UNIVERSITI TEKNOLOGI MARA

ARABIC HANDWRITTEN LETTER RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

MUHAMMAD ARFAN BIN MOHD DIN @ MOHD NORDIN

BACHELOR OF INFORMATION SYSTEM (HONS.)

INTELLIGENT SYSTEM ENGINEERING

FEBRUARY 2024

UNIVERSITI TEKNOLOGI MARA

Arabic Handwritten Letter Recognition Using Convolutional Neural Network

Muhammad Arfan Bin Mohd Din @ Mohd Nordin

Proposal Report submitted in fulfillment of the requirements for Bachelor of Information System (HONS.)

Intelligent System Engineering

College of Computing, Informatics and Media

FEBRUARY 2024

SUPERVISOR APPROVAL

ARABIC HANDWRITTEN LETTER RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

By

MUHAMMAD ARFAN BIN MOHD DIN @ MOHD NORDIN

2022947297

This thesis was prepared under the supervision of the project supervisor, CT Munnirah Niesha Binti Mohd Shafee. It was submitted to the College of Computing, Informatics and Media and was accepted in partial fulfilment of the requirements for the degree of Bachelor of Information System (Hons.) Intelligent System Engineering.

Approved by

[illegible]

....

CT Munnirah Niesha Binti Mohd Shafee

Project Supervisor

JANUARY 23, 2024

STUDENT DECLARATION

I certify that this thesis and the project to which it refers is the product of my own work and that any idea or quotation from the work of other people, published or otherwise are fully acknowledged in accordance with standard referring practices of the discipline.

Arfan

…………………………………………………

MUHAMMAD ARFAN BIN MOHD DIN @ MOHD NORDIN

2022947297

JANUARY 23, 2024

ACKNOWLEDGEMENT

Alhamdulillah, praises and thanks to Allah because of His Almighty and His utmost blessings, I was able to finish this research within the given time.

First and foremost, my special thanks go to my supervisor, CT Munnirah Niesha Binti Mohd Shafee, for being patient with me and always guiding me from time to time in making this project. I also give utmost thanks to my beloved parents, Mohd Din @ Mohd Nordin Bin AB Rahman and Asiah Binti HJ Mohammad, for their prayers, love and for tolerating and understanding my situation during this quarantine season.

Last but not least, I am extremely grateful to my friends who kept me company and helped by giving me a positive atmosphere during these challenging times. I surely could not have done it without all of the encouragement to keep me going in completing the project.

ABSTRACT

Arabic script has a distinct visual appeal due to its artistic complexity and historical significance. We examine the subtleties of these antiquated characters in this investigation, fusing custom with modern insight to recognise the cultural depth embodied in every line and curve. In response to the inherent difficulty in distinguishing Arabic letters due to their intricate visual similarities, this research endeavours to present an innovative solution. The proposed system not only draws but also inserts images of Arabic letters into a comprehensive framework that facilitates their classification into 28 distinct characters. Addressing the intricacies of Arabic script, this project employs state-of-the-art Convolutional Neural Network (CNN) technology, a powerful tool in image recognition and classification. To ensure optimal performance, the images are subjected to a meticulous preprocessing phase and resized to a standardized 32x32 pixel format, thereby establishing a consistent and manageable dataset. The dataset itself is derived from the well-curated Arabic Handwritten Characters Dataset available on Kaggle, providing a diverse and representative collection for training and evaluation. The model architecture chosen for this project is a CNN with a 7 by 3 split, a configuration designed to capture the nuanced features of Arabic letters. Training the model over 50 epochs result in a commendable accuracy of 96.37%, showcasing the robustness and effectiveness of the proposed approach. The model's ability to accurately classify images into their respective Arabic character classes highlights its potential utility in addressing the challenge of distinguishing visually similar characters. Beyond its technical achievements, this prototype holds promise as an educational tool. By aiding users in learning and discerning the subtle differences between Arabic characters that share visual characteristics, it contributes to the broader goal of promoting language understanding and proficiency. This research marks a significant stride in the realm of Arabic script recognition, offering a valuable resource for both learners and enthusiasts seeking to navigate the intricacies of this ancient and visually complex writing system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MNIST	Modified National Institute of Standards and Technology database.
CNN	Convolutional Neural Network
MSA	Modern Standard Arabic
AI	Artificial Intelligence
NAE	National Archives of Egypt
NAs	National Archives
IoT	Internet of Things
ML	Machine Learning
4IR	Fourth Industrial Revolution
ROC	Receiver Operating Characteristic
ANN	Artificial Neural Network
SVM	Support Vector Machine.
HAC	Hierarchical Agglomerative Clustering
PHoG	Pyramid Histogram of Oriented Gradients
DCNNs	Deep Convolution Neural Networks

OCR	Optical Character Recognition.
HMM	Hidden Markov Model
ReLu	Rectified Linear Activation Function
AHCD	Arabic Handwritten Characters Dataset
BLSTM	Bidirectional Long Short-Term Memory.
AHDB	Arabic Handwriting Data Base
HOG	Histogram-Oriented Gradient
DRB	Deep Rule Based
HCCR	Handwritten Chinese Character Recognition
SGD	Stochastic Gradient Descent
RNN	Recurrent Neural Network
LSTM	Long short-term memory
KNN	K-nearest neighbors
UI	User Interface
IEEE	Institute of Electrical and Electronics Engineers
HTML	Hypertext Markup Language

CHAPTER 1

INTRODUCTION

This chapter provides an overview and encouragement for creating an Arabic Handwritten Letter Recognition system for students of the language. This chapter covers the project's background, problem statement, objectives, scope, and significance. To fully comprehend the project as a whole, you must understand this chapter. Arabic letter recognition systems have gained increasing importance in the field of Arabic language processing, owing to the growing demand for accurate and efficient methods of analyzing Arabic text.

1.1 Background of Study

The use of handwriting in daily life has been proven to increase memory retention, boost performance, and improve comprehension, according to psychological science. Writing by hand enables us to rephrase ideas in our own words. The priceless handwritten manuscripts left by our ancestors need to be digitally preserved. Due to the similarity of the characters and the wide range of writing styles, Arabic character recognition in isolation is much more difficult than Latin character recognition. (Ghanim et al., 2020).

Because of its unique challenges compared to other languages, Arabic handwriting is a topic that many researchers' studies. These difficulties are caused by a variety of elements, including the cursive writing style, characters that overlap, have ascending and descending letters, word morphology, diacritical marks, and dots that change the meaning of words. (Alzrrog et al., 2022).

Arabic is an only language that use a distinct set of 28 letters that are written from right to left. One of the most widely used Semitic languages is Arabic. Each letter has a unique shape that can change depending on where it appears in a word. (Ullah & Jamjoom, 2022). Ancient civilizations made important contributions and left behind priceless handwritten manuscripts that must be digitally preserved. (Ghanim et al., 2020).

Arabic is a script used in about 27 different languages. With more than 420 million native speakers, Arabic is the sixth most spoken language in the world. (Ghanim et al., 2020) . In addition, numerous languages, including Urdu, Pashto, Persian, and Jawi, uses Arabic letters, words, and sentence constructions. (Ullah & Jamjoom, 2022).

Arabic calligraphy is a central part of the Arabic cultural heritage and has been used since its introduction, with the first writing of the Holy Quran, up until the present. It is famous for the artistic and complicated ways that letters and words interweave and intertwine to express textual statements.

Learning Arabic letters is a necessary step towards mastering the language. As a beginner, it is critical to learn the names of the letters as well as their various forms. Learners can begin to recognise and write Arabic words with practise, allowing them to read and communicate in the language. So, by understanding every distinctive feature they can learn the words properly.

1.2 Problem Statement

Arabic letter handwritten recognition system is multifaceted and encompasses several challenges. Firstly, the Arabic alphabet contains letters that are remarkably similar in appearance, making it difficult to differentiate between them solely based on their visual features. These visually similar letters can only be distinguished by the number and position of dots added to the letter forms. Consequently, accurately recognizing and distinguishing these visually similar letters becomes a complex task, requiring advanced recognition algorithms. (Ullah & Jamjoom, 2022)

The diversity in human handwriting styles poses a significant obstacle in accurately recognizing Arabic letters. Different individuals exhibit unique handwriting styles, resulting in substantial variations in the shape and form of the letters. Furthermore, Arabic script allows for flexible letter shapes corresponding to their spatial localities within words. This variability challenges the development of a robust recognition system that can effectively differentiate between different handwriting styles and adapt to the changing letter shapes based on their positional context. (Ghanim et al., 2020)

Beginners or individuals with limited knowledge of Arabic script may face difficulties in distinguishing the different strokes and stroke lengths of certain letters. Arabic letters can possess varying stroke lengths, which can significantly impact their visual representation. Lack of awareness or understanding of these stroke variations can lead to inaccuracies in letter recognition, especially for those learning or encountering Arabic script for the first time.

Addressing these challenges is crucial for the development of an effective Arabic letter handwritten recognition system. Overcoming the complexities of visually similar letters, diverse handwriting styles, and stroke length differentiations requires the exploration and implementation of advanced algorithms, feature extraction techniques, and machine learning approaches.

1.3 Research Questions

The main questions of this project are:

- What are the outcomes to be achieved within the letter recognition?
- Which machine learning algorithm is most suitable for accurately recognizing and classifying Arabic letters in handwritten text?
- How does the developed model for Arabic letter handwriting recognition perform in terms of accuracy, precision, and recall?

1.4 Research Objectives

The main objectives of this project are:

- To identify the features and handwriting styles of every Arabic letters.
- To design a machine learning model through CNN for the Arabic Letter Handwritten Recognition system.
- To develop Arabic letter handwriting recognition prototype that can accurately recognize and classify Arabic letters.

1.5 Research Scope

The scope of the project entails developing an Arabic handwritten letter recognition system that focuses on recognizing and classifying the 28 Arabic letters. The system will be trained and tested on a dataset comprising 16,800 Arabic letters, written by 60 participants that is obtained from the Arabic MNIST dataset and 47,434 characters written by 591 participants from the handwritten Hijja dataset. The development of the system will be implemented using Python, utilizing the Convolutional Neural Network (CNN) algorithm.

The primary aim of the system is to cater to individuals who are learning Arabic letters, particularly targeting young children and individuals who are newly converted to the Arabic script. By focusing on this demographic, the system aims to provide a user-friendly interface and accurate recognition capabilities to aid in the learning and recognition of Arabic letters.

The system's scope encompasses the accurate recognition and classification of all 28 Arabic letters, considering the unique shapes and forms of each letter. It will consider the variations in letter connectivity, cursive handwriting styles, and the presence of diacritical marks. The system will be designed to handle the challenges posed by visually similar letters, diverse handwriting styles, and changes in letter shapes corresponding to different spatial localities within words.

The project scope includes the development and optimization of the CNN-based recognition system to achieve high accuracy in recognizing Arabic handwritten letters. Additionally, the system's performance will be evaluated using various metrics to ensure its effectiveness. The evaluation will involve testing the system on real-world Arabic handwritten samples to assess its robustness and reliability.

1.6 Research Significance

The significance of the Arabic handwritten letter recognition system lies in its ability to assist individuals in accurately recognizing and differentiating various types of Arabic letters within reading materials. By providing a reliable and efficient system, the project aims to prevent misreading or misinterpretation of Arabic letters, particularly due to their intricate shapes and strokes. This has significant implications for a wide range of users, including students, researchers, and individuals who regularly engage with Arabic texts.

One of the key beneficiaries of the system is those who are starting to learn the Al-Quran, the holy book of Islam written in Arabic. The accurate recognition of Arabic letters can greatly facilitate their understanding and proficiency in reading the Quranic text, ensuring that they comprehend the intended meanings and teachings without confusion or errors. This can have a profound impact on their spiritual journey and deepening their connection with their faith.

Furthermore, the digitization and preservation of Arabic texts are vital for the cultural heritage of Arabic-speaking communities. By enabling the digital preservation and accessibility of Arabic texts through accurate letter recognition, the system contributes to the conservation and dissemination of Arabic literature, history, and linguistic traditions. It facilitates the digitization of handwritten manuscripts, historical documents, and literary works, ensuring their long-term preservation and making them accessible to a wider audience.

Additionally, the recognition system has practical implications for everyday applications, such as automated transcription, document analysis, and language processing. It can significantly enhance the efficiency and accuracy of tasks involving

Arabic handwritten text, reducing manual effort, and increasing productivity in various domains.

1.7 Summary

In conclusion, this chapter highlights the background of the study, discussing the importance of handwriting in personal life and the need for digital preservation of handwritten manuscripts. The challenges of Arabic letter recognition, such as similarity between characters and variability in writing styles, are acknowledged. The significance of the project is emphasized, including its potential to aid learners of the Arabic language, preserve cultural heritage, and facilitate digital accessibility of Arabic texts. The problem statement addresses the difficulties in differentiating visually similar letters, diverse handwriting styles, and stroke length variations. The research questions focus on measures of discovery, expected outcomes, and the model's performance across different subsets of Arabic letters. The research objectives include identifying letter features, designing a machine learning-based recognition system, and evaluating its performance. The project scope encompasses the development of a recognition system for the 28 Arabic letters, considering various challenges and targeting learners. The use of Python and CNN algorithm is mentioned. The significance of the project lies in assisting individuals in recognizing Arabic letters, particularly for learning the Quran and preserving cultural heritage. It also has practical applications in transcription, document analysis, and language processing.

CHAPTER 2

LITERATURE REVIEW

This chapter outlines the prior research that has been conducted by other scientists who have contributed to projects like this one. Arabic characters, Arabic manuscripts, and their history are the first subtopics that will be covered in this chapter. Next, various artificial intelligence (AI) methods will be discussed, including convolutional neural networks (CNN), image classification, and machine learning. Discussions of related works using the same techniques in a different domain as well as those using the same techniques in related works within the same domain are covered at the chapter's conclusion.

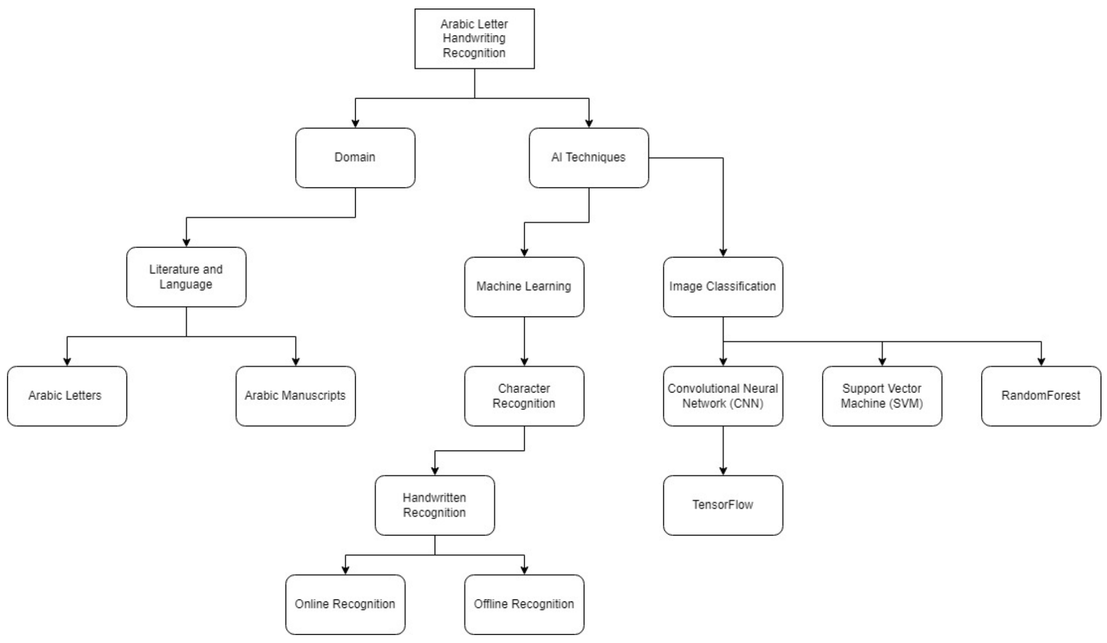


Figure 2.0: Literature Review Mind map

2.1 Arabic Characters

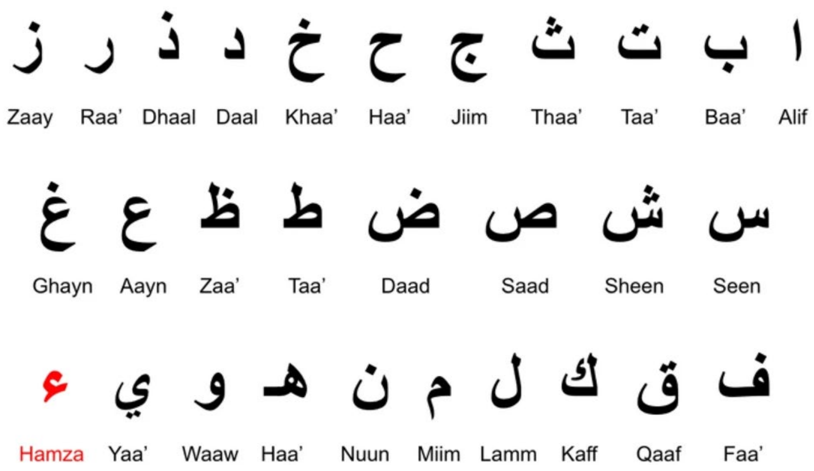


Figure 2.1: Image of Arabic characters

420 million people around the world speak Arabic, an old, ancient language. There are 28 nations that recognise Arabic as a formal language. The official language of communication between Arab nations is Modern Standard Arabic (MSA). Nowadays, every language has both a handwritten and a digital (machine print) writing style. Arabic is known for its right-to-left writing style, cursive, ligatures (the joining of two or more letters), and letters with between two and four shapes. Arabic character writing presents a variety of difficulties in terms of morphology and writing style. Arabic uses 28 characters total. To distinguish between letters with the same loop or shape, such as (ب BAA), (ت TAA), (ث THAA), (ع AIN), (غ KHAIN), a total of 16 letters has one to three dots. There are formal varieties of Arabic called Modern Standard Arabic (MSA) and regionally specific Arabic dialects. Both employ the same characters to create words of various shapes. There are actually multiple PAWs in Arabic; some letters do not connect to one another but nonetheless form a word. For example, the letter (ذ) that makes up the word for for (ذَهَبَ went) does not connect to the word itself but is one of the letters that do so. This word only has one PAW. Some words from letters, like (زَارَ visited), (أَرُ زَ rice), and (وَرَ قَ papers), are not connected, but they still form words with two PAWs in Arabic. Some words, like (زردة picnic) and (الزروق author's last name), have more than two PAWs. The Arabic diacritical marks (vowels) determine the true meaning of words (Alzrrog et al., 2022).

Depending on where it appears in the word at the start, middle, or end of each Arabic letter can take on different forms. Each letter is illustrated in three different ways when it is connected to other letters in the word and written alone in an isolated form. Arabic also has eighteen distinct shapes in addition to its twenty-eight basic letters. The Arabic language is composed of 18 shapes that represent 28 phonetic sounds using diacritical marks. Depending on its placement in a word, an Arabic letter can produce a variety of sounds. Like English's lowercase i and j, Arabic letters must have dots, which are an essential component of each letter. fourteen letters resulting in ط ظ /ع غ /ف ق /د ذ/ ر ز س ش/ ص ض/ (a pairs), six خ ج ح six (double triplets), and ت ث ب or having a common letter shape known as rasm (Yassin et al., 2020).

2.2 Arabic Manuscripts

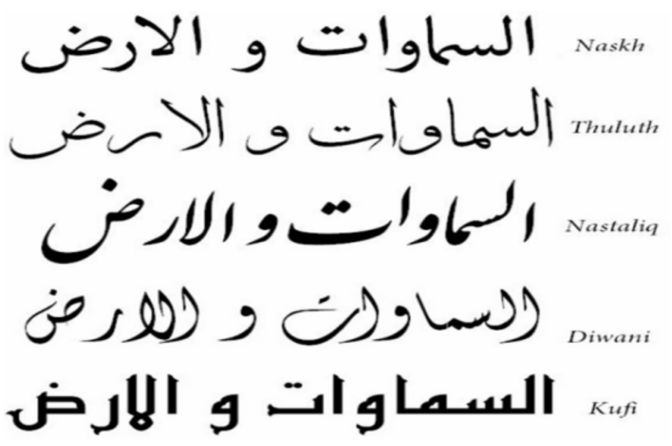


Figure 2.2: Image of Arabic Script Styles

AMs, or Arabic manuscripts, are regarded as historical sources that span several centuries and provide evidence of human literary and cultural advancement. Globally, national archives (NAs), museums, and libraries place a high value on AMs. In spite of this, several nations' NAs continue to handle manuscripts according to the old-fashioned methods. To manually process and handle manuscripts, however, they rely on the skills of specialists. Considering that we are dealing with a legacy spanning thousands of years of human writing, this is a major worry. Prior to engaging in any document learning techniques, deteriorated manuscripts undergo restoration and preservation operations in order to maintain their integrity. As an illustration, the National Archives of Egypt (N.A.E.) has a number of digitization initiatives that focus on manuscripts in order to provide better outcomes. Recognising the AMs' style is the goal of this effort. As a result, we create a model that is trained on labelled data. Then, before deploying the model, we utilise the model to identify a test set that has never been exposed to it in order to assess its performance (Ezz et al., 2019).

Considering that the writing styles used in Arabic manuscripts vary depending on the region, nation, occasion, and materials. They also share several traits with writing instruments for handwritten scripts, such as calligraphy pens and writing orientation (right to left). Consider the Arabic text horizontal projection profiles, which feature a single peak in the centre of the text line, as well as the different forms of the alphabet's letters depending on whether they are at the start, middle, or end of a word. The fact that Arabic manuscripts differ through time and from writer to writer, which increases unpredictability (in the learned traits), adds another level of complexity. There are several publications that break down each style into its two component parts: complete words and isolated letters. For instance, Mosoet Al Khat Al Araby is available in two volumes: Vol. I for Naskh and Vol. II for Reqaa. The two volumes deal with each letter in its respective circumstances, i.e., words and solitary letters (Ezz et al., 2019).

The "Thuluth" style was the source of Naskh style, which developed into its own form in the tenth century. Additionally, Naskh style is more readable and straightforward, particularly in tiny letter sizes. Additionally, its lines are naturally narrow and circular. Additionally, Naskh is now the most widely used literary genre in Arabic literature in general and the holy Quran in particular. The most straightforward handwriting style for casual, everyday use is Reqaa. The 9th century saw the introduction of this flowing, rounded design. Reqaa has gained popularity as the preferred writing style for everyday writing in the eastern Arab world due to its simplicity. Its words have short horizontal strokes, a thick baseline, and a rich ligature pattern(Ezz et al., 2019).

2.3 Character Recognition

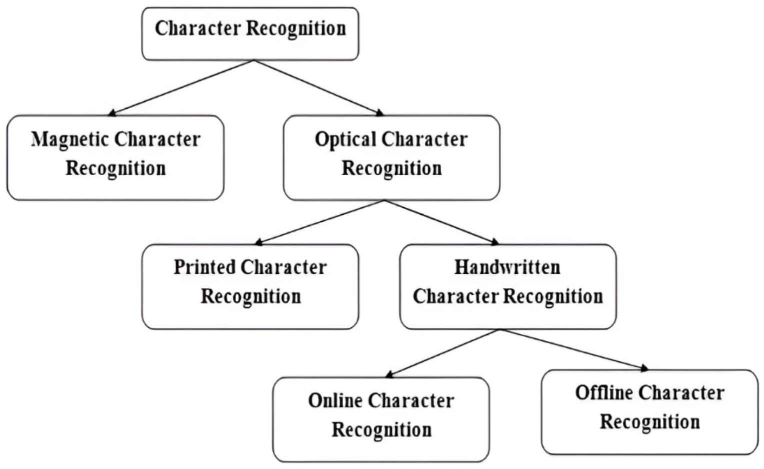

Character Recognition
Magnetic Character Recognition
Optical Character Recognition
Printed Character Recognition
Handwritten Character Recognition
Online Character Recognition
Offline Character Recognition

Figure 2.3: Image of Character Recognition Hierarchy

Source: (Punia, 2023)

Character recognition is a field of active research in image analysis and pattern recognition. It is the process of converting a visual representation of a file to an electronic one. Character recognition comes in two forms: optical character recognition (OCR) and magnetic character recognition. In fact, handwriting recognition is a cutting-edge invention that will prove useful in the twenty-first century. It can act as a starting point for the creation of new necessities (Punia, 2023).

The process of offline character recognition involves translating a file image into a string of characters. Online character recognition is a difficult process. It's a dynamic process. It entails recognizing character statistics while the writer is drafting. It requires the use of an electronic pen and a specific writing page. The handwritten character is identified based on the pen's action (Punia, 2023).

2.3 Handwritten Recognition

The ability of a system to recognise human-written input is known as handwriting recognition. Computers can now interpret and digitise handwritten text thanks to the fascinating field of handwriting recognition. In this procedure, handwritten characters are translated into machine-readable text by examining their shapes, strokes, and patterns. To produce precise and effective results, handwriting recognition systems use cutting-edge algorithms and machine learning techniques. According to (Altwaijry & Al-Turaiki, 2021a), A system's capacity to recognise human-written input is known as handwriting recognition. The handwriting can be found in a variety of places, including paper documents, photographs, touchscreens, and other technology. Handwriting input via scanning is regarded as offline, whereas input via a pen tip is regarded as online.

According to (Altwaijry & Al-Turaiki, 2021a), in computer vision, handwriting recognition is regarded as a difficult problem. The writing styles of various people typically differ from one another. Additionally, a single writer's handwriting may vary slightly from time to time. Feature extraction is crucial because distortions and pattern variability are the main challenges in handwriting recognition. If features are chosen manually, there might not be enough data to correctly predict the character class. A large number of features, however, typically leads to issues because of increased dimensionality. Handwriting recognition involves computer transcription of handwritten documents. Due to the high degree of diversity in human handwriting, the scarcity of large, publicly available datasets, the nature of Arabic script, such as its cursive form, variable letter sizes, and different letter shapes depending on where they are located, Arabic handwriting recognition presents many challenges. (Ullah & Jamjoom, 2022)

The difficulty in identifying and categorising handwritten characters stems from the variations in handwriting styles among writers. Everybody writes differently. Some people have very difficult to read handwriting. Character recognition in some manuscripts may be challenging due to degradation. Font styles and thicknesses may differ. Furthermore, data is frequently displayed as scanned images or photographs, which necessitates taking into account sources of variability such lighting, distortion-causing camera locations, and the colour of the backdrop and foreground.

2.3.1 Online Character Recognition

An essential use of pattern recognition is the identification of handwritten text on digital surfaces by extracting the spatial and temporal information of handwritten patterns and interpreting the handwritten text. This includes the recognition of online handwriting (Singh et al., 2021).

Digital writing support like tablets, smartphones, pens, and writing pads are needed for online recognition because the user's writing is digitalized into the recognition system in real-time as a two-dimensional sequence of points. Online handwriting recognition uses the number of strokes, speed at which each stroke is written, direction that each stroke is written in, and order that each stroke is written to identify the character or word. However, since humans are accustomed to writing with a digital pen rather than a keyboard, it is simpler for them to do so (Youssef & Abd-Alsabour, 2022).

The handwriting process is monitored by the online recognition approach, which immediately inputs data pertaining to the spatial and temporal characteristics of handwriting, including handwriting coordinates and time series obtained from sensors. There are various uses for online recognition systems, including digital learning. One possible technological advancement for online handwriting recognition is deep learning (Li, n.d.).

2.3.2 Offline Character Recognition

In computer vision, offline handwriting recognition is an image-based sequence recognition job. Conventional methods rely on extensive linguistics knowledge, sophisticated feature extraction methods, and lexical segmentation (Tran et al., 2019).

The core of offline recognition is the conversion of text from a picture to codes so that text editors or other computer programmes may process it. Images of handwritten papers are accepted as input by the offline handwritten recognition system, which outputs the characters that are identified from the document image (Joe et al., 2019).

Numerous research avenues remain unexplored in the context of the offline handwriting recognition problem. The algorithms must determine which sections of the documents include handwritten text, maybe segment lines and words, and identify the words by turning the images into character sequences in order to recognise the handwritten text in a document (Juan & Revuelta, 2021).

It is a difficult undertaking to achieve high accuracy in offline handwritten character recognition. The rate of character identification is affected by a number of variables, including background noise, the writers' distinct writing styles, variances in character size, pen width, pen ink, character spacing, skew and slant, and the similarity of some characters' shapes and sizes. Other important elements, such as the writer's removal of the header line or the division of modifiers and touching characters, also play a vital role in the creation of an effective offline handwritten character recognition system (Yadav & Jain, 2019).

2.4 Machine Learning

A major trend in contemporary, highly skilled technological applications in the industrial sector is machine learning. The effects of machine learning techniques can be seen in a variety of fields, including the security, industrial, and medical sectors. (Tumpa & Dey, 2022). A branch of artificial intelligence called machine learning gathers data from various sources and in different formats (Kushwaha & Kumaresan, 2021).

The digital world has a wealth of data in the Fourth Industrial Revolution (4IR or Industry 4.0) era, including Internet of Things (IoT) data, cybersecurity data, mobile data, business data, social media data, health data, etc. Knowledge of artificial intelligence (AI), and in particular, machine learning (ML), is essential to intelligently analyse these data and create the corresponding smart and automated applications. There are numerous types of machine learning algorithms in the field, including supervised, unsupervised, semi-supervised, and reinforcement learning. In addition, deep learning, which belongs to a larger family of machine learning techniques, has the ability to intelligently analyse a lot of data (Sarker, 2021).

Machine learning is now recognised as being crucial to AI. Instead of having to spend time teaching computers everything they need to know, the idea is to give them the ability to learn on their own. Algorithms are used in machine learning applications. These algorithms are powerful in that they can learn from the data and run on their own without any help from a human. Computational statistics are a major factor in machine learning when developing the basic algorithms. By revealing its complex patterns and fundamental algorithms, the data actually instructs the computer (Tumpa & Dey, 2022).

Statistics, data processing, knowledge analytics, and many other multidisciplinary fields could all benefit from machine learning (Kushwaha & Kumaresan, 2021). The most well-known newest technologies in the fourth industrial revolution (4IR or Industry 4.0) are typically referred to as machine learning (ML), which gives systems the capacity to learn and improve from experience automatically without being specifically programmed (Sarker, 2021).

2.5 Image Classification

With the growth of data in various industries, including e-commerce, automotive, healthcare, and gaming, image classification has recently gained popularity among technology developers. Facebook is where this technology is used in the most obvious way. With just a few tagged images, Facebook can now identify your face with up to 98% accuracy, classifying it into your Facebook album. The ability of the technology to classify or recognise images is almost superior to that of the human eye. Deep learning is one of the most popular methods for this technology. Deep learning is a type of artificial intelligence that can behave or think just like a person. In order to make the system itself more effective and quicker during the "training" session, hundreds or even thousands of input data are typically set into the system. It begins by providing some kind of 'training' using all the input data (Abu et al., 2019).

When evaluating image classification tasks, accuracy and reliability are the main factors considered. The confusion matrix and the receiver characteristic operator (ROC) curve are the two primary methods of verification (Liu et al., 2020). The systems that are frequently used for image classification include machine learning. Machine learning still has room for improvement in some areas, though. Therefore, the deep learning system will be involved in image classification (Abu et al., 2019).

Image classification has its own context when it comes to machine vision. This technology has the capacity to identify people, objects, places, actions, and writing in images. Machine vision and artificial intelligence software work together to produce the impressive classification of images. Making sure that all of the images are classified according to their distinct sectors or groups is the primary goal of image classification. Classification is simple for humans, but machines have had significant difficulties with it. As opposed to detecting an object, which should be classified into the appropriate categories, it consists of unidentified patterns. the many uses for image classification technology, including remote sensing, robot navigation, and vehicle navigation. It continues to require difficult work, and it needs to be improved with few resources (Abu et al., 2019).

2.5.1 Convolutional Neural Network (CNN)

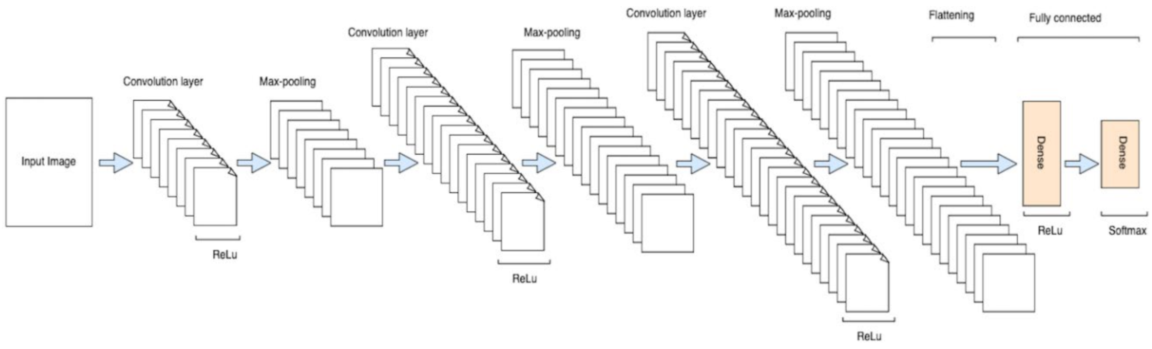


Figure 2.5.1: Image of CNN architecture

Source: (Ullah & Jamjoom, 2022)

Over time, a wider range of disciplines have become applicable to meaningful mastering. In large-scale mastering, convolutional neural networks (CNN) are used for the analysis of visual vision. Convolutional Neural Network Systems should be capable of modelling tasks like object revelation, face confirmation, high level mechanics, video examination, department, plan affirmation, everyday language managing, junk mail disclosure, topic recreation plan, backslide assessment, communicate affirmation, and picture gathering. The accuracy of these fields extends to handwritten digits (Sakthimohan et al., 2023). Deep convolutional neural network (CNN) models are widely used today for image classification tasks (Liu et al., 2020). Deep learning has come to be associated with convolutional neural networks because of their robustness and versatility. For problems involving the spatial arrangement of patterns, such as handwritten character identification, the approach works well (Ramesh et al., 2019).

In computer vision tasks, deep convolutional neural network (CNN) models are frequently used. Deep CNN models for image classification tasks fall into two major categories. One is enhancing deep CNN model performance. The MNIST image dataset, which had ten categories and 70,000 images in the past, has evolved into the ImageNet, which now has thousands of categories and more than 14 million images. It is more challenging to classify those image datasets due to the size of the datasets and the sheer volume of categories. The development of deep learning algorithms has accelerated recently, and deep CNN models have outperformed conventional machine learning techniques in image classification tasks (Liu et al., 2020). Deep Convolutional Neural Networks (CNNs) have validated the human level accuracy of the confirmation. The human visible structure herbal model is the one that executes the CNN strategy (Sakthimohan et al., 2023).

The thinning of deep CNN models is the other branch. Mobile image classification applications are one of the many embedded devices, like mobile phones, that are frequently used in daily life. However, because of their deep structure and high computational demands, deep CNN models are unable to meet the real-time requirements of embedded devices with limited power and storage. To address this issue, a number of research studies on deep CNN model compression and acceleration have been suggested. These techniques seek to reduce the weight of the complex CNN models without sacrificing accuracy (Liu et al., 2020).

CNN follows a similar methodology to ANN. Every layer of an ANN contains some neurons. The weighted measure of the layer's numerous neurons is altered by the dedication of a neuron from the partner layer that offers an uneven admiration. CNN's layer has three angles. This region's neurons are not entirely interconnected (Sakthimohan et al., 2023). Convolutional neural networks (CNN) have recently gained a lot of popularity thanks to their dramatic performance improvement over manually created features. In numerous applications where processing of visual information is necessary, such as image classification, object detection, semantic segmentation, colon cancer classification, depth estimation, and face antispoofing, the CNN has demonstrated very promising performance (Roy et al., 2020).

2.5.1.1 TensorFlow

An open-source library called Tensor Flow is used for large-scale machine learning and numerical computation. In deep learning, tensors are the standard data representation method. Although machine learning is a complicated science, using Tensor Flow to create machine learning models makes the process much simpler (Gupta & Goyal, 2021).

A wide collection of functions and classes offered by TensorFlow enable users to create sophisticated models from the ground up. TensorFlow 2.0 is the most recent version, and it has significant enhancements over its predecessor. TensorFlow's Python interface, model deployment capabilities on web browsers and mobile devices, and ease of use have all contributed to its rapid adoption by the machine learning community. TensorFlow may be used to create non-machine learning jobs that need numerical computation using data flow graphs, even if its main application is machine learning. The contents of the models' distribution packages differ greatly from those of executable binaries (Janardhanan, 2020).

Tensor flow characteristics that distinguish it from the competitors are adopted Keras because its high-level APIs, which facilitate machine learning and offer greater flexibility, eager execution for instant iteration and debugging. Tensor flow kernels that permit eager execution run instantly instead of building graphs that will run at a later time (Gupta & Goyal, 2021).

A machine learning system's representation of what it has learnt from the training set of data is called a model. A graph that describes the structure of how a prediction will be calculated is called a model in TensorFlow. During the training phase, the TensorFlow graph's weights and biases are established. All of the variables and the network graph should be saved to a file for later use once the training is over. TensorFlow saves and retrieves model artefacts in a format unique to it. Higher-level systems and tools can create, use, and modify TensorFlow models since the model artefacts are stored in a language-neutral, recoverable serialisation format (Janardhanan, 2020).

2.5.2 Support Vector Machine (SVM)

SVM, or supervised learning models, are used in artificial intelligence to analyse data, spot patterns, classify data, and perform regression and classification. They are often linked to learning algorithms. Because of its accuracy, robustness, and impact on even tiny datasets, SVM has its origins in statistics and has become popular (Gontumukkala et al., 2021). In order to solve classification problems, Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) have been shown to produce cutting-edge outcomes. SVM serves as an output predictor in the proposed method while a pre-trained CNN model serves as an automated feature extractor (Juliana et al., 2022).

Machine learning's most effective classifier is SVM. A surprising degree of precision may be assembled with a minimum of processing using the rapid speed and supervised learning mechanism known as the support vector machine. SVM determines which hyperplane fits the characteristics the best in order to separate them. In this case, the number of features from the CNN extracted data that were fitted into the SVM model matched the number of classes (Biswas & Islam, 2021).

Support Vector Machine (SVM) is one of the most well-known machine learning algorithms in supervised learning. It can be used for issues involving both regression and classification. SVM looks for a hyperplane that can be utilised to distinguish across classes by classifying data points in a distinct way (Dutta et al., 2021). SVM is an effective method for organising high dimensional data. By enhancing the arrangement edge, SVM learns the optimal hyper plane that separates training models into different classes, in contrast to the nearest neighbour classifier. By using a technique called the part stunt, which extends the info information to a higher layered include space where a straight isolating hyperplane may be established, it is also applicable to informative collections with nonlinear choice surfaces (Madireddy, 2018).

To assist in forming the hyperplane, SVM selects extreme vectors. Support Vector Machine is the name given to this technique since these extreme situations are known as support vectors. Mainly, SVMs can be classified as either linear or non-linear. In the SVM process of issue solving, choosing a kernel function is a crucial stage. In contrast to numerous other methods, SVM proves effective in resolving classification challenges(Chychkarov et al., 2021).

Support vector machines provide an efficient and effective way to learn text classifiers from models, as demonstrated by a few developers. High precision could be used to identify documents related to a particular subject. Support Vector Machine (SVM) is a controlled learning approach used to determine the straight isolating hyperplane that increases the edge between the two informative collections and augments the edge, i.e., the ideal isolating hyperplane (OSH) .

2.5.3 RandomForest

A random forest is a collection of unpruned regression or classification trees that use random feature selection in the tree imitation process. They are triggered from bootstrap samples of the training data. By summing up the ensemble's predictions through superiority voting for categorization, the prediction is made. It is more robust against noise and yields the generalisation error rate. Nonetheless, RF might experience the same problems as most classifiers when it comes to learning from a very unbalanced training set. Because its design aims to reduce the overall error rate, it frequently prioritises the majority class's prediction efficiency, which leaves the minority class with subpar accuracy (Shamim et al., 2018).

Breiman created the ensemble learning technique known as RF to address problems with regression and classification. A machine learning technique called ensemble learning combines several models to solve a single issue in order to increase accuracy. Specifically, ensemble classification involves several classifiers working together to produce findings that are more accurate than those of a single classifier. Put differently, combining several classifiers reduces variation and can lead to more dependable outcomes, particularly when dealing with unstable classifiers (Sheykhmousa et al., 2020).

A collection of decision trees is called a random forest. The decision trees serve as the forest's foundational classifiers. Histogram of Oriented Gradient (HOG), greyscale multi-resolution pyramid, and other techniques can be used to extract features. For every tree in the random forest, a random selection of features must be made which is the number of features must be greater than 1 and less than 20. Each forest tree's prediction is taken into consideration while classifying a sample. The class label of the sample is assigned to the class that receives the majority of votes (Dutta et al., 2020).

2.6 Related Works

Character recognition has been the subject of numerous research. (Ghanim et al., 2020)proposed a multi-stage cascading system to serve the field of offline Arabic handwriting recognition. Starting with dividing the database into clusters that are only loosely related to one another, the method employs the Hierarchical Agglomerative Clustering (HAC) technique. It is possible to represent the database as a large search tree model thanks to the relationships between the built-in clusters, and these relationships also make it simpler to match each test image to a cluster. Then, cluster members are ordered using the new ranking formula we have suggested. To determine divergence, the Kullback-Leibler method is used after computing the Pyramid Histogram of Oriented Gradients (PHoG), the first step in this ranking algorithm. Only the highly ranked matching classes are ultimately subjected to the classification process. Six different deep convolution neural networks (DCNNs) are compared to determine how they affect the final recognition rates of the suggested system. The Arabic database of the IFN/ENIT is used for experiments. Only 11% of the entire database is utilised in the proposed clustering and ranking stages for classifying test images. In comparison to more recent existing systems, this results in more reduced computation complexity and improved classification results.

Next, (Sokar et al., 2019)proposed a generic optical character recognition (OCR) system based on deep Siamese convolution neural networks (CNNs) and support vector machines (SVM). Deep CNNs under supervision are highly accurate at classifying objects. To get around the issue of dataset bias, a trained model must be adjusted for a new set of classes using a substantial amount of data. When the available dataset is insufficient, deep neural networks' (DNNs') classification accuracy suffers. Additionally, altering the network architecture and retraining the model are required when using a trained deep neural network to categorise a new class. The suggested system handles all of these constraints. The deep Siamese CNN has been taught to extract distinguishing features. One training session is held with a collection of classes. Then, without retraining or fine-tuning the deep Siamese CNN model, different classes are recognised using the OCR system. For classification, only a small number of samples are required from any target class. The proposed OCR system is tested using test sets that include printed and handwritten letters and numerals in the Arabic, Eastern-Arabic, Hindu-Arabic, and Farsi languages. Without the need for retraining, the proposed system achieves a very promising recognition accuracy that is comparable to the outcomes obtained by CNNs trained for particular target classes and recognition systems. In a one-shot classification task, the system performs about 12% better than the state-of-the-art technique that uses Siamese CNN.

In order to address the issue of Arabic handwritten word recognition, (Amrouch et al., 2018) developed an integrated architecture based on CNN and HMM classifiers. By using the HMM baseline system as a recognizer and a CNN architecture resembling the LeNet-5 as a salient features extractor, the CNN-based HMM model was put into practise. As a result, pertinent characteristics could be extracted without pre-processing or putting a lot of emphasis on the feature extraction procedure. This model's recognition accuracy was tested using the "abc-d" and "abcd-e" scenarios from the IFN/ENIT database, and it achieved 88.95% and 89.23%, respectively. However, the offline Arabic handwritten numbers and characters were not included in the generalisation scope of the model recognition tests, and the recognition accuracy needs to be increased.

By completing research employing the Arabic language in recognition systems and developing a dataset called Hijja, which comprises Arabic letters written by children between the ages of 7 and 12, (Altwaijry & Al-Turaiki, 2021b) investigated this topic. 47,434 characters total from 591 individuals were included in their dataset. They also put up a CNN-based methodology for automated handwriting recognition. Three convolutional layers, three pooling layers, and four fully linked layers made up the model. In order to prevent overfitting, the ReLU activation function and 80% dropout in the fully connected layers were used after each convolutional layer. To train this model, the Hijja and AHCD datasets were consulted.

(Khan et al., 2020) proposed an OCR strategy for recognising isolated offline Pashto characters and offer a sizable database with 4488 samples of handwritten Pashto characters. Convolution Neural Network (CNN) was used to perform the proposed OCR approach, and it outperformed other conventional models with an accuracy rate of 80.7%. On the Hijja and AHCD datasets, the suggested CNN model fared better than the other model, reaching 88% and 97% accuracy, respectively.

(Maalej & Kherallah, 2019) proposed a hybrid CNN-BLSTM model. The CNN is utilised to automatically extract features from raw images. Then, for sequence labelling, a connection is made between a bidirectional long short-term memory (BLSTM) and a temporal classification layer (CTC). According to the experiments performed on the IFN/ENIT Database, the hybrid model's recognition rate is 92.21%.

(Ali & Suresha, 2019) made a decision to use machine learning techniques to recognise Arabic handwritten letters. The accuracy analysis and classifiers' performance were evaluated using an integrated recognition system built on CNN, k-Nearest Neighbour (kNN), and SVM. The AHDB and IFN/ENIT databases were used to test the integrated recognition system, which reported a 98.4% accuracy rate. It is important to note that the datasets used were not those used in our study.

For the purpose of identifying handwritten Arabic letters, (Djaghbellou et al., 2020) has proposed a new framework based on multi-scale histogram-oriented gradient (HOG) features and the deep rule-based classifier (DRB). During feature extraction, the framework incorporates multi-scale HOG, and DRB is then used to categorise the extensive HOG features and predict the class labels. The recognition rate reported by the framework for the AHCD dataset was 74.61%, which was significantly higher than that of other studies.

A deep convolutional neural network built on the foundation of HCCR-GoogLeNet was used by (Xu et al., 2018)to conduct research on the recognition of Mandarin characters from screen-rendered images. A synthetic character engine is used to create and validate a dataset with a total of 52 and 3755 characters, respectively, for Chinese and English characters. The study's findings demonstrate that the proposed 16 method, which can achieve the best results with an average accuracy of 94.83% and a startling difference from the other methods, is preferable to using HCCR-AlexNet and HCCR-GoogLeNet. HCCR-AlexNet has an average accuracy of 85.74% and 91.88% for HCCR-GoogLeNet.

For the purpose of recognising Chinese characters written in cursive, Hong et al (2019) presents a convolutional neural network. They used a dataset of authentic, antique books' cursive characters from within the company to train and validate the model. The collection includes 38878 scans of common cursive characters that are used in everyday life. They employed DenseNet-based recognition models from the DenseNet201-Adam, DenseNet201-Aug, DenseNet201-Gray-SGD, and DenseNet201-Gray-SGD-Aug families. The DenseNet201-Gray-SGD-Aug model, which converts image data to grayscale and uses augmentation method, has been shown to have the greatest accuracy of 91.60% among all the models.

2.7 Summary

In conclusion, this chapter describes studies carried out by researchers to gain a better knowledge of artificial intelligence techniques, methodologies, and how a system for recognising Arabic characters may be created based on those qualities. After examining the related works, it is clear that character recognition has attracted the attention of several researchers, with the rate of recognition accuracy steadily increasing. Numerous techniques have been used to achieve the maximum accuracy. In conclusion, a CNN model will be created for more accuracy, and the approach has been chosen to be used in building the project since it reduces training time.

CHAPTER 3

RESEARCH METHODOLOGY

The study technique and the procedures used to carry out the project's goals are covered in this chapter. The study methodology discussed and described how and how the project was completed in accordance with the project's objectives. The stages of the research structure are preliminary studies, knowledge gathering and acquisition, data collection, data preprocessing, model development, model design, model evaluation, system design, system development, system testing and evaluation, and the last stage is the documentation of the final report.

3.1 Research Outline

This research involves four phases with their activities. Phase 1 is the process for the first objective where it includes the preliminary studies, knowledge gathering and acquisition, data collection, data preprocessing. Next, phase 2 is the process for the second objective where phase 2 describes the system design, knowledge representation, system development. The last phase, which is phase 3 is where the system is tested, and evaluation followed by the documentation of the final year report.

3.2 Research Design

The research design is the framework for all of the research approaches and techniques a researcher chooses to use to conduct a study. The design makes it possible for researchers to concentrate on creating research methods relevant to the subject and sets up our studies for success. For each of the many elements, it has a specific objective, phase, activities, and deliverables that should be worked on.

Research Objectives	Phases	Activities	Deliverables
To identify the features and handwriting styles of every Arabic letter	Preliminary study	Read and understand articles regarding the methods that are used to implement for Arabic letter recognition	• Background study • Problem Statement • Research questions • Project Scope • Significance
	Knowledge gathering and acquisition	Gaining knowledge and accumulate the results gain from articles on the preferred methods	• Literature Review
	Data Collection	Search for Arabic letters dataset and collect data of images of different handwriting styles from different people through Kaggle	• Data of Arabic letters and images of different handwriting styles
	Data Preprocessing	Transforming images of different handwritten Arabic letters by classification, splitting, converting class vector, reshaping, resizing, and data augmentation.	• Processed handwritten Arabic letters
To design a machine learning model through CNN for the Arabic Letter Handwritten Recognition system	Model Development	• Choosing CNN as the machine learning model for Arabic letter recognition. • Splitting the dataset into training and validation sets. Training selected model using the training data. Use the validation data	• Documentation of the hyperparameter tuning process, listing the selected hyperparameters and the performance impact on the model.

		to tune hyperparameters and prevent overfitting. • Fine-tune the model by adjusting hyperparameters such as learning rate, batch size, and epoch to achieve better performance.	
	Model Design	• Design a Convolutional Neural Network for image processing. • Design the model with different types of parameters such as layers and optimizers. • Testing the model with different split ratios.	• Model of the Arabic letter Recognition system
	Model Evaluation	• Evaluate the trained model on a separate test dataset to assess its performance. • The models are compared to see the best accuracy among them.	• A detailed evaluation report showcasing the model's performance metrics on the test dataset, including accuracy, precision, training and validation graphs, recall, F1-score, and confusion matrix.
To develop Arabic letter handwriting	System Development	• Develop a system for the	• A simple Prototype of the Arabic

recognition prototype that can accurately recognize and classify Arabic letters.		prototype through python	Handwritten letter recognition system
	System Design	• Designing an interface to obtain written letters that are drawn live and receive images from user input.	• UI Design and system interface design through Python • System Architecture
	System testing and evaluation	• Testing the prototype • Test the models and compare the accuracy of the methods. • Test the functionality of the project by prototyping the actual dashboard of interface.	• Testing and result evaluation
	Documentation	Writing the report	• Final year report

Table 3.1: Project Phase Timeline

3.3 PHASE 1

Phase 1 focused to identify the features and handwriting styles of every Arabic letter. Phase 1's goal is to recognise Arabic letters characteristics. Phase 1 activities include preliminary study, knowledge acquisition, data collecting, and pre-processing of the data.

3.3.1 Preliminary Study

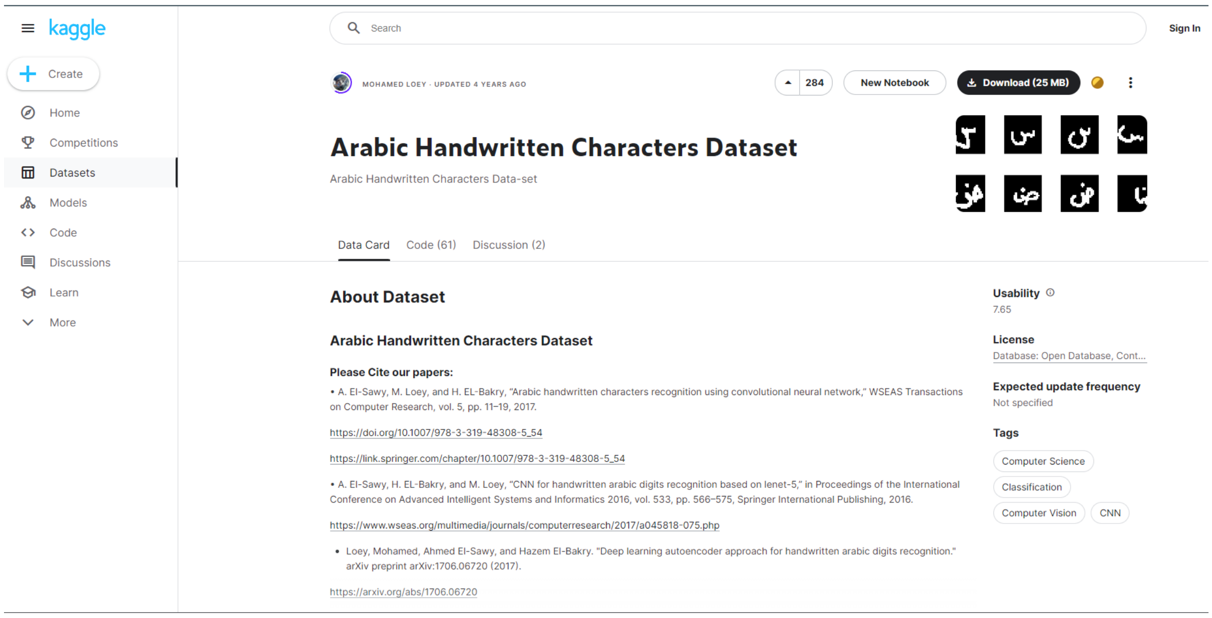
Preliminary study has been done to help gain a better knowledge of the materials that are actually present and the opinions that are being expressed that is also known as preliminary research. This kind of research has enlarged or helped me narrow down topics regarding Arabic Handwritten Letter Recognition and strengthening it. Using the data acquired, it has provided me with better keywords. Preliminary research, sometimes referred to as the first step in the research process, evaluates the entire project by taking into account its context, defining the issue, and observing and comprehending how Arabic Letter Handwritten Recognition is related to other topics such as Optical Character Recognition, digit recognition, and handwriting recognition which gave access to the data that is needed to effectively plan and carry out the research. Additionally, it provides background information about the key subjects that is essential for assessing the research's progress. It also helps in decision-making and background research for the subtopics in letter recognition. The preliminary study has been helpful in determining the project's background study, problem statement, scope, research questions, and significance. In order to stay updated of the most recent developments in the field of Arabic letter recognition, additional online study was done.

3.3.2 Knowledge Gathering and Acquisition

The process of extracting, categorising, and analysing data from a single source typically a human expert is known as knowledge acquisition. However, for this project, knowledge acquisition was done using Google Scholar, IEEE Xplore, Scopus, Web of Science, and other open-source articles. The issue was identified, together with its goal, range, and importance in connection to the associated topic, which is letter recognition, by applying the literature analysis technique. It refers to the process of gathering all the facts and information. Earlier studies conducted by other analysts were examined in the evaluation of the literature, which has helped to further understand the requirements in related subtopics of Arabic Handwritten Letter Recognition. Since there are numerous ways to frame through findings and address each issue, this phase enabled the acquisition of new knowledge. The methods and approaches, or method algorithms, that has been chosen and used on this project will be supported by this phase based on their efficiency and accuracy in the development of the models. The analysis of the project, method, and approaches has also been determined based on how they are described in the previous research that are done related to letter recognition and on how to carry out each of the procedures in developing the model for Arabic Handwritten Letter Recognition. To better understand how to select the best method for letter recognition, research is also conducted using the same technique in various domains. This was accomplished by reviewing related deep learning works on letter recognition. In conclusion, CNN has been extensively utilised for image recognition in order to categorise each image into a unique class. Consequently, CNN is selected to be used in conjunction with SVM in order to increase accuracy for this project. The CNN model is used to categorise Arabic letter images into their respective classes once it has been trained.

3.3.3 Data Collection

In the data collection phase, the data on Arabic letters was searched and taken from a website named Kaggle as shown in Figure 3.1. Under Google LLC, Kaggle is a platform for data science competitions and an online community for data scientists and machine learning practitioners. Users can discover and publish datasets on Kaggle, explore and develop models in a web-based data science environment, collaborate with other data scientists and machine learning experts, and participate in competitions to address data science challenges. The dataset contains 16,800 letters written by 60 participants, the age range is between 19 to 40 years, and 90% of participants are right-hand. The database is partitioned into two sets: a training set (13,440 letters to 480 images per class) and a test set (3,360 letters to 120 images per class). Writers of training set and test set are exclusive. These images have 32 × 32 resolution pixels and are in the Portable Network Graphics (PNG) format.



Arabic Handwritten Characters Dataset

Arabic Handwritten Characters Data-set

About Dataset

Arabic Handwritten Characters Dataset

Please Cite our papers:

• A. El-Sawy, M. Loey, and H. EL-Bakry, "Arabic handwritten characters recognition using convolutional neural network," WSEAS Transactions on Computer Research, vol. 5, pp. 11–19, 2017.

https://doi.org/10.1007/978-3-319-48308-5_54

https://link.springer.com/chapter/10.1007/978-3-319-48308-5_54

• A. El-Sawy, H. EL-Bakry, and M. Loey, "CNN for handwritten arabic digits recognition based on lenet-5," in Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016, vol. 533, pp. 566–575, Springer International Publishing, 2016.

https://www.wseas.org/multimedia/journals/computerresearch/2017/a045818-075.php

- Loey, Mohamed, Ahmed El-Sawy, and Hazem El-Bakry. "Deep learning autoencoder approach for handwritten arabic digits recognition." arXiv preprint arXiv:1706.06720 (2017).

https://arxiv.org/abs/1706.06720

Usability
7.65

License
Database: Open Database, Cont...

Expected update frequency
Not specified

Tags
Computer Science
Classification
Computer Vision
CNN

Figure 3.1: Screenshot of Arabic letter dataset

3.3.4 Data Preprocessing

Data pre-processing is the final phase 1 step that is covered. To train AI and machine learning models and run inferences against them, data preparation techniques have lately undergone modifications. Data preprocessing transforms raw data into information so that data mining and machine learning can process it more quickly and effectively. The methodologies are frequently used at the very beginning of the machine learning and AI development pipeline in order to get dependable results. In this step, the dataset is cleaned in order to create a clean and balanced dataset for later analysis stage. These processes reshaping the images, resizing the images, data augmentation, and renaming the labels of every image to fit the classes of every Arabic Letter. Here, the dataset of Arabic letters that was gathered during the data collection activity was processed and cleaned. This activity's deliverable was an Arabic letter dataset that has been processed.

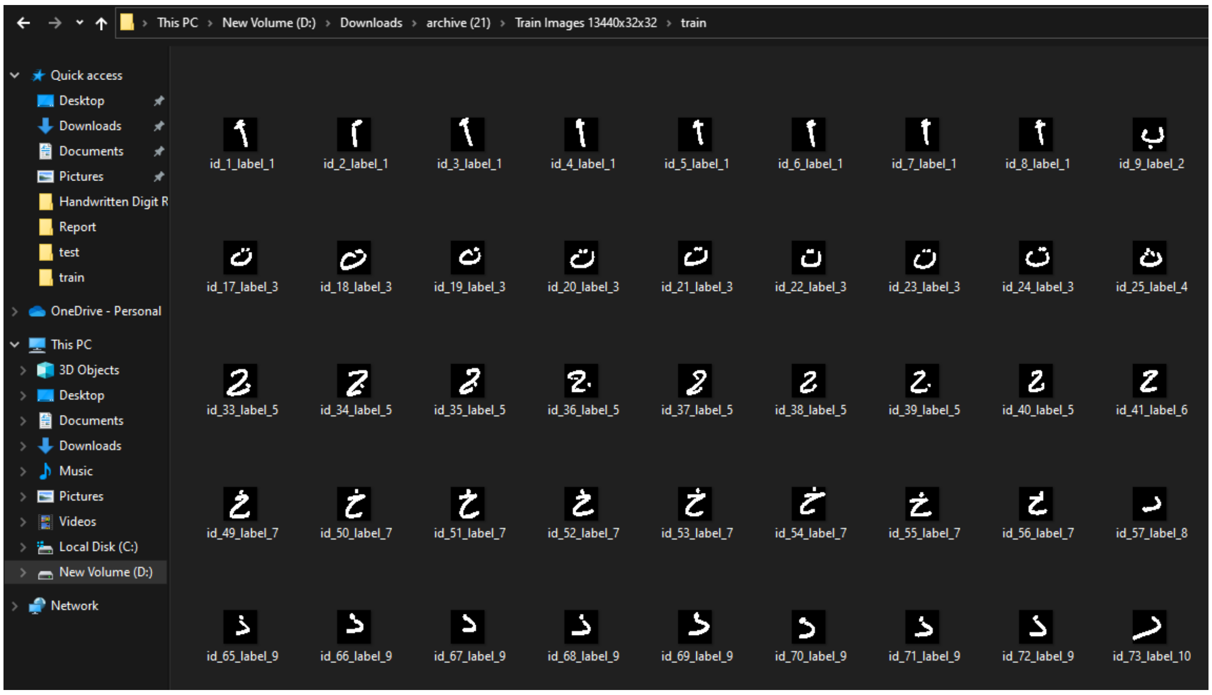


Figure 3.2: Screenshot of Arabic letter before classification

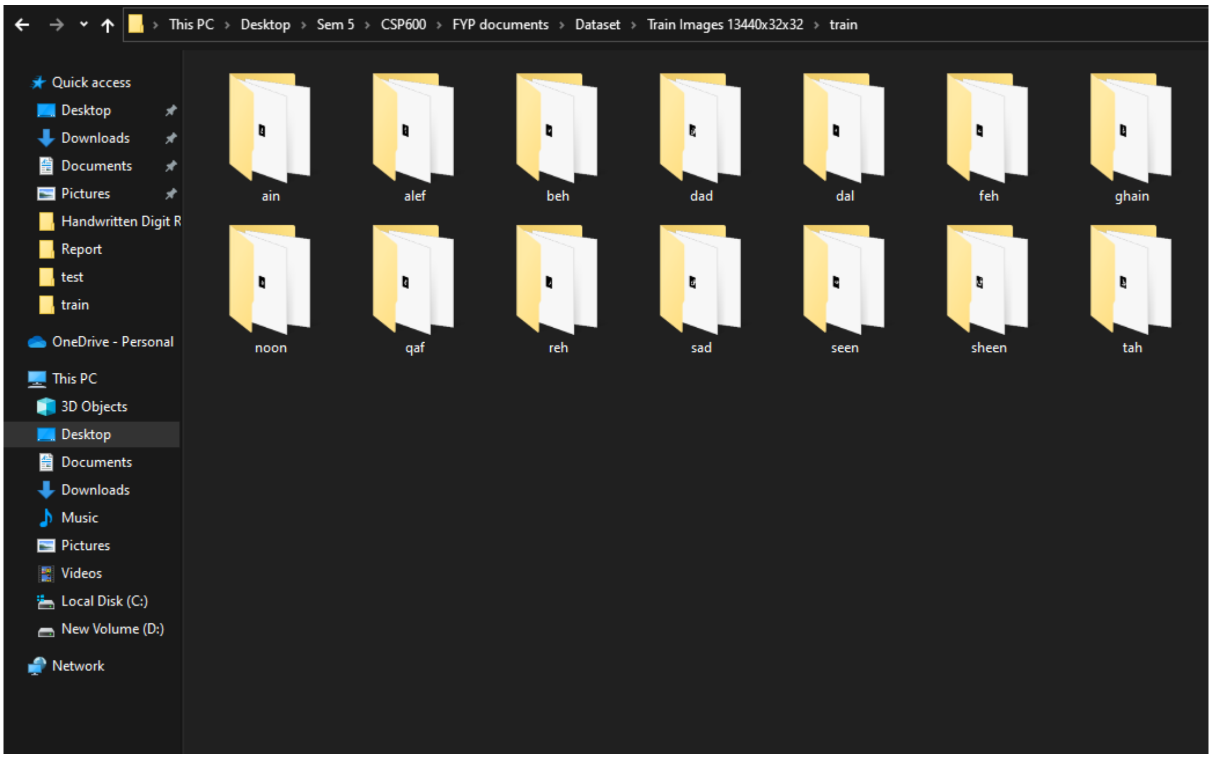
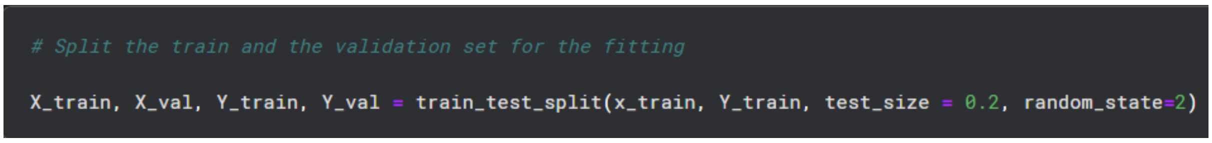

This PC > Desktop > Sem 5 > CSP600 > FYP documents > Dataset > Train Images 13440x32x32 > train
ain
alef
beh
dad
dal
feh
ghain
noon
qaf
reh
sad
seen
sheen
tah

Figure 3.3: Screenshot of Arabic letter after classification

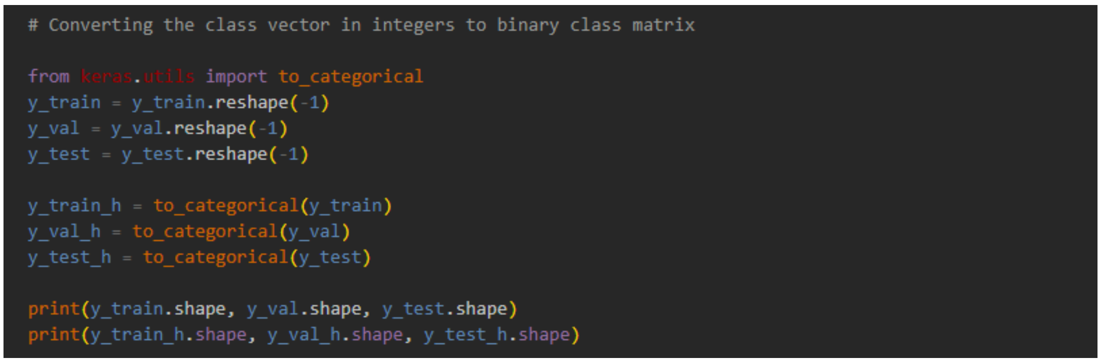
First of all, the images of the Arabic letters are classed according to their own letters into files. Then the name of the folders was changed for an easier understanding and can be read as letters with label. Next, the name of every image was too long which is then renamed to make it easier to be coded in python.



```
# Split the train and the validation set for the fitting

X_train, X_val, Y_train, Y_val = train_test_split(x_train, Y_train, test_size = 0.2, random_state=2)
```

Figure 3.4: Code for splitting dataset into training and testing set.

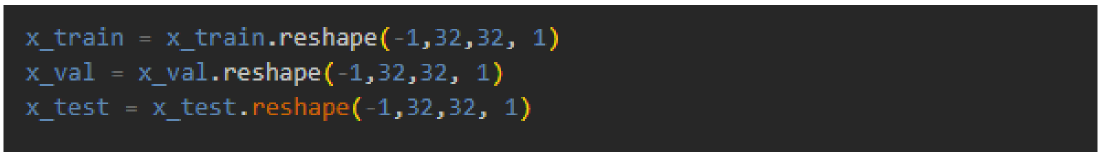
The dataset was already split between test and train, so further data splitting is done through the process of coding in python. Twenty percent of the data from the original training dataset were used for validation, while eighty percent were used for training. The Sklearn model selection tool's "train_test_split" method was used to do this. Data arrays are frequently divided into two subsets: training data and validation data. The proportion that was specified in the coding will be used by the system to divide it. The coding of the dataset was divided into two sets of data, as shown in Figure 3.4. For the testing data, the "test_size" was set to 0.20, or 20%. The "random_state" function is utilised to distribute the dataset in an unbiased manner, as a fixed value ensures that the same set of random numbers is generated each time the code is executed. Two datasets, test and train, each containing 28 Arabic Letters, are the process's output. The testing dataset was later then used to test the prediction of the model that was trained using training and validation set.



```
# Converting the class vector in integers to binary class matrix

from keras.utils import to_categorical
y_train = y_train.reshape(-1)
y_val = y_val.reshape(-1)
y_test = y_test.reshape(-1)

y_train_h = to_categorical(y_train)
y_val_h = to_categorical(y_val)
y_test_h = to_categorical(y_test)

print(y_train.shape, y_val.shape, y_test.shape)
print(y_train_h.shape, y_val_h.shape, y_test_h.shape)
```

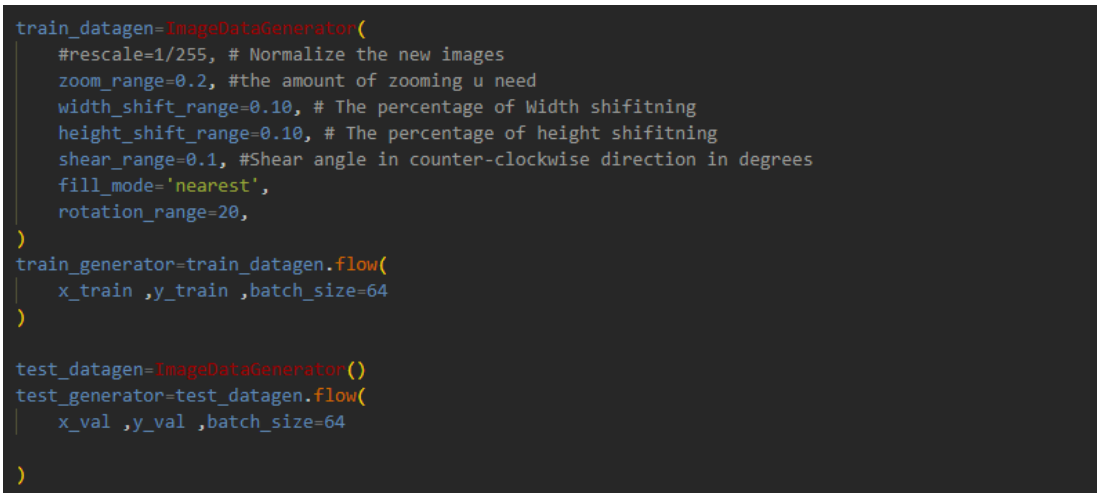
Figure 3.5: Code for converting class vector.

The images are categories according to its classes by converting the class vector in integers to binary class matrix for the model to class the images easier.



```python
x_train = x_train.reshape(-1,32,32, 1)
x_val = x_val.reshape(-1,32,32, 1)
x_test = x_test.reshape(-1,32,32, 1)
```

Figure 3.6: Code for reshaping image



```python
train_datagen=ImageDataGenerator(
    #rescale=1/255, # Normalize the new images
    zoom_range=0.2, #the amount of zooming u need
    width_shift_range=0.10, # The percentage of Width shifitning
    height_shift_range=0.10, # The percentage of height shifitning
    shear_range=0.1, #Shear angle in counter-clockwise direction in degrees
    fill_mode='nearest',
    rotation_range=20,
)
train_generator=train_datagen.flow(
    x_train ,y_train ,batch_size=64
)

test_datagen=ImageDataGenerator()
test_generator=test_datagen.flow(
    x_val ,y_val ,batch_size=64

)
```

Figure 3.7: Code for data augmentation.

The reshaping of the images is done and then proceeded to data augmentation where the images are then randomly flipped to create new samples within the same class of images. Finally, data augmentation is used to expand the dataset's image count. In addition to increasing the dataset's size, data augmentation produces a variety of newly altered image types. This enables the model to identify hidden features in the images and make better sense of the data. As a result, the model's accuracy may be raised. The dataset was supplemented using the ImageDataGenerator function found in the Python package Keras, as seen in Figure 3.7. In order to improve the dataset's resilience and store it in overhead memory, ImageDataGenerator augments it in real-time. The images in the dataset were augmented by flipping, rotating, and zooming them while adjusting their width and height within a 0.1 range. However, some of the image's pixels were moved outside of the image, leaving an empty space, when the image was rotated and shifted. The nearest pixel values were used to fill the empty area of the image using the fill mode. Every transformation was applied to the dataset at random.

3.4 PHASE 2

Phase 2 focused to design Arabic letter handwriting recognition using a machine learning algorithm that can accurately recognize and classify Arabic letters. The system was created using a deep learning algorithm, and the Python programming language was used to create the system's web application. Phase 2 activities include model development, model design, and model evaluation.

3.4.1 Model Development

The model development will utilize neural network architectures such as Convolutional Neural Networks (CNNs) or Deep Convolutional Neural Networks (DCNNs) to learn the representations directly from the input features. These models can automatically learn hierarchical and abstract representations of the letters, enhancing recognition accuracy. The choice of knowledge representation depends on the complexity of the recognition task, available data, and computational resources. Once the knowledge is represented, it will be employed on Convolutional Neural Network model to recognized Arabic Letters. The model will undergo model training which is splitting the dataset into training and validation sets. Training selected model using the training data. Using the validation data to tune hyperparameters and prevent overfitting. The model is also fine-tuned by adjusting hyperparameters such as learning rate, batch size, network architecture, epoch to achieve better performance. The model was also tested with different types of method such as SVM and Random Forest. Both of these models use Grid Searching to help find the best parameters to help fit into the model to obtain the best performance. Using a dataset as a training set, support vector machines (SVMs) are trained to determine the ideal decision boundary for efficiently dividing several letter classes. The underlying relationships in the data determine which kind of kernel function, such as radial basis function (RBF), linear, or polynomial, to use. The model's performance is further improved via hyperparameter tuning, and assessment measures like accuracy and precision are used to measure the model's efficacy on an independent test dataset. Next, Random Forest works by building several decision trees during training, each of which picks up patterns and features on its own from various data subsets. These trees work together to create a strong and varied model using a procedure called bagging. To generate an ensemble decision that greatly enhances accuracy and generalisation in the context of letter recognition, the method evaluates many characteristics and pixel values.

3.4.2 Model Design

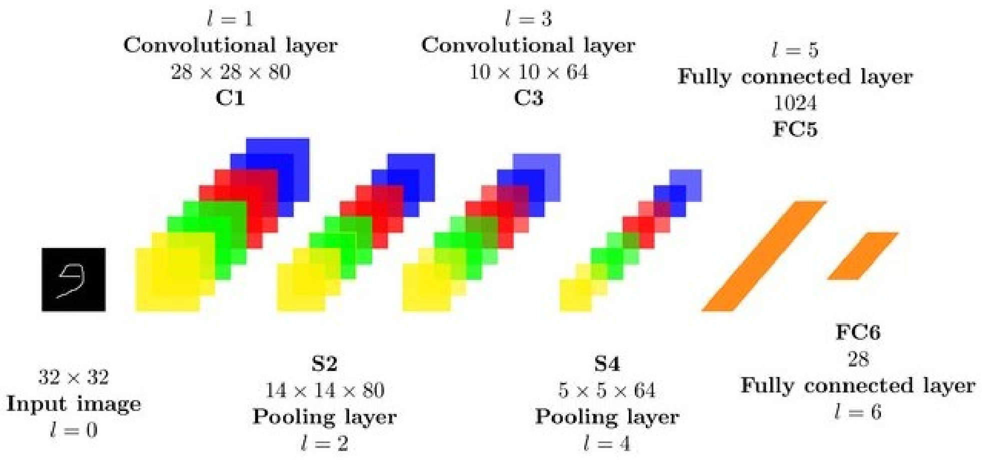


Figure 3.8: Image of CNN architecture

. A machine learning or deep learning model is selected based on the specific requirements of the recognition task and image processing. Models like Convolutional Neural Networks (CNNs) will be used for Arabic letter recognition. The chosen model is trained using the annotated dataset, adjusting hyperparameters and optimizing the model's performance.

Convolutional Neural Networks (CNNs) are constructed as part of the model design in order to recognise and categorise handwritten letters in images. Data collection is the first step in the process, during which photographs of handwritten letters are obtained to create a dataset that represents various writing styles and variants. Resizing, normalisation, and perhaps augmentation are preprocessing operations that make sure the data is in a format that the model can use. The CNN architecture is made up of several convolutional layers, each of which uses learnable filters to analyse the input images and identify regional patterns and properties particular to the letters. By adding more convolutional layers, the model may gradually learn hierarchical representations of the letters and extract increasingly intricate information as the layers are added. By pooling layers, it is possible to sample the feature maps, which lowers computational effort and improves translational invariance. The first layer of the CNN model accepts images that are sized (32x32) as its width and height. The images go through 4 different layers which are convolutional and pooling layers and using Relu as its activation function. The layers are then flattened into a fully connected layer.

Fully connected layers are used for letter classification after the convolutional and pooling layers. To generate predictions for each letter class, the learnt attributes are integrated. A SoftMax activation function is employed in the output layer to produce probabilities for each letter class, indicating the chance that a given input image belongs to a specific letter.

3.4.3 Model Evaluation

Model evaluation for CNN image processing in letter recognition is a crucial step to assess the model's performance and its ability to accurately classify handwritten letters. The evaluation process involves measuring the model's predictions against the ground truth labels in the test dataset. To begin the evaluation, we feed the test images through the trained CNN model, SVM model and Random Forest model. The model predicts the letter classes for each image based on the learned features from the training data. We then compare the predicted classes with the true labels of the test images.

Accuracy, precision, recall, and the F1-score are frequently used evaluation metrics for letter recognition. Out of all test samples, the accuracy is the percentage of correctly classified letters. When comparing all predicted positive cases, precision is defined as the percentage of correctly identified positive predictions (correctly classified letters). The percentage of correctly identified positive cases among all actually positive cases is known as recall, also known as sensitivity. The F1-score provides a balanced metric for models that might have unbalanced class distributions because it is the harmonic mean of precision and recall. A confusion matrix is also created to see how well the model performs across various letter classes. It provides information on the model's advantages and disadvantages by displaying the number of true positive, true negative, false positive, and false negative predictions for each class. Out of all the models, it seemed that CNN provided the best classification performance in terms of accuracy.

3.5 PHASE 3

Phase 3 is the last phase where it describes the third objective which is to evaluate the performance of the model for the Arabic letter handwriting recognition. These phases are divided into which is system testing and evaluation, and documentation.

3.5.1 System Development

The development process starts with implementing the preprocessing techniques to clean and normalize the Arabic letter images. This includes resizing, denoising, and contrast enhancement. Next, the feature extraction algorithms are implemented to extract the relevant features from the pre-processed images. The chosen feature representation is encoded into a suitable format for further processing. The selected machine learning or deep learning model for Arabic letter recognition is then implemented. This involves constructing the network architecture, defining the layers, and setting the parameters. The model is trained using the prepared training data, employing optimization algorithms and backpropagation techniques to update the model's parameters iteratively. After training, the developed system undergoes rigorous testing using the separate test set. The system's recognition accuracy, speed, and robustness are evaluated. Performance metrics are calculated to assess the system's effectiveness, and any necessary adjustments or refinements are made. Using Python's extensive library and toolkit, creating a letter recognition system requires a methodical and thorough approach. Python modules like PIL and OpenCV are used and quite helpful in managing image processing jobs and extracting pertinent characteristics. Convolutional neural networks (CNNs), which are well-liked for their efficacy in image identification tasks, are implemented using the scikit-learn, Keras, and TensorFlow libraries. The system is developed mainly using Python which is a programming language commonly used for machine learning. The python environment is integrated together with a python library called Tkinter and CustomTkinter. A library that helps the development of graphical interface for Python. The best Convolutional neural networks (CNNs) model is saved as a file and is later called as a function to help integrate together with the application that is developed through Tkinter.

3.5.2 System Design

3.5.2.1 System Architecture

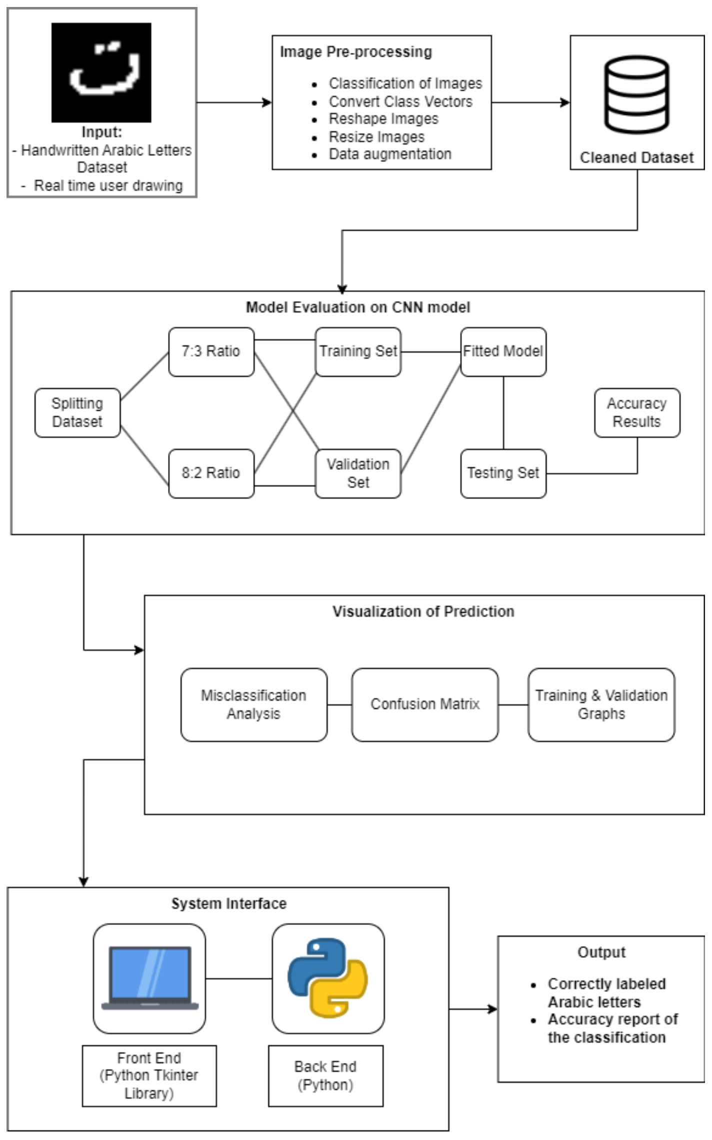
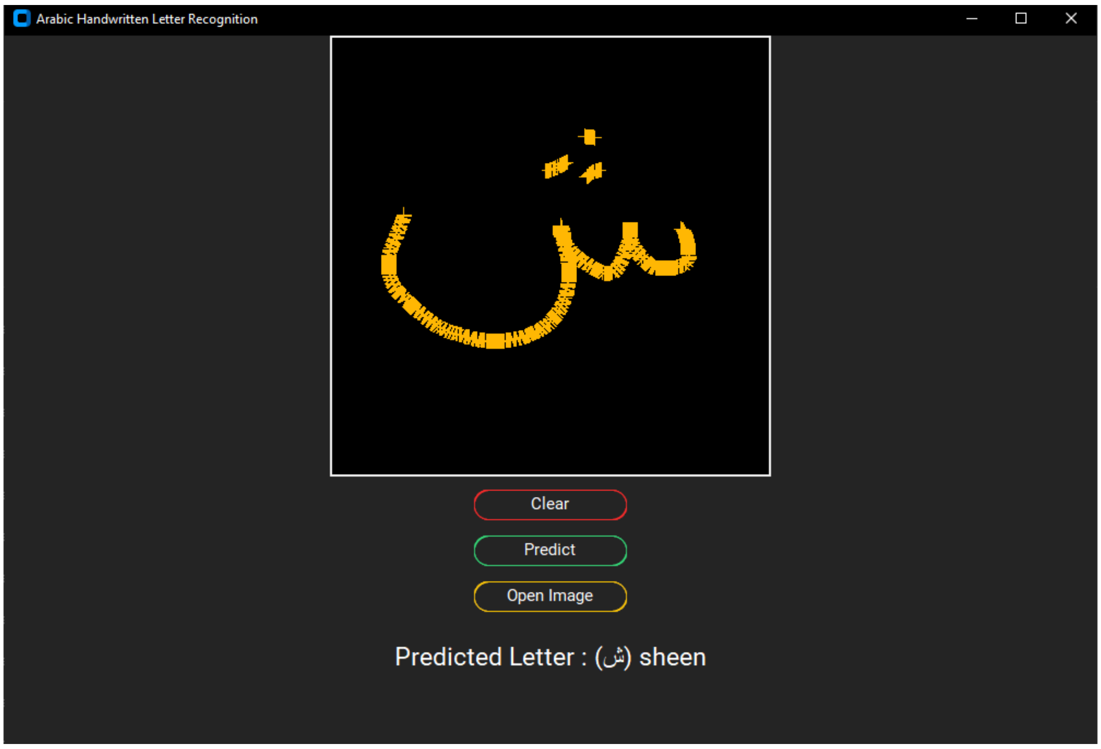


Figure 3.9: Image of System Architecture

The system architecture is a diagram that shows the structure, behaviour, and viewpoints of the system. The project's system architecture design is depicted in the picture below. There are four primary components to the architecture that must be completed in order to finish the system. The Arabic letter dataset's input images were first and foremost pre-processed. The pre-processed data includes classification, converting class vector, reshaped, and augmented images. A validation set and a testing set were split from the final, cleaned dataset. The split of the dataset uses ratio of 7 by 3 and 8 by 2. The different types of CNN models were then used to obtain the accuracy result. The evaluation is then visualized through misclassification analysis, confusion matrix, and training and validation graphs. Finally, Python was used to code both the system's front end and back end. A library in Python is used to improve the interface's appearance, Python code was combined with Tkinter and CustomTkinter. The system will also be able to receive real time drawing of handwritten Arabic letters. The output is then produced as correctly labelled Arabic letters and accuracy report from the training of models.

3.5.2.2 System Interface



Predicted Letter : (ش) sheen

Figure 3.10: Image of System Interface

The user interface (UI) of the system serves as a conduit for user interaction and communication with the system. Therefore, by having a user interface, the system can also be used by end users who lack the technical skills necessary to input the images. Python programming language is used to develop and show the system on a prompt window to create the system interface. The prompt window is called through the Windows original command prompt. The system page's structure, which includes a canvas to allow the user to draw the Arabic letter, a home page to provide description of the system, and also an option to allow users to input images from files and process is to obtain prediction label. The interface is integrated where users can draw letters using a mouse. The system also has buttons that provide navigation to browse through the system and buttons that activate functions to predict and clear the drawings on the canvas. There is a button to open file to allow users to input raw images. The handwritten letter will be detected through the prediction model. The prediction model is called as a file through file paths to help with the recognition of the letters that are inserted through the system. Finally, the system will display the predicted letter according to letter provided by the user. The interface will be adapted through a Python library which is Tkinter and CustomTkinter that allows to customize features and the design of the GUI in Python.

3.5.3 System Testing and Evaluation

In order to guarantee the efficiency and accuracy of an Arabic letter recognition system, system testing, and evaluation are essential steps. In testing, the effectiveness of the system is evaluated using a broad and representative dataset that are processed through different types of splitting and parameters. The system is assessed by comparing the recognised letters with the ground truth labels, using metrics like accuracy, precision, recall, or F1 score. Error analysis directs improvements by identifying patterns and issues that the system faces. An evaluation of performance measures the system's responsiveness, effectiveness, and resource usage. The system's resistance to changes is tested through robustness testing. The generalisation abilities of the system are thoroughly evaluated by cross-validation. In order to gain knowledge about usability and effectiveness, user feedback is gathered. The system was tested with a variety of different types of models using different types of methods such as CNN, SVM and Random Forest. The competitiveness of the system is assessed through comparisons with benchmarks or existing techniques. The correctly predicted label is proof of the efficiency of the system, most of the drawing done through the canvas of the system provides a well classification of the letters. When the system shows a wrongly classified label, it can be defined that the model is still inaccurate as the drawing of letters can differ from every user.

3.5.4 Documentation

The documentation for the Arabic letter recognition system serves as a comprehensive resource, offering a detailed understanding of the system's architecture, components, usage, and implementation. It begins with an introduction that highlights the significance and applications of Arabic letter recognition. The system overview section provides a high-level description of the system's architecture, emphasizing the interplay between its major components and the data flow. System requirements are listed, including hardware specifications, software dependencies, and necessary configurations. The installation and setup instructions guide users through the process of installing and configuring the system on various platforms.

The user guide section outlines the system's usage, including input/output formats and provides examples to illustrate its functionality. Each system component is meticulously explained, delving into its purpose, functionality, and dependencies. Preprocessing techniques are described, covering reshaping, resizing, and data augmentation. The feature extraction methods employed to extract pertinent features from the pre-processed Arabic letter images are also explained. A comprehensive understanding of the system's model architecture is offered, highlighting the layers, parameters, and algorithms utilized. The training and evaluation section delves into the training process, encompassing the dataset, hyperparameter settings, and optimization algorithms.

Performance metrics, such as accuracy, precision, recall, and F1 score, are presented alongside a comparative analysis with existing methods or baselines. Troubleshooting tips are provided to address common issues, ensuring smooth operation and user experience. The documentation concludes with future improvement suggestions, highlighting potential research directions and areas for further development. Overall, this documentation provides users and developers with the necessary information to comprehend, deploy, and harness the full potential of the Arabic letter recognition system.

3.6 Summary

This chapter has described and explained the three phases to complete the project. Phase 1 consists of preliminary studies, knowledge gathering and acquisition, data collection, and preparation. Phase 2 then consists of knowledge representation, system design, system development. Phase 3 explains about system testing and evaluation, and documentation of the system. All these phases coexist to make sure that the right steps are to be taken in order for the system to use the right methods and algorithms to help develop the Arabic letter recognition system.

CHAPTER 4

RESULT AND FINDINGS

This chapter covers every experiment conducted as well as the outcomes and conclusions from the study following the application of the methodology. Every chart and result will be displayed and examined. The outcomes of the dataset preparation, the model training and evaluation, model experiment comparison, system testing, and the chapter summary are all included in this chapter.

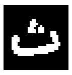
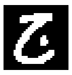
4.1 Dataset Preparation

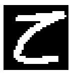
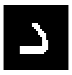
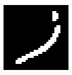
As mentioned in previous chapters, the dataset that is used is a Arabic Handwritten Letter Dataset by Mohamed Loey that is obtained from Kaggle. The dataset consists of 16,800 characters written by 60 individuals, 90% of whom are right-handed, and whose ages range from 19 to 40. Two sets of data are extracted from the database: a test set with 3,360 characters and 120 images per class, and a training set with 13,440 characters and 480 images per class. The dataset also contains CSV files that notify the labels of every character in the dataset. The original dataset that is obtain from Kaggle is separated between train and test set which requires rearrangement to help better classify the image in its own categories. The images are rearranged into 28 different classes which are 'alef', 'beh', 'teh', 'theh', 'jeem', 'hah', 'khah', 'dal', 'thal', 'reh', 'zain', 'seen', 'sheen', 'sad', 'dad', 'tah', 'zah', 'ain’, 'ghain', 'feh', 'qaf', 'kaf', 'lam', 'meem', 'noon', 'heh', 'waw', and 'yeh'. The images are arranged according to its label id on the images which ranges from 1 to 28 following the name of the class by order.

Further preprocessing is done through data augmentation, resizing, and reshaping in the python environment which are through coding. There are 3 types of models used which are Convolutional Neural Networks (CNN), SVM and Random Forest. The datasets that are processed through all these models are split into 2 types of ratios which is 7 by 3 also known as 70% for training and 30% for validation and the second split is 8 by 2 also known as 80% for training and 20% for validation. The original training dataset which consists of 13,440 characters is used for this purpose to split into training and validation set. The original test set is used later on for the model prediction once the model has been fitted with the parameters.

Data Augmentation Properties	**Value**
Reshape and resizing.	32 x 32 pixels
Zoom range	0.2
The percentage of width shifting	0.1
The percentage of height shifting	0.1
Shear angle in counterclockwise direction (degrees)	0.1
Fill mode	Nearest
Rotation range (degrees)	20

Table 4.1: *Table of Data Augmentation properties.*

Character	**Number of Images before splitting**	**Images after (7:3) ratio splitting**	**Images after (8:2) ratio splitting**
Alef	480	336	384
Beh	480	336	384
Teh	480	336	384
Theh 	480	336	384
Jeem 	480	336	384

Hah 	480	336	384
Khah	480	336	384
Dal 	480	336	384
Thal	480	336	384
Reh	480	336	384
Zain 	480	336	384
Seen	480	336	384
Sheen	480	336	384
Sad	480	336	384
Dad	480	336	384
Tah	480	336	384
Zah	480	336	384
Ain	480	336	384

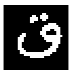
Ghain	480	336	384
Feh	480	336	384
Qaf 	480	336	384
Kaf	480	336	384
Lam	480	336	384
Meem	480	336	384
Noon	480	336	384
Heh	480	336	384
Waw	480	336	384
Yeh	480	336	384

Table 4.2: *Table for Number of Images in train set after data spliting*

4.2 Model Training and Evaluation

There are a total of 8 experiments that are done to evaluate this project. The experiments are done using 3 different types of machine learning techniques which are CNN, SVM and Random Forest. Each of these techniques are divided into two types of experiments which uses the 7 by 3 and 8 by 2 dataset ratio. Lastly, the CNN model is split into 4 experiments which are separated according to 30 and 50 iterations and on top of the dataset splitting. This section of the chapter will explain the experiment details and the result gathered through model training. All of the experiments use the same testing set to predict the model's accuracy.

4.2.1 Experiment 1: SVM model with 7 by 3 split

a) Model Configuration

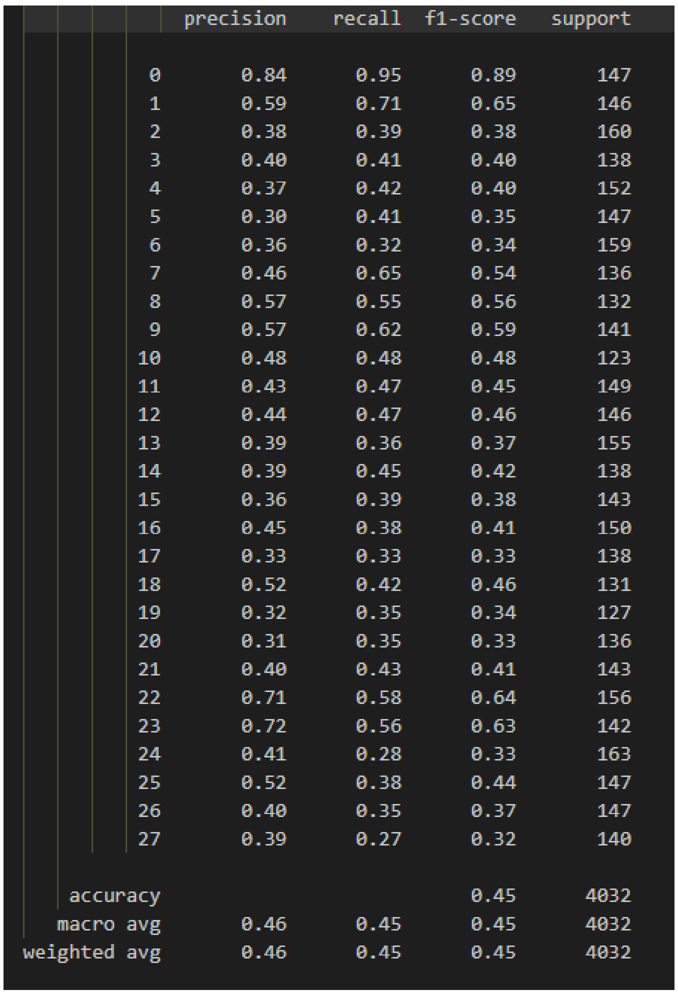
The first experiment uses SVM technique for the model and the 7 by 3 training dataset which consists of 9408 characters for training and 4032 characters for validation. The model uses the GridSearch function in SVM environment to obtain the best parameters to model the dataset.

Parameters	**Value**
Regularization parameter (C)	10
Gamma	0.0001
Kernel	linear

Table 4.3: *Table of SVM parameters 7 by 3 split*

b) Model Experiment Results

The model is trained using SVM function which is SVC to help model the training set using the mentioned parameters. After the model training is done, it is then predicted using the validation set containing 4032 characters. The newly predicted model achieved an accuracy of 45.36% which is relatively low to classify the characters for each image. An accuracy report is generated and through all the letter, the average of its accuracy is obtained, and it can only predict 46% of the letter out of 100%. This model is not really reliable since the performance of the model is not reaching the required standard.



```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89       147
           1       0.59      0.71      0.65       146
           2       0.38      0.39      0.38       160
           3       0.40      0.41      0.40       138
           4       0.37      0.42      0.40       152
           5       0.30      0.41      0.35       147
           6       0.36      0.32      0.34       159
           7       0.46      0.65      0.54       136
           8       0.57      0.55      0.56       132
           9       0.57      0.62      0.59       141
          10       0.48      0.48      0.48       123
          11       0.43      0.47      0.45       149
          12       0.44      0.47      0.46       146
          13       0.39      0.36      0.37       155
          14       0.39      0.45      0.42       138
          15       0.36      0.39      0.38       143
          16       0.45      0.38      0.41       150
          17       0.33      0.33      0.33       138
          18       0.52      0.42      0.46       131
          19       0.32      0.35      0.34       127
          20       0.31      0.35      0.33       136
          21       0.40      0.43      0.41       143
          22       0.71      0.58      0.64       156
          23       0.72      0.56      0.63       142
          24       0.41      0.28      0.33       163
          25       0.52      0.38      0.44       147
          26       0.40      0.35      0.37       147
          27       0.39      0.27      0.32       140

    accuracy                           0.45      4032
   macro avg       0.46      0.45      0.45      4032
weighted avg       0.46      0.45      0.45      4032
```

Figure 4.1: *Accuracy Report of SVM model using 7 by 3 split*

4.2.2 Experiment 2: SVM model with 8 by 2 split

a) Model Configuration

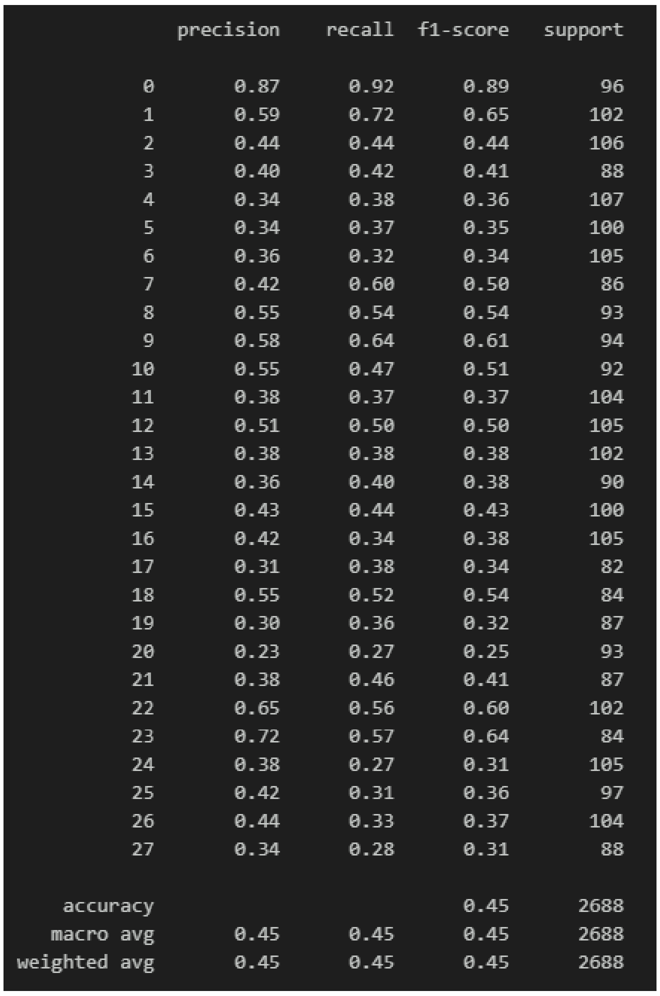
The second experiment uses SVM technique for the model and the 8 by 2 training dataset which consists of 10,752 characters for training and 2688 characters for validation. The model uses the GridSearch function in SVM environment to obtain the best parameters to model the dataset. The same parameters are used for the second experiment as it is the best parameter obtained through GridSearch function.

Parameters	**Value**
Regularization parameter (C)	10
Gamma	0.0001
Kernel	linear

Table 4.4: *Table of SVM parameters 8 by 2 split*

b) Model Experiment Results

The model is trained using SVM function which is SVC to help model the training set using the mentioned parameters. After the model training is done, it is then predicted using the validation set containing 2688 characters. The newly predicted model achieved an accuracy of 44.64% which is relatively low to classify the characters for each image. An accuracy report is generated and through all the letter, the average of its accuracy is obtained, and it can only predict 45% of the letter out of 100%. This model is not really reliable since the performance of the model is not reaching the required standard. The accuracy of the second experiment is lower by 1% which means that the model performance for experiment 2 is slightly decreasing.



```
              precision    recall  f1-score   support

           0       0.87      0.92      0.89        96
           1       0.59      0.72      0.65       102
           2       0.44      0.44      0.44       106
           3       0.40      0.42      0.41        88
           4       0.34      0.38      0.36       107
           5       0.34      0.37      0.35       100
           6       0.36      0.32      0.34       105
           7       0.42      0.60      0.50        86
           8       0.55      0.54      0.54        93
           9       0.58      0.64      0.61        94
          10       0.55      0.47      0.51        92
          11       0.38      0.37      0.37       104
          12       0.51      0.50      0.50       105
          13       0.38      0.38      0.38       102
          14       0.36      0.40      0.38        90
          15       0.43      0.44      0.43       100
          16       0.42      0.34      0.38       105
          17       0.31      0.38      0.34        82
          18       0.55      0.52      0.54        84
          19       0.30      0.36      0.32        87
          20       0.23      0.27      0.25        93
          21       0.38      0.46      0.41        87
          22       0.65      0.56      0.60       102
          23       0.72      0.57      0.64        84
          24       0.38      0.27      0.31       105
          25       0.42      0.31      0.36        97
          26       0.44      0.33      0.37       104
          27       0.34      0.28      0.31        88

    accuracy                           0.45      2688
   macro avg       0.45      0.45      0.45      2688
weighted avg       0.45      0.45      0.45      2688
```

Figure 4.2: *Accuracy Report of SVM model using 8 by 2 split*

4.2.3 Experiment 3: Random Forest model with 7 by 3 split

a) Model Configuration

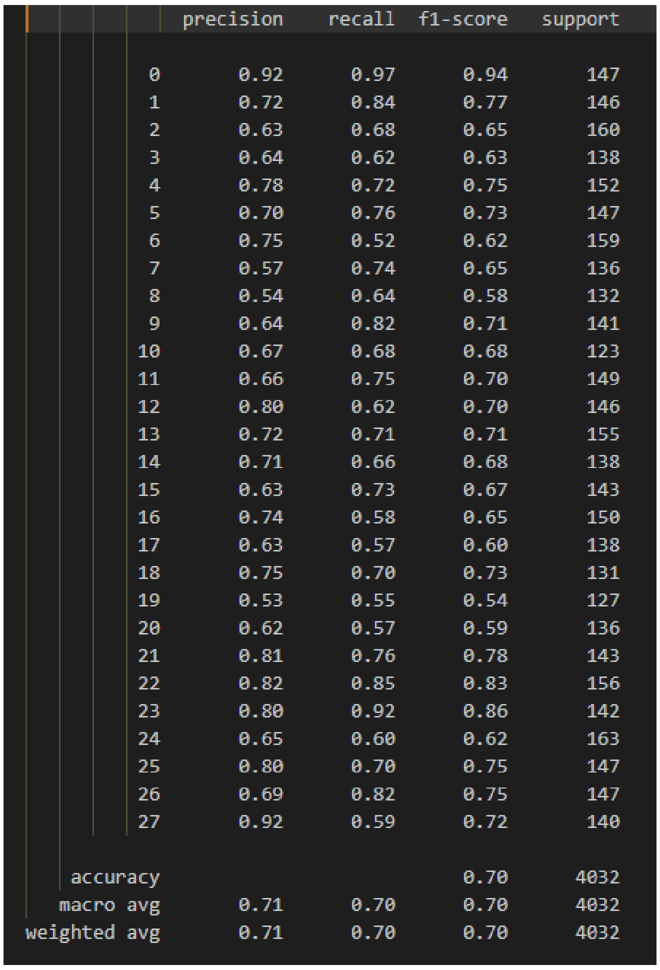
The third experiment uses Random Forest technique for the model and the 7 by 3 training dataset which consists of 9408 characters for training and 4032 characters for validation. The model uses the GridSearch function in Random Forest environment to obtain the best parameters to model the dataset.

Parameters	**Value**
Number of trees in forest	100, 500
Random state of sample	42

Table 4.5: *Table of Random Forest parameters 7 by 3 split*

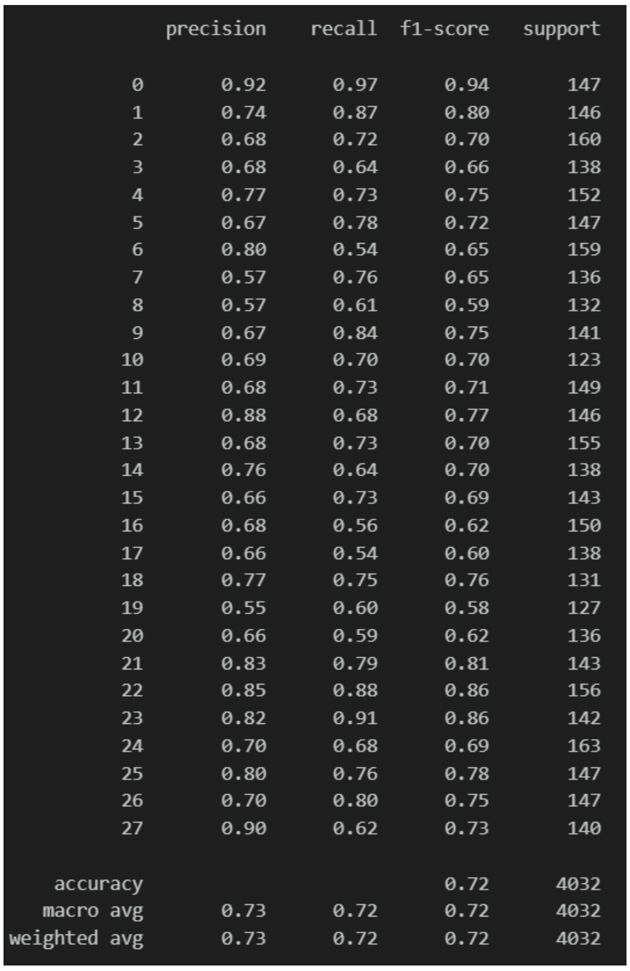
b) Model Experiment Results

The model is trained using Random Forest function which is Random Forest Classifier to help model the training set using the mentioned parameters. There were 2 types of number of trees used which is 100 and 500. After the model training is done, it is then predicted using the validation set containing 4032 characters. The newly predicted model achieved an accuracy of 72.14% for 500 number of trees which is better than the previous experiments with SVM to classify the characters for each image but still is not considered as a good performance model. For the model that uses 100 number of trees, it received 70.23% accuracy which is lower than the model with 500 trees. An accuracy report is generated and through all the letter, the average of the 500 trees model's accuracy is obtained, and it can only predict 73% of the letter out of 100%. This model is already reliable since the performance of the model is close to reaching the required standard. For some of the letters it can be predicted easily but for some may not be due to certain conditions or shapes that are likely to be similar



```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       147
           1       0.72      0.84      0.77       146
           2       0.63      0.68      0.65       160
           3       0.64      0.62      0.63       138
           4       0.78      0.72      0.75       152
           5       0.70      0.76      0.73       147
           6       0.75      0.52      0.62       159
           7       0.57      0.74      0.65       136
           8       0.54      0.64      0.58       132
           9       0.64      0.82      0.71       141
          10       0.67      0.68      0.68       123
          11       0.66      0.75      0.70       149
          12       0.80      0.62      0.70       146
          13       0.72      0.71      0.71       155
          14       0.71      0.66      0.68       138
          15       0.63      0.73      0.67       143
          16       0.74      0.58      0.65       150
          17       0.63      0.57      0.60       138
          18       0.75      0.70      0.73       131
          19       0.53      0.55      0.54       127
          20       0.62      0.57      0.59       136
          21       0.81      0.76      0.78       143
          22       0.82      0.85      0.83       156
          23       0.80      0.92      0.86       142
          24       0.65      0.60      0.62       163
          25       0.80      0.70      0.75       147
          26       0.69      0.82      0.75       147
          27       0.92      0.59      0.72       140

    accuracy                           0.70      4032
   macro avg       0.71      0.70      0.70      4032
weighted avg       0.71      0.70      0.70      4032
```

Figure 4.3: *Accuracy Report of Random Forest model using 7 by 3 split (100 trees)*



```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       147
           1       0.74      0.87      0.80       146
           2       0.68      0.72      0.70       160
           3       0.68      0.64      0.66       138
           4       0.77      0.73      0.75       152
           5       0.67      0.78      0.72       147
           6       0.80      0.54      0.65       159
           7       0.57      0.76      0.65       136
           8       0.57      0.61      0.59       132
           9       0.67      0.84      0.75       141
          10       0.69      0.70      0.70       123
          11       0.68      0.73      0.71       149
          12       0.88      0.68      0.77       146
          13       0.68      0.73      0.70       155
          14       0.76      0.64      0.70       138
          15       0.66      0.73      0.69       143
          16       0.68      0.56      0.62       150
          17       0.66      0.54      0.60       138
          18       0.77      0.75      0.76       131
          19       0.55      0.60      0.58       127
          20       0.66      0.59      0.62       136
          21       0.83      0.79      0.81       143
          22       0.85      0.88      0.86       156
          23       0.82      0.91      0.86       142
          24       0.70      0.68      0.69       163
          25       0.80      0.76      0.78       147
          26       0.70      0.80      0.75       147
          27       0.90      0.62      0.73       140

    accuracy                           0.72      4032
   macro avg       0.73      0.72      0.72      4032
weighted avg       0.73      0.72      0.72      4032
```

Figure 4.4: *Accuracy Report of Random Forest model using 7 by 3 split (500 trees)*

4.2.4 Experiment 4: Random Forest model with 8 by 2 split

a) Model Configuration

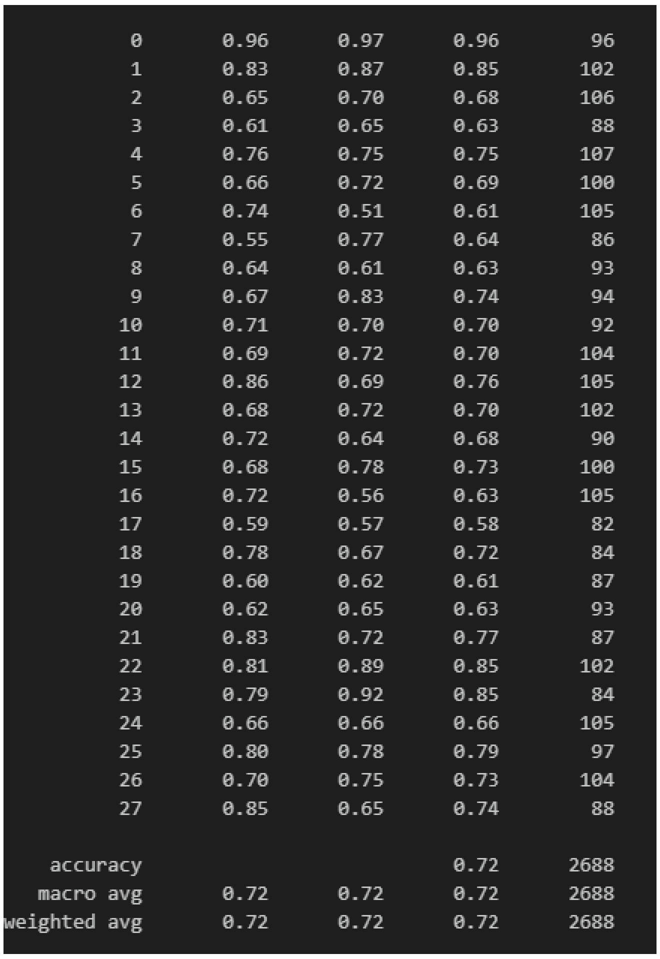
The fourth experiment uses Random Forest technique for the model and the 8 by 2 training dataset which consists of 10,752 characters for training and 2688 characters for validation. The model uses the GridSearch function in Random Forest environment to obtain the best parameters to model the dataset.

Parameters	Value
Number of trees in forest	100, 500
Random state of sample	42

Table 4.6: *Table of Random Forest parameters 8 by 2 split*

b) Model Experiment Results

The model is trained using Random Forest function which is Random Forest Classifier to help model the training set using the mentioned parameters. There were 2 types of number of trees used which is 100 and 500. After the model training is done, it is then predicted using the validation set containing 2688 characters. The newly predicted model achieved an accuracy of 73.32% for 500 number of trees which is better than the previous Random Forest experiment but good enough to classify the characters for each image. For the model that uses 100 number of trees, it received 71.68% accuracy which is lower than the model with 500 trees. An accuracy report is generated and through all the letter, the average of the 500 trees model's accuracy is obtained, and it can only predict 74% of the letter out of 100%. This model is already reliable since the performance of the model is close to reaching the required standard. For some of the letters it can be predicted easily but for some may not be due to certain conditions or shapes that are likely to be similar. The accuracy of the second experiment for Random Forest is higher by 1% which means that the model performance for experiment 2 is slightly increasing.



```
           0       0.96      0.97      0.96        96
           1       0.83      0.87      0.85       102
           2       0.65      0.70      0.68       106
           3       0.61      0.65      0.63        88
           4       0.76      0.75      0.75       107
           5       0.66      0.72      0.69       100
           6       0.74      0.51      0.61       105
           7       0.55      0.77      0.64        86
           8       0.64      0.61      0.63        93
           9       0.67      0.83      0.74        94
          10       0.71      0.70      0.70        92
          11       0.69      0.72      0.70       104
          12       0.86      0.69      0.76       105
          13       0.68      0.72      0.70       102
          14       0.72      0.64      0.68        90
          15       0.68      0.78      0.73       100
          16       0.72      0.56      0.63       105
          17       0.59      0.57      0.58        82
          18       0.78      0.67      0.72        84
          19       0.60      0.62      0.61        87
          20       0.62      0.65      0.63        93
          21       0.83      0.72      0.77        87
          22       0.81      0.89      0.85       102
          23       0.79      0.92      0.85        84
          24       0.66      0.66      0.66       105
          25       0.80      0.78      0.79        97
          26       0.70      0.75      0.73       104
          27       0.85      0.65      0.74        88

    accuracy                           0.72      2688
   macro avg       0.72      0.72      0.72      2688
weighted avg       0.72      0.72      0.72      2688
```

Figure 4.5: *Accuracy Report of Random Forest model using 8 by 2 split (100 trees)*

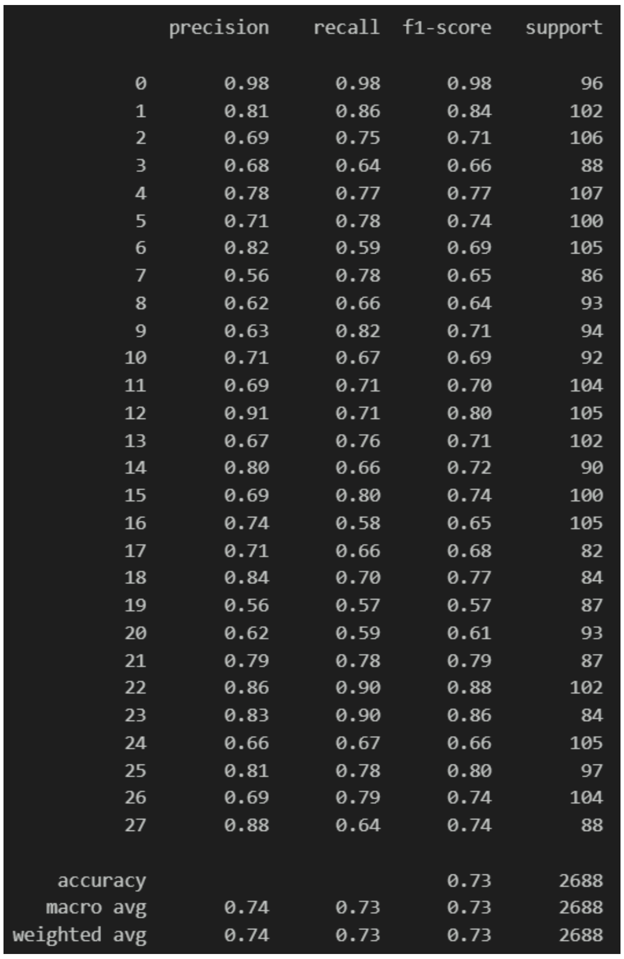

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        96
           1       0.81      0.86      0.84       102
           2       0.69      0.75      0.71       106
           3       0.68      0.64      0.66        88
           4       0.78      0.77      0.77       107
           5       0.71      0.78      0.74       100
           6       0.82      0.59      0.69       105
           7       0.56      0.78      0.65        86
           8       0.62      0.66      0.64        93
           9       0.63      0.82      0.71        94
          10       0.71      0.67      0.69        92
          11       0.69      0.71      0.70       104
          12       0.91      0.71      0.80       105
          13       0.67      0.76      0.71       102
          14       0.80      0.66      0.72        90
          15       0.69      0.80      0.74       100
          16       0.74      0.58      0.65       105
          17       0.71      0.66      0.68        82
          18       0.84      0.70      0.77        84
          19       0.56      0.57      0.57        87
          20       0.62      0.59      0.61        93
          21       0.79      0.78      0.79        87
          22       0.86      0.90      0.88       102
          23       0.83      0.90      0.86        84
          24       0.66      0.67      0.66       105
          25       0.81      0.78      0.80        97
          26       0.69      0.79      0.74       104
          27       0.88      0.64      0.74        88

    accuracy                           0.73      2688
   macro avg       0.74      0.73      0.73      2688
weighted avg       0.74      0.73      0.73      2688
```

Figure 4.6: *Accuracy Report of Random Forest model using 8 by 2 split (500 trees)*

4.2.5 Experiment 5: CNN model with 7 by 3 split using 30 epochs.

a) Model Configuration

The fifth experiment uses CNN technique for the model and the 7 by 3 training dataset which consists of 9408 characters for training and 4032 for validation. The model uses 30 iterations for training.

Parameters	Value
Model	CNN
Dataset	9408 training, 4032 validation, 3360 testing
Iterations (Epochs)	30
Learning Rate	0.001
Optimizer	Adam

Table 4.7: *Table of CNN parameters 7 by 3 split using 30 epochs*

b) Model Experiment Results

The model is trained using CNN layers and 30 iterations through the training and validation set. A result from training the fitted model can be shown as accuracy, loss, validation accuracy, and validation loss. The model finished training at 30 iterations with values such as 93.22% accuracy, 0.27% loss, 95.19% validation accuracy, and 21.80% validation loss.

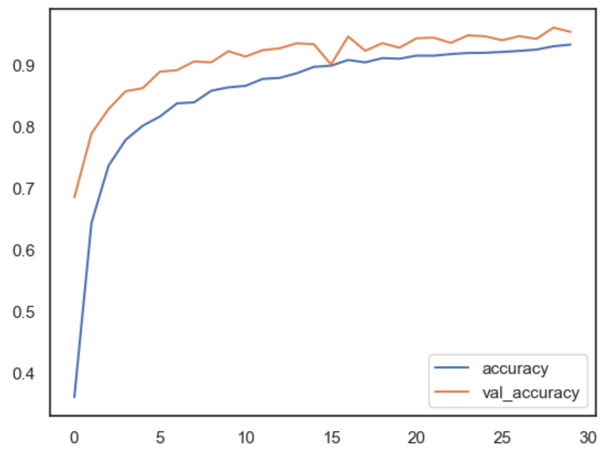


Figure 4.7: *Image of Learning curve graph*

The model is proven to be well fitted as the value of the training and validations is near with the difference of 1.97% in terms of accuracy. The model is not over fitted as the value of train accuracy is still below the validation accuracy.

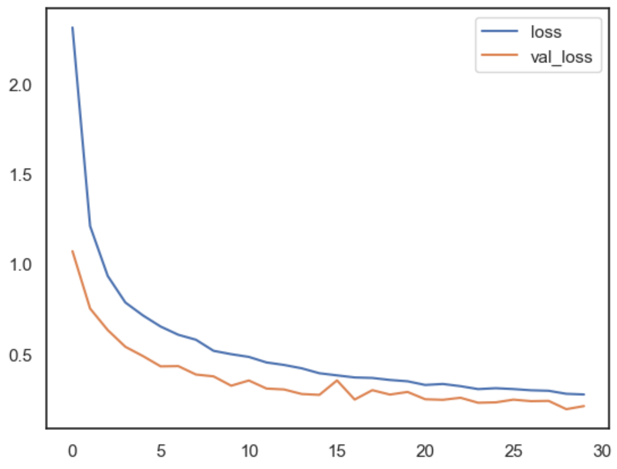
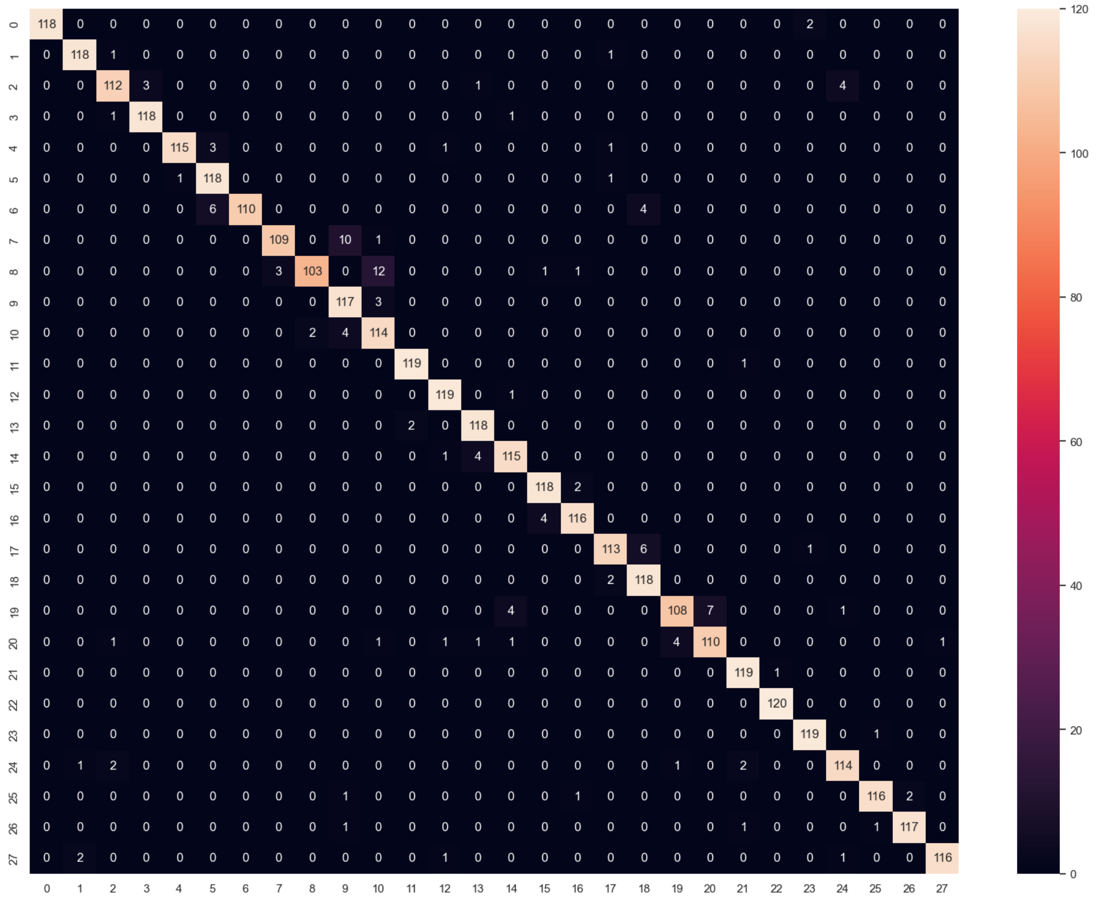


Figure 4.8: *Image of loss Learning curve graph*

The loss starts off with 231% at epoch 1 and gradually decreases until 27% at epoch 30. For validation loss it starts off with 107% at epoch 1 and gradually decreases until 21% at epoch 30. After the model training is done, it is then predicted using the testing set containing 3360 characters.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
1	0	118	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	112	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4	0	0	0
3	0	0	1	118	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	115	3	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	118	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	6	110	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	109	0	10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	103	0	12	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	117	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	4	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	119	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	119	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	2	0	118	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	4	115	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	2	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	116	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	113	6	0	0	0	0	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	118	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	108	7	0	0	0	1	0	0	0
20	0	0	1	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	4	110	0	0	0	0	0	0	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	1	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	1	0	0
24	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	114	0	0	0
25	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	116	2	0
26	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	117	0
27	0	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	116

Figure 4.9: *Confusion Matrix*

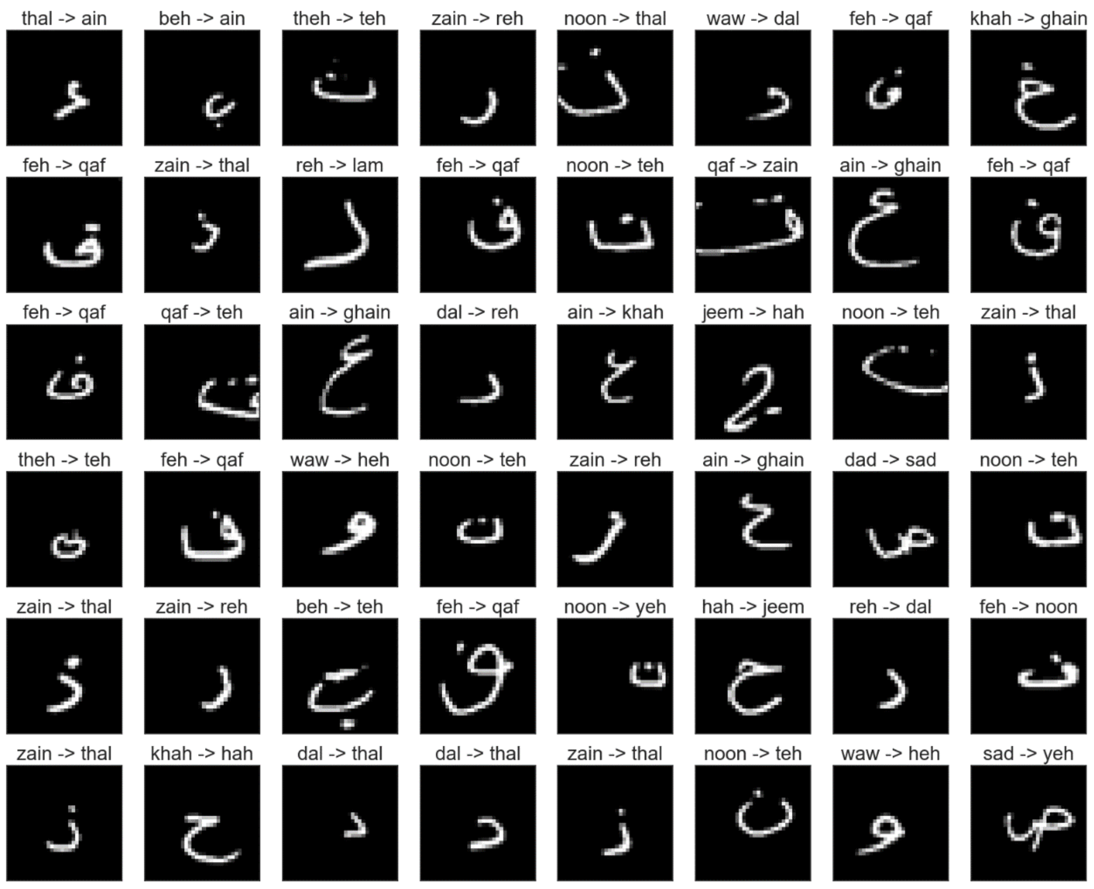
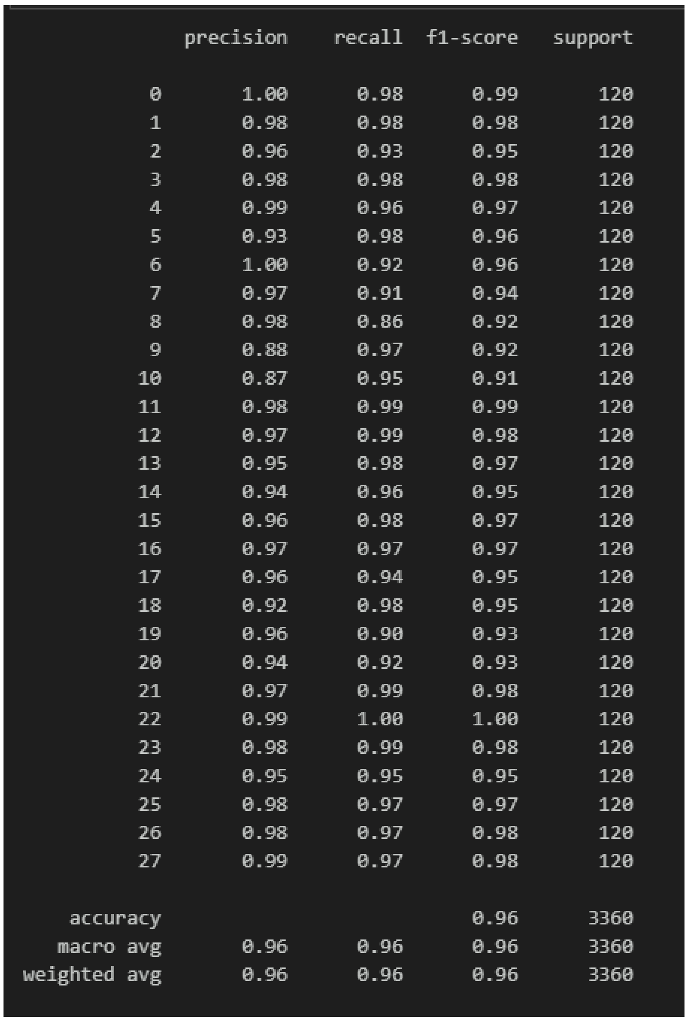


Figure 4.10: *Misclassification Analysis*

Through the above confusion matrix and misclassification analysis, it can be seen that there are a few classes that are predicted wrongly to its actual label but are classified differently. The newly predicted model achieved an accuracy of 96.18% for 30 epochs which is better than the previous experiments and is good enough to classify the characters for each image.



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       120
           1       0.98      0.98      0.98       120
           2       0.96      0.93      0.95       120
           3       0.98      0.98      0.98       120
           4       0.99      0.96      0.97       120
           5       0.93      0.98      0.96       120
           6       1.00      0.92      0.96       120
           7       0.97      0.91      0.94       120
           8       0.98      0.86      0.92       120
           9       0.88      0.97      0.92       120
          10       0.87      0.95      0.91       120
          11       0.98      0.99      0.99       120
          12       0.97      0.99      0.98       120
          13       0.95      0.98      0.97       120
          14       0.94      0.96      0.95       120
          15       0.96      0.98      0.97       120
          16       0.97      0.97      0.97       120
          17       0.96      0.94      0.95       120
          18       0.92      0.98      0.95       120
          19       0.96      0.90      0.93       120
          20       0.94      0.92      0.93       120
          21       0.97      0.99      0.98       120
          22       0.99      1.00      1.00       120
          23       0.98      0.99      0.98       120
          24       0.95      0.95      0.95       120
          25       0.98      0.97      0.97       120
          26       0.98      0.97      0.98       120
          27       0.99      0.97      0.98       120

    accuracy                           0.96      3360
   macro avg       0.96      0.96      0.96      3360
weighted avg       0.96      0.96      0.96      3360
```

Figure 4.11: *Accuracy Report of CNN using 7 by 3 split (30 epochs)*

An accuracy report is generated and through all the letter, the average of the model's accuracy is obtained, and it can only predict 96% of the letter out of 100%. This model is already reliable since the performance of the model has already reached its requirement to classify the letters accurately. Most of the letters can be predicted easily even for those that have certain conditions or shapes that are likely to be similar will be able to detect its class.

4.2.6 Experiment 6: CNN model with 7 by 3 split using 50 epochs.

a) Model Configuration

The fifth experiment uses CNN technique for the model and the 7 by 3 training dataset which consists of 9408 characters for training and 4032 for validation. The model uses 30 iterations for training.

Parameters	**Value**
Model	CNN
Dataset	9408 training, 4032 validation, 3360 testing
Iterations (Epochs)	50
Learning Rate	0.001
Optimizer	Adam

Table 4.8: *Table of CNN parameters 7 by 3 split using 50 epochs*

b) Model Experiment Results

The model is trained using CNN layers and 50 iterations through the training and validation set. A result from training the fitted model can be shown as accuracy, loss, validation accuracy, and validation loss. The model finished training at 50 iterations with values such as 94.89% accuracy, 0.21% loss, 96.63% validation accuracy, and 0.16% validation loss.

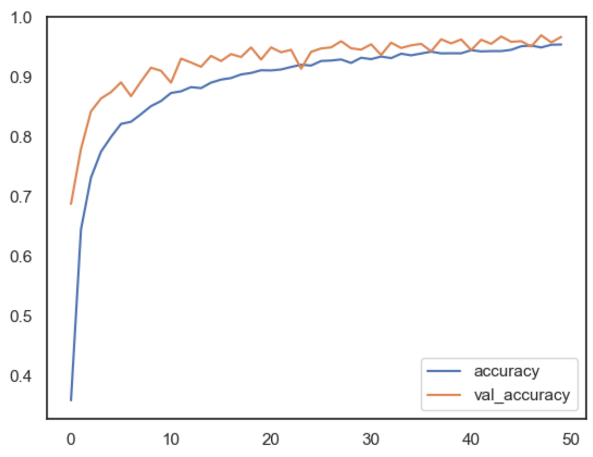


Figure 4.12: *Image of Learning curve graph*

The model is proven to be slightly over fitted as the value of the training and validations has overlapping segments from the learning curve graph and has difference of 1.49% in terms of accuracy. The model is not too over fitted and still can be considered as a reliable model and as the value of train accuracy is still below the validation accuracy.

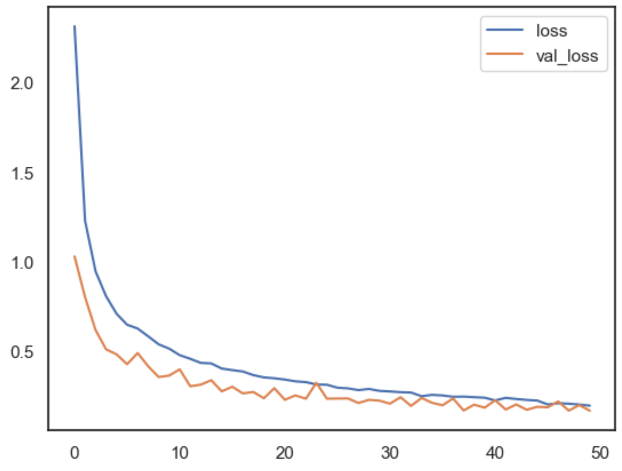
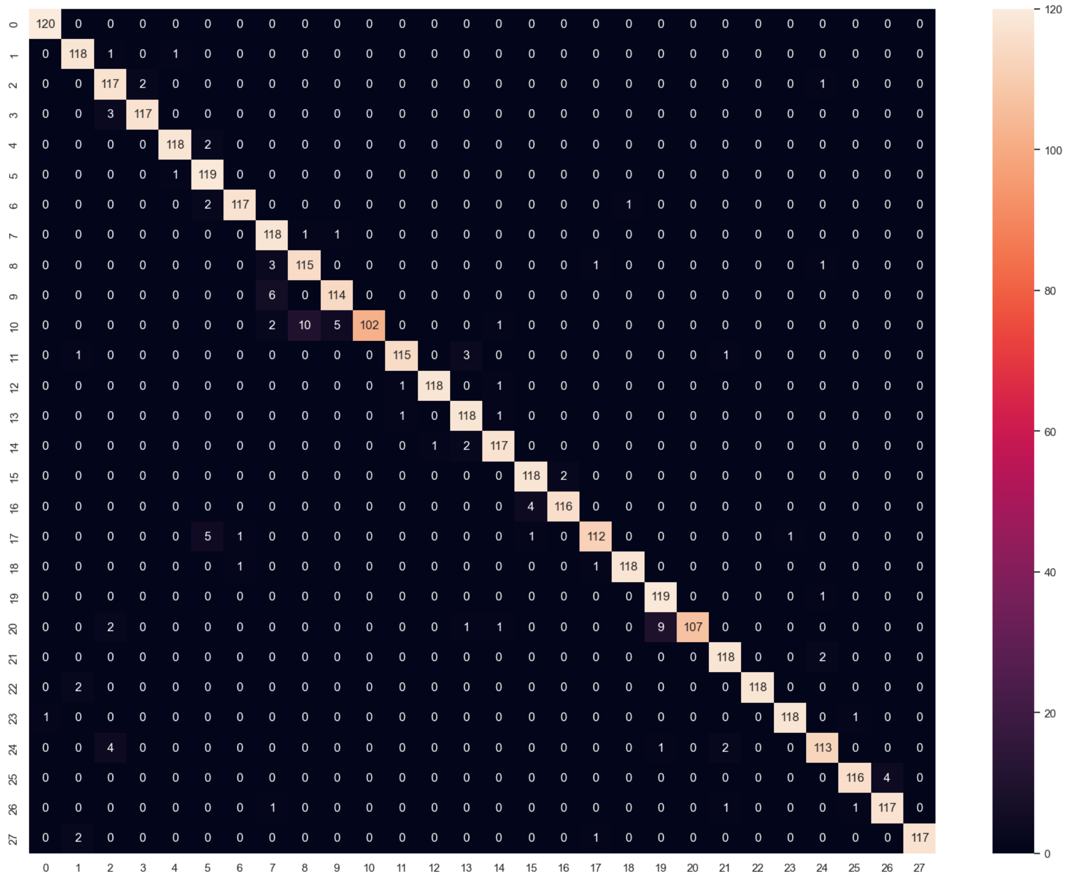


Figure 4.13: *Image of loss Learning curve graph*

The loss starts off with 236% at epoch 1 and gradually decreases until 21% at epoch 50. For validation loss it starts off with 111% at epoch 1 and gradually decreases until 16% at epoch 50. After the model training is done, it is then predicted using the testing set containing 3360 characters.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	118	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	117	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	3	117	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	118	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	119	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	2	117	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	118	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	115	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	6	0	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	2	10	5	102	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0	0	0	115	0	3	0	0	0	0	0	0	0	1	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	118	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0	118	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	2	117	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	2	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	116	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	1	0	112	0	0	0	0	0	1	0	0	0	0
18	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	118	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	0	0	0	1	0	0	0
20	0	0	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	9	107	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	0	0	2	0	0	0
22	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	0	1	0	0
24	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	113	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	116	4	0
26	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	117	0
27	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	117

Figure 4.14: *Confusion Matrix*

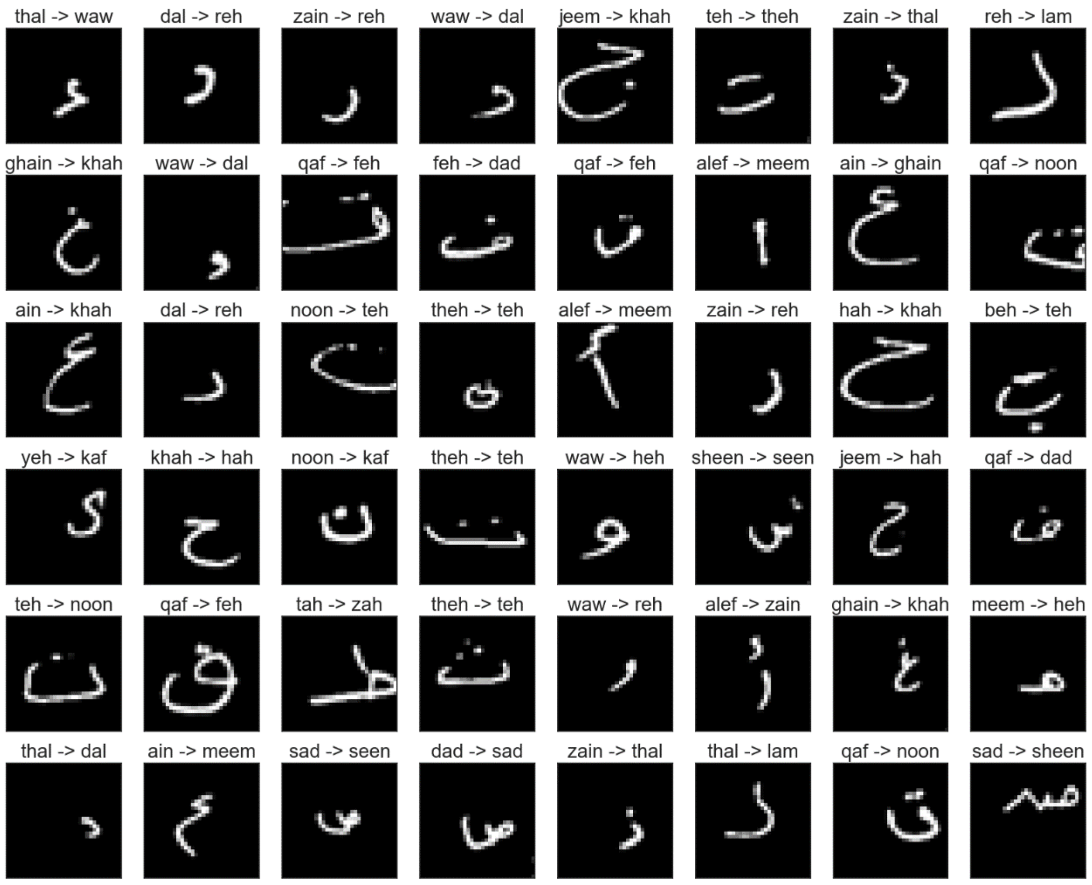


Figure 4.15: *Misclassification Analysis*

Through the above confusion matrix and misclassification analysis, it can be seen that there are a few classes that are predicted wrongly to its actual label but are classified differently. The newly predicted model achieved an accuracy of 96.37% for 50 epochs which is better than the previous CNN experiment and is good enough to classify the characters for each image.

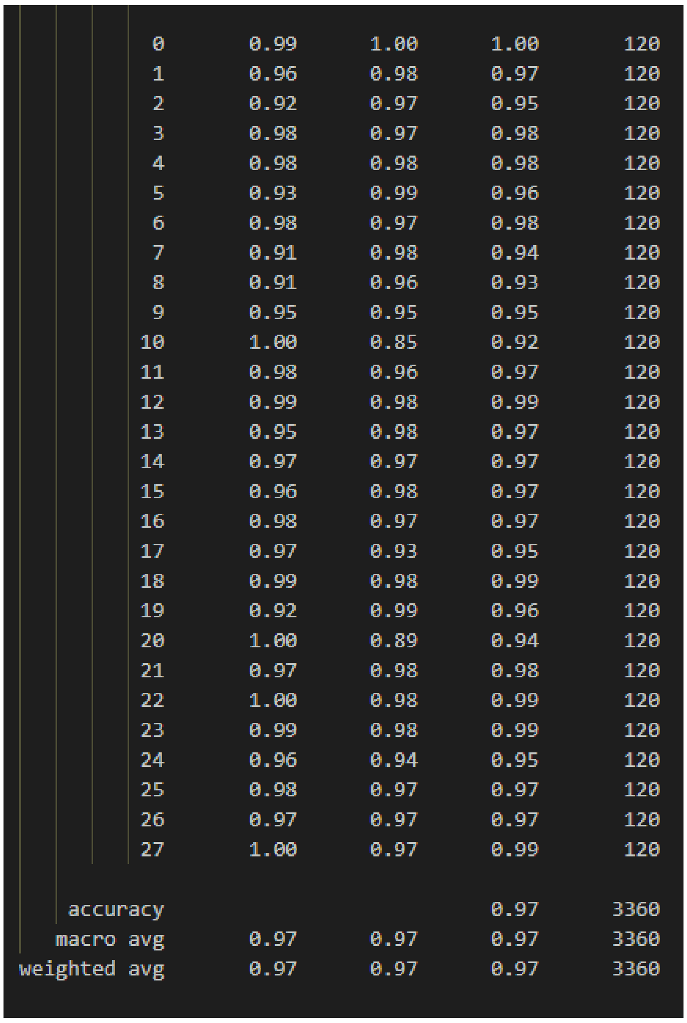

```
           0       0.99      1.00      1.00       120
           1       0.96      0.98      0.97       120
           2       0.92      0.97      0.95       120
           3       0.98      0.97      0.98       120
           4       0.98      0.98      0.98       120
           5       0.93      0.99      0.96       120
           6       0.98      0.97      0.98       120
           7       0.91      0.98      0.94       120
           8       0.91      0.96      0.93       120
           9       0.95      0.95      0.95       120
          10       1.00      0.85      0.92       120
          11       0.98      0.96      0.97       120
          12       0.99      0.98      0.99       120
          13       0.95      0.98      0.97       120
          14       0.97      0.97      0.97       120
          15       0.96      0.98      0.97       120
          16       0.98      0.97      0.97       120
          17       0.97      0.93      0.95       120
          18       0.99      0.98      0.99       120
          19       0.92      0.99      0.96       120
          20       1.00      0.89      0.94       120
          21       0.97      0.98      0.98       120
          22       1.00      0.98      0.99       120
          23       0.99      0.98      0.99       120
          24       0.96      0.94      0.95       120
          25       0.98      0.97      0.97       120
          26       0.97      0.97      0.97       120
          27       1.00      0.97      0.99       120

    accuracy                           0.97      3360
   macro avg       0.97      0.97      0.97      3360
weighted avg       0.97      0.97      0.97      3360
```

Figure 4.16: *Accuracy Report of CNN using 7 by 3 split (50 epochs)*

An accuracy report is generated and through all the letter, the average of the model's accuracy is obtained, and it can only predict 97% of the letter out of 100%. This model is already reliable since the performance of the model has already reached its requirement to classify the letters accurately. Most of the letters can be predicted easily even for those that have certain conditions or shapes that are likely to be similar will be able to detect its class.

4.2.7 Experiment 7: CNN model with 8 by 2 split using 30 epochs.

a) Model Configuration

The fifth experiment uses CNN technique for the model and the 8 by 2 training dataset which consists of 10,752 characters for training and 2688 for validation. The model uses 30 iterations for training.

Parameters	**Value**
Model	CNN
Dataset	10,752 training, 2688 validation, 3360 testing
Iterations (Epochs)	30
Learning Rate	0.001
Optimizer	Adam

Table 4.9: *Table of CNN parameters 8 by 2 split using 30 epochs*

b) Model Experiment Results

The model is trained using CNN layers and 30 iterations through the training and validation set. A result from training the fitted model can be shown as accuracy, loss, validation accuracy, and validation loss. The model finished training at 30 iterations with values such as 93.67% accuracy, 0.26% loss, 95.94% validation accuracy, and 0.19% validation loss.

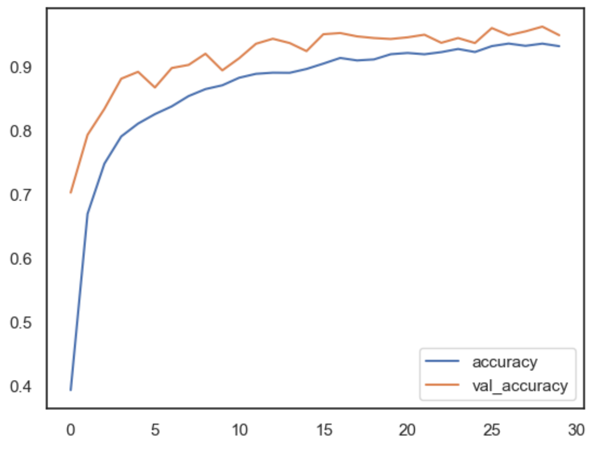


Figure 4.17: *Image of Learning curve graph*

The model is proven to be well fitted as the value of the training and validations is near with the difference of 2.27% in terms of accuracy but is lower than the previous experiments. The model is not over fitted as the value of train accuracy is still below the validation accuracy.

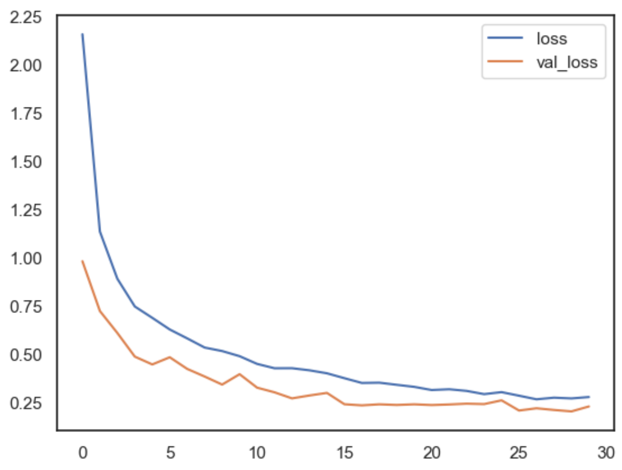
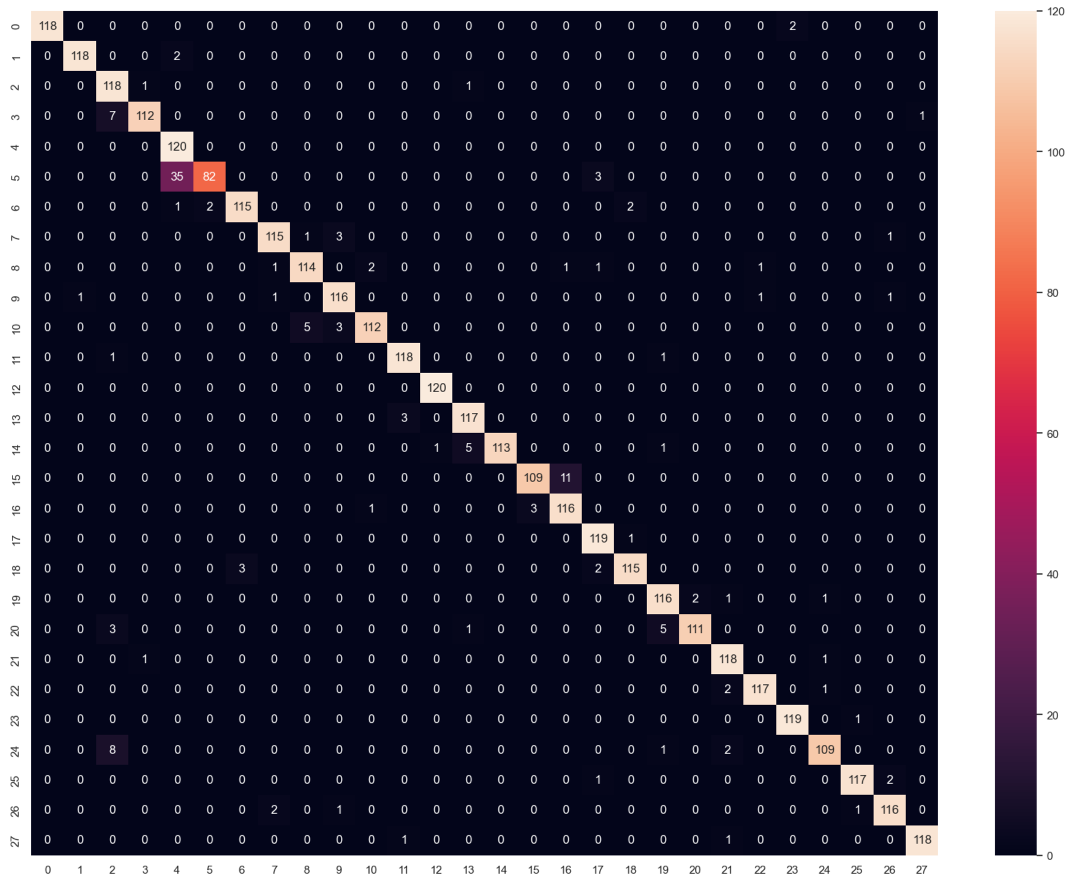


Figure 4.18: *Image of loss Learning curve graph*

The loss starts off with 216% at epoch 1 and gradually decreases until 26% at epoch 30. For validation loss it starts off with 95% at epoch 1 and gradually decreases until 19% at epoch 30. After the model training is done, it is then predicted using the testing set containing 3360 characters.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
1	0	118	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	118	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	7	112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	35	82	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	2	115	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	115	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1	114	0	2	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
9	0	1	0	0	0	0	0	1	0	116	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
10	0	0	0	0	0	0	0	0	5	3	112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	118	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	3	0	117	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	5	113	0	0	0	0	1	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	109	11	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3	116	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	1	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	2	115	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	116	2	1	0	0	1	0	0	0
20	0	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	5	111	0	0	0	0	0	0	0
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	0	0	1	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	117	0	1	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	1	0	0
24	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	109	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	117	2	0
26	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	116	0
27	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	118

Figure 4.19: *Confusion Matrix*

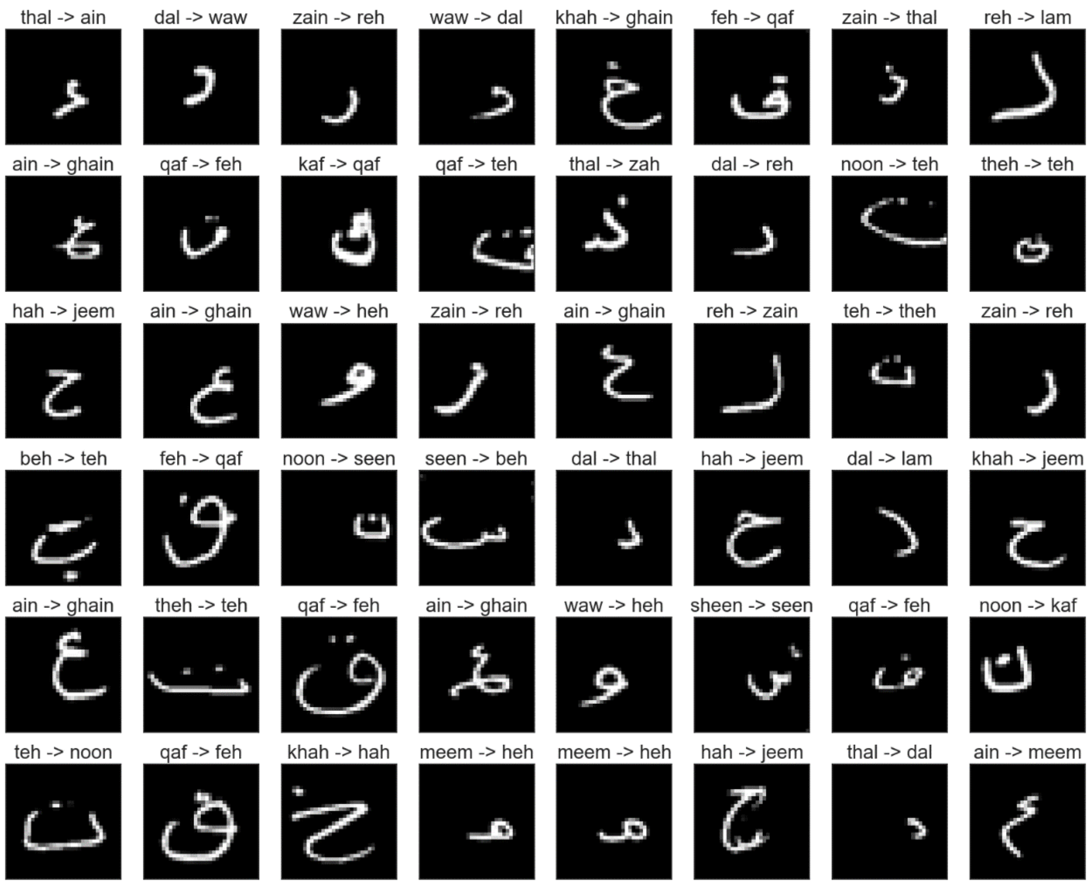
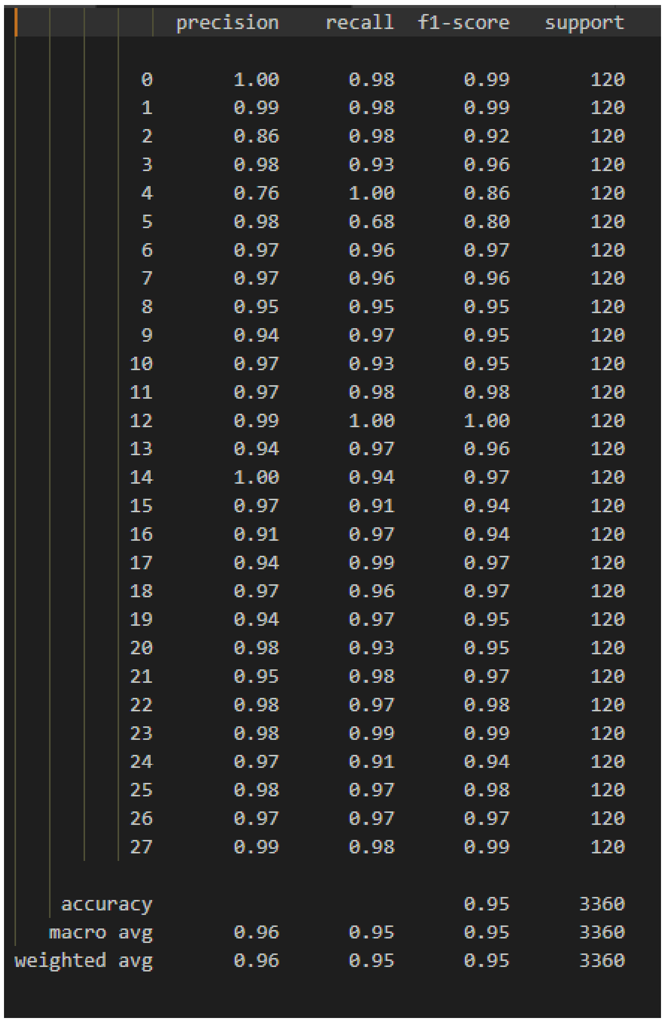


Figure 4.20: *Misclassification Analysis*

Through the above confusion matrix and misclassification analysis, it can be seen that there are a few classes that are predicted wrongly to its actual label but are classified differently. The class 'jeem' and 'hah' shows a high number of misclassification in the confusion matrix. The newly predicted model achieved an accuracy of 95.94% for 30 epochs which is lower than the previous CNN experiment and is good enough to classify the characters for each image.



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       120
           1       0.99      0.98      0.99       120
           2       0.86      0.98      0.92       120
           3       0.98      0.93      0.96       120
           4       0.76      1.00      0.86       120
           5       0.98      0.68      0.80       120
           6       0.97      0.96      0.97       120
           7       0.97      0.96      0.96       120
           8       0.95      0.95      0.95       120
           9       0.94      0.97      0.95       120
          10       0.97      0.93      0.95       120
          11       0.97      0.98      0.98       120
          12       0.99      1.00      1.00       120
          13       0.94      0.97      0.96       120
          14       1.00      0.94      0.97       120
          15       0.97      0.91      0.94       120
          16       0.91      0.97      0.94       120
          17       0.94      0.99      0.97       120
          18       0.97      0.96      0.97       120
          19       0.94      0.97      0.95       120
          20       0.98      0.93      0.95       120
          21       0.95      0.98      0.97       120
          22       0.98      0.97      0.98       120
          23       0.98      0.99      0.99       120
          24       0.97      0.91      0.94       120
          25       0.98      0.97      0.98       120
          26       0.97      0.97      0.97       120
          27       0.99      0.98      0.99       120

    accuracy                           0.95      3360
   macro avg       0.96      0.95      0.95      3360
weighted avg       0.96      0.95      0.95      3360
```

Figure 4.21: *Accuracy Report of CNN using 8 by 2 split (30 epochs)*

An accuracy report is generated and through all the letter, the average of the model's accuracy is obtained, and it can only predict 96% of the letter out of 100%. This model is already reliable since the performance of the model has already reached its requirement to classify the letters accurately. Most of the letters can be predicted easily even for those that have certain conditions or shapes that are likely to be similar will be able to detect its class.

4.2.8 Experiment 8: CNN model with 8 by 2 split using 50 epochs.

a) Model Configuration

The fifth experiment uses CNN technique for the model and the 8 by 2 training dataset which consists of 10,752 characters for training and 2688 for validation. The model uses 50 iterations for training.

Parameters	Value
Model	CNN
Dataset	10,752 training, 2688 validation, 3360 testing
Iterations (Epochs)	50
Learning Rate	0.001
Optimizer	Adam

Table 4.10: *Table of CNN parameters 8 by 2 split using 50 epochs*

b) Model Experiment Results

The model is trained using CNN layers and 50 iterations through the training and validation set. A result from training the fitted model can be shown as accuracy, loss, validation accuracy, and validation loss. The model finished training at 50 iterations with values such as 94.76% accuracy, 0.21% loss, 96.21% validation accuracy, and 0.17% validation loss.

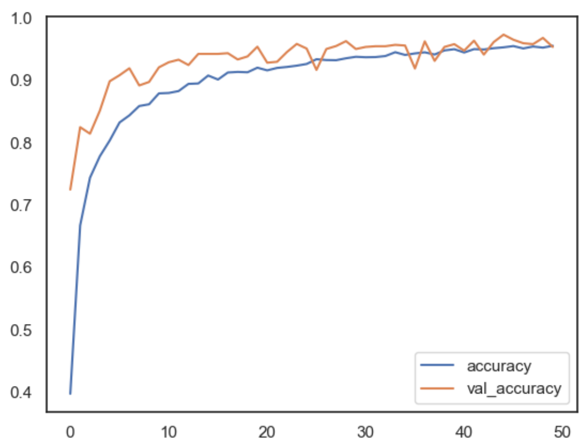


Figure 4.22: *Image of Learning curve graph*

The model is proven to be slightly over fitted as the value of the training and validations has overlapping segments from the learning curve graph and has difference of 1.45% in terms of accuracy. The model is not too over fitted and still can be considered as a reliable model and as the value of train accuracy is still below the validation accuracy.

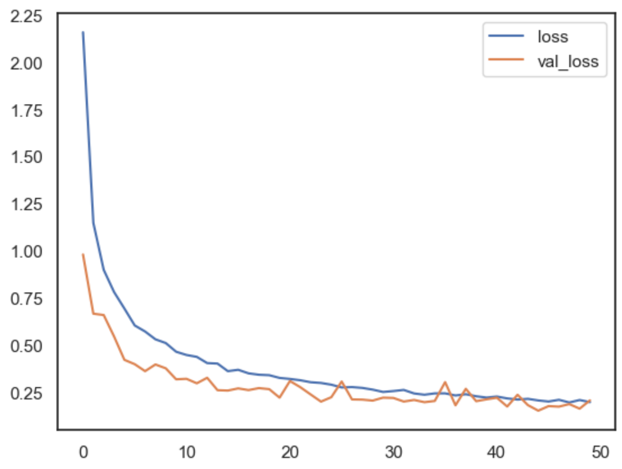
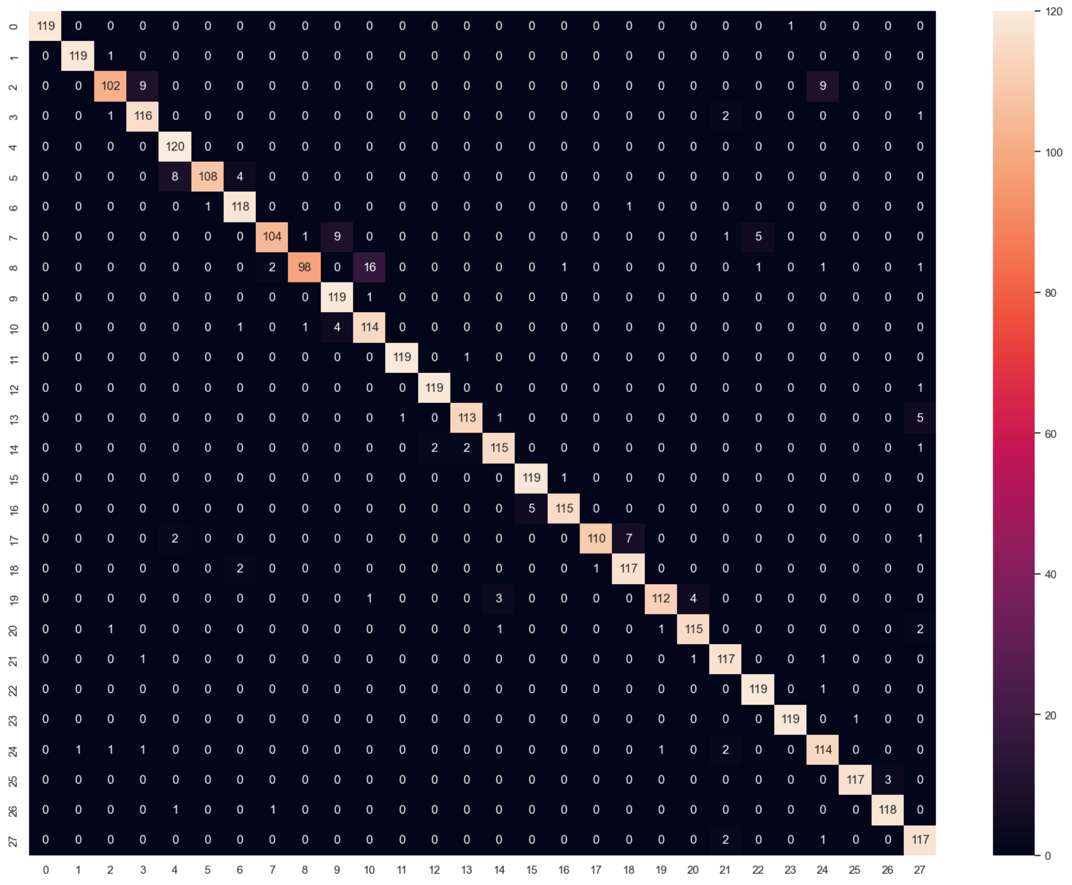


Figure 4.23: *Image of loss Learning curve graph*

The loss starts off with 228% at epoch 1 and gradually decreases until 21% at epoch 30. For validation loss it starts off with 106% at epoch 1 and gradually decreases until 17% at epoch 50. After the model training is done, it is then predicted using the testing set containing 3360 characters.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	119	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	119	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	102	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0
3	0	0	1	116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1
4	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	8	108	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	118	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	104	1	9	0	0	0	0	0	0	0	0	0	0	0	1	5	0	0	0	0	0
8	0	0	0	0	0	0	0	2	98	0	16	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1
9	0	0	0	0	0	0	0	0	0	119	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	1	4	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	119	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	119	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	1	0	113	1	0	0	0	0	0	0	0	0	0	0	0	0	5
14	0	0	0	0	0	0	0	0	0	0	0	0	2	2	115	0	0	0	0	0	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	1	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	115	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	110	7	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	1	117	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0	0	112	4	0	0	0	0	0	0	0
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	115	0	0	0	0	0	0	2
21	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	117	0	0	1	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	1	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	1	0	0
24	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	114	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	117	3	0
26	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	118	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	117

***Figure 4.24:** Confusion Matrix*

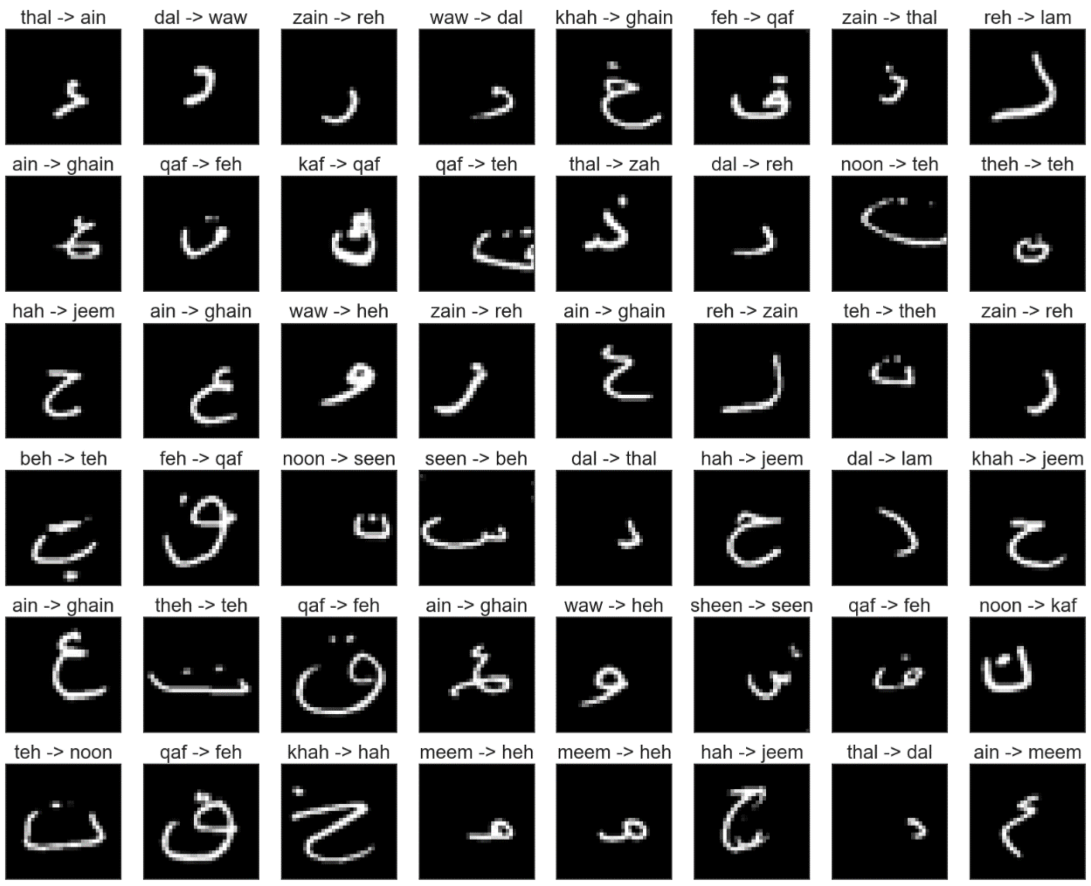
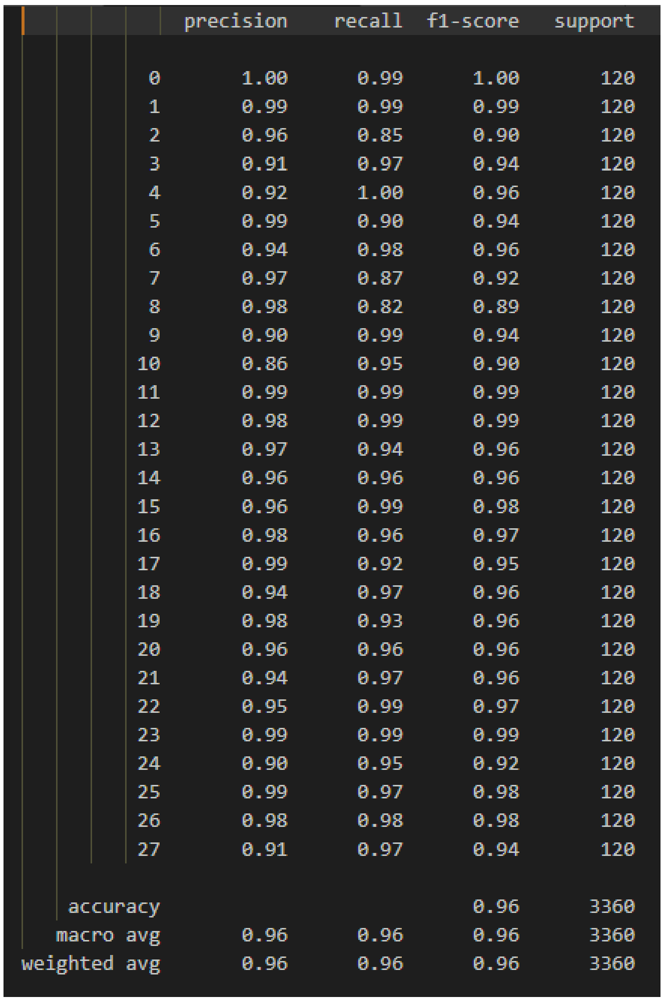


Figure 4.25: *Misclassification Analysis*

Through the above confusion matrix and misclassification analysis, it can be seen that there are a few classes that are predicted wrongly to its actual label but are classified differently. The class ‘jeem’ and ‘hah’ shows a lower number of misclassification in the confusion matrix compared to experiment 7 but there a few more classes that are now misclassified. The newly predicted model achieved an accuracy of 96.20% for 50 epochs which is higher than the previous CNN experiment and is good enough to classify the characters for each image.



```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       120
           1       0.99      0.99      0.99       120
           2       0.96      0.85      0.90       120
           3       0.91      0.97      0.94       120
           4       0.92      1.00      0.96       120
           5       0.99      0.90      0.94       120
           6       0.94      0.98      0.96       120
           7       0.97      0.87      0.92       120
           8       0.98      0.82      0.89       120
           9       0.90      0.99      0.94       120
          10       0.86      0.95      0.90       120
          11       0.99      0.99      0.99       120
          12       0.98      0.99      0.99       120
          13       0.97      0.94      0.96       120
          14       0.96      0.96      0.96       120
          15       0.96      0.99      0.98       120
          16       0.98      0.96      0.97       120
          17       0.99      0.92      0.95       120
          18       0.94      0.97      0.96       120
          19       0.98      0.93      0.96       120
          20       0.96      0.96      0.96       120
          21       0.94      0.97      0.96       120
          22       0.95      0.99      0.97       120
          23       0.99      0.99      0.99       120
          24       0.90      0.95      0.92       120
          25       0.99      0.97      0.98       120
          26       0.98      0.98      0.98       120
          27       0.91      0.97      0.94       120

    accuracy                           0.96      3360
   macro avg       0.96      0.96      0.96      3360
weighted avg       0.96      0.96      0.96      3360
```

Figure 4.26: *Accuracy Report of CNN using 8 by 2 split (50 epochs)*

An accuracy report is generated and through all the letter, the average of the model's accuracy is obtained, and it can only predict 96% of the letter out of 100%. This model is already reliable since the performance of the model has already reached its requirement to classify the letters accurately. Most of the letters can be predicted easily even for those that have certain conditions or shapes that are likely to be similar will be able to detect its class.

4.3 Model Experiment Comparison

Model	Epochs / Trees	Dataset split	Accuracy
CNN	30	7:3	96.18%
	50	**7:3**	**96.37%**
	30	8:2	95.94%
	50	8:2	96.20%
SVM	-	7:3	45.36%
SVM	-	8:2	44.64%
RandomForest	100	7:3	70.23%
RandomForest	500	7:3	72.14%
RandomForest	100	8:2	71.68%
RandomForest	500	8:2	73.32%

Figure 4.11: *Table of Model Comparison*

Based on all the model experiments that has been done, it can be seen that the CNN method produces the best accuracies among SVM and RandomForest. SVM obtains a relatively low accuracy due to the fact the dataset has a large number of images. SVM works better with a smaller dataset. Out of all the CNN model, it is clear that the model that has the highest accuracy is the model with 50 epoch and 7 by 3 split obtaining an accuracy of 96.37%.

4.4 System Testing

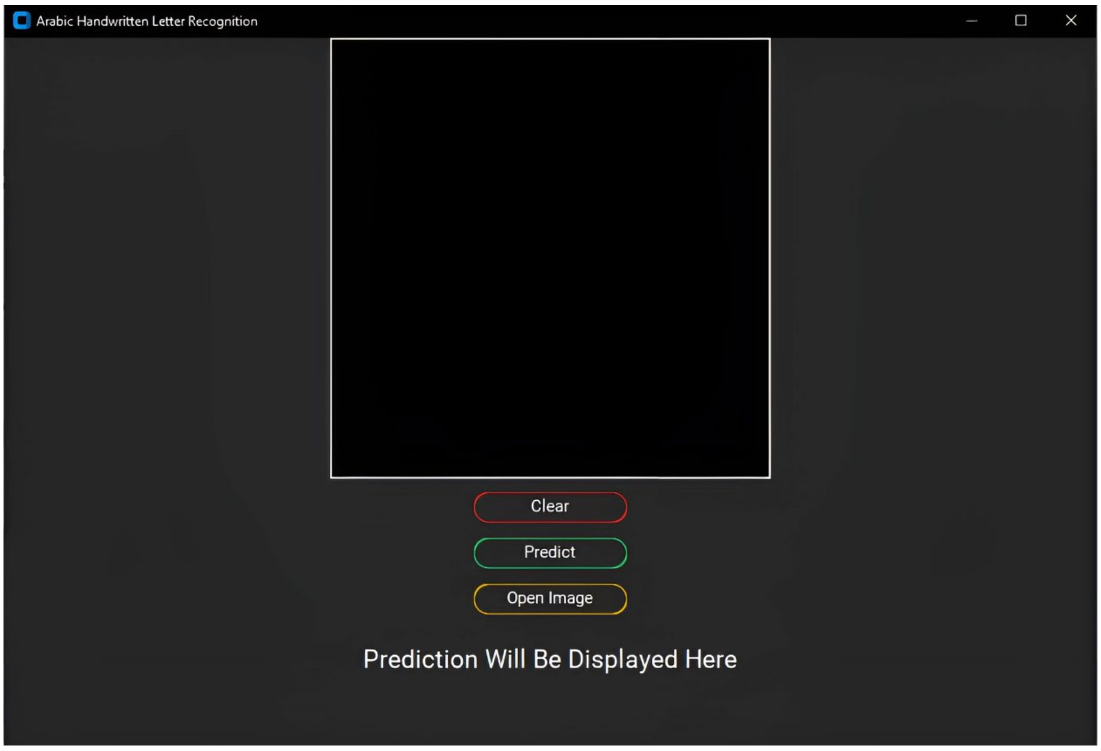
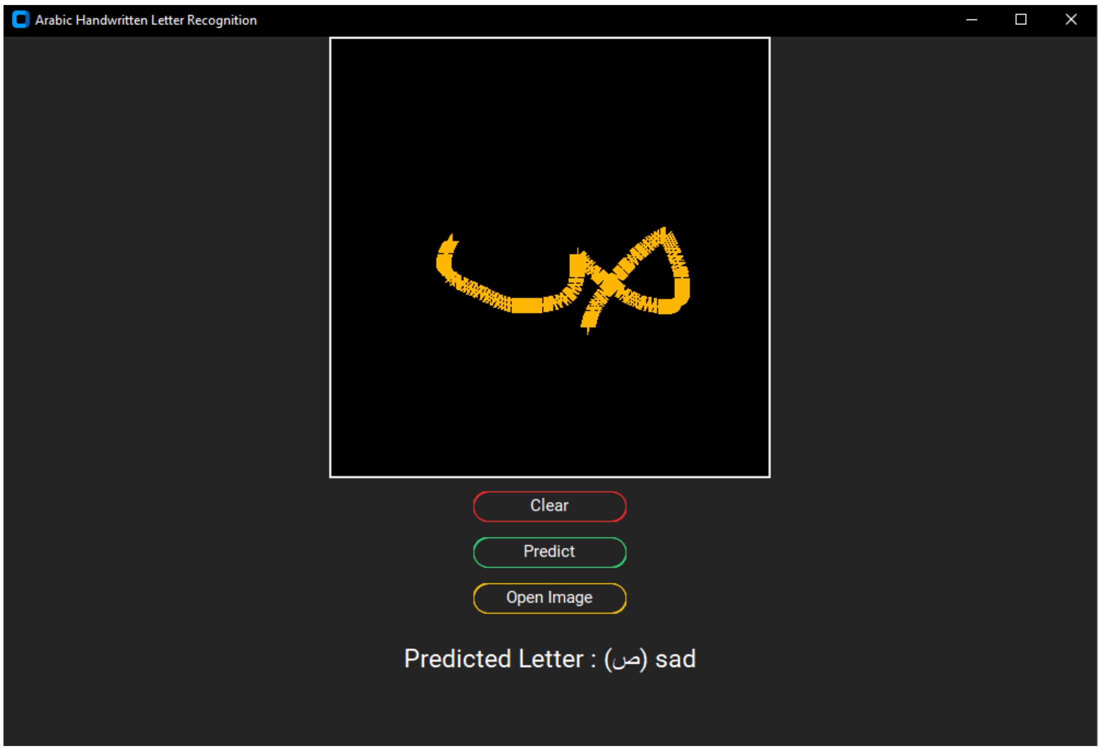

Arabic Handwritten Letter Recognition
Clear
Predict
Open Image
Prediction Will Be Displayed Here

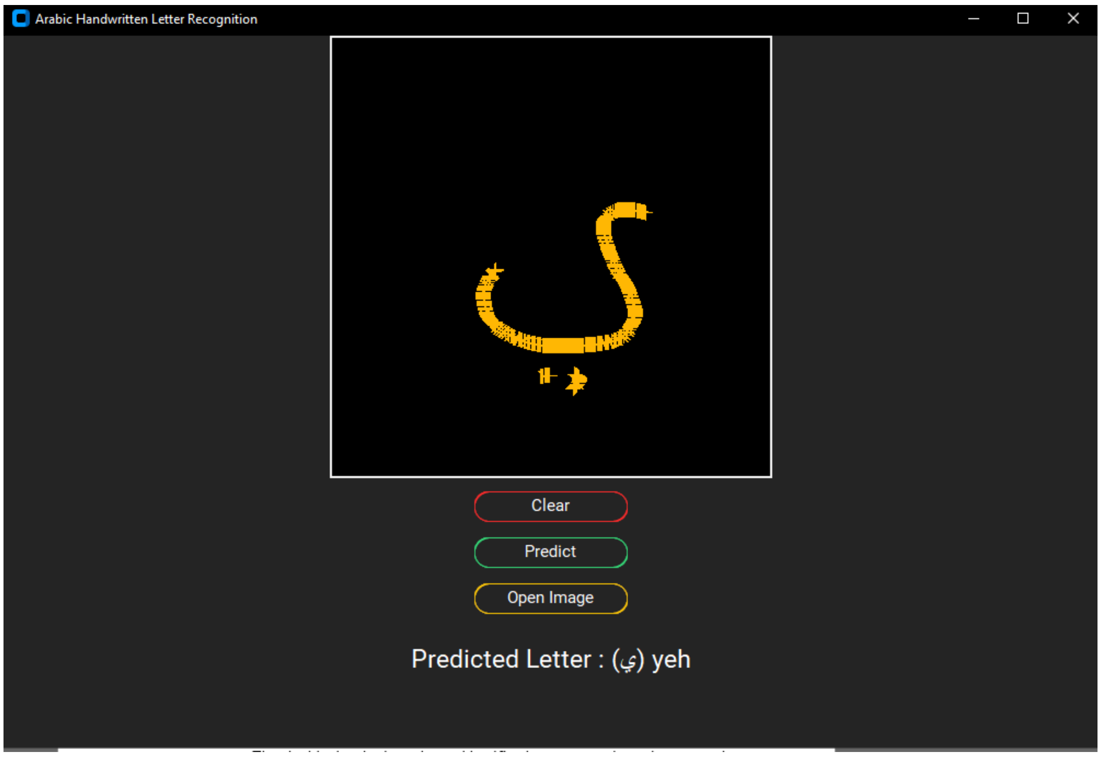
Figure 4.27: *Image of Arabic Handwritten Letter Recognition System*

Figure 4.24 shows the interface of the system that has been developed to access in accepting handwritten images and live handwritten drawings of Arabic letters from user. The system is integrated with the chosen model which is CNN that uses 7 by 3 split dataset with 50 iterations. The system allows user to draw the shape of Arabic letters on the black canvas in the system and the features of the letters will be transformed into image form through the backend coding of Python together with Tkinter library. The button "Clear" is used to erase any drawing that is drawn on the canvas. Next, the button "predict" is used to predict the drawn Arabic letter using the system prediction model. Other than that, the "Open Image" button is used to insert image from the file directories. Overall testing has been done to the system and most of the letters drawn are classified correctly but some tend to be misclassified if the letters are drawn at a different angle.



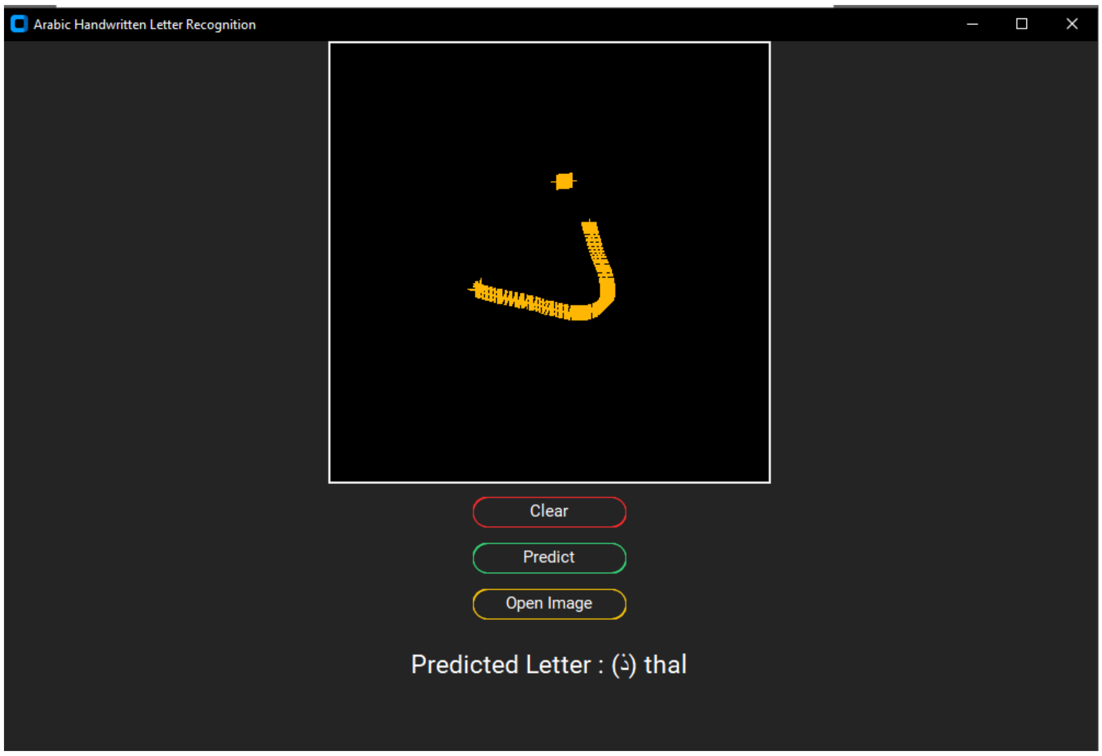
Predicted Letter : (ص) sad

Figure 4.28: *Image of prediction for letter Sad*



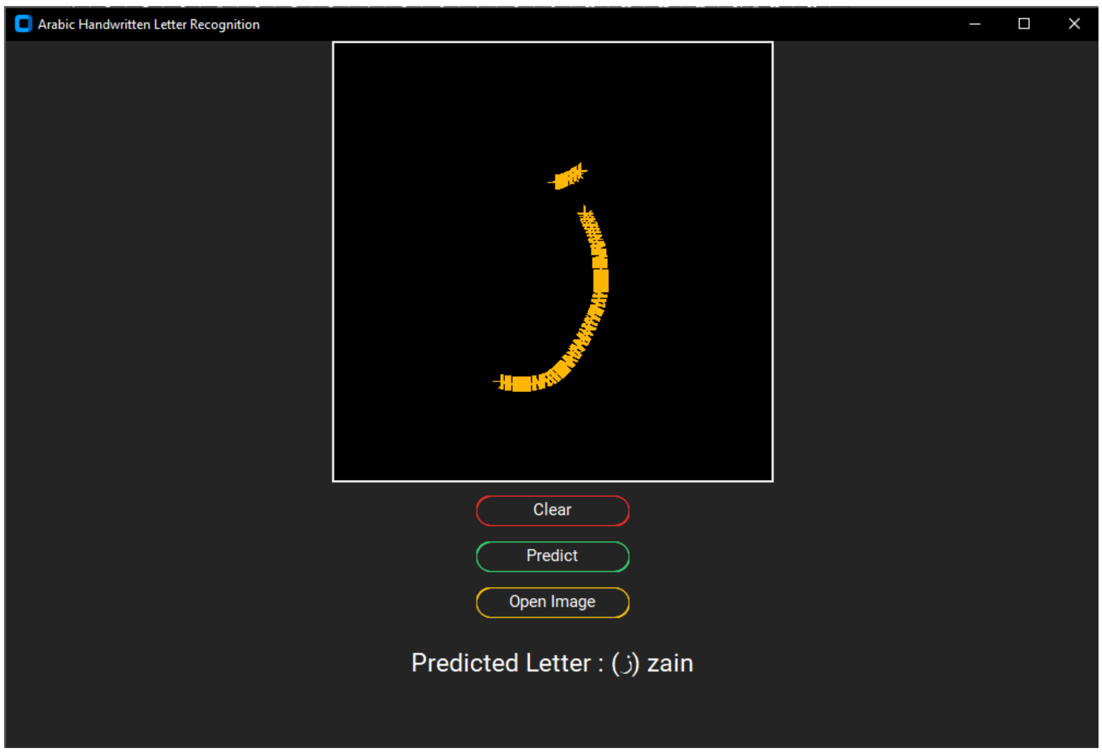
Predicted Letter : (ي) yeh

Figure 4.29: *Image of prediction for letter Yeh*



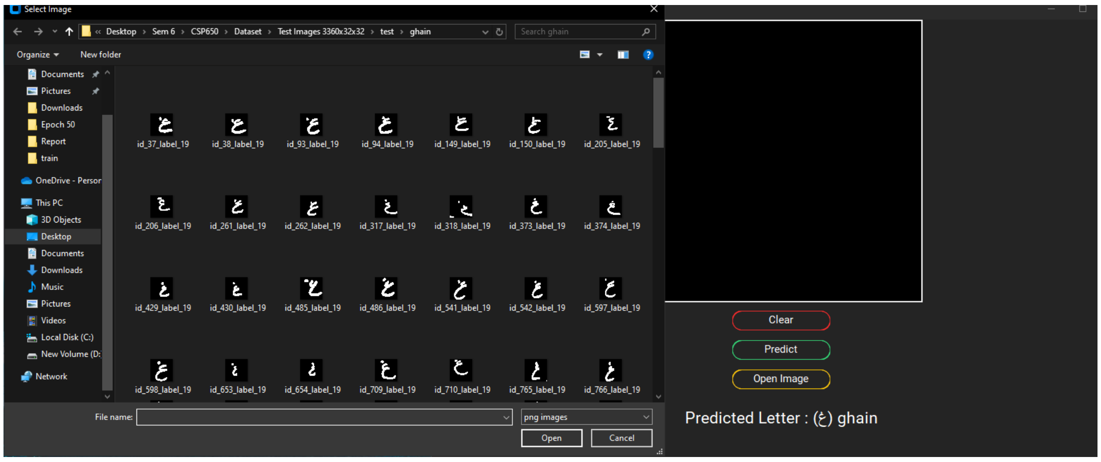
Predicted Letter : (ذ) thal

Figure 4.30: *Image of prediction for letter Thal*



Predicted Letter : (ز) zain

Figure 4.31: *Image of prediction for letter Zain*



Predicted Letter : (غ) ghain

Figure 4.32: *Image of prediction using "Open Image" button*

4.5 Summary

The Arabic handwritten letter identification system showed encouraging outcomes and perceptive conclusions during its assessment. The algorithm demonstrated an impressive level of accuracy in recognising and categorising handwritten Arabic letters after undergoing extensive testing and analysis. The precision, recall, and F1 score, which are measures of overall performance, demonstrated how well the system handled a variety of writing styles and variants.

A noteworthy discovery was that the system could adjust to various handwriting subtleties, which added to its adaptability and usefulness. A key factor in improving the system's performance was the integration of cutting-edge machine learning algorithms and feature extraction methods. The study also identified important issues, like handling ambiguity in certain letter forms and optimising for real-world situations with different writing settings.

CHAPTER 5

CONCLUSION, LIMITATION AND FUTURE WORKS

The results of the experimentation, the conclusion, any project constraints, and the work on future trends were all compiled into this chapter.

5.1 Conclusion

Identifying the unique characteristics and handwriting styles connected to every Arabic letter was the main goal of this study. With the goal of capturing the subtle nuances of Arabic writing, a state-of-the-art machine learning model was created utilising convolutional neural networks (CNNs). In addition to identification, the model's performance in Arabic letter handwriting recognition was evaluated through the creation of a thorough assessment framework.

A strong prototype that can correctly recognise and classify Arabic letters was produced as a consequence of these efforts coming together. This prototype is proof of the effective fusion of sophisticated machine learning methods, feature identification, and real-world CNN implementation to acquire accurate Arabic handwritten letter recognition.

The Arabic handwritten letter recognition system performed well, correctly recognising and categorising handwritten Arabic letters with a high degree of accuracy. Notable strengths included its capacity to adapt to various handwriting styles and its efficient use of sophisticated machine learning algorithms. Because of its stability, the system can be used in real-world settings for text digitization, document analysis, and educational tools. Regardless of several obstacles were detected, like dealing with vagueness in specific letter forms, indicating potential directions for further improvement. All things considered; the method has potential to further Arabic handwritten letter recognition.

5.2 Limitations

Even though Arabic handwritten letter recognition algorithms have advanced significantly, there are still certain inherent drawbacks that need be taken into account. The complexity and diversity of Arabic calligraphy, with its numerous styles and variants, is a significant difficulty. Handwritten letters that differ significantly from normal forms may be difficult for the algorithm to read correctly, which could result in incorrect classifications.

Furthermore, the quality of the input data may have an impact on the system's performance, especially if the language is unintelligible or poorly written. Ambiguities in certain letter forms and contextual differences present additional difficulties that could result in recognition errors. Limited training datasets may further limit the system's efficacy by making it less flexible to handle a wider variety of handwriting styles.

The system's sensitivity to noise and distortions, which can result from smearing, incomplete scanning, or other external circumstances, is another significant drawback. All of these difficulties highlight the necessity of continued study and improvement to deal with the complexities of Arabic calligraphy and improve the system's resilience when processing a variety of handwritten inputs from the real world.

The difficulty of the Arabic handwritten letter recognition system to process raw photos directly without requiring preprocessing places limitations on it. This restriction suggests that before identification can occur, input photos must go through particular preprocessing processes. This could introduce another step into the workflow and reduce the system's ability to handle raw or sparingly processed image data.

5.3 Future Works

Looking forward, the evolution of Arabic handwritten letter recognition systems could benefit from several strategic future works and upgrades. One key focus lies in refining the system's capacity to handle noisy and varied input, allowing it to excel in real-world conditions where handwriting quality may differ. Incorporating self-learning mechanisms and continuous training approaches could contribute to the system's adaptability and resilience over time, accommodating dynamic changes in writing patterns.

Moreover, exploring interpretability features within the model architecture could enhance transparency and provide valuable insights into the decision-making process, fostering trust and usability. Collaboration with linguists and cultural experts could further enrich the system's understanding of context-specific nuances, ensuring accurate recognition in diverse linguistic contexts.

The integration of user feedback mechanisms and iterative model updates is essential for maintaining relevance and effectiveness. As data continues to evolve, staying abreast of emerging handwriting trends and continually updating the system's database will be crucial for sustained accuracy.

Efforts can also be directed towards making the system more accessible, perhaps by developing user-friendly dashboard interfaces or mobile applications, expanding its reach and impact across various user demographics. Other than that, the system could potentially be open to more recognition types such as Aerial letter recognition, where users use hand gestures to draw a certain letter.

REFERENCES

Abu, M. A., Hazirah Indra, N., Halim, A., Rahman, A., Sapiee, A., & Ahmad, I. (2019). A study on Image Classification based on Deep Learning and Tensorflow. *International Journal of Engineering Research and Technology*, *12*(4), 563–569. http://www.irphouse.com563

Ali, A. A. A., & Suresha, M. (2019). Arabic Handwritten Character Recognition Using Machine Learning Approaches. *Proceedings of the IEEE International Conference Image Information Processing*, *2019-November*, 187–192. https://doi.org/10.1109/ICIIP47207.2019.8985839

Altwaijry, N., & Al-Turaiki, I. (2021a). Arabic handwriting recognition system using convolutional neural network. *Neural Computing and Applications*, *33*(7), 2249–2261. https://doi.org/10.1007/S00521-020-05070-8/TABLES/3

Altwaijry, N., & Al-Turaiki, I. (2021b). Arabic handwriting recognition system using convolutional neural network. *Neural Computing and Applications*, *33*(7), 2249–2261. https://doi.org/10.1007/S00521-020-05070-8/TABLES/3

Alzrrog, N., Bousquet, J. F., & El-Feghi, I. (2022). Deep Learning Application for Handwritten Arabic Word Recognition. *Canadian Conference on Electrical and Computer Engineering*, *2022-September*, 95–100. https://doi.org/10.1109/CCECE49351.2022.9918375

Amrouch, M., Rabi, M., & Es-Saady, Y. (2018). Convolutional feature learning and CNN based HMM for Arabic handwriting recognition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10884 LNCS*, 265–274. https://doi.org/10.1007/978-3-319-94211-7_29/TABLES/4

Biswas, A., & Islam, M. S. (2021). Hybrid CNN-SVM Model for Brain Tumor Classification utilizing Different Datasets. *Proceedings of International Conference on Electronics, Communications and Information Technology, ICECIT 2021*. https://doi.org/10.1109/ICECIT54077.2021.9641201

Chychkarov, Y., Serhiienko, A., Syrmamiikh, I., & Kargin, A. (2021). *Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks*.

Djaghbellou, S., Akhtar, Z., Bouziane, A., & Attia, A. (2020). ARABIC HANDWRITTEN CHARACTERS RECOGNITION VIA MULTI-SCALE HOG FEATURES AND MULTI-LAYER DEEP RULE-BASED CLASSIFICATION. *ICTACT JOURNAL ON IMAGE AND VIDEO PROCESSING*, *10*, 4. https://doi.org/10.21917/ijivp.2020.0315

Dutta, K. K., Sunny, S. A., Victor, A., Nathu, A. G., Ayman Habib, M., & Parashar, D. (2020). Kannada alphabets recognition using decision tree and random forest models. *Proceedings of the 3rd International Conference on Intelligent*

Sustainable Systems, ICISS 2020, 534–541. https://doi.org/10.1109/ICISS49785.2020.9315972

Dutta, K. K., Swamy, S. A., Banerjee, A., Divya Rashi, B., Chandan, R., & Vaprani, D. (2021). Kannada character recognition using multi-class SVM method. *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*, 405–409. https://doi.org/10.1109/CONFLUENCE51648.2021.9376883

Ezz, M., Sharaf, M. A., & Hassan, A. A. A. (2019). Classification of Arabic Writing Styles in Ancient Arabic Manuscripts. *International Journal of Advanced Computer Science and Applications*, *10*(10), 409–414. https://doi.org/10.14569/IJACSA.2019.0101056

Ghanim, T. M., Khalil, M. I., & Abbas, H. M. (2020). Comparative Study on Deep Convolution Neural Networks DCNN-Based Offline Arabic Handwriting Recognition. *IEEE Access*, *8*, 95465–95482. https://doi.org/10.1109/ACCESS.2020.2994290

Gontumukkala, S. S. T., Godavarthi, Y. S. V., Gonugunta, B. R. R. T., Subramani, R., & Murali, K. (2021). Analysis of Image Classification using SVM. *2021 12th International Conference on Computing Communication and Networking Technologies, ICCCNT 2021*. https://doi.org/10.1109/ICCCNT51525.2021.9579803

Gupta, N., & Goyal, N. (2021). Machine Learning Tensor Flow Based Platform for Recognition of HandWritten Text. *2021 International Conference on Computer Communication and Informatics, ICCCI 2021*. https://doi.org/10.1109/ICCCI50826.2021.9402622

Janardhanan, P. S. (2020). Project repositories for machine learning with TensorFlow. *Procedia Computer Science*, *171*, 188–196. https://doi.org/10.1016/J.PROCS.2020.04.020

Joe, K. G., Savit, M., & Chandrasekaran, K. (2019). Offline Character recognition on Segmented Handwritten Kannada Characters. *2019 Global Conference for Advancement in Technology, GCAT 2019*. https://doi.org/10.1109/GCAT47503.2019.8978320

Juan, J., & Revuelta, S. (2021). *Continuous Offline Handwriting Recognition using Deep Learning Models*. https://arxiv.org/abs/2112.13328v1

Juliana, H. Q. A., Sakundarini, N., & Seong, L. C. (2022). Classification of Contaminants in Glass Recycling Using Hybrid CNN-SVM Model. *4th IEEE International Conference on Artificial Intelligence in Engineering and Technology, IICAIET 2022*. https://doi.org/10.1109/IICAIET55139.2022.9936857

Khan, S., Hafeez, A., Ali, H., Nazir, S., & Hussain, A. (2020). Pioneer dataset and recognition of Handwritten Pashto characters using Convolution Neural

Networks. *Measurement and Control*, *53*(9), 2041–2054. https://doi.org/10.1177/0020294020964826

Kushwaha, P. K., & Kumaresan, M. (2021). Machine learning algorithm in healthcare system: A Review. *Proceedings of International Conference on Technological Advancements and Innovations, ICTAI 2021*, 478–481. https://doi.org/10.1109/ICTAI53825.2021.9673220

Li, Z. (n.d.). *Design of Online Handwriting Recognition System Using Simplified CNN Approach*. https://doi.org/10.2139/SSRN.4382864

Liu, Y., Dou, Y., & Qiao, P. (2020). Beyond top-N accuracy indicator: a comprehensive evaluation indicator of CNN models in image classification. *IET Computer Vision*, *14*(6), 407–414. https://doi.org/10.1049/IET-CVI.2018.5839

Maalej, R., & Kherallah, M. (2019). Convolutional Neural Network and BLSTM for Offline Arabic Handwriting Recognition. *ACIT 2018 - 19th International Arab Conference on Information Technology*. https://doi.org/10.1109/ACIT.2018.8672667

Madireddy, V. R. (2018). Content Based Image Classification Using Support Vector Machine Algorithm. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.4289407

Punia, A. (2023). Recognition of Handwritten Character using Recognition Model based on SVM. *7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), I-SMAC 2023 - Proceedings*, 444–449. https://doi.org/10.1109/I-SMAC58438.2023.10290409

Ramesh, G., Sharma, G. N., Balaji, J. M., & Champa, H. N. (2019). Offline kannada handwritten character recognition using convolutional neural networks. *2019 5th IEEE International WIE Conference on Electrical and Computer Engineering, WIECON-ECE 2019 - Proceedings*. https://doi.org/10.1109/WIECON-ECE48653.2019.9019914

Roy, S. K., Krishna, G., Dubey, S. R., & Chaudhuri, B. B. (2020). HybridSN: Exploring 3-D-2-D CNN Feature Hierarchy for Hyperspectral Image Classification. *IEEE Geoscience and Remote Sensing Letters*, *17*(2), 277–281. https://doi.org/10.1109/LGRS.2019.2918719

Sakthimohan, M., Elizabeth Rani, G., Naveneethakrishnan, M., Abinaya, M., Karthigadevi, K., & Siddhartha, P. V. (2023). Digit Recognition of MNIST Handwritten Using Convolutional Neural Networks (CNN). *Proceedings of the 2023 International Conference on Intelligent Systems for Communication, IoT and Security, ICISCoIS 2023*, 328–332. https://doi.org/10.1109/ICISCOIS56541.2023.10100602

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3), 1–21. https://doi.org/10.1007/S42979-021-00592-X/FIGURES/11

Shamim, S. M., Badrul, M., Miah, A., & Sarker, A. (2018). Handwritten Digit Recognition Using Machine Learning Algorithms. *Article in Indonesian Journal of Science and Technology*. https://doi.org/10.17509/ijost.v3i1.10795

Sheykhmousa, M., Mahdianpari, M., Ghanbari, H., Mohammadimanesh, F., Ghamisi, P., & Homayouni, S. (2020). Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 6308–6325. https://doi.org/10.1109/JSTARS.2020.3026724

Singh, S., Sharma, A., & Chauhan, V. K. (2021). Online handwritten Gurmukhi word recognition using fine-tuned Deep Convolutional Neural Network on offline features. *Machine Learning with Applications*, *5*, 100037. https://doi.org/10.1016/J.MLWA.2021.100037

Sokar, G., Hemayed, E. E., & Rehan, M. (2019). A Generic OCR Using Deep Siamese Convolution Neural Networks. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, 1238–1244. https://doi.org/10.1109/IEMCON.2018.8614784

Tran, H. P., Smith, A., & Dimla, E. (2019). Offline Handwritten Text Recognition using Convolutional Recurrent Neural Network. *Proceedings - 2019 International Conference on Advanced Computing and Applications, ACOMP 2019*, 51–56. https://doi.org/10.1109/ACOMP.2019.00015

Tumpa, E. S., & Dey, K. (2022). A Review on Applications of Machine Learning in Healthcare. *2022 6th International Conference on Trends in Electronics and Informatics, ICOEI 2022 - Proceedings*, 1388–1392. https://doi.org/10.1109/ICOEI53556.2022.9776844

Ullah, Z., & Jamjoom, M. (2022). An intelligent approach for Arabic handwritten letter recognition using convolutional neural network. *PeerJ Computer Science*, *8*, e995. https://doi.org/10.7717/peerj-cs.995

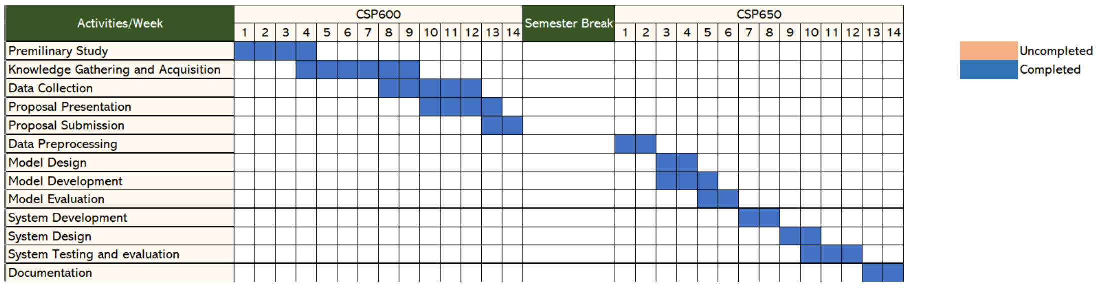
Xu, X., Zhou, J., Zhang, H., & Fu, X. (2018). Chinese characters recognition from screen-rendered images using inception deep learning architecture. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10735 LNCS*, 722–732. https://doi.org/10.1007/978-3-319-77380-3_69

Yadav, H., & Jain, S. (2019). Offline Handwritten Character Recognition using Neural Networks. *International Journal of Computer Sciences and Engineering*, *7*(5), 838–845. https://doi.org/10.26438/IJCSE/V7I5.838845

Yassin, R., Share, D. L., & Shalhoub-Awwad, Y. (2020). Learning to Spell in Arabic: The Impact of Script-Specific Visual-Orthographic Features. *Frontiers in Psychology*, *11*, 518168. https://doi.org/10.3389/FPSYG.2020.02059/BIBTEX

Youssef, N. I., & Abd-Alsabour, N. (2022). A REVIEW ON ARABIC HANDWRITING RECOGNITION. *Journal of Southwest Jiaotong University*, *57*(6), 745–764. https://doi.org/10.35741/ISSN.0258-2724.57.6.66

APPENDICES



Activities/Week	CSP600														Semester Break	CSP650													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Premilinary Study	🟦	🟦	🟦	🟦																									
Knowledge Gathering and Acquisition				🟦	🟦	🟦	🟦	🟦	🟦																				
Data Collection								🟦	🟦	🟦	🟦	🟦																	
Proposal Presentation										🟦	🟦	🟦	🟦																
Proposal Submission													🟦	🟦															
Data Preprocessing																🟦	🟦												
Model Design																		🟦	🟦										
Model Development																		🟦	🟦	🟦									
Model Evaluation																				🟦	🟦								
System Development																						🟦	🟦						
System Design																								🟦	🟦				
System Testing and evaluation																									🟦	🟦	🟦		
Documentation																												🟦	🟦

🟧	Uncompleted
🟦	Completed

Figure 6.1: *Gant Chart for FYP*