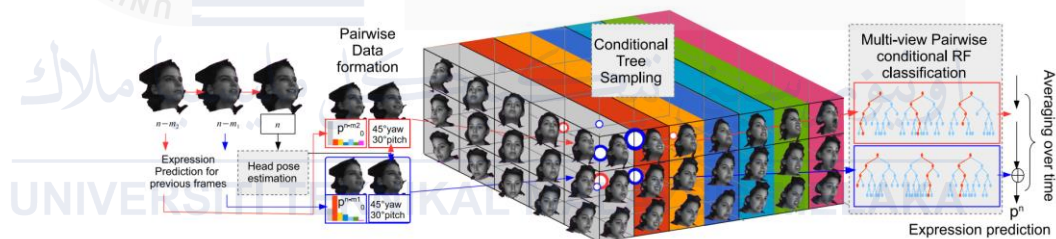


Figure 2.13: Flowchart of MVPCRF FER method [22]

2.1.3.3 Facial Expression Recognition using Deep Learning Technique (CNN)

Deep learning methods are used for FER, which involves training neural networks to recognize and categorize facial expressions automatically from image data or videos. By making it possible to extract high-level characteristics from facial data and to recognize complex patterns, deep learning has improved FER. It was shown in the study by [23] that face feature extraction and classifier construction can be combined to improve the efficiency of the two stages of expression recognition. Deep

learning techniques, especially CNN architectures, a multi-stage design inspired by biology that automatically acquire hierarchies of invariant features, have been effectively used to extract features and perform classification.

Deep learning-based techniques that have significantly advanced the field of FER. In the paper by [24], the author states that deep learning networks have progressively improved our comprehension of low-dimensional features that can distinguish high-dimensional complex face patterns from low-dimensional features. CNN, Recurrent Neural Networks (RNN), and long-short-term Memory (LSTM) are models for deep learning techniques to classify the output expression. All types of deep learning models can be used for various tasks, such as image recognition. This paper also presented an illustration of arousal and valence domain emotional labelling, as shown in Figure 2.14. The valence-arousal space is a popular and adaptable paradigm. The valence-arousal model classifies different sorts of emotions according to how valuable their emotional elements are.

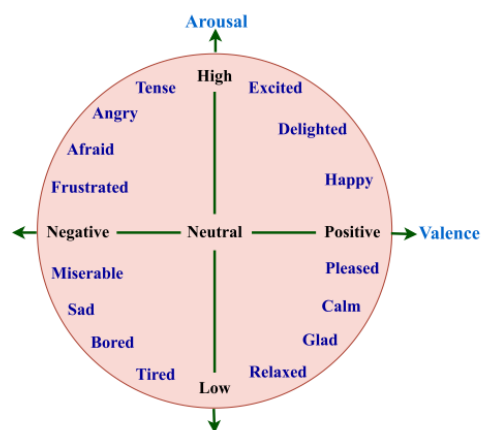


Figure 2.14: The illustration of arousal and valence domain emotional labeling [24]

One of the best-known variations of traditional multilayer neural networks for image processing is the CNN. CNNs are a subset of deep neural networks that are widely used in computer vision tasks such as image and video segmentation, classification, and recognition. CNN uses convolutional layers to learn and automatically extract features from raw data by applying several filters to the input data. CNN has an important role in deep learning because of its better feature extraction ability. CNN consists of several layers, such as an input layer, a hidden layer, and an output layer, as shown in Figure 2.15. For the hidden layer, it consists of a convolution layer, pooling, and fully connected to teach the representation of input features. As for the output layer, it is used to perform classification or regression tasks using the output from the convolution layer.

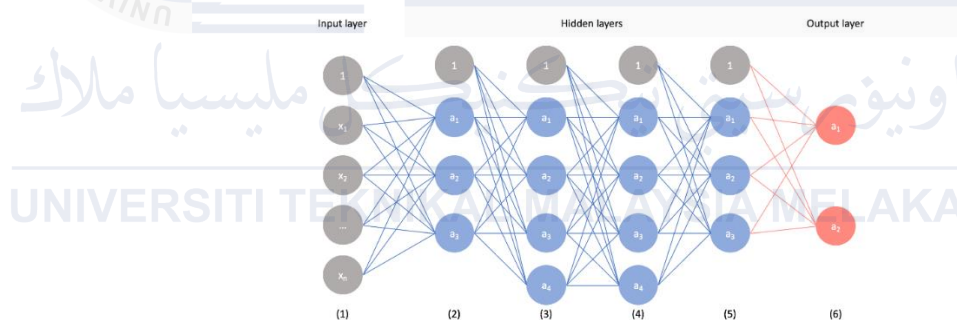


Figure 2.15: The basic CNN architecture

Based on the paper [25], the author suggests a resource for automatic emotion recognition by using deep learning techniques. It uses the CNN method to identify six basic emotions using MATLAB. It uses the JAFFE dataset, which consists of 213 images with dimensions of 256x256 pixels and six different emotions. It uses the CNN method because convolution is powerful in finding the features of the input image if the kernel used is correct. Each convolution result is added to the following layer in a

hidden node. Kernel design is an art form that has been enhanced to achieve some incredible things with images. The nodes in the hidden layer, as shown in Figure 2.16, correspond to each feature of the convoluted image. This paper shows that there are several types of layers for CNN architecture, such as the input layer, where the input image is in 2D and grayscale, and then it is converted to a pattern. In addition, the convolution layer, where it takes the image using zero padding, provides output when the kernel is ready to use. As for the pooling layer, it divides the input image into a set of rectangles, and for each sub-region, it outputs a value. For the fully connected layer, where it is a 1D vector and each neuron in the layer will be connected to the previous volume. Finally, as a result, the author obtained an overall accuracy of 91.6%.

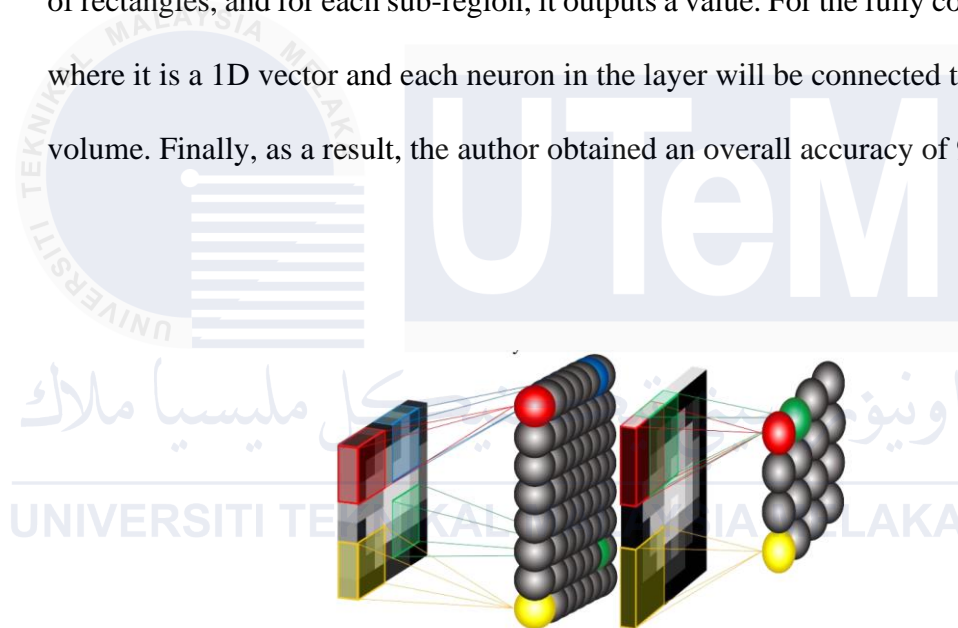


Figure 2.16: Hidden layer nodes in neural network [25]

In the paper [12], the author presented facial expression recognition using deep learning techniques, namely CNN, using JAFFE and KDEF datasets. The CNN architecture enabled many degrees of characteristics to be extracted from the input image. It emphasizes the process of extracting important features using a pre-trained convolutional neural network called the AlexNet model. The extracted learned features are then trained using Extreme Learning Machine (ELM) and Back

Propagation (BP) methods. The author stated that AlexNet is a useful model to replace manual feature extraction, and the ELM trained with the extracted features can work faster and offer superior performance to BP. For the architecture of the CNN model as shown in Figure 2.17, it uses RELU as a non-linear activation function for the convolution operation. It is used because it is suitable for CNN architecture because it reduces the problem of gradients caused by neural networks. In addition, it is also able to accelerate the learning process faster than other activation functions. For the hidden CNN layer, it uses max-pooling because it is more efficient than the mean and sum pooling methods and is faster in selecting the most relevant pixels.

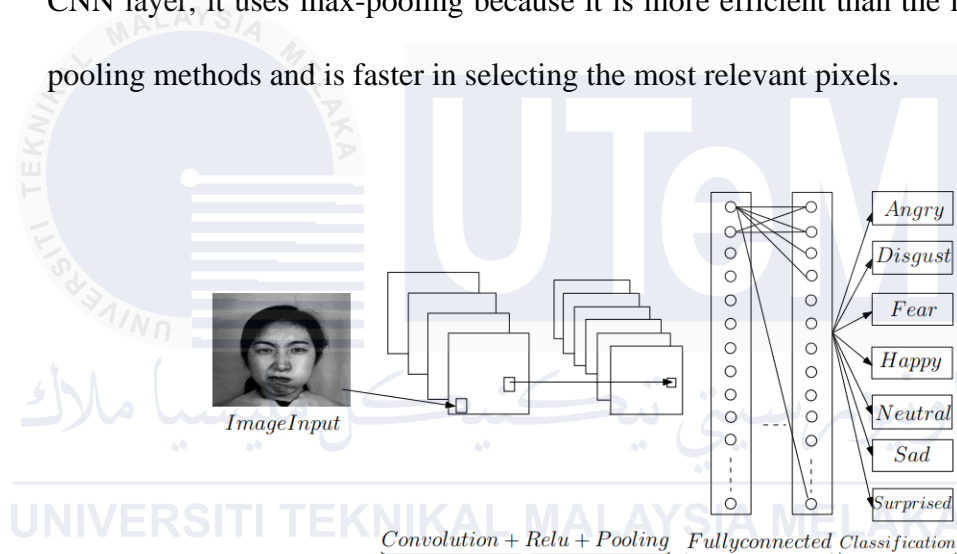


Figure 2.17: The CNN architecture for FER algorithm [12]

The author focused on two main parts, namely implementing a pre-trained AlexNet model to detect facial features and, in the second part, using the extracted features to train the BP algorithm. However, the BP model converges very slowly. With that, the author implemented ELM to improve facial expression recognition performance. Based on Figure 2.18, it shows CNN+BP and CNN+ELM for the FER model, which is a process for facial expression recognition. Based on the two models,

the accuracy of BP approaches between 91% and 94% for recognizing emotions, while the accuracy of ELM can exceed 95% in classifying facial expressions in a short time.

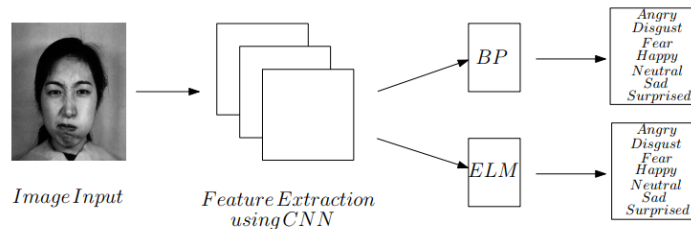


Figure 2.18: Illustration CNN+BP and CNN+ELM of FER model [12]

The paper by [14] shows how to classify FER from a static image without requiring feature extraction work. It also provides pre-processing methods such as face detection and lighting correction to improve accuracy in recognizing facial expressions. The aim of this paper is to develop a novel architecture from scratch to classify faces into emotional categories using CNN and improve accuracy on the FER2013 dataset through preprocessing tasks. The author says the image preprocessing process involves the detection and alignment of faces, poses, occlusion, data augmentation, and others. To detect faces, it uses the Haar Cascade Classifier. To correct the lighting in the picture, the researchers propose to use histogram equalization. Based on Figure 2.19, the author uses CNN models to solve FER problems such as translation, rotation, subject independence, and scale invariance. It consists of 6 convolution layers using RELU as an activation function: 3 max-pooling, 2 drop-out with value 0.2, 1 flattened layer, and 2 dense layers, one dense with RELU and the other with SoftMax as an activation function. The total number of parameters is 1.2 million.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 128)	1280
conv2d_2 (Conv2D)	(None, 44, 44, 128)	147584
max_pooling2d_1 (MaxPooling2)	(None, 21, 21, 128)	0
conv2d_3 (Conv2D)	(None, 19, 19, 128)	147584
conv2d_4 (Conv2D)	(None, 17, 17, 128)	147584
max_pooling2d_2 (MaxPooling2)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_5 (Conv2D)	(None, 6, 6, 128)	147584
conv2d_6 (Conv2D)	(None, 4, 4, 128)	147584
max_pooling2d_3 (MaxPooling2)	(None, 2, 2, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 1024)	525312
dropout_2 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 7)	7175
Total params: 1,271,687		
Trainable params: 1,271,687		
Non-trainable params: 0		

Figure 2.19: Illustration of CNN architecture [14]

Based on Figure 2.20, this shows the confusion matrix table done by the author to analyze the overall accuracy of the FER system. The author uses different batch sizes of 512 and 10 epochs to get the best test accuracy. As a result, the author achieved an overall accuracy on FER2013 test data of 61.7% without involving pre-processing tasks, while the state-of-the-art test accuracy for 7 emotion categories using ensemble CNN was 75.2%.

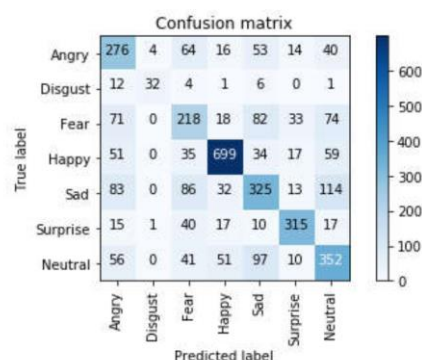


Figure 2.20: The confusion matrix on the FER2013 test dataset [14].

2.2 Summary

In summary of Chapter 2, SER involves recognizing and categorizing emotions using speech signals, is crucial for realistic human-computer interaction. Additionally, as body gesture can communicate non-verbal emotional signals that cannot be expressed through the face or speech, the study of emotion identification through body motions is another area of interest. In addition, the goal of FER is to identify, evaluate, and interpret the emotions and facial expressions of people. FER technology analyses and decodes human emotions from facial expressions using computer vision and artificial intelligence algorithms. Databases are essential to FER analysis because they offer a variety of pictures of people's faces expressing various emotions. The most widely utilized laboratory-controlled database to evaluate FER systems is the FER2013 database. The next area of AI such machine learning, focuses on developing algorithms and models that let computers learn from data and make predictions or decisions without having to be explicitly programmed. Moreover, FER which involves training neural networks to automatically recognize and classify facial expressions from picture data or videos, makes use of deep learning techniques. RELU are used by the authors as a non-linear activation function for the convolution process, and max-pooling is used in the CNN layer since it is faster and more effective than mean and sum pooling at identifying meaningful. Therefore, this project aims to recognize emotions using the facial expression database FER2013. The technique employed is deep learning, specifically CNN, which offer several advantages, including the ability to handle complex datasets.

CHAPTER 3



In this chapter, will be discussed about the methodology for implementing FER using CNN. It will begin by exploring the versatile simulation environment provided by Jupyter Notebook and Google Colab and explained the essential role of open-source libraries such as TensorFlow, KERAS, OpenCV, Matplotlib, and NumPy in simplifying various aspects of the FER project, from data manipulation and visualization to machine learning model development. Subsequently, will detail the proposed FER algorithm employing CNN, covering data preparation, model architecture design, hyperparameter tuning, training, testing, and performance analysis. Finally, will discussed the integration of the FER system into a webcam for real-time facial expression analysis, offering a comprehensive understanding of the entire process.

3.1 Modern Tools

In this section, we discuss the use of modern tools in this simulation to achieve project results. It uses open-source software and libraries for each different task. For tasks such as acquiring datasets. This project uses the Kaggle website while for model design, simulation and optimization and model comparison, we use modern tools in terms of open-source libraries such as TensorFlow, KERAS, OpenCV, Matplotlib and NumPy. Open-source libraries such as TensorFlow and KERAS are used in this project to build the model architecture and train the model. For OpenCV library is used as identify face in image or video stream while Matplotlib library allows visualization of model performance and confusion matrix to evaluate the performance. For NumPy library is used for manipulating and processing numerical data during generating confusion matrix. For the CNN model simulation task, this project uses the Google Colab platform which is Google's free cloud infrastructure that allows to write and execute python code.

3.1.1 Jupyter Notebook

Jupyter notebook is an open-source tool that operates in web applications, it allows users to write python code for various tasks, as well as create visual representations, and mathematical formulas. It is structured based on cells where users can separate their code or text to divide the coding part. In this project, Jupyter notebook is used to implement real time monitoring by loaded the proposed model. Each cell is divided into code such as code to define the library, and code to perform real time monitoring using a webcam. It makes it easy for users to organize programs. In addition, Jupyter Notebook also caters to different types of users and their

preferred languages. It excels in analyzing data, research efforts, educational purposes and collaborative initiatives based on visualization capabilities. In addition, Jupyter Notebook allows users to easily share their work and increases reproducibility and facilitates collaborative exploration.

3.1.2 Google Colab

Google Colab, is cloud-based platforms that offers several advantages for FER system. Google Colab provides free access to GPU and TPU resources, significantly speeding up model training and enabling efficient handling of large datasets and complex architectures. This is because the amount of model parameters and large datasets causes the training process to be longer and requires a good GPU to train the model. Therefore, this project uses Google Colab as a platform to process images, build CNN models, train models and evaluate model performance.

Additionally, it integrates seamlessly with popular machine learning libraries, such as TensorFlow and KERAS, making it easy to import and use pre-built models, datasets, and libraries.

3.1.3 Open-Source Library

This project uses open-source libraries to provide pre-implemented functions, classes or algorithms that users can leverage to perform various tasks. In this project, libraries such as Matplotlib are used for numerical calculations, data manipulation and data visualization. Machine learning libraries such as TensorFlow and OpenCV offer functions and classes for training and deploying machine learning models. This

library allows users to access powerful tools and functions that simplify complex tasks and speed up their development process. To use this open-source library, it is necessary to install the library using a package manager such as “pip” or “conda”. Once a library is installed, it can be imported into Notebook using a standard import statement such as "import library-name" which allows the user to use the library. With that, users can use open-source libraries to program code for various tasks.

3.1.3.1 TensorFlow

TensorFlow is an open-source machine learning framework. The task of facial expression identification can be successfully solved using the open-source library TensorFlow which is a powerful deep learning framework. A CNN can provide the data. It is good enough to extract important aspects from facial images. KERAS, the high-level API provided by TensorFlow, makes it easier to create and train CNN architectures. When training the model, the prepared data set is fed into the CNN, and the model parameters are optimized to reduce the desired loss function. TensorFlow offers a variety of optimization techniques, including Adam and Stochastic Gradient Descent (SGD), enabling effective training. Therefore, TensorFlow can create reliable and accurate facial expression recognition models.

3.1.3.2 KERAS

KERAS is an open-source learning framework used for the process of building and training our deep learning models. Based on this project, KERAS is used to import layers from KERAS that represent different neural network

components such as input layer, convolution layer, maxpooling2D, fully connected layer and others. This library can also separate the dataset into two, namely training dataset and validation dataset. For model training, KERAS offers a simple loading method that makes training easy where users can monitor training progress and use callbacks to stop early or save the best model.

3.1.3.3 OpenCV

OpenCV is an open-source library for computer vision tasks. It offers a variety of algorithms and features which include deep learning-based techniques, which allow it to find and identify faces in images or video streams. In addition, OpenCV can also be combined with deep learning or machine learning frameworks to detect facial expressions. To train a deep learning model, it can use the extracted features as input. OpenCV is an important library for performing real time monitoring. To make the system's output easier to understand, it can display the expected expression, draw a bounding box around the face, or overlay a recognized face landmark on the screen or video frame.

3.1.3.4 Matplotlib

Matplotlib is an open-source library used to provide various functions such as creating plot-style visualizations. In this project, the Matplotlib library is used for plotting model accuracy and loss graphs for the model that has been trained. With that, it makes it easier to analyze training accuracy, validation accuracy, training loss and validation loss. Matplotlib was also used to plot the distribution of facial

expression labels, providing insight into the balance of different expressions in the data set. In addition, Matplotlib allows visualization of model performance such as confusion matrix to evaluate the performance of different models or compare performance on different data sets. A confusion matrix, showing the predicted and true distribution of labels, can also be created using Matplotlib to gain insight into the model's classification performance.

3.1.3.5 NumPy

NumPy is an important tool that manipulates and analyses numerical data. It is an open-source library. NumPy's capacity to manage big datasets effectively is one of its key features. Several mathematical functions that are necessary for this face expression recognition challenge are also provided by NumPy. These include signal processing, statistical computations, linear algebra procedures, and fundamental arithmetic operations. This approach is used to image processing tasks including image normalization and feature extraction. Even when working with big datasets, this library can guarantee quick and precise computations.

3.2 Proposed FER Algorithm using CNN.

Based on this project algorithm, the FER project is divided into several tasks, as shown in Figure 3.1. First, this project selects the FER2013 dataset as the input image for the training model. It is taken from the Kaggle website. Next, the dataset is loaded on Google Colab as a database preparation. Next, before designing and training the model, the image on the database must be processed such as normalizing

the image, applying data augmentation technique, separating the data set into training and validation data sets of 70% and 30% of the train dataset, respectively. Next, the project chooses a deep learning technique which is CNN to classify emotions into seven emotions. This project needs to design a custom CNN model according to the appropriate parameters. There are several parameters involved in designing a custom CNN model, including input shape, convolution layer, pooling layer, activation function, fully connected layer, output layer, and the other. After completing the CNN model design, the CNN model needs to be trained on several epochs. Next, the validation accuracy is observed to ensure that the model predicts well on images that the model has never seen during training. If the accuracy of the model is not satisfactory, the CNN custom model needs to tune the hyperparameters which are learning rate, batch size and epoch. The values for tuning these hyperparameters are 0.01, 0.001, 0.0001 for the tuning learning rate, for batch sizes 8, 16, 32, 64 and 128, and epochs 50, 75, 100, 125, and 150. After modifying the hyperparameters the model should be retrained until achieving satisfactory verification accuracy. If it can achieve satisfactory training and verification accuracy, this project tests the model that has been trained with some sample images from the FER2013 test dataset where the model never sees that image during the training process. If the images that have been predicted by the model obtained more incorrect predictions than correct predictions, the model needs to be retrained by tuning hyperparameter. If the model succeeds in predicting images with more correct than incorrect the model will be tested for performance such as model accuracy, precision, recall and F1-score that have been trained with the test dataset in the FER2013 dataset. Finally, this FER system will be used in real time using a webcam.

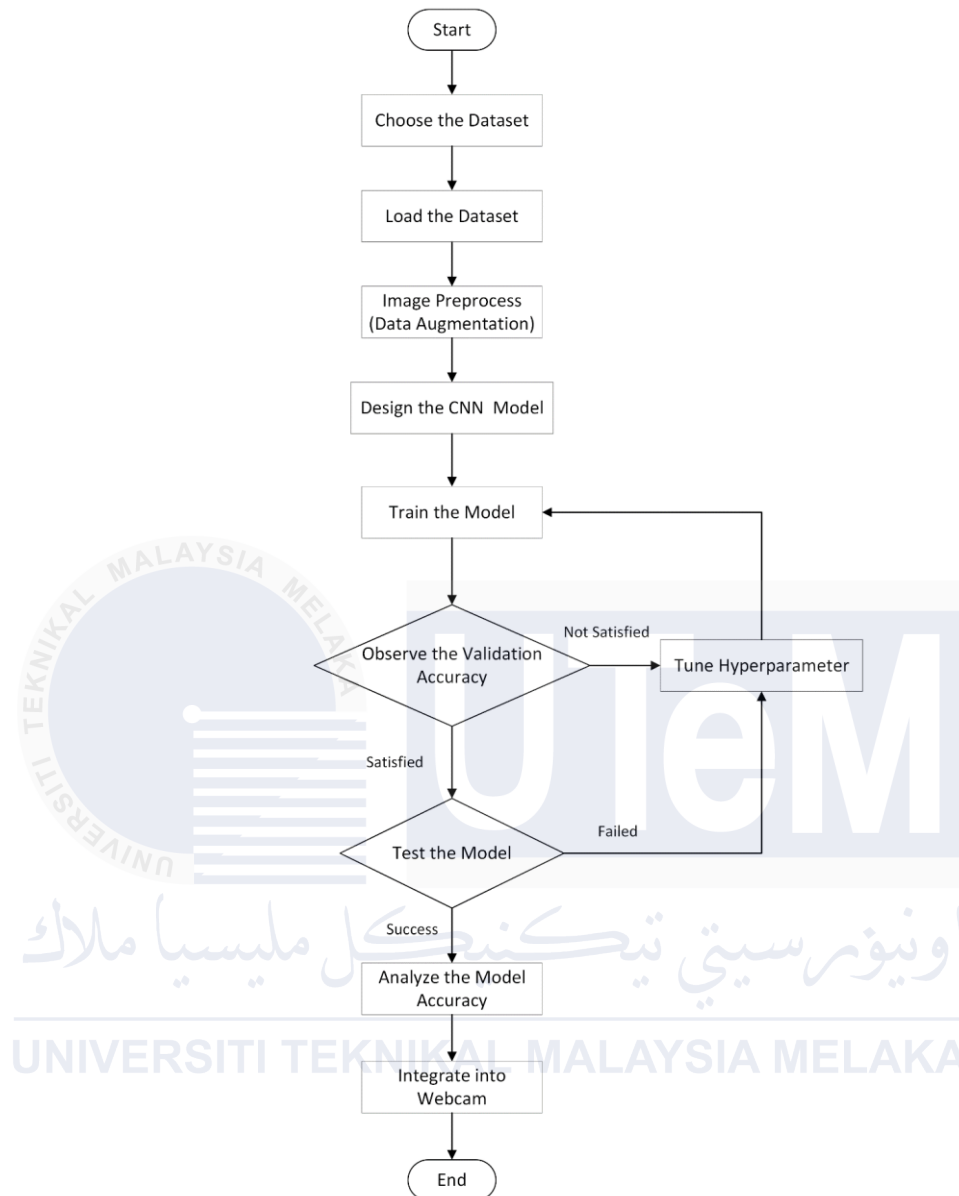


Figure 3.1: The flowchart diagram in FER project

3.2.1 Preparing and Processing Images in Database

The project starts by selecting a dataset that has the seven basic emotions which available in Kaggle. This project has selected the FER2013 dataset for model training and testing. Since the FER2013 dataset includes images with seven different emotions such as angry, disgust, fear, happy, neutral, sad and surprise. It is important

to have access to large datasets of facial images displaying various emotions and expressions with different persons. As a result, the FER2013 dataset will be used because it has 35887 images. To ensure that images are in a format according to deep learning architecture, the dataset must first be preprocessed. During image preprocessing, the train dataset is split into training and validation datasets where 70% for training and 30% for validation and normalized the pixel values because neural networks tend to perform better when the input data is scaled to a smaller range. It is essential to help in stabilizing and speeding up the training process. Normalization prevents large input values from dominating the learning process and makes it easier for the model to learn meaningful patterns. Besides that, this project applied data augmentation techniques to reduce the overfitting problem and obtain better model accuracy by generating diverse variations of the existing training images. These techniques involve rotations, shear, width shift, height shifts, and horizontal flip as shown in Figure 3.2. Therefore, enriching the dataset and allowing the model to learn more generalized and robust features, resulting in improved performance on unseen data.

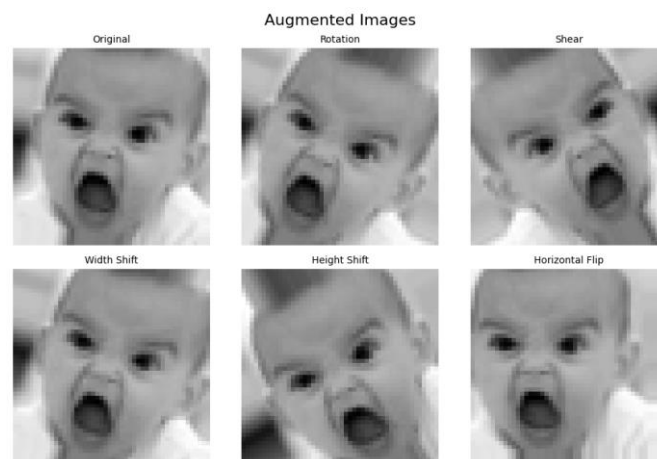


Figure 3.2: The augmented images

3.2.2 Design the CNN Custom Model

Next, to design a custom CNN model, there are several parameters and layers to consider such as input shape, convolution layer, activation function, pooling layer, fully connected layer and output layer. As shown in Figure 3.3 the input shape used is 48x48x3 is the same as the image input shape from the FER2013 dataset and the architecture of this model consists of 3 CNN modules, 4 fully connected layers, and the output layer with the activation function used is SoftMax. For the first module, it consists of 2 convolutional layers with a feature map size of 46x46x256, where the convolutional layer is used to extract relevant features from the input image. After that, the maxpooling2D layer is used to reduce the spatial dimension. In the second CNN module, there are 2 convolutional layers with a smaller feature map size of 23x23x128. Again, the maxpooling2D layer is used to reduce the spatial dimension. As the network grows, the dimension of the feature map decreases due to a combination of convolution and pooling operations. Moving to the third CNN module, it consists of 2 convolutional layers with a smaller 11x11x64 feature map size. Similarly, the maxpooling2D layer is used to reduce the spatial dimension. Each CNN module uses the RELU activation function. Regarding the fully connected layer, it consists of 4 layers, each with a different number of neurons, namely 512, 256, 128, and 64. All these layers use the RELU activation function. Finally, the output layer with activation function SoftMax consists of 7 neurons for the classification of 7 emotions.

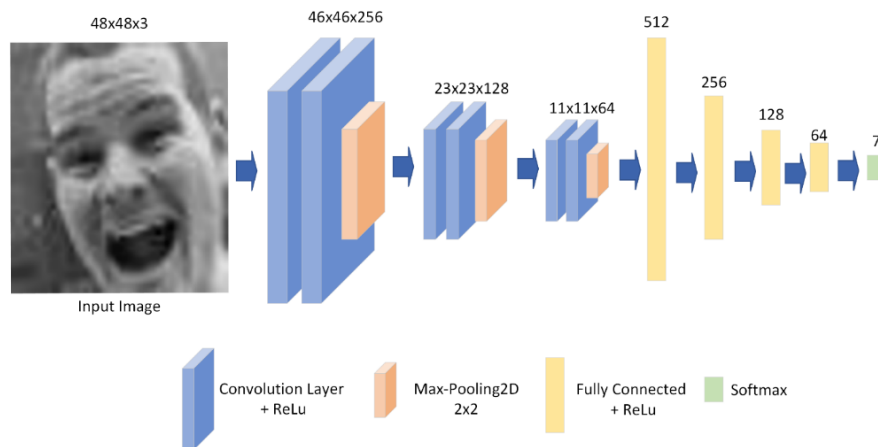


Figure 3.3: Designed the CNN custom model.

3.2.2.1 Input Shape

One of the characteristics in the CNN model is the shape of input images where it is important to determine the dimensions of the input image or data that will be processed by the CNN model. This project uses a 48x48x3 image format where 48x48 pixels while 3 is the number of channels in the images according to the size from the FER2013 dataset. Therefore, the CNN model will expect an input image of 48 x 48 pixels. The input shape parameters are important in establishing the layer size and structural compatibility of the CNN model. A CNN model will be made to handle and process images of that size if the input format is set to 48x48. This ensures that the model is properly prepared to examine the facial expressions recorded for the datasets.

3.2.2.2 Convolutional 2D Layer

The fundamental component of the CNN architecture that extracts various levels of features from the input image is the convolutional 2D layer. The input data is convoluted by a convolution layer to apply a collection of learnable filters, often known as kernels or feature detectors. It is based on this project and extracts local features or patterns from the input data using 3 to 3 kernel filters. Figure 3.4 illustrates how each filter convolutions over the input data in a sliding window manner during the convolution procedure.

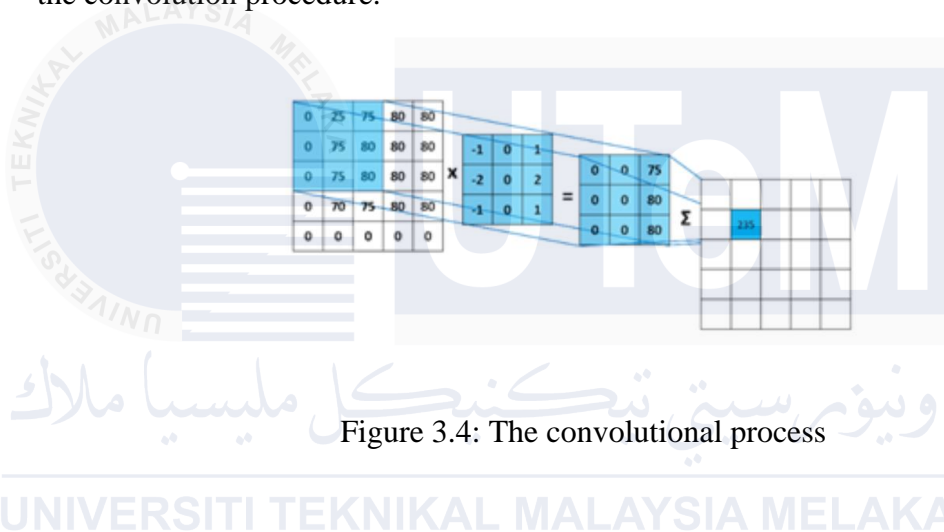


Figure 3.4: The convolutional process

The filter adds the result after multiplying its weight element by element by the appropriate input value at each place of its acceptance field. A feature map is produced as a result of this process, and it shows how a particular filter responds to various patterns and features contained in the input. The main goal of the convolution layer is to identify important features at various spatial locations of the input and capture local spatial dependencies. Convolutional layers can automatically learn and extract different low-level and high-level features, such as edges, vertices, textures and more complex patterns specific to the job at hand, by learning filter weights through training. Edges, colors and lines are detected at a low level by the first-round

operation, while the high-level features are extracted by the second-round operation.

The winding operation is represented as follows:

$$A[i, j] = \sum_{m=0}^{m-1} \sum_{n=0}^{n-1} F(m, n) \times I(i - m, j - n) \quad (3.1)$$

Where:

$A[i, j]$: The output feature map

i, j : i and j denote by the spatial position on the feature map.

m : m is the number of rows in the filter.

n : n is the number of columns in the filter.

The convolution layer produces several feature maps, each of which shows the activation of a different filter. A RELU is then applied to these feature maps to create non-linearities and increase the capacity of the network to model complex relationships.

3.2.2.3 Pooling Layer

Following the convolution layer, the parameter pooling layer is a critical component. The main goal of this operation is to preserve the most critical information while reducing the spatial extent of the feature map obtained from the convolution layer, pooling, which works as a form of downsampling, is used to reduce the computational complexity of the network, control the attachment and improve the obtained features. Pooling operations are usually applied independently to each feature map, which is achieved by dividing it into non-overlapping regions

referred to as pooling windows. maxpooling2D was used to reduce the spatial dimension and downsampling the feature maps in line with this effort. maxpooling2D is generally more efficient than the mean and sum pooling methods because it allows selection of the most relevant pixels and operates at the shortest speed. Following collection, the feature map output dimensions are as follows:

$$W_c = \left(\left(\frac{(W_{in} - F_w + 2P)}{S} \right) + 1 \right) \quad (3.2)$$

$$H_c = \left(\left(\frac{(H_{in} - F_h + 2P)}{S} \right) + 1 \right) \quad (3.3)$$

$$D_c = K \quad (3.4)$$

Where:

W_c : The width of the output feature map after the pooling operation

W_{in} : The width of the input feature map to the pooling layer

F_w : The width of the pooling window

P : The amount of padding

S : The stride of the pooling operation

H_c : The height of the output feature map after the pooling operation

H_{in} : The height of the input feature map to the pooling layer

F_h : The height of the pooling window

D_c : The depth of the output feature map after the pooling operation

K : The number of filters

3.2.2.4 Activation Function

The output of every neuron or node in a CNN is subjected to an activation function. The activation function imparts non-linearity to the model, facilitating its ability to discern and acquire knowledge of complex correlations that exist between inputs and outputs. Selected activation functions have a substantial impact on the capacity of the network to approximate and represent complex functions. The input and hidden layers of this framework employ RELU, while the SoftMax activation function transforms the previous layer's output into probabilities denoting the proportion of times an input is assigned to each class. The nonlinear RELU activation function that follows the convolution procedure is illustrated in Figure 3.5.

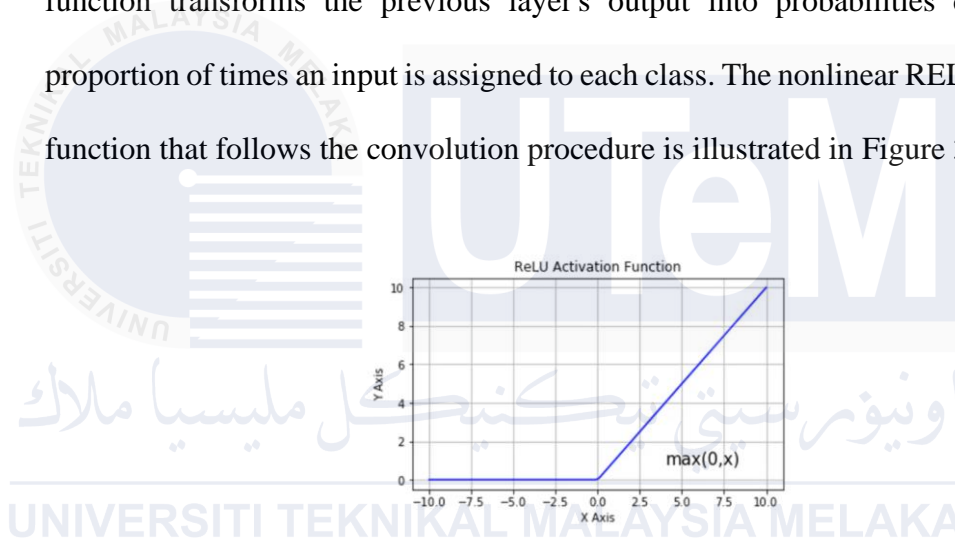


Figure 3.5: The RELU activation function

It is the optimal function for a CNN architecture due to the fact that it aids in mitigating the gradient problem introduced by deep neural networks. In the same way that it can expedite the learning process in comparison to alternative activation methods. Its definition is as follows:

$$f(x) = \begin{cases} \max(0, x), & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (3.5)$$

Where:

$f(x)$: The RELU function

$\max(0, x)$: The maximum of 0 and x

x : The output of a neuron

3.2.2.5 Fully Connected Layer

Fully connected layers, also known as dense layers in CNN architecture, are used in the project to be responsible for creating the final predictions based on features learned from previous layers. It creates a fully connected network structure by connecting every neuron in the previous layer to every neuron in the current layer. Learning the non-linear combination of information obtained by convolution and fusion layers is the main purpose of fully connected layers. A fully connected layer, which functions as a conventional neural network layer with a learnable set of weights and biases, takes these flattened features and feeds them into it. To create the final output, the fully connected layer runs a sequence of matrix multiplications and uses activation functions. Based on this project, the fully connected layer, it consists of 4 layers, each with a different number of neurons, namely 512, 256, 128, and 64. All of these layers use the RELU activation function while the output layer consists of seven classes, and it uses SoftMax as the activation function and is placed after the layer is fully connected. This is to enable the model to generate predictions, this activation function transforms the output of a fully connected layer into a probability distribution over the classes.

3.2.3 Training and Observe the Validation Accuracy

The CNN model needs to be trained in several epochs and this project trains the model with 50 epochs as an initial as shown in Figure 3.6 and then observes for the accuracy of validation to determine whether it is satisfactory or not. If the accuracy of the model is not satisfactory, the CNN model needs to tune hyperparameters such as learning rate, batch size and number of epochs. After modifying the hyperparameters, the model needs to be retrained until it achieves satisfactory validation accuracy. If the validation accuracy is satisfactory, the model is evaluated by analyzing model performance such as analyzing overall accuracy, precision, recall and F1-score. Training the model to recognize and extract important facial features that show different expressions is an important step in the facial expression recognition process. CNN model was used to analyze images and identify relevant features. The CNN models need to be trained after the dataset has been preprocessed. The preprocessed image is fed into the model, and its parameters are modified. After learning the deep features, the FER system will classify the provided faces into one basic emotion. It is necessary to create training and test sets to train, evaluate and validate our model on test data. The training procedure is important because it teaches the model what to look for and predict. The FER system cannot produce an accurate model without it. By examining discrete regions of the input image at a time, the network trains to recognize patterns in the input data, such as edges and shapes. One frequent use of CNNs is FER. In this task, a CNN is trained using a train dataset, each of which has an associated facial emotion identified in each image. The network gained the ability to recognize features associated with various facial expressions, including the shape of the mouth and eyebrows, eye

placement, and forehead wrinkles. Applying learned features to new input images allows the network to predict facial expressions after they have been trained.

```

Epoch 30/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.9323 - accuracy: 0.6516 - val_loss: 0.9963 - val_accuracy: 0.6303
Epoch 31/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.9172 - accuracy: 0.6590 - val_loss: 1.0133 - val_accuracy: 0.6289
Epoch 32/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.9173 - accuracy: 0.6577 - val_loss: 1.0690 - val_accuracy: 0.6056
Epoch 33/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.9107 - accuracy: 0.6595 - val_loss: 0.9771 - val_accuracy: 0.6348
Epoch 34/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.9083 - accuracy: 0.6626 - val_loss: 0.9754 - val_accuracy: 0.6348
Epoch 35/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8968 - accuracy: 0.6674 - val_loss: 0.9834 - val_accuracy: 0.6403
Epoch 36/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8912 - accuracy: 0.6668 - val_loss: 0.9818 - val_accuracy: 0.6395
Epoch 37/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8806 - accuracy: 0.6709 - val_loss: 0.9946 - val_accuracy: 0.6343
Epoch 38/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8773 - accuracy: 0.6720 - val_loss: 1.0230 - val_accuracy: 0.6267
Epoch 39/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8727 - accuracy: 0.6726 - val_loss: 0.9966 - val_accuracy: 0.6346
Epoch 40/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8709 - accuracy: 0.6774 - val_loss: 0.9986 - val_accuracy: 0.6346
Epoch 41/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8620 - accuracy: 0.6813 - val_loss: 0.9660 - val_accuracy: 0.6436
Epoch 42/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8491 - accuracy: 0.6844 - val_loss: 1.0128 - val_accuracy: 0.6354
Epoch 43/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8499 - accuracy: 0.6827 - val_loss: 0.9933 - val_accuracy: 0.6357
Epoch 44/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8386 - accuracy: 0.6918 - val_loss: 1.0233 - val_accuracy: 0.6290
Epoch 45/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8381 - accuracy: 0.6869 - val_loss: 1.0395 - val_accuracy: 0.6275
Epoch 46/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8310 - accuracy: 0.6905 - val_loss: 1.0027 - val_accuracy: 0.6367
Epoch 47/50
1256/1256 [=====] - 41s 32ms/step - loss: 0.8244 - accuracy: 0.6920 - val_loss: 1.0475 - val_accuracy: 0.6271
Epoch 48/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8141 - accuracy: 0.7004 - val_loss: 0.9842 - val_accuracy: 0.6400
Epoch 49/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8125 - accuracy: 0.6991 - val_loss: 1.0204 - val_accuracy: 0.6352
Epoch 50/50
1256/1256 [=====] - 40s 32ms/step - loss: 0.8095 - accuracy: 0.6984 - val_loss: 0.9558 - val_accuracy: 0.6533

```

Figure 3.6 The training processes captured from 30 to 50 epochs.

3.2.3.1 Tuning Hyperparameter Batch Size

Batch size is a hyperparameter that plays an important role in the model training process. It is also important to get better validation accuracy. Batch size is the number of data samples used in each iteration of the training algorithm. It specifies the number of examples from the training data set that the model processes in one forward and backward pass before updating the model parameters. The smaller the batch size value, the model processes fewer instances in each iteration. This indicates that a significant number of iterations are required to finish an epoch, or a complete run of the training dataset. The smaller batch makes each iteration quicker, but since more iterations are required to view the complete dataset, the overall

training time can increase. In addition, a large batch size value means more instances are processed by the model in each iteration. As a result, each epoch may be completed with fewer iterations, which may speed up the training process overall. Based on this project, several batch sizes are used to determine the validation accuracy that is optimal and suitable for the CNN model that was constructed. Among the batch sizes that have been tuned are 8, 16, 32, 64, and 128. Tuning the batch size is important because it helps strike the right balance between training speed and model generalization.

3.2.3.2 Tuning Hyperparameter Learning Rate

The learning rate controls the size of the step used by the model to update its parameters when training in a CNN architecture. It regulates the rate at which the model learns from the training data. When training a CNN model, it is crucial to keep the learning rate in mind as it influences the convergence speed and the learned model quality. If the learning rate is set too high, the model can end up wildly off course and never converge. If the learning rate is excessively slow, the model may stall out before reaching convergence or settle for an imperfect solution. It uses a learning rate of 0.001 to keep the training process stable, according to this project. It reduces the possibility that the loss function may vary substantially due to major weight changes. Thus, to ensure the successful training and optimization of the CNN model, this project requires the adjustment of the learning rate. Several learning rate values have been fine-tuned, including 0.01, 0.001, and 0.0001.

3.2.3.3 Tuning Hyperparameter Epochs

The number of epochs is a hyperparameter that controls how often the full training data set is sent back and forth across the network during training. Input is fed into the network in the forward pass to provide predictions, and the model then adjusts its weights in the backward pass depending on the calculated losses and the chosen optimization strategy. The number of epochs chosen influences the effectiveness of the model learning from the training data. A model may become less appropriate if the number of epochs is set too low, fail to recognize complex patterns and perform poorly on both the training and test data sets. Conversely, increasing the number of epochs too much may result in overfitting, when the model becomes over-adapted to the training set and struggles to perform well on new, untried data. Therefore, this project needs to utilize different epoch values to ensure that it can achieve both good validation accuracy and optimize performance for the CNN model. Various epoch values have been tuned, ranging from 25 to 125, including 25, 50, 75, 100, and 125.

3.2.4 Testing the Model

After successfully achieving validation with higher accuracy of trained CNN model, it should be tested using some sample of images from the FER2013 test dataset, as illustrated in Figure 3.7. This test aims to evaluate the CNN model's ability to correctly predict human expressions. The figure displays image samples that the model has predicted incorrectly and those it has predicted correctly. If the model makes more incorrect predictions than correct ones, it is an indication that the CNN model needs to be retrained by tuning its hyperparameters. Conversely, if the CNN

model predicts more images correctly than incorrectly, it signifies success, and further analysis of the model's performance becomes necessary.



Figure 3.7: The images of predicted emotion from the test dataset

3.2.5 Analyzing the Model Performance

After training a deep learning model, it is necessary to test the model and then evaluate the performance of the model in terms of precision, recall, F1-score and model accuracy. In FER, the confusion matrix is the best tool used to evaluate the model's performance because it provides a visual representation of the model's prediction compared to the actual ground truth label. A confusion matrix is a table that compares the deep learning model's predicted expression with the actual expression for the test image array. The table has a row for each actual expression and a column for each predicted expression. The component of the confusion matrix indicates the

number of correctly or incorrectly classified model images. By understanding the confusion matrix, it is possible to evaluate and fine-tune the trained model, address its strengths and weaknesses, and make informed decisions to improve its performance for specific classification problems. This tool can analyze such things as model accuracy, precision, recall, and F1-score, where the formulas are:

For accuracy:

$$Accuracy = \frac{\text{Correctly predicted Images}}{\text{Total images}} \quad (3.6)$$

For precision:

$$Precision = \frac{TP}{TP+FP} \quad (3.7)$$

For recall:

$$Recall = \frac{TP}{TP+FN} \quad (3.8)$$

For F1-score:

$$F1 \text{ score} = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (3.9)$$

Where:

True Positive (TP) : The number of samples that were correctly predicted as positive.

True Negative (TN) : The number of samples that were correctly predicted as negative.

False Positive (FP) : The number of samples that were incorrectly predicted as positive.

False Negative (FN) : The number of samples that were incorrectly predicted as negative.

In this project, to evaluate the model performance, this project writes the program code to display the confusion matrix and a classification report where it shows parameters to evaluate the performance of the model as shown in Figure 3.8. The table has a row for each actual expression and a column for each predicted expression. Therefore, this project will use a confusion matrix that can be used to analyze the performance of the model in terms of model accuracy, precision, recall and F1-score, which provides more detailed information about the performance of the model for each expression.

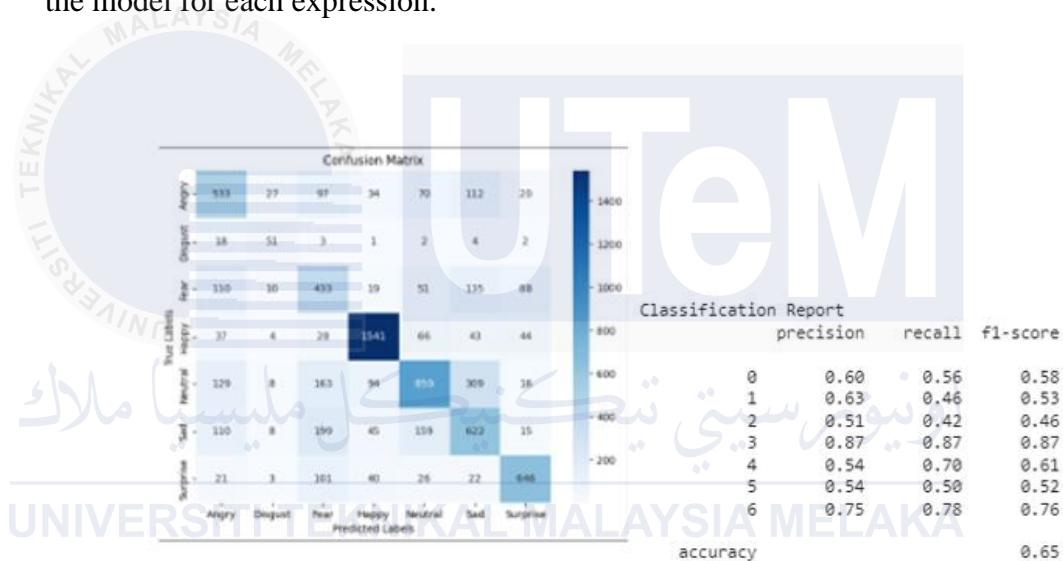


Figure 3.8: The confusion matrix and classification report.

3.2.6 Integrate the FER system in Real Time

After analyzing the performance of the model, the project continued by integrating the FER system into a webcam. This integration enables real-time facial expression monitoring and recognition from live video streams captured by web cameras. The integration process usually involves several steps. First, the proposed CNN model was loaded, along with any necessary dependencies such as image processing libraries and webcam interfaces. Next, the webcam feed is accessed, and

frames are captured continuously. Each frame is then pre-processed to prepare it to be fed into the FER model. This preprocessing may include resizing, normalization and any other transformations necessary to ensure compatibility with the model's input requirements. To detect faces in each frame, this project uses the "Haar Cascade Classifier" code. This aims to reduce the noise on each frame and make it easier for the uploaded model to predict the detected facial expressions. Once the frames are preprocessed, they are sent through the FER model for prediction. The model analyzes facial features and predicts the corresponding emotion or expression.

3.3 Summary

In summary of Chapter 3, discussed about the modern tools used in this project, and proposed FER algorithm using CNN. For simulation environment, Google Colab is a cloud based Jupyter Notebook platform that enables users to run simulations and check results in a simple and collaborative environment. It supports programming languages like Python and provides resources like CPUs for speeding up calculations. Besides, open-source libraries like NumPy, and Matplotlib are used for numerical calculations, data manipulation, and visualization. Machine learning libraries like TensorFlow and OpenCV offer functions and classes for training and using models, saving time and effort compared to building complex services from scratch. Moreover, Matplotlib is an open-source library used for facial expression recognition, providing functions and plot styles. NumPy is a powerful Python library for facial expression recognition, offering efficient handling of large datasets and manipulation of multidimensional data. For the part of proposed FER algorithm using CNN, the FER project focuses on a deep learning technique called CNN to classify

emotions into seven emotions. The project uses the FER2013 dataset, consisting of 35887 facial image images, to train and test deep learning models. The dataset is preprocessed by dividing it into training and test sets, normalizing images and apply data augmentation technique. The CNN model is chosen due to its good classification accuracy in deep learning techniques. Parameters such as input shape, convolutional layer, pooling layer, activation function, fully connected layer and output layer are set to ensure the model's performance and generalization capacity. The number of epochs in a CNN architecture controls the frequency of training dataset sent back and forth. The trained CNN model is tested with image samples from dataset test FER2013 to evaluate its accuracy in predicting human expressions. If the model predicts less than satisfactory accuracy, it is retrained by tuning hyperparameters. The project then integrates the FER system into a webcam to analyze and recognize facial expressions from live video streams. The integration process involves loading a trained CNN model, preprocessing frames, and using the "Haar Cascade Classifier" code to detect faces. The FER model analyzes facial features and predicts corresponding emotions, which can be displayed in real-time through overlays, bounding boxes, or other visual representations.

CHAPTER 4

RESULTS AND DISCUSSION



In this chapter, discussed about the analysis of data augmentation for custom CNN model which is important for addressing the issue of overfitting problem.

Besides that, this chapter will explore the process of hyperparameter tuning such as learning rate, batch size and epochs to achieve the highest possible model accuracy.

Moreover, this chapter will include presentation of our proposed custom CNN model by using appropriate hyperparameter and confusion matrix for analyzing the model performance. Besides that, in this chapter will discuss the model accuracy and loss for three pretrained model which are VGG16, AlexNet and MobileNet to compare with proposed custom CNN model by using the same hyperparameter value. The comparison involves model accuracy, precision, recall and F1-score.

4.1 Data Augmentation Analysis for Custom CNN Model

In deep learning, data augmentation is an important technique used to increase the variety of available training data. The results of this experiment are illustrated in Figure 4.1, with epochs represented along the x-axis and model accuracy and loss along the y-axis. Without using data augmentation, accuracy generally shows an upward trend during the training phase but may reach a plateau or even begin to decline during the validation phase. At 10 epochs, the model begins to overfit, as evidenced by a significant increase in training accuracy peaking at 90.13 percent after 14 epochs. On the other hand, the validation accuracy showed a small increase to 54.89% after 5 epochs, but then leveled off between 18 and 50 epochs. Overfitting is indicated by the fact that the model's performance on new unseen data does not improve significantly past a certain point, as this difference shows. The continued reduction in training loss indicates that the model has been more accurately matched to the training data. Typically, this decrease occurs between 0 and 20 epochs.

However, the validation loss behaves differently. At seven epochs, it decreases to a minimum of 1.2090, indicating that the model has reached its maximum performance on the validation set. Once the validation loss exceeds this minimum value, it starts to increase, indicating that the effectiveness of the model decreases with the progression of training on unobserved data. Increasing validation loss following the minimum is a clear indication of overfitting. By implementing data augmentation techniques such as shear, rotation, width shift, height shift and horizontal flip, which helps in improving their robustness, generalization, and ability to learn features that are invariant to such transformations. As a result of the data augmentation process, the gap between training accuracy and validation accuracy is reduced compared to models that do not use augmentation. This indicates that by using augmentation data, the model has obtained

a more general representation, thus reducing the tendency towards overfitting. A steady increase in training accuracy is usually observed throughout the epochs. Validation accuracy exhibits a comparable pattern to training accuracy. The initial growth is more gradual, but the trend is toward greater stability at higher levels when compared to the non-augmented. Over epochs, the training loss exhibits a consistent downward trend as the model gains proficiency in fitting the training data. Nevertheless, the rate of decline may be marginally delayed in comparison to training that does not involve augmentation. Additionally, it may stabilize at a greater loss value as a result of the increased diversity of the augmented data. Similarly to the validation accuracy, the validation loss may initially resemble training without augmentation. However, stability may be achieved at a reduced loss value in comparison to non-augmented. The model may exhibit indications of enhanced generalization as it is exposed to a variety of instances throughout the training process.

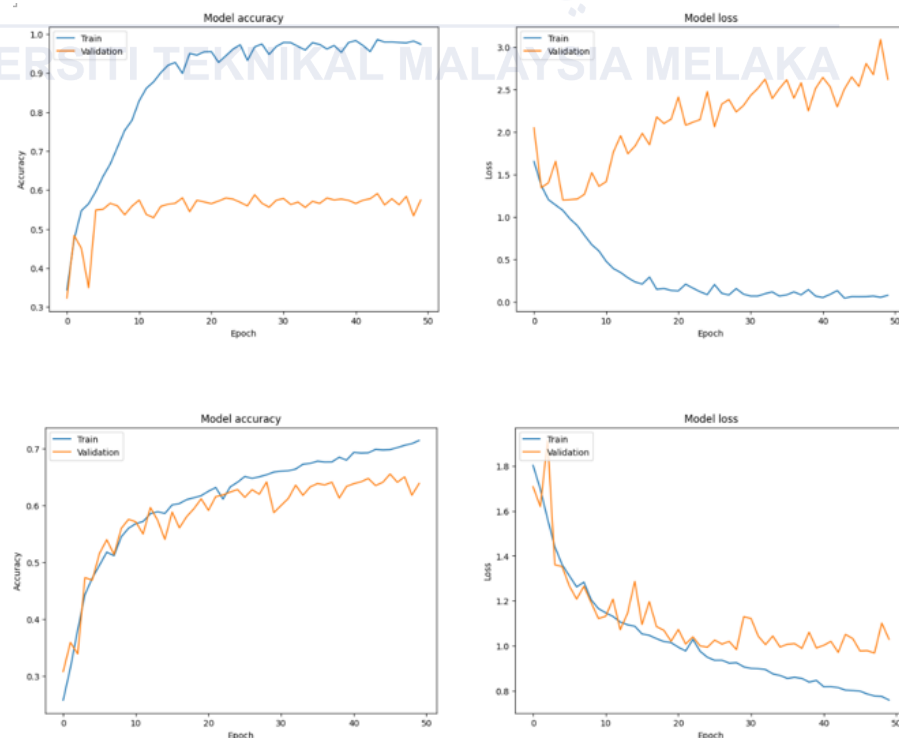


Figure 4.1: The model accuracy and loss without and with data augmentation

4.2 Hyperparameter Tuning for Custom CNN Model

When employing a custom CNN model for face emotion recognition task, hyperparameter tuning is an essential stage in any machine learning project. Finding the ideal number of values to maximize the model's performance involves adjusting the model's hyperparameters.

4.2.1 Tuning Learning Rate and Batch size

Hyperparameter tuning is a critical step in optimizing the performance of a neural network model. The learning rate controls the size of the step used by the model to update its parameters when training in a CNN architecture. It regulates the rate at which the model learns from the training data while batch size determines how many data samples are processed together during each training iteration. The objective of tuning process is to obtain the combination of that hyperparameters that maximizes the accuracy of the model for FER2013 dataset.

Table 4.1: Analyzing the accuracy of custom CNN model by tuning the hyperparameter learning rate and batch size.

Batch Size	Learning Rate	Model Accuracy (%)
8	0.01	63.00
	0.001	63.33
	0.0001	63.00
16	0.01	64.15
	0.001	65.27
	0.0001	64.84
32	0.01	63.03
	0.001	64.91
	0.0001	64.61
64	0.01	62.64
	0.001	63.39
	0.0001	63.93
128	0.01	61.95
	0.001	40.00
	0.0001	25.83

Based on Table 4.1 shows the results of tuning two important hyperparameters which are learning rate, and batch size, for a custom CNN model. The column indicates the batch size used for training the CNN model. For the column represents the learning rate used for training the model. For the column model accuracy (%) shows the accuracy achieved by the custom CNN model on the test dataset for different combinations of batch size and learning rate. Based on the observations, the

table shows the performance of a custom CNN model by varying two hyperparameters which are batch size and learning rate which are 8, 16, 32, 64, and 128 batch size while learning rate 0.01, 0.001, and 0.0001. For batch size 8, the model accuracy varies with different learning rates. The accuracy achieved is 63.33% when the learning rate is 0.001, while a learning rate of 0.01 and 0.0001 both result in slightly lower accuracies of 63.00%. At batch size 16, the learning rate of 0.001 achieved the highest model accuracy in this project which is 65.27%. This is the highest accuracy observed across all the tested configurations, suggesting that a moderate learning rate coupled with this batch size provides an effective balance for learning in this custom CNN model. Increasing the learning rate to 0.01 or decreasing it to 0.0001 both result in lower accuracies which are 64.15% and 64.84% respectively. This suggests that 0.001 is near the optimal learning rate for this batch size, providing enough step size to converge to a good solution without overshooting, while still being large enough to avoid getting trapped in local minima. For batch size 32, the model accuracy remains relatively stable across the different learning rates, with the highest accuracy of 64.91% at a learning rate of 0.001. This further supports the notion that a learning rate of 0.001 is effective for this custom CNN model. The smaller and larger learning rates produce slightly lower accuracies which are 63.03% and 64.61% respectively, indicating that there is less sensitivity to learning rate changes at this batch size compared to batch size 16. For batch size 64, there is a slight decrease in accuracy across all learning rates compared to batch size 32. The model performs best with the learning rate of 0.0001 and achieving an accuracy of 63.93%. The accuracy at a learning rate of 0.01 drops to 62.64% which the lowest among the learning rates for this batch size, suggesting that a higher learning rate may be less effective as the batch size increases. For batch size 128, a significant drop in accuracy is observed when the batch size

increases to 128. At a learning rate of 0.01, the accuracy is 61.95%, which is lower compared to smaller batch sizes. This could be due to the model not having enough updates per epoch to adequately learn from the data. At a learning rate of 0.001, the accuracy plummets to 40.00% such a severe drop might indicate that the step size is too small when combined with the large batch size preventing the model from effectively adjusting the weights. The accuracy further drops to 25.83% with a learning rate of 0.0001 reinforcing the idea that this learning rate is too conservative for the model to make meaningful progress especially when paired with a large batch size. Therefore, batch size 16 combined with a learning rate of 0.001 yields the highest accuracy at 65.27%, indicating a mutually enhancing relationship between batch size and learning rate. Smaller and larger learning rates show reduced accuracy, and while batch size 32 performs well, batch size 64 exhibits a slight dip in accuracy but benefits from a smaller learning rate which is 0.0001. Conversely, batch size 128 leads to a significant accuracy drop, particularly with learning rates 0.001 and 0.0001 which are 40.00% and 25.83% respectively. These observations emphasize the sensitivity of model performance to learning rate, especially with larger batch sizes, and underscore the importance of finding a balanced combination.

4.2.2 Tuning Epochs

Tuning hyperparameter epochs is crucial to identify the highest validation accuracy after obtained the hyperparameter learning rate and batch size that fits our CNN model. An epoch is one complete cycle through the entire training data set. During each epoch, the model will go through all the training data once.

Table 4.2: Analyzing the custom CNN model by tuning the hyperparameter epochs.

Batch Size	Learning Rate	Epochs	Model Accuracy (%)
16	0.001	25	61.05
		50	65.27
		75	64.02
		100	64.52
		125	63.96

In this experiment, Table 4.2 shows the results of analyzing the CNN model by tuning the hyperparameter epoch where the learning rate and batch size hyperparameter are set to 0.001 and 16 respectively. Based on the results in Table 4.2, when the number of epochs is set to 25, regardless of cluster size and learning rate, the model achieves an accuracy of around 61.05%. Increasing the number of epochs to 50 leads to an increase in accuracy which obtains the highest accuracy of 65.27%. Continuing to increase the number of epochs to 75 or 100 also leads to better accuracy compared to 25 epochs but with less fluctuation. At 75 epochs, the accuracy is around 64.02%, and at 100 epochs it is around 64.52%. Finally, at 125 epochs, the model's accuracy dropped slightly to 63.96%. From these results, it appears that increasing the number of epochs beyond 50 generally improves model accuracy up to a point around 100 epochs, after which there may be diminishing returns or even a slight decrease in accuracy. The optimal number of epochs may depend on other factors such as data set, model architecture and others. Therefore, this project achieved the highest proposed model accuracy at 65.27% when the hyperparameter batch size, learning rate and epochs were tuned.

4.3 Analyzing the Performance of the Proposed Custom CNN Model

Analyzing the performance of the proposed CNN model is crucial because it helps assess how well the custom CNN model is performing in recognizing facial expressions. By measuring its accuracy, it can determine the model's ability to correctly identify and classify different facial expressions.

4.3.1 Model Accuracy and Loss

In this experiment, this project analyzes the model accuracy and loss during the training process at several epochs, which are iterations over the entire dataset to show the performance of the model during the training process. The result of this experiment is given in Figure 4.2 where the x-axis represents the number of epochs, while the y-axis represents the accuracy and loss. Generally, accuracy is the fraction of model predictions that are obtained correctly. In this graph, the training accuracy is consistently higher than the validation accuracy, which is typical because the model learns directly from the training data. Validation accuracy is important because it provides an indication of how well the model generalizes to unseen data. The gap between train and validation accuracy suggests several attachments, where the model learns the training data very well but does not perform as well on the validation data. For the model loss graph, which plots the loss on the training and validation data sets over the same epochs. Loss is a numerical value calculated by the loss function, which measures how well the model predictions match the actual labels. The goal of the exercise is to minimize this value. In this graph, both the training and validation losses decrease over time, which is good because it shows that the model is learning. Therefore, this result is as expected where the measured data is consistent with the theoretical results shown in Figure 4.3 [26].

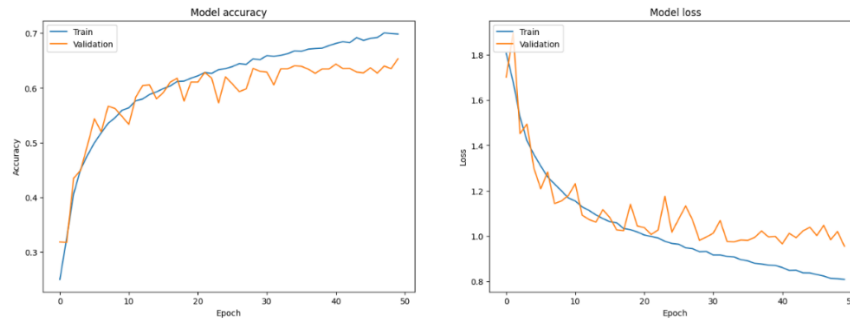


Figure 4.2: The plot graph illustrating the model accuracy and loss for the proposed model.

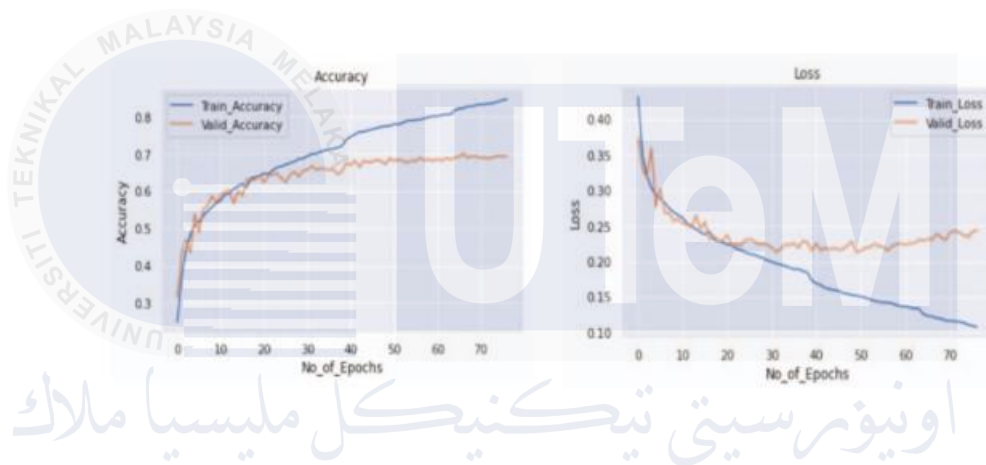


Figure 4.3 The plot graph illustrating the model accuracy and loss for CNN model in paper [26]

4.3.2 Confusion Matrix

In this experiment, this project also analyzes the confusion matrix shown in Figure 4.4 which is important to evaluate the performance of the classification model that appears to classify emotions into categories such as Angry, Disgust, Fear, Happy, Neutral, Sad and Surprise in terms of precision, recall, F1-score and model accuracy. This experiment aims to get more true positive values shown on the diagonal of the confusion matrix. This is because true positives represent correct predictions for each class. To analyze true positives, this project found that the proposed model correctly predicted Angry, which is 533 images, Disgust which is 51 images, Fear which is 433

images, Happy which is 1541 images, Neutral which is 859 images, Sad which is 622 images, and Surprise which is 646 images. The highest number of correct predictions was for Happy which was 1541, indicating that the model was most effective at identifying this emotion. Disgust has the lowest number of correct predictions of only 51 images, which may be due to under-representation in the training data or similarities to other emotions that confound the model. For misclassification analysis, this number shows how often an emotion is misclassified as another category. For example, Angry is most often confused with Surprise, which is 112 images, while Sad is often mistaken for Neutral with 159 images. Misclassification can indicate that certain emotions are being confused with each other, which may be due to similarities in the features representing these emotions. To analyze False Negatives, each row represents an instance of the true class. For example, on the Angry row, there were 533 correct predictions, there were misclassifications across other emotions, with Surprise being the most common error. The sum of the off-diagonal elements in a row indicates the total number of false negatives for that emotion. To analyze False Positives, each column represents an occurrence of the predicted class. For example, the Angry column shows that the model predicts Angry not only for true Angry events but also incorrectly for other emotions. The sum of the off-diagonal elements in the column indicates the total number of false positives for that emotion. Accuracy for each class can be calculated as the number of true positives divided by the total number of true positives and false positives of the entire column. Therefore, this proposed model performs well for the Happy and Neutral classes, with a high number of true positives. This model has difficulty with Disgust, possibly due to the fewer training samples in the FER2013 dataset. There were significant confounds between Angry and Surprise, as well as Sad and Neutral, suggesting that these emotions may have

similar characteristics in the context of the data or that the model needs further refinement to better distinguish them. With that, to prove this experiment is correct, it is compared with the paper [14] where the measured data is consistent with the theoretical results shown in Figure 4.5. Therefore, this result is as expected where there are more true positive values in the confusion matrix.

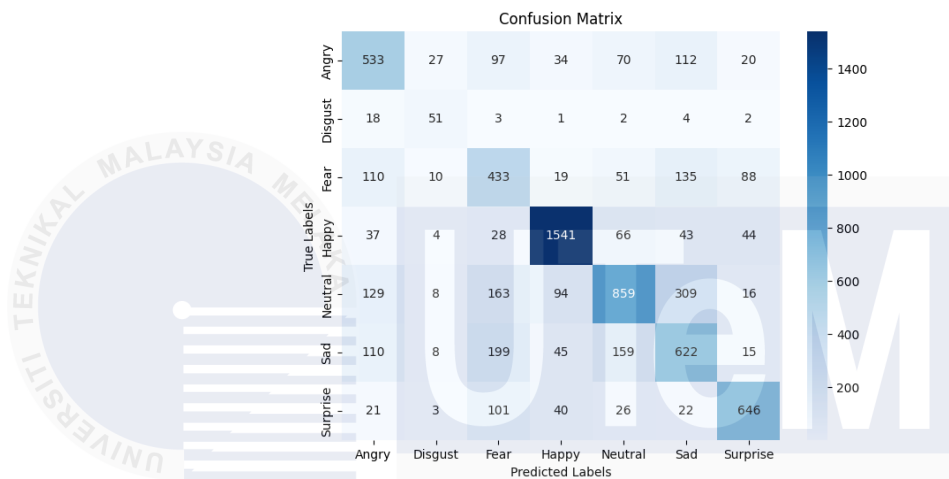


Figure 4.4: The Confusion Matrix for Proposed CNN Model

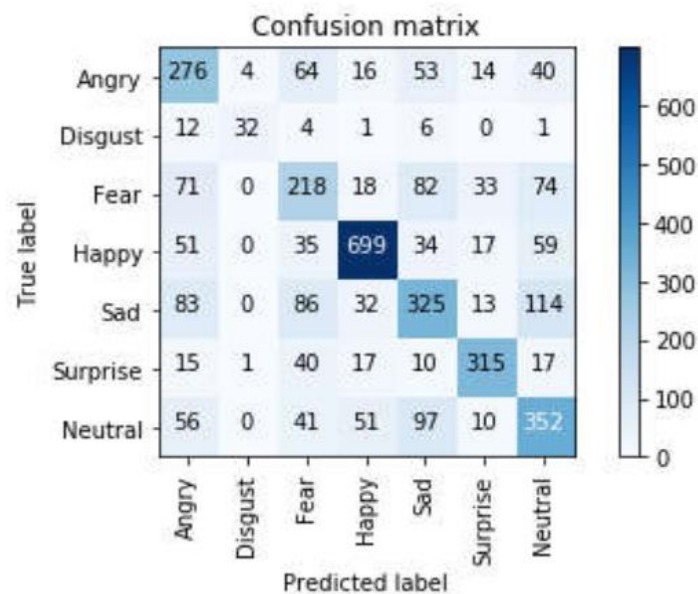


Figure 4.5: The Confusion Matrix for CNN architecture in paper [14]

In order to achieve one of the outcomes of this project, an experiment to analyze the proposed model's performance was conducted by analyzing the parameters of precision, recall, F1-score and model accuracy. It aims to determine whether the proposed model can classify the seven emotions well or poorly.

Table 4.3: Performance Analysis for Proposed CNN Model

Parameter	Emotion Class						
	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.60	0.63	0.51	0.87	0.54	0.54	0.75
Recall	0.56	0.46	0.42	0.87	0.70	0.50	0.78
F1-score	0.58	0.53	0.46	0.87	0.61	0.52	0.76
Overall Accuracy (%)	65.27						

Based on Table 4.3 shows the performance metrics for the proposed CNN model across various emotion classes namely Angry, Disgust, Fear, Happy, Neutral, Sad and Surprise. As a result of this experiment, the overall accuracy of the proposed CNN model, is 65.27%. It is as expected where the accuracy does not reach better because the FER2013 dataset has image imbalance in each class. Also, for the precision parameter, precision is the ratio of correctly predicted positive observations to the number of predicted positives. High precision is associated with a low false positive rate. The model performed best in the Happy class with an accuracy of 0.87, indicating that when it predicted Happy, it was correct 87% of the images. The Fear class had the lowest accuracy at 0.51, indicating a relatively high number of false

positives for this class. Also, for the recall parameter, is the ratio of correctly predicted positive observations to all observations in the actual class. A high recall is associated with a low false negative rate. Thus, Happy has the highest recall at 0.87, meaning that the model is good at identifying the Happy emotion from all actual Happy cases. The Disgust class had the lowest recall at 0.46, indicating that the model missed more than half of the true Disgust cases. Also, for the F1-score parameter, the F1-score is the harmonic mean of Precision and Recall. The F1-score reaches its best value at 1 for perfect precision and recall and worst at 0. The Happy class has the highest F1-score at 0.87, which suggests a good balance between precision and recall for this class. The Fear class had the lowest F1-score at 0.46, indicating that both precision and recall were low for this class. Based on the observations of this experiment, this model is very strong in detecting the Happy emotion, with high scores across accuracy, recall and F1-scores. The Fear and Disgust classes performed the weakest, with the lowest precision and recall, respectively. This is due to various factors, such as fewer training samples for this class, features that are not clear enough for the model to learn effectively, or similarities between these emotions and other emotions that confuse the model. Balanced F1-scores for Angry, Neutral and Surprise suggest that the model is well tuned for this class even with moderate precision and recall values.

4.4 Pretrained Models

Several pretrained models have been trained using the FER2013 dataset and using the same hyperparameter values of 16 batch size, 0.001 learning rate and 50 epochs in the proposed model. This is because this experiment compares our proposed model and pretrained models such as VGG16, AlexNet and MobileNet models in

terms of graph model accuracy and loss and some parameters to evaluate model performance namely precision, recall, F1-score and model accuracy.

4.4.1 Model Accuracy and Loss for VGG16 Model

In this experiment, the VGG16 model was trained and its accuracy and loss after the training process were plotted, as shown in Figure 4.6, so that training and validation accuracy and loss could be observed. The accuracy graph of the model illustrates its performance across epochs, specifically in terms of accuracy, on both the training and validation sets. An increase in both training and validation accuracy is indicative of learning. Nevertheless, the validation accuracy exhibits notable fluctuations and lacks smoothness, culminating in a precipitous decline at specific intervals, particularly between epochs 20 and 40. It is expected that the training accuracy would consistently surpass the validation accuracy, given that the model acquires knowledge exclusively from the training data. Nevertheless, variations and declines in validation accuracy serve as indicators that the model is overfitting and fails to adequately generalize to novel data. The right loss plot on the model loss graph illustrates the progressive reduction of the model's prediction error. A decline in the training loss signifies that the model's performance on the training set is improving. However, the rate of validation loss exhibits significant variability, peaking at specific epochs, specifically between 15 and 40 epochs. The observed surges in validation loss are concerning. This occurs because the learning rate of 0.001 is excessively high for the VGG16 model, resulting in improper weight updates. Additionally, a batch size of 16 may result in significant fluctuations in both precision and loss, whereas a batch size of 16 may oversimplify critical signals. This could be the result of an improper

learning rate that causes the model weights to be abruptly updated, validation data processing errors, and batch size issues.

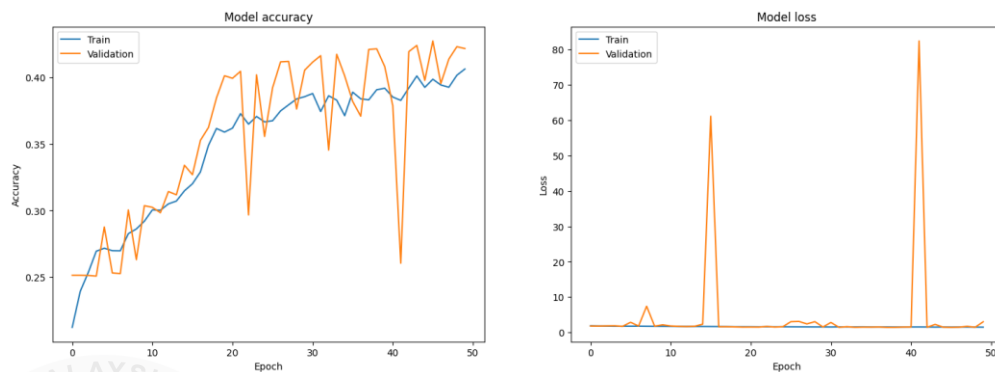


Figure 4.6: The plot graph illustrating the model accuracy and loss for the VGG16 model.

4.4.2 Model Accuracy and Loss for AlexNet Model

The experiment involved training the AlexNet model and graphing its accuracy and loss after the training process, as shown in Figure 4.7. This allowed for the observation of model accuracy and loss during the training. When examining the model accuracy graph, it is observed that both the training and validation accuracy exhibit a nearly horizontal trend, with only a marginal degree of variation centered around a particular value, which corresponds to an approximate accuracy of 25.1%. The model's ineffective learning is indicated by the close overlap between the training and validation accuracy lines, one would anticipate that the training accuracy would be greater and continue to rise as time progresses. Assuming a multi-class classification problem with over four classes, the model's performance is marginally superior to random, as proof by the low accuracy values. This is because the learning rate for the AlexNet model, which is 0.001, is excessively high and causes the weights to update incorrectly. The loss plot for the model loss analysis demonstrates an initial

significant decline during the first epoch for both the training and validation losses. Subsequently, the loss exhibits slight variations and lacks a distinct downward. The training and validation losses exhibit a high degree of similarity, maintaining a nearly constant value starting from approximately epoch 5. It appears that the model is not acquiring knowledge efficiently, as evidenced by the absence of any progressive improvement in accuracy and reduction in loss. In light of the model's low accuracy and flat loss curves, it is possible that it has been underfitted. This could be the result of an excessive level of regularization, inadequate feature extraction, or insufficient model complexity.

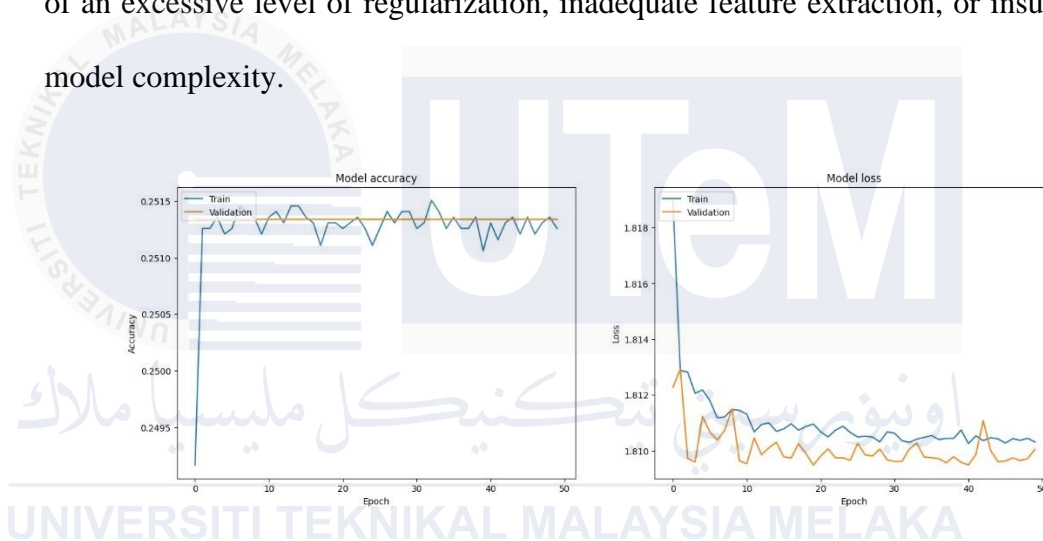


Figure 4.7: The plot graph illustrating the model accuracy and loss for AlexNet model.

4.4.3 Model Accuracy and Loss for MobileNet Model

The experiment involved training the MobileNet model and plotting its accuracy and loss after the training process, as shown in Figure 4.8. This allowed for the observation of model accuracy and loss after the training process. When examining the model accuracy graph, initially, the training accuracy improve relatively steadily, while the validation accuracy fluctuates. After around 20 epochs, the training accuracy continues to improve but with significant fluctuations, indicating some instability in

the learning process. The validation accuracy, shows high variability and does not improve consistently, suggesting the model not be generalizing well to unseen data. The gap between the training and validation accuracy suggests the model may be overfitting, as it performs better on the training data compared to the validation data. This is due to the learning rate where it determines the size of the steps the optimizer takes during training. A learning rate of 0.001 is generally considered moderate, however it still be too high for this pretrained model, causing the model parameters to update in large, erratic jumps. This can prevent the model from converging to a more stable and accurate set of parameters, leading to the high variability observed in the validation accuracy and loss. Besides that, this is due to batch size of 16 is relatively small, which can lead to higher variance in the gradient estimates during training. While smaller batch sizes help escape local minima, they can also cause the training process to be less stable, as seen in the fluctuations of the accuracy and loss curves. For model loss graph analysis, the training loss decreases over time, which is expected as the model learns from the training data. The validation loss, on the other hand, does not show a clear downward trend and has significant spikes, especially past 20 epochs. This behavior in the validation loss indicates that the model is not learning effectively from the validation dataset, which could be due to overfitting and a too high learning rate that causes the model parameters to change too drastically between epochs.

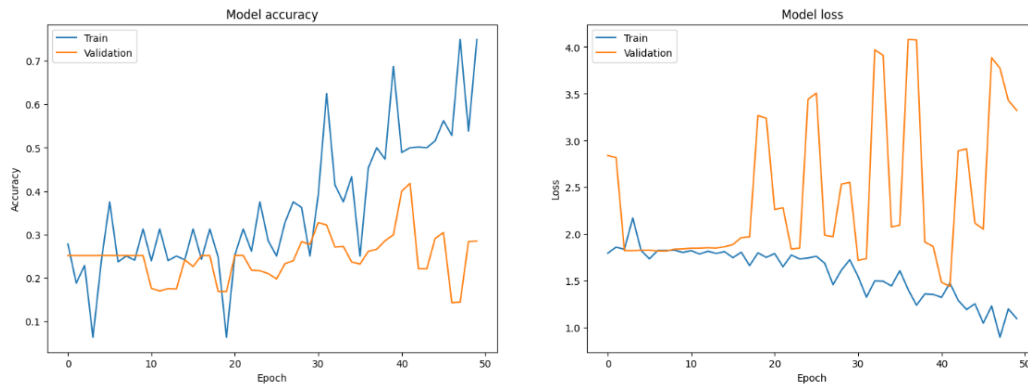


Figure 4.8: The plot graph illustrating the model accuracy and loss for the MobileNet model.

4.4.4 Comparison Between a Proposed Model and All Pretrained Models

One of the main aspects focused on this project is the comparison of model performance between the proposed model and all pretrained models which are VGG16, AlexNet, MobileNet using the same hyperparameters as in the proposed model, with a learning rate of 0.001 and a batch size of 16.

Table 4.4: The Comparison Between a Proposed Custom CNN Model and Three Pretrained Models

Model	Parameter	Emotion Class						
		Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Custom CNN Model	Precision	0.60	0.63	0.51	0.87	0.54	0.54	0.75
	Recall	0.56	0.46	0.42	0.87	0.70	0.50	0.78
	F1-score	0.58	0.53	0.46	0.87	0.61	0.52	0.76
	Overall Accuracy (%)	65.27						
VGG16	Precision	0.00	0.00	0.30	0.69	0.23	0.27	0.61
	Recall	0.00	0.00	0.08	0.74	0.20	0.65	0.68
	F1-score	0.00	0.00	0.13	0.72	0.22	0.38	0.64
	Overall Accuracy (%)	42.11						
AlexNet	Precision	0.00	0.00	0.00	0.25	0.00	0.00	0.00
	Recall	0.00	0.00	0.00	1.00	0.00	0.00	0.00
	F1-score	0.00	0.00	0.00	0.40	0.00	0.00	0.00
	Overall Accuracy (%)	24.71						
MobileNet	Precision	0.00	0.00	0.40	0.26	0.00	0.00	0.79
	Recall	0.00	0.00	0.00	1.00	0.00	0.00	0.32
	F1-score	0.00	0.00	0.00	0.41	0.00	0.00	0.45
	Overall Accuracy (%)	28.35						

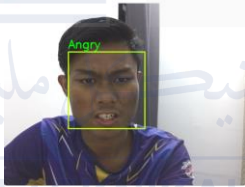
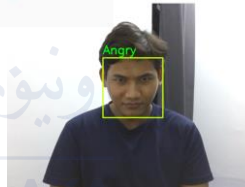





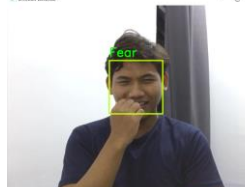


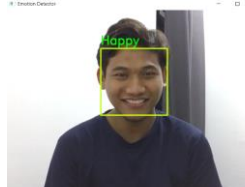
Based on Table 4.4, our proposed model shown the highest overall accuracy at 65.27% and balanced performance across precision, recall and F1-scores for various emotion classes. This project found that the emotions Happy and Surprise with high accuracy, which is 0.87 and 0.75, and recall which is 0.87 and 0.78, produce the highest F1-score for this class. This is because in the FER2013 dataset, there are the most images which are 7214 images for the Happy class and 3171 images for the Surprise class. Therefore, this model can recognize more features on different images during the training process. For comparison with other models, the VGG16 model exhibits a low overall accuracy at 42.11%, showing a lower performance than our proposed model. This is because the hyperparameter learning rate and batch size used are not suitable for the VGG16 model. As for precision, recall and F1-scores are generally low for most emotion classes, except for Happy, where it achieves a good F1-score of 0.72. VGG16 struggles to recognize emotions such as Anger, Disgust and Sadness. Next for the AlexNet model, AlexNet performed the worst among all models with an overall accuracy of 24.71%. It fails to provide meaningful precision, recall or F1-scores for most emotion classes, except for Happy, where it achieves a somewhat higher recall of 1.00, but still struggles with a low precision of 0.25. This is because the hyperparameter learning rate and batch size used are not suitable for the AlexNet model. Finally, for the MobileNet model, MobileNet offers better overall accuracy than AlexNet but is still lower than the custom model at 28.35%. It showed moderate precision of 0.40 and recall of 0.26 for Fear and high recall of 1.00 for Happy. However, F1-scores are generally low, indicating difficulty in correctly classifying most emotion classes. Therefore, our proposed model outperforms all three pre-trained models in terms of overall accuracy and balanced accuracy, recall and F1-scores across various emotion classes. VGG16, although better than AlexNet and MobileNet,

still falls short of the performance of the custom model. AlexNet and MobileNet, on the other hand, exhibit the lowest accuracy and insufficient emotion class recognition.

4.5 Real Time Monitoring using Webcam.

To achieve the project outcome, we have implemented and integrated our proposed emotion recognition model into a webcam application to enable real-time emotion analysis of individuals. Our model's performance has been evaluated based on the results presented in Table 4.5.

Table 4.5: Real time monitoring using webcam.

Expression	Person 1	Person 2	Person 3
Angry			
Disgust			
Fear			
Happy			



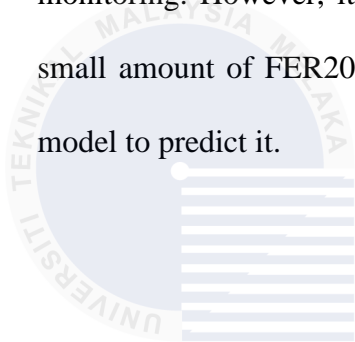
Based the Table 4.5, shows that our proposed model performed well in correctly recognizing seven different emotions for individuals labeled as person 1 and person 3. These emotions likely include common feelings like angry, disgust, fear, happy, neutral, sad, and surprise. This suggests that our model is effective understanding facial expressions in real-time. However, for person 2, our model struggled to identify the emotion of disgust and instead predicted surprise. This is due to the limitations of the dataset used to train our model, known as FER2013. This dataset has a relatively small amount of data for the disgust emotion class, which makes it harder for the model to learn and make accurate predictions in such cases. The shortage of data for the disgust category means that the model does not have enough examples to reliably distinguish it from other emotions. Consequently, the model may have trouble correctly recognizing disgust in real-time.

4.6 Summary

In summary, data augmentation techniques are important in this project to solve the issue of model overfitting which shows how data augmentation techniques, including shear, rotation, width shift, height shift and horizontal flip, help improve the robustness and generalization of the model by reducing the gap between training and validation accuracy. Analysis reveals that data augmentation leads to a more balanced and stable training process. This chapter continues to investigate hyperparameter tuning, focusing on learning rate and batch size. Experiments demonstrate the effect of varying these hyperparameters on the performance of custom CNN models. The optimal combination of hyperparameters was determined by evaluating the accuracy of the model on the FER2013 dataset. The findings show that a batch size of 16 and a learning rate of 0.001 results in the highest model accuracy at 65.27%. This experiment also revealed that different hyperparameter configurations can have a significant impact on model performance, emphasizing the importance of fine-tuning these parameters to achieve optimal results. Further analysis involves evaluating the proposed model. Model accuracy and loss are tracked throughout the training process, demonstrating the model's ability to learn and generalize.

An important component of the performance analysis is the confusion matrix, which evaluates the model's ability to accurately classify emotions. It provides insight into true positives, misclassifications, false negatives and false positives across different emotion classes. The analysis revealed that the model performed well in recognizing happiness but struggled with emotions such as disgust and fear. It also highlights misclassifications between anger and surprise, as well as between sadness and neutrality, indicating the need for further refinement. To benchmark the proposed custom CNN model, a comparison is made with three trained models namely VGG16,

AlexNet and MobileNet. The pretrained model is trained using the same hyperparameters as the custom model. The results show that our proposed model outperforms all three pre-trained models in terms of overall accuracy and balanced accuracy, recall and F1-scores across various emotion classes. VGG16 achieves lower accuracy due to inappropriate hyperparameters, while AlexNet and MobileNet struggle with emotion recognition, with the lowest accuracy among the models. In real time monitoring, correctly predicted the seven different emotions in real time monitoring. However, it is difficult to predict the emotions of disgust because the small amount of FER2013 dataset for the disgust classes makes it difficult for the model to predict it.



اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CHAPTER 5

CONCLUSION AND FUTURE WORKS



In this chapter, will conclude our proposed model was successfully designed and correctly predict seven different emotions by implementing CNN model in real time monitoring. It highlights the hyperparameter tuning, data augmentation to achieve the highest accuracy of 65.27% compared to pretrained models. This chapter also discussed the future work by suggesting that applying feature extraction techniques, addressing class imbalance in the FER2013 dataset, applying illumination correction, and implementing 468 facial landmarks to enhance face detection and recognition accuracy in the future.

5.1 Conclusion

In conclusion, our proposed model for facial expression recognition was successfully designed using the CNN architecture as well as setting the parameters such as convolution layer, pooling layer, activation function, fully connected layer and so on that have been investigated to fulfill the first objective of this project. Next, this project addresses the overfitting problem by demonstrating the importance of data augmentation techniques which help to improve the robustness and generalization of the model by reducing the gap between training and validation accuracy. With that, it can improve model accuracy.

These findings highlight the importance of using CNN as a powerful architecture for recognizing the facial expression. In order to achieve the best accuracy, therefore this project also tuning the Hyperparameter such as learning rate, batch size, and the number of epochs, allowed us to identify optimal configurations for our model. Furthermore, this project also trains the pretrained models such as VGG16, AlexNet and MobileNet by using the same hyperparameters as the proposed model which is intended to be compared with the proposed model. As the result, our proposed model achieved the highest accuracy with 65.27% compared to the pretrained models where VGG16 with 42.11%, AlexNet with 24.71% and MobileNet with 28.35% on the FER2013 test dataset.

By achieving an accuracy of 65.27% for the proposed model, this project successfully achieved the second objective where our proposed model was successfully deployed into a webcam with correctly predicted the seven different emotions such as angry, disgust, fear, happy, sad, neutral and surprise in real time monitoring. However, it is difficult to predict the emotions of disgust because the

small amount of FER2013 dataset for the disgust classes makes it difficult for the model to predict the classes. Furthermore, this project also succeeded in achieving the third objective where at the end of this project, this project also evaluates model performance in terms of model accuracy, precision, recall, and F1-score for each model that has been trained. This is because to ensure that the model can correctly predict the seven emotions. Overall, this project addressing the problem which stated in the problem statement and achieved the overall outcomes where our proposed model can be designed, evaluated model performances and integrate into a webcam for the purpose of real time monitoring.

5.2 Future Work

In future work, there exist several approaches that can be investigated to improve the performance of this project. These approaches are intended to address specific limitations or areas in need of enhancement that have been identified during the present project.

5.2.1 Applying feature extraction techniques before training models

In future work, we will implement feature extraction techniques to capture relevant information from input images before model training. This process aims to distinguish important image features that are important for accurate facial expression classification. Typically, feature extraction is carried out by a CNN, a deep learning model commonly used for image classification tasks. A CNN consists of multiple layers that perform various operations on the input image, extracting features such as edges, vertices and textures. Before CNN training, the input image will undergo preprocessing steps such as face detection, histogram equalization, image

normalization, and face alignment [27]. Various techniques for feature extraction, including multilevel feature extraction and fusion, will be explored to obtain multilevel features from the entire output of the extraction network.

5.2.2 Address class imbalance in the FER2013 dataset

In future work concerning facial expression recognition through deep learning, we can tackle the issue of class imbalance within the FER2013 dataset by employing techniques like upsampling, downsampling, or implementing loss weighting to give more significance to underrepresented classes [28]. These strategies aim to balance the sample distribution across different classes, preventing the model from exhibiting bias towards the majority class. By effectively addressing the class imbalance in the FER2013 dataset, deep learning models can enhance their ability to recognize facial expressions accurately and efficiently. This improvement can lead to superior performance in real-world applications, including emotion recognition in human-computer interaction, security systems, and healthcare. The selection of the appropriate technique should be based on the specific project requirements, considering factors such as overfitting risks with upsampling, potential information loss with downsampling, and computational demands with loss weighting [28]. Thus, the choice should align with the project's unique needs and constraints.

5.2.3 Applying illumination correction techniques during pre-processes images

In future work, we can explore and apply illumination correction techniques to enhance the accuracy and robustness of facial expression recognition systems when dealing with varying lighting conditions [29]. These methods encompass advanced approaches within deep learning, such as Generative Adversarial Networks (GANs) or CNNs, which are specifically designed to address illumination variations in facial images. By delving into and implementing these illumination correction techniques, deep learning models can refine their ability to precisely and efficiently identify facial expressions, making them better suited for real-world applications.

5.2.4 Implement 468 facial landmarks.

In future work, the utilization of facial landmarks can be considered to enhance the precision of face detection by accurately pointing and representing crucial facial regions during real-time monitoring. These facial landmarks denote specific points on the face, facilitating the identification and tracking of features like the eyes, nose, mouth, and eyebrows. The incorporation of 468 facial landmarks can substantially improve the deep learning models' ability to recognize facial expressions with greater accuracy and efficiency. Facial landmark detection plays a pivotal role in facial expression recognition, enabling the precise localization and tracking of facial features over time. Various techniques for facial landmark detection, including shape regression, template matching, and deep learning-based methods, can be explored [30].

REFERENCES

- [1] J. Guo, "Deep learning approach to text analysis for human emotion detection from big data," *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 113–126, Jan. 2022, doi: 10.1515/jisys-2022-0001.
- [2] J. R. Lee, L. Wang, and A. Wong, "EmotionNet Nano: An Efficient Deep Convolutional Neural Network Design for Real-Time Facial Expression Recognition," *Front Artif Intell*, vol. 3, Jan. 2021, doi: 10.3389/frai.2020.609673.
- [3] D. Kalita, "Designing of facial emotion recognition system based on machine learning," *8th International Conference on Reliability*, 2020.
- [4] R. and S. S. Appasaheb Borgalli, "Deep Learning Framework for facial emotion recognition using CNN architectures," *International Conference on Electronics and Renewable Systems (ICEARS)*, 2022.
- [5] J. Sultana and M. Naznin, "Breaking the Barrier with a Multi-Domain SER," in *Proceedings - 2022 IEEE 46th Annual Computers, Software, and Applications*

- Conference, COMPSAC 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 456–461. doi: 10.1109/COMPSAC54236.2022.00088.
- [6] T. M. et al, Wani, “Speech emotion recognition using convolution neural networks and deep stride convolutional neural networks,” *6th International Conference on Wireless and Telematics (ICWT)*, 2020.
- [7] Z. et al, Shen, “Emotion recognition based on multi-view body gestures,” *IEEE International Conference on Image Processing (ICIP)*, 2019.
- [8] S. D. A. K. and R. J. S. Saha, “A study on emotion recognition from body gestures using Kinect sensor,” *International Conference on Communication and Signal Processing*, pp. 056-060, 2014.
- [9] K. et al, Mohan, “Facial expression recognition using local gravitational force descriptor-based deep convolution neural networks,” *IEEE Transactions on Instrumentation and Measurement*, 70, pp. 1–12, 2021.
- [10] S. and D. W. Li, “Deep facial expression recognition: A survey,” *IEEE Transactions on Affective Computing*, 13(3), pp. 1195–1215, 2022.
- [11] N. M. Abdullah and A. F. AL-Allaf, “Facial Expression Recognition (FER) of Autism Children using Deep Neural Networks,” in *4th International Iraqi Conference on Engineering Technology and Their Applications, IICETA 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 111–116. doi: 10.1109/IICETA51758.2021.9717550.
- [12] I. Jammoussi, M. Ben Nasr, and M. Chtourou, “Facial Expressions Recognition through Convolutional Neural Network and Extreme Learning Machine,” in

Proceedings of the 17th International Multi-Conference on Systems, Signals and Devices, SSD 2020, Institute of Electrical and Electronics Engineers Inc., Jul. 2020, pp. 162–166. doi: 10.1109/SSD49366.2020.9364189.

- [13] D. E. P. L. C. A. C. M. M. B. H. W. C. Y. T. D. T. and D.-H. & others L. I. J. Goodfellow, “Challenges in representation learning: A report on three machine learning contests,” *International Conference on Neural Information Processing*, 2015.
- [14] S. , & N. F. Singh, “Facial Expression Recognition with Convolutional Neural Networks,” *10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020.
- [15] M. K. Christopher Pramerdorfer, “Facial Expression Recognition using Convolutional Neural Networks: State of the Art,” *IEEE Trans Affect Comput*, 2016.
- [16] D. E. P. L. C. A. C. M. M. B. H. W. C. Y. T. D. T. D.-H. L. et al I. J. Goodfellow, “Challenges in representation learning: A report on three machine learning contests,” *International Conference on Neural Information Processing*. Springer, pp. 117–124, 2013.
- [17] V. and R. R. Verma, “Recognition of facial expressions using a deep neural network,” *8th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2021.
- [18] Batta Mahesh, “Machine Learning Algorithms– A Review,” *International Journal of Science and Research (IJSR)*, vol 9, ISSN: 2319-7064, 2020.

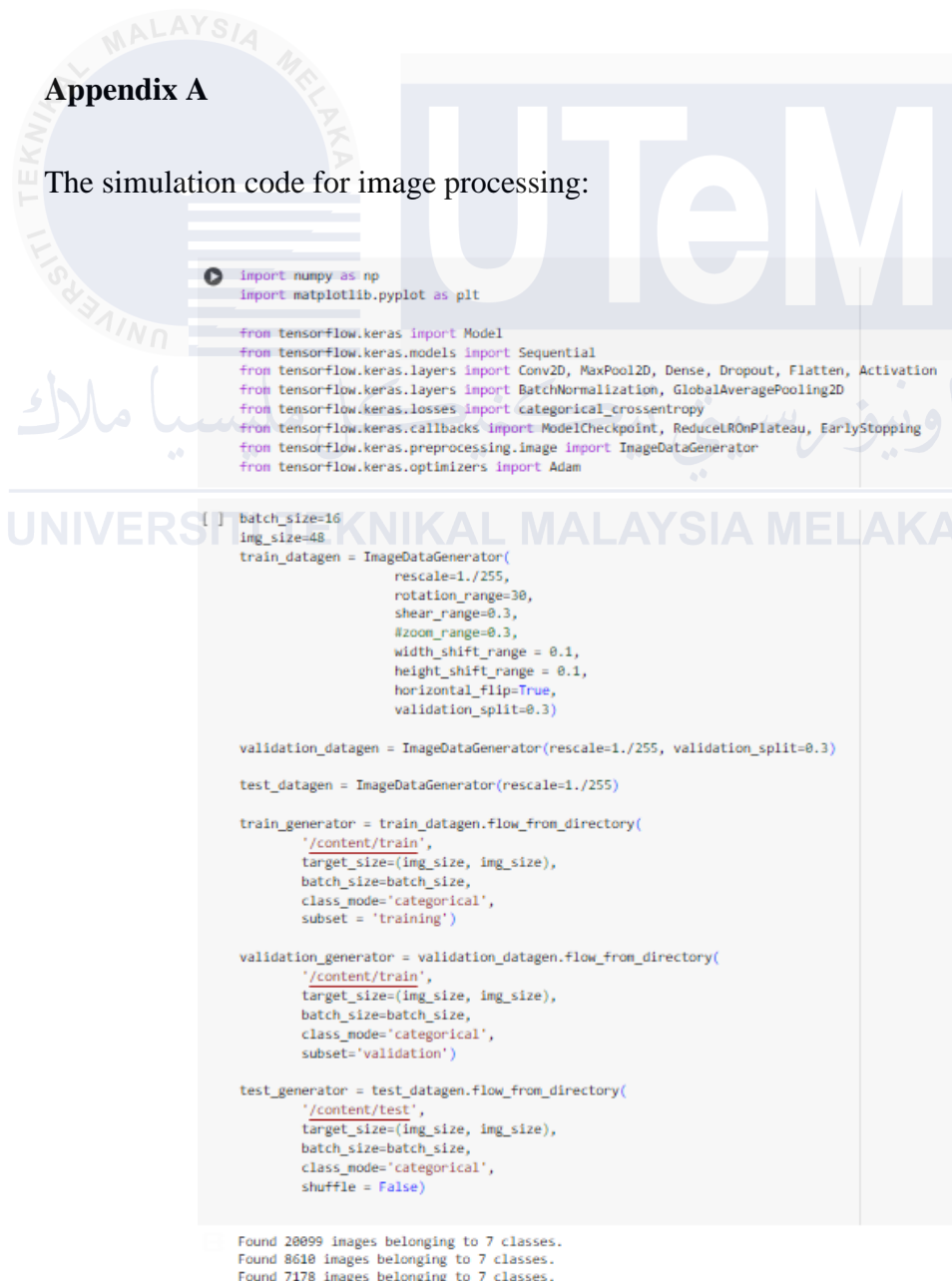
- [19] A. , & S. V Singh, “Comparative study of facial expression recognition using SVM and KNN,” *International Journal of Computer Applications*, 168(1), 10-13, 2017.
- [20] M. et al Fatchan, “Support Vector Machine and neural network algorithm approach to classifying facial expression recognition,” *Fifth International Conference on Informatics and Computing (ICIC)*, 2020.
- [21] J. and P. U. S. Mathur, “Facial expression recognition using wavelet based support Vector Machine,” *Recent Developments in Control, Automation & Power Engineering (RDCAPE)*, 2017.
- [22] A. , B. K. and D. S. Dapogny, “Dynamic pose-robust facial expression recognition by multi-view pairwise conditional random forests,” *IEEE Transactions on Affective Computing*, pp. 167–181, 10(2), 2019.
- [23] A. , A. L. and D. A. Fathallah, “Facial expression recognition via deep learning,” *IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 2017.
- [24] M. et al. Karnati, “Understanding deep learning techniques for recognition of human emotions using facial expressions: A comprehensive survey,” *IEEE Transactions on Instrumentation and Measurement*, 72, pp. 1–31, 2023.
- [25] S. et al. Harshitha, “Human facial expression recognition using deep learning technique,” *2nd International Conference on Signal Processing and Communication (ICSPC)*, 2019.

- [26] A. , & W. J. Savoiu, "Recognizing Facial Expressions Using Deep learning," 2017.
- [27] M. et al Bie, "Facial expression recognition from a single face image based on Deep Learning and Broad Learning," *Wireless Communications and Mobile Computing*, pp. 1–10, 2022.
- [28] Q. T. and Y. S. Ngo, "Facial expression recognition based on weighted-cluster loss and deep transfer learning using a highly imbalanced dataset," *Sensors*, 20(9), p. 2639, 2020.
- [29] C. M. and A. N. Z. Refat, "Deep learning methods for facial expression recognition," *7th International Conference on Mechatronics Engineering (ICOM)*, 2019.
- [30] Y. and W. Y. Qiu, "Facial expression recognition based on landmarks," *IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2019.

APPENDICES

Appendix A

The simulation code for image processing:



```

import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Dropout, Flatten, Activation
from tensorflow.keras.layers import BatchNormalization, GlobalAveragePooling2D
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

[ ] batch_size=16
img_size=48
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    shear_range=0.3,
    zoom_range=0.3,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    horizontal_flip=True,
    validation_split=0.3)

validation_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.3)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    '/content/train',
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset = 'training')

validation_generator = validation_datagen.flow_from_directory(
    '/content/train',
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')

test_generator = test_datagen.flow_from_directory(
    '/content/test',
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle = False)

Found 28099 images belonging to 7 classes.
Found 8618 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.

```


Appendix B

The simulation code for designing CNN model:

```

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten, Activation
#initialize parameters
num_features = 64

model = Sequential()

#module 1
model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), input_shape=(48,48,3), data_format='channels_last'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Conv2D(2*2*num_features, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

#module 2
model.add(Conv2D(2*num_features, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Conv2D(2*num_features, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

#module 3
model.add(Conv2D(num_features, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Conv2D(num_features, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

#flatten
model.add(Flatten())

#dense 1
model.add(Dense(2*2*2*num_features))
model.add(BatchNormalization())
model.add(Activation('relu'))

#dense 2
model.add(Dense(2*2*num_features))
model.add(BatchNormalization())
model.add(Activation('relu'))

#dense 3
model.add(Dense(2*num_features))
model.add(BatchNormalization())
model.add(Activation('relu'))

#dense 4
model.add(Dense(num_features))
model.add(BatchNormalization())
model.add(Activation('relu'))

#output layer
model.add(Dense(7, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-7),
              metrics=['accuracy'])

```

Appendix C

The simulation code for plotting model accuracy and loss and generating confusion matrix:

```

import matplotlib.pyplot as plt
fig, axes = plt.subplots(1,2, figsize=(18, 6))
# Plot training & validation accuracy values
axes[0].plot(history.history['accuracy'])
axes[0].plot(history.history['val_accuracy'])
axes[0].set_title('Model accuracy')
axes[0].set_ylabel('Accuracy')
axes[0].set_xlabel('Epoch')
axes[0].legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
axes[1].plot(history.history['loss'])
axes[1].plot(history.history['val_loss'])
axes[1].set_title('Model loss')
axes[1].set_ylabel('Loss')
axes[1].set_xlabel('Epoch')
axes[1].legend(['Train', 'Validation'], loc='upper left')
plt.show()

from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

class_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Evaluate the model on the test data
test_predictions = model.predict(test_generator)
y_test = test_generator.classes
# Convert predictions and true labels to one-hot encoded format
test_preds_classes = test_generator.classes
test_true_classes = [np.argmax(probas) for probas in test_predictions]

# Generate confusion matrix
conf_matrix = confusion_matrix(test_true_classes, test_preds_classes)

# Plot confusion matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()

```

Appendix D

The simulation code for generating classification report:

```

from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score

# Assuming you have the actual labels (y_true) and predicted labels (y_pred)
# These labels should be based on the confusion matrix from your problem

# Calculate the confusion matrix
cm = confusion_matrix(test_preds_classes, test_true_classes)

# Calculate overall accuracy
accuracy = accuracy_score(test_preds_classes, test_true_classes)

# Calculate F1-score
f1 = f1_score(test_preds_classes, test_true_classes, average='micro')

# Calculate precision
precision = precision_score(test_preds_classes, test_true_classes, average='micro')

# Print the results
print("Confusion Matrix:")
print(cm)
print("Overall Accuracy: {:.2f}%".format(accuracy * 100))
print("F1-Score: {:.2f}".format(f1))
print("Precision: {:.2f}".format(precision))

#Plot the confusion matrix. Set Normalize = True/False

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix', cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting normalize=True.
    """
    plt.figure(figsize=(7,7))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
        print("Normalized confusion matrix")
    else:
        print("Confusion matrix, without normalization")
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

#Print the Target names
target_names = []
for key in train_generator.class_indices:
    target_names.append(key)
# print(target_names)

#Confusion Matrix
Y_pred = model.predict(test_generator)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
cm = confusion_matrix(test_generator.classes, y_pred)
#plot_confusion_matrix(cm, target_names, title='Confusion Matrix')

#Print Classification Report
print('Classification Report')
print(classification_report(test_generator.classes, y_pred, target_names=target_names))

```

Appendix E

The simulation code for testing the trained model with a sample image:

```
import numpy as np
from keras.models import load_model
from keras.preprocessing import image
import matplotlib.pyplot as plt

# Load the pre-trained CNN model saved in .h5 format
model = load_model('custom_modelv47.h5')

# Load and preprocess the image you want to test
img_path = 'test/6/PrivateTest_14522193.jpg' # Replace with the path to your test image
img = image.load_img(img_path, target_size=(48, 48)) # Resize the image to match the model's input size
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0) # Add a batch dimension
img /= 255.0 # Normalize pixel values to the range [0, 1]

# Make predictions using the pre-trained model
predictions = model.predict(img)

# Define a list of facial expression labels (e.g., happy, sad, angry)
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Get the index of the predicted label with the highest probability
predicted_label_index = np.argmax(predictions)
predicted_label = emotion_labels[predicted_label_index]

# You can also add the actual label (ground truth) if available
actual_label = 'Surprise' # Replace with the actual label if you have it

# Display both labels with modified font sizes
fig, ax = plt.subplots()
ax.imshow(img[0])
ax.axis('off')
ax.set_title("Actual Emotion: " + actual_label, fontsize=14) # Fontsize for actual label
plt.figtext(0.5, 0.06, "Predicted Emotion: " + predicted_label, ha="center", fontsize=14) # Fontsize for predicted label
plt.show()
```

Appendix F

The simulation code for integrating our proposed model using a webcam:

```
In [12]: import numpy as np
import tensorflow as tf
from keras.models import load_model
import cv2
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from time import sleep

In [21]: #input image must 48x48x3
faceCascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_default.xml')
classifier = load_model('custom_modelv47.h5')
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
#emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255), 2)
        roi_color = frame[y:y+h, x:x+w]
        #roi_gray = cv2.resize(roi_color, (224, 224), interpolation=cv2.INTER_AREA)

        roi_gray = gray[y:y+h, x:x+w]
        roi_gray = cv2.resize(roi_color, (48, 48), interpolation=cv2.INTER_AREA)

        if np.sum([roi_gray]) != 0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi, axis=0)

            prediction = classifier.predict(roi)[0]
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0))
            label = emotion_labels[prediction.argmax()]
            label_position = (x, y-10)
            cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        else:
            cv2.putText(frame, 'No Faces', (30, 80), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    cv2.imshow('Emotion Detector', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```