Deep Learning Applications to Offline Arabic Handwriting Words Recognition Using Convolutional Neural Network

by

Nori Alzrrog

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
January 2023

DEDICATION PAGE

I dedicate this thesis to God Almighty my creator, my strong pillar, my source of inspiration, wisdom, knowledge and understanding. He has been the source of my strength throughout this program. I also dedicate this dissertation to the memory of my parents, my wife Najat Musa, and Prof. Mohamed El-Hawary. Although they were my inspiration to pursue my doctoral degree, they were unable to see my graduation. This is for them. This dissertation is dedicated to my wife Rowaeda Mosa who encouraged me to pursue my dreams and finish my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Automatic handwriting recognition is the process of converting online and offline letters or words as a graphical form into its text format. Automatic Arabic Handwriting words recognition using deep learning neural networks is still in the early stages in terms of research. There are no general, complete, and reliable Arabic Handwritten Words (AHW) database (lexicon) that can be used as a reference or a benchmark for all researchers who want to extend the work on automatic Arabic handwriting word recognition. Also, many historic Arabic manuscripts have deteriorated because of inappropriate storage and most of them have not been digitized due to the lack of reliable database that can be used to recognize the words of Arabic manuscripts.

Deep Convolutional Neural Networks (DCNNs) can be used to solve the problems of automatic Arabic handwriting words recognition. In this work, a new DCNN algorithm applied to a new dataset of Handwritten Arabic words representing the seven days of the week named Arabic Handwritten Weekdays Dataset (AHWD) has been programmed, tested, and analyzed. Our dataset contains 21,357 words equally distributed between the seven classes and prepared by 1000 people. So, it can be used for training and testing on a reliable DCNN model that will be able, after training, to generalize to new datasets.

The model works by training a (DCNN) model on a balanced-randomly-selected dataset using different structures. The results are improved by adding drop-out, image regularization, proper learning rate to avoid overfitting of the data. Finally, a blind test has been performed on the hidden test set and the performance was reported using a confusion matrix and learning curves as a validation tool for the model.

Results show that our model's performance is promising, achieving accuracy rate of 99.76% with error rate of 0.0230 using AHWD dataset, accuracy rate of 99.87% with error rate of 0.0181 using IFN/ENIT dataset, and accuracy rate of 99.90% with error rate of 0.0074 using augmented AHWD.

LIST OF ABBREVIATIONS USED

AdaGrad	Adaptive Gradient algorithm
API	Application Programming Interface
AHTR	Arabic Handwritten Text Recognition
AHWD	Arabic Handwritten Weekdays dataset
AHW	Arabic Handwritten Words
AOCR	Arabic Optical Characters Recognition
AOTR	Arabic Optical Text Recognition
ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short-Term Memory
BBN	Bolt Beranek and Newman Technologies
Batch Size	Batch Size
CMATER	Center for Microprocessor Applications for Training Education and Research
CV	Computer Vision
CM	Confusion Matrix
CNN	Convolutional neural network
DCNN	Deep Convolutional Neural Network
DARPA/SAIC	Defense Advanced Research Projects Agency/ Science Applications International Corporation
GTE	General Telephone and Electronic corporation
GER	Generalization Error Rate
HMM	Hidden Markov Model
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IFN/ENIT	I Institut für Nachrichtentechnik / Ecole Nationale d'Ingénieurs de Tunis
ICDAR	International Conference on Document Analysis and Recognition dataset
IRAC	Iterative Recognition of Arabic Character
k-NN	K Nearest Neighbor
KFUPM	King Fahd University of Petroleum and Minerals

KHATT	King Fahd University of Petroleum and Minerals Handwritten Arabic TexT
LC	Learning Curve
LR	Learning Rate
MAC	Macintosh
MSA	Modern Standard Arabic
OCR	Optical Character Recognition
PC	Personal Computer
PLB	Potential Letter Boundaries
ReLU	Rectified Linear Unit
ResNet110	Residual Network 110
ResNet18	Residual Network 18
ResNet50	Residual Network 50
RBM	Restricted Boltzmann Machine
RMSprop	Root Mean Square Propagation algorithm
SGD	Stochastic Gradient Descent algorithm
SVM	Support Vector Machine
TanH	Tangents Hyperbolicus activation function
TDNN	Time Delay Neural Network
VGG16	Visual Geometry Group 16
VGG19	Visual Geometry Group 19

ACKNOWLEDGEMENTS

All praises due to God who has permitted us and has given us the ability to complete this research work.

I would like to thank and express my gratitude to my supervisor, Prof. Jean-François Bousquet, and my co-supervisor Prof. Idris El-Feghi for invaluable patience, feedback, and for allowing me to undertake on such a journey through my doctoral research program. Their ongoing assistance and support enabled me to explore a challenging problem with confidence and courage. I would also like to thank the members of my supervisory committee for their helpful and constructive comments and contributions on my thesis work.

I would like to thank my family members; my wife, my sons, and my daughters for all the love and support they have provided. Special worm thanks go to my wife Rowaeda. Her constant support is always my power to make progress. Thanks also go to all my brothers, my sisters, my relatives, and my friends who helped me by recommendations and by collecting the data.

I would like to thank and express my gratitude to Teri Colter (the principal of Westmount Elementary School, Halifax) for reviewing this thesis.

I would like to express my deepest appreciation to my Defense Committee Members for their valuable time, their valuable comments about my PhD thesis, and express my gratitude to them.

Finally, I would like to thank all the students and people who helped me by writing the dataset and appreciate their patience with me.

Obtaining a PhD is undoubtedly a difficult journey; therefore, this assistance and support is critical to me. Obtaining a PhD also improved my knowledge and broadened my horizons. Every time I think I've solved a problem; an unknown frontier appears. It is now time to draw a conclusion, and this dissertation is a compilation of my years of work. I hope it contributes to technological advancement.

CHAPTER 1 INTRODUCTION

1.1 Background

Arabic language is an old, ancient language spoken by 420 million people across the world [1]. Modern Standard Arabic (MSA) is the standardized language that is used to communicate officially between Arabic communities. Nowadays, each language has a handwritten language style and a digital language style [1] [2]. The following sections explain the general characteristics and features of Arabic letters and words,

1.1.1 General Characteristics of Arabic Letters and Words

There are many challenges in Arabic character writing in terms of morphology and the way of writing, Arabic letters (characters) and words are written from right to left in cursive way. It applies ligature (combining two letters or more), and letters have between 2 to 4 shapes [2]. There are 28 characters in Arabic language, 16 letters have from one to three dots which differentiate between letters that have the same loop or shape such as (ب BAA), (ت TAA), (ث THAA), (ع AIN), (غ KHAIN) and so on as shown in Figure 1.1 and Figure 1.2 [4]. The real meaning of the words in Arabic language depends on the diacritics marks (vowels) and Nunation, it is called it in Arabic (TASH_KEEL التشكيل) Such as: Fat_Hah (,(فتحة Dha_Mmah (,(ضمة Ka_Srah(كسرة) and Soo_Koon(سكون) as shown in Figure 1.1

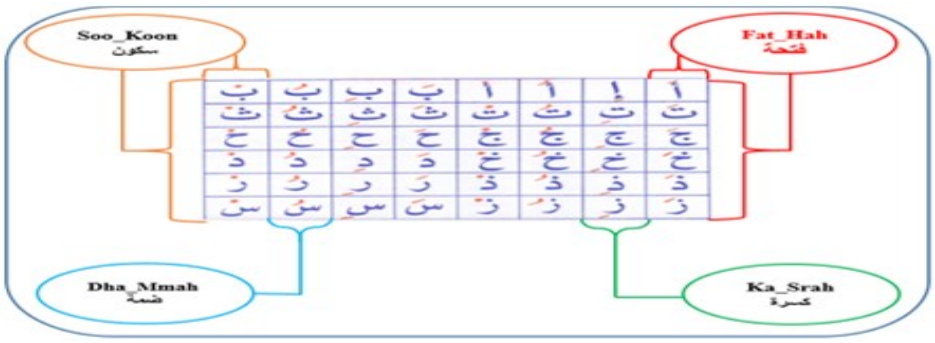


Figure 1.1 Diacritics marks on Arabic letters with single (Fat_Hah, Dha_Mmah, Ka_Srah, and Soo_Koon).

Nunation in Arabic has three forms: double Fat_Hah(,(تنوين بالفتحة double Dha_Mmah (تنوين بالضمة) and double Ka_Srah (تنوين بالكسرة) as shown in Figure 1.2.

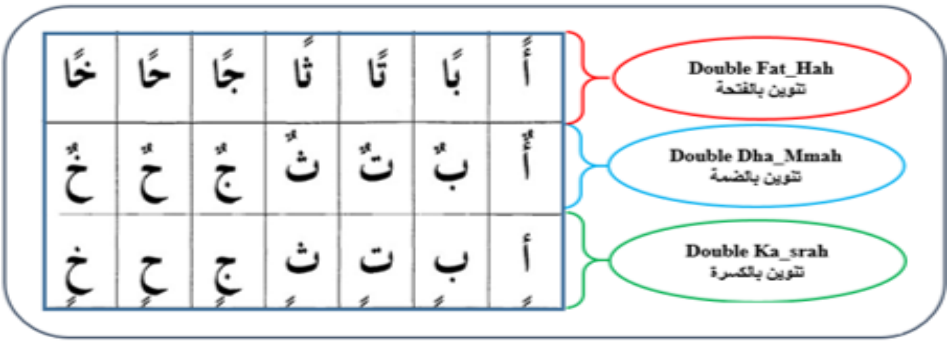


Figure 1.2 Diacritics marks on Arabic letters with double (Fat_Hah, Dha_Mmah, and Soo_Koon).

A character may have up to four positions in the word as shown in Table 1.1. Words in Arabic language are composed of connected letters and each word is separated by space; however, some letters are not connected with the word, but they compose the word such as (ذَهَب went), the letter (ذ) is not connected with the word, but it is one of the letters composing the word (ذَهَب went). Some letters of a word are not connected such as (زَار visited), (أُرْزْ Rice) and (وَرَقْ Papers) but they make words in Arabic language.

Table 1.1 The Arabic letters' positions and shapes.

Arabic letter name in English	Arabic letter	Arabic letters positions and shapes (forms)			
		Isolated	Beginning	Middle	End
ALEEF	ا	ا	ا	ـا	ـا
BAA	ب	ب	بـ	ـبـ	ـب
TAA	ت / ة	ت / ة	تـ	ـتـ	ـت / ـة
THAA	ث	ث	ثـ	ـثـ	ـث
JEEM	ج	ج	جـ	ـجـ	ـج
HAA	ح	ح	حـ	ـحـ	ـح
KHAA	خ	خ	خـ	ـخـ	ـخ
DAL	د	د	د	ـد	ـد
THAL	ذ	ذ	ذ	ـذ	ـذ
RAA	ر	ر	ر	ـر	ـر
ZAIN	ز	ز	ز	ـز	ـز
SEEN	س	س	سـ	ـسـ	ـس
SHEEN	ش	ش	شـ	ـشـ	ـش
SAAD	ص	ص	صـ	ـصـ	ـص
DHAD	ض	ض	ضـ	ـضـ	ـض

Arabic letter name in English	Arabic letter	Arabic letters positions and shapes (forms)			
		Isolated	Beginning	Middle	End
TTAA	ط	ط	طـ	ـطـ	ـط
TTHAA	ظ	ظ	ظـ	ـظـ	ـظ
AIEN	ع	ع	عـ	ـعـ	ـع
GHAIN	غ	غ	غـ	ـغـ	ـغ
FAA	ف	ف	فـ	ـفـ	ـف
QAAF	ق	ق	قـ	ـقـ	ـق
KAAF	ك	ك	كـ	ـكـ	ـك
LAAM	ل	ل	لـ	ـلـ	ـل
MEEM	م	م	مـ	ـمـ	ـم
NOON	ن	ن	نـ	ـنـ	ـن
HAA	ه	ه	هـ	ـهـ	ـه
WAAW	و	و	و	ـو	ـو
YAA	ي	ي	يـ	ـيـ	ـي

Note. This amended table demonstrates how many positions each letter can have [3].

1.1.2 Feature Extraction Classification

It is very important to know the features of the character or the word to use them as an input into all classifiers in the traditional methods depicted in Figure 1.3. These features are classified into classes, Structural features classification, Statistical features classification and neural network features classification. Structural features classification includes dots, concave loops, convex loops, endpoints, and branch points. Statistical features classification like number of image pixels, intensity histogram, and the pixels neighbor relationships, means, variance, energy, and diagonal moments, etc. Neural network features work as a black box method which depends on training the neural network to make it learn the pattern for right classification to reach the appropriate interconnection between the input and the output. Deep Convolutional Neural Networks (DCNN) extract features from raw image pixels automatically [4] - [5].

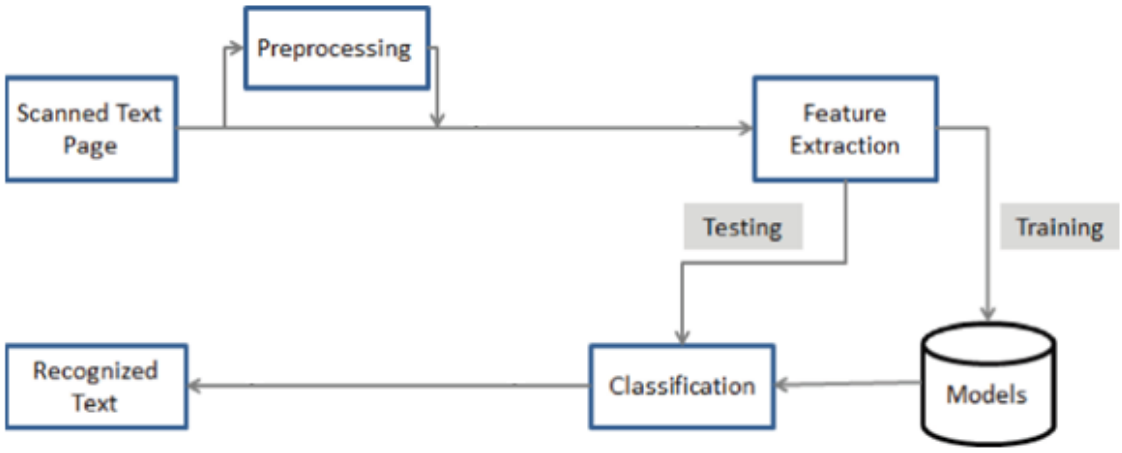


Figure 1.3 A general model of Arabic offline handwritten text and word recognition [4].

1.2 Problem Statement

Arabic language is a cursive style, written from right to left. It contains similar letters and can be written using assorted styles. These properties create many challenges that prevent recognizing text in Arabic manuscript. In fact, Automatic Arabic Handwriting Word Recognition using deep learning neural network is still at the early stages. Most of the research using Deep Neural Networks were done on Arabic Optical Characters Recognition (AOCR) and digital number recognition [6] [7]. There are no general, complete, and reliable Arabic Handwritten words database that can be used as a reference to all researchers who want to extend the work on Automatic Arabic Handwriting Word Recognition. There are some efforts of using Arabic sub words to synthesize many words as labeled dataset [8]. This is not a complete representation of the Arabic handwritten words because it does not reflect the reality of the natural Arabic language. By having this complete database (large number of Arabic words), a typical model can be created using DCNN to solve the problems of Automatic Arabic Handwriting Words Recognition.

Arabic has different writing styles such as Naskh, Ruq'a, and decorating styles (diwani, thuluth, andalusi). Theis leads to the misinterpretation of the words. For example, there are superimposed letters in the same word as shown in Table 1.2; errors associated with dots positions as shown in Table 1.3; errors associated with diacritics marks positions as shown in Table 1.4; ascending and descending letters on the baseline and unrecognized

words by deleted or hidden letters. All these errors are problematic for segmentation and challenging for traditional methods, such as Hidden Markov Model (HMM), Artificial Neural Network (ANN), Support Vector Machine (SVM), K Nearest Neighbor (k-NN), and syntactical methods, all of them depend on feature extraction. A survey was done by Parvez et al. [4] that explains all these traditional methods.

Table 1.2 Arabic writing style and superimposed letters, these words are from AHWD.

Andalusi	Thuluth	Diwani	Ruq'a	Naskh	Superimposed letters
الأحد	الأربعاء	الجمعة	السبت	السبت	الجمعة
الأحد	الأحد	الأربعاء	الاثنين	الاثنين	الخميس
الخميس	الأربعاء	الخميس	الثلاثاء	الثلاثاء	الجمعة
الجمعة	الجمعة	الجمعة	الخميس	الأربعاء	الأربعاء

Many Arabic manuscripts are deteriorating because of inappropriate storage, and most of them have not been digitized. This is due to the lack of reliable databases that can be used to recognize the words of Arabic manuscripts by using a typical model using deep learning. Most of the models are restricted to a specific dataset and cannot be generalized [6].

Table 1.3 Writing Positions of the three types of dots, these words are from AHWD.

Right position	Wrong Position	Wrong writing	Right Writing	Description
الثلاثاء	الثلاثاء	الثلاثاء	الثلاثاء	Three Dots
الاثنين	الاثنين	الاثنين	الاثنين	Two Dots
الاثنين	الاثنين	الاثنين الاثنين	الاثنين	One Dot

Table 1.4 Writing positions of diacritic marks, these words are from AHWD.

Right position	Wrong Position	Wrong writing	Right Writing	Description
الخميس	الخميس	الخميس	الخميس	Fat_Hah (فتحة)
الثلاثاء	الثلاثاء	الجمعة	الجمعة	Dha_Mmah(ضمة)
الأربعاء	الأربعاء	الأربعاء	الأربعاء	Ka_Srah(كسرة)
الأربعاء	الأربعاء	الأربعاء	الأربعاء	Soo_Koon(سكون)

1.3 Objectives

A number of studies have been done on offline and online handwritten recognition whether on characters, words or text line in English languages and lot of results were satisfactory results. The Arabic language still suffers from the lack of typical handwriting digital dataset [10] and from a learning algorithm for Arabic Handwritten Word (AHW) recognition. This is a good motivation to design a complementary system that overcomes all the problems that were investigated in the previous section.

The overarching aim of this work is to create a database that contains many Arabic writing styles and handwriting variations as shown in Table 1.2, starting by Arabic weekdays as a preliminary study. By amassing data from a variety of Arabic people to create a large dataset, then a model can be designed to satisfy the purpose of this research. The dataset that were collected depends on calligraphy, which means every person writes the weekdays on the paper naturally. Since Arabic language has many writing styles, people would choose the way that they feel comfortable to express their skills in terms of writing. Many Arabic handwritten weekdays words that were collected represent many Arabic writing styles, this would make the collected dataset varied and includes most of the writing format that is needed in this thesis. More than 21000 Arabic handwritten weekday words divided into seven classes, each class containing more than 3000 words. For example, the Saturday class contains more than 3000 words of Arabic Handwritten "Saturdays" with different writing styles. This is the same for all other classes. More on our database is in the dataset chapter.

The specific objectives of this research are:

1- To create a large train dataset of Arabic handwriting of weekday words coming from many people to capture all the expected variations of the handwriting. Then, prepare the data in binary image or gray image format so it will be used for training and testing the model.

2- Develop a new DCNN model on a balanced-randomly-selected set using different DCNN structures and improve the results by adding drop-out, image regularization, and learning rate to avoid overfitting of data. Finally, perform a blind test on the hidden testing dataset and report results using the confusion matrix and learning curves.

1.4 Outline

This dissertation shows how Deep Convolutional Neural Network (DCNN) can help solve the problems of automatic Arabic handwriting word recognition by creating a new Arabic dataset and by designing a new DCNN model, then test the new model using this dataset. This thesis is organized as following:

Chapter 1 is the introduction, which discusses the nature of Arabic language and its widespread characteristics and its general features.

Chapter 2 presents the state of the art in Arabic handwriting image recognition.

Chapter 3 explains the analysis, the organization, and the design of datasets that are used in the proposed DCNN model.

Chapter 4 describe the optimization of DCNN model.

Chapter 5 demonstrates the Experimental setup.

Chapter 6 explains the analysis of the result.

Chapter 7 explains the conclusion and the future work.

CHAPTER 2 LITERATURE REVIEW

2.1 Background

The history of pattern recognition is very interesting to analyze from the first spark where an Optical Character Recognition (OCR) was discovered until nowadays where the Deep Neural Network is considered the dominant technique for pattern recognition and other applications.

2.1.1 Optical Character Recognition

In the early 1950s, the first commercial machine was invented called OCR (Optical Character Recognition), this machine was hardware based. Then in 1970, machines that were software based were invented [11]. The OCR is the process of converting any written text, either handwritten or printed, into computer language format to increase the interaction by enhancing the interfacing between humans and computers in automatic ways [12].

In 1975, Nazif. [13] produced the first Arabic Recognition System called Arabic Optical Text Recognition (AOTR). His system was based on the idea of extracting strokes (20 Radicals) to recognize the Arabic letters. The work on Arabic characters recognition carried on [14] to include the recognition of separated handwritten Arabic characters which led to an online system named Iterative Recognition of Arabic Character (IRAC). Then Amin et al. [15] produced a new work for recognizing multi-font Arabic characters in offline mode. Moreover, a segmentation stage was tested and done on the cursive Arabic writing [16] [17] [18] they have built up a system that recognizes isolated offline cursive words by using many approaches such as letter and word segmentation by applying local minima with low vertical profiles and detecting base line.

In 1987, Almuallim et al. [19] built a structural system technique that recognizes offline Arabic cursive handwriting by segmenting words (preprocessing) to strokes (sub-words) was designed and created, where later these strokes were classified based on their geometrical and topological properties into strokes with loop, stroke without loop and complementary characters. In the nineties, the commercial OCR for English were available to be used by computers Personal Computer (PC) or Macintosh (MAC) and the systems had the ability to read handwritten and printed writing even in other language such as

Chinese, Korean, Cyrillic, Arabic and Japanese [20] , [21]. In 1990, an on-line system was developed for Arabic handwritten recognition by El-Sheikh et al. [22]. This system is based on segmentation where the character position has four sets within the word (beginning, middle, ending, and isolated) and each set is classified into another sub-set called strokes. In terms of statistical technique, an approach was developed by Al-Yousefi et al. [23]. This approach recognizes Arabic handwritten characters by using vertical and horizontal projections momentarily.

In 1996, Olivier et al. [24] developed a system which dealt with segmentation and handwritten word coding by individual monitoring to automate the processing of the handwritten Arabic script, image, or document. This system was composed of three stages. The first stage segmented the word into its characters (graphemes), the second stage analyzed these characters (graphemes) by a series of attention or observations which is like human processing, and finally, they collected the outcome from both stages and utilized them in the recognition stage. This system worked under two main predefined cases to keep the minima in safe side. The first case is that there is no loop under the minima, and second case is that the mean width of the word must be greater than the sub-word (stroke) width in the minima area.

According to [25] [26], recognition of an Arabic system was developed; the system depends on shape primitives by using mathematical operation in terms of morphology. Chen et al. [27] developed a system to recognize handwritten words by using Hidden Markov Model (HMM). HMM has two parameters, transition probability and emission probability (output probability). Each time the system in state x produces y observation based on the probability that is correlated with state x. In1995, Emam [28] developed an OCR system that recognized Arabic handwritten script by using the feature of border transition descriptor. Motawa et al. [9] used the projection-based algorithm and produced a technique that used mathematical morphology based on the theory that most of the time the Arabic characters are connected by horizontal lines.

In [29] the contour-based algorithms were used where the local minima points are located for all the upper contour and the local maxima points are located for all the lower contour of each word in the text. All these points are considered as Potential Letter Boundaries (PLB) by using some rules on lower and upper PLB they remove any bad PLB

that might affect the right matching technique between lower and upper PLB. Sakher and OmniPage products have developed an Arabic OCR system by using Defense Advanced Research Projects Agency/Science Applications International Corporation (DARPA/SAIC) database and the accuracy was 86.89 as a real observation as shown by Kanungo et al [30].

As demonstrated in [31] Bolt Beranek and Newman Technologies (BBN), General Telephone and Electronic corporation (GTE) have developed a new methodology for OCR using continuous speech recognition. That resulted in successful technology that depends on Hidden Markov Models (HMM) and shows many features such as script-independent feature extraction and speech recognition. The new system was tested using DARPA Arabic OCR Corpus. Natarajan et al. [32] produced porting the BBN BYBLOS OCR System to other languages, such as Arabic, in three steps. These steps collect the required data, choose the right training model and system optimization. In 2001, Trenkle et al. [33] produced many enhanced improvements to a system which has being used to recognize Arabic and Farsi script in low resolution, low quality, and binary images by using ensembles of decision trees as recognition method instead of neural nets.

A new OCR Arabic system was created and developed by Hamami and Berkani [34] [35]where they could handle multi font and multi style characters. The problem of over segmented characters was solved for some of them by using a structural approach which didn't need to use skeleton portioning (time killing), and the proposed system could be used in Arabic and Latin because the geometrical characters were adapted to the two languages. In 2003, Pechwitz et al. [36] developed an offline system for Arabic recognition depending on a semi-continuous 1- dimensional HMM by using the height, length, and baseline skew as normalized parameters with features that were gathered by using the sliding windows method. They have accomplished and obtained 89% of word recognition by using IFN/ENIT database.

In 2003, Amin [37] developed a system to recognize Arabic characters using machine learning automatically, the system achieved 86.65% of character recognition by using handwriting character database which was written from different people with low to high quality. A new OCR system called; An Automatic Arabic Handwritten Text Recognition (AHTR) was designed as shown by Jannoud [38] . The system used the

segmentation as the main stage, where the word or sub word must be thinned, and the base line is calculated by horizontal projection. For more details, a survey was conducted as shown by Althobaiti and Lu [39], summarizing the complications and challenges of Arabic Optical Character Recognition (OCR). They divided their difficulties analysis into three categories, "general challenges, handwritten text challenges and Arabic text challenges".

2.1.4 Deep Neural Network

Deep Neural Networks (DNN) have been the dominant star for a long time compared to the visual recognition models such as character and text recognition [46]. Automatic handwritten, Image recognition [8], [42], [47], [48], [49] and Face Recognition [50], [51]. Convolutional neural network (CNN) is one of (DNN) that consists of three main parts, Convolutional layers, Max-pooling layers, and fully connected layers. Wshah et al. [52] has proposed a method using CNN for lexicon size reduction. This method applies the dot descriptor with a piece of Arabic words to eliminate unlike words. They used the IFN/ENIT database of 26459 Arabic Handwritten Word images in their experimental work and got 87% as a reduction rate and 93% as an accuracy rate.

The first successful work for Visual recognition (Computer Vision) was in 2017 by AlexNet which employed Convolutional Neural Network as an architecture for image classification [53]. They achieved a top-1 test error rate of 37.5% and top-5 error rate of 17.0% by using ImageNet Large Scale Visual Recognition Challenge (ILSVRC) -2010 dataset. A new benchmark was developed by Wang et al. [46], where they took advantage of multilayers neural network with unsupervised feature learning to create a model that acutely trains the network. They achieved a classification accuracy rate of 82.2% with recognition model of 180 filters, classification accuracy rate of 83.4% with recognition model of 360 and filters classification accuracy rate of 83.9% with recognition model of 720 filters on International Conference on Document Analysis and Recognition dataset (ICDAR) 2003.

A new system was proposed by Mars et al. [54] that recognizes online Arabic handwriting (letters and words) based on Time Delay Neural Network (TDNN) and Multi-layer perceptron. They applied the system on their database which contains 6090 characters and 1080 words. By using their own dataset, they achieved a high accuracy rate of 98.50% for characters recognition and 96.90% for words recognition. A new method in Arabic

handwritten recognition was proposed by Alani [55], the method was proposed in two stages. The first stage extracted the features from the raw data by using Restricted Boltzmann Machine (RBM), and then the second stage fed the extracted features into Convolutional Neural Network (CNN). They trained and tested their model on Center for Microprocessor Applications for Training Education and Research (CMATER 3.3.1) and the Arabic handwritten digit dataset and achieved an accuracy rate of 98.59%. In 2017, Ashiquzzaman et al. [56], a new system was proposed by using two ways for enhancing the Arabic offline handwritten recognition. First, they used Rectified Linear Unit (ReLU) as an activation function in their model (Input layer, hidden layer and softmax 'classifier' layer) with the use of dropout regularization which chooses a random number of neurons in each layer in order not to update their gradient to avoid overfitting. Second, they used Convolutional Neural Network which employed Backpropagation algorithm for training to update the weights and the bias. They achieved an accuracy rate of 97.4% as a new state-of-the-art on CMATER database.

Upgrading work on Arabic Handwritten Words Recognition was done by Alexnet by Almodfer et al. [57]. This work reduced error and avoided overfitting through using the dropout regularization technique. They used the IFN/ENIT Dataset to train and test their model with many settings of experiments to achieve classification accuracy rate of 92.13% and 92.55% as a new state-of-the-art. A new consecutive method for Offline Arabic Handwritten Recognition was proposed by Ghanim et al. [7] where they employed the Hierarchical Agglomerative Clustering (HAC) method to divide the IFN/ENIT database into associate clusters (a, b, c, d) for training and (e) for testing to show the database as a large search tree to cut down the complications while comparing every test image with a cluster. They proposed a system to evaluate the outcome of six various Deep Convolutional Neural networks (AlexNet, VGG-16, GoogleNet, Res50 Net, ResNeXt Net, DenseNet) on the recognition rate and the accuracy. At the end they concluded that their proposed method using CNN as features extraction and AlexNet as a classifier reached 95.6% as a recognition rate, and AlexNet had the best accuracy rate by applying three different learning rates on each of the six different DCNN. 89% as an accuracy rate with 0.0001 as a learning rate, 90% as an accuracy rate with 0.001 as a learning rate and 99% as an accuracy rate with 0.01 as a learning rate. The drawback of this method is, they only use

11% of the total 859 database class to reduce recognition complexity while in deep learning more data is needed to get valid results.

Ashiquzzaman at el. [59] have used seven types of deep learning transfer models, AlexNet, GoogleNet, Residual Network 18(ResNet18), Residual Network 50(ResNet50), Residual Network 110(ResNet110), Visual Geometry Group 16(VGG16), and Visual Geometry Group 19(VGG19). The purpose of using these seven types is to determine which model is good to be used for classification using two Arabic handwritten images datasets written by native and non-native people. They have used the original datasets and augmented datasets for (training 60%) and (testing 40%) by all seven types and the GoogleNet had the best performance. The performance was measured based on accuracy, sensitivity, and specificity. Accuracy represents the correctness of the deep learning classifier; 93.2% for the original data and 95.5% for the augmented data. Sensitivity (Sens) which represent the correctness of non-native language classification; 92.4% for the original data and 93.9% for the augmented data. Specificity (Spec) which represents the correctness of native language classification; 93.9% for the original data and 97.0% for the augmented data.

CHAPTER 3 ANALYSIS AND DESIGN OF DATASETS

3.1 Datasets

A perfect dataset needs to meet all the requirements for problem solving to get a beneficial and long usage. Also, to be valuable and long-lasting, a dataset must reflect a sufficiently difficult problem [60]. Datasets are critical and a necessary component of any pattern recognition, image classification, computer vision work. Because a single dataset may only cover a single job, having a large and diverse range of datasets is critical for taking a more comprehensive approach to measuring and reviewing algorithm performance. By creating a benchmark dataset, a classification and comparison work would be created and used by researchers on variant machine learning methods, and the work would be quite easy, fast, and precise [61].

3.2 Arabic Handwritten Weekdays Dataset (AHWD)

In this section, a new dataset is represented and called the Arabic Handwritten Weekdays Dataset (AHWD). AHWD is the collection of Arabic handwritten weekday words that was written in Libya by different Arabic speakers (Libyan, Tunisian, Algerian, Saudi, Palestinian, Syrian, Omani, and Sudani) with different range of ages: students in grades 5 to 9, as well as high school students, undergraduate and postgraduate university students. Also, people on the street, senior people and many other Arabic nationalities were asked to participate in this database. The aim was to collect between 20,000 to 30,000 Arabic handwritten weekday words. More than 20,000 words were collected and divided into seven classes, each one containing more than 3,000 words of the same weekday words with different morphological and calligraphic styles. For collecting the data, more than 1,000 A4 papers (form with 70 empty blocks) were used, each paper is divided into seventy empty equal rectangular shapes, ten empty rows in seven empty columns. The participants would write at least 3 rows including all weekdays as shown in Figure 3. 1 with excellent quality and Figure 3. 2 with low quality. AHWD is a dataset that can be used to study Arabic handwritten words recognition and, in this work, specifically.

Figure 3. 1 Sample of collected data before preprocessing with good quality.

Many schools, and universities were targeted to collect the data from. More than five primary schools, seven junior high schools, ten high schools, Zawia university, Tripoli university, and Sabratha university were visited to collect the data.

Figure 3. 2 Sample of collected data before preprocessing with low quality.

Regarding the data that was collected from the public, supermarkets, government administration offices, sport clubs, and workshop places were visited. In general, many of them were helpful and happy to participate while only a few of them declined.

3.1.1 Preprocessing the AHWD

More than 1,000 people filled out the Arabic handwritten weekdays words forms the first stage. This was followed by stage two with some work needed to be done such as scanning all the Arabic handwritten weekday words forms as an image using MS-paint program and some preliminary work on each word. This included getting a clear resolution, without touching the milestone of the original word and saving the form that contained the preprocessed seventy-word image as .JPG format as shown in Figure 3. 3. The JPG image file format was created by Joint Photographic Experts Group (JPEG) in 1992, the group recognized the need to reduce the size of large photographic files so that they could be shared more easily. Figure 3. 4 shows the preprocessing work that was done for one sample as an example. Figure 3. 3 is the preprocessed of Figure 3. 1. Around 70% of the data was preprocessed, and 30% did not require preprocessing.

Figure 3. 3 Sample of collected data after preprocessing Figure 3. 1

The preprocessing work included removing any noise, fixing the position of the dots, fixing the position of the diacritic marks, fixing superimposed letters as shown in Figure 3. 3 and Figure 3. 4 and fixing some letters to look acceptable to get clean and clear images. All the work was done carefully and precisely without changing the meaning of the original word. The preprocessing took more than seven months to get the data ready for stage three, where each weekday word was separated and saved in its proper folder.

The preprocessing here is to fix some week words which were written by the participants in unreadable manner to avoid reading them wrongly by the model and lose the words. As mentioned earlier, this dataset is the first step toward creating a complete database for handwritten Arabic words to be used in all research purposes. Once this complete database is done the model would be ready to classify any handwritten Arabic words.

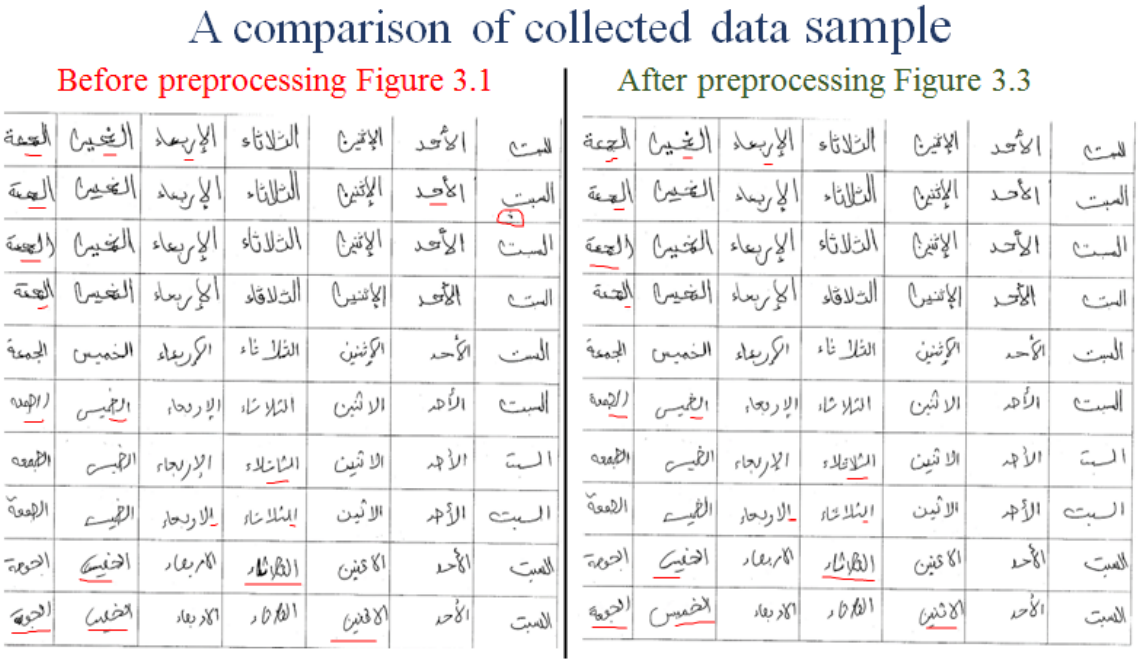


Figure 3. 4 comparison of collected data between Figure 3. 1 before processing and Figure 3. 3 after preprocessing.

The third stage was opening each saved seventy-word form image file using MS-paint and cut each Arabic handwritten weekday word and save it in MS-word program as shown in Figure 3. 5 to do more work on it later in stage four. Seven MS-word files were created, each file contained the same weekdays word name. For example, all cropped preprocessed Saturday words would be saved under Saturday.doc file as shown in Figure 3. 5, and the same applied to all the other six files.

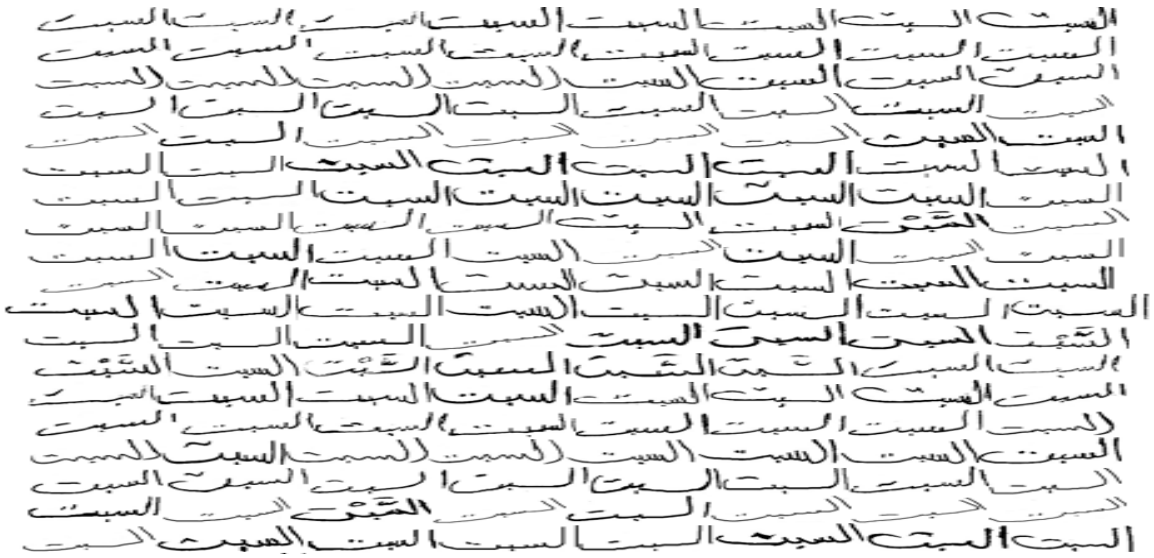


Figure 3. 5 Sample of preprocessed Arabic handwritten Saturdays after cropping.

Another example of cropped preprocessed weekdays is Monday's words would be saved under Monday.doc file as shown in Figure 3. 6.

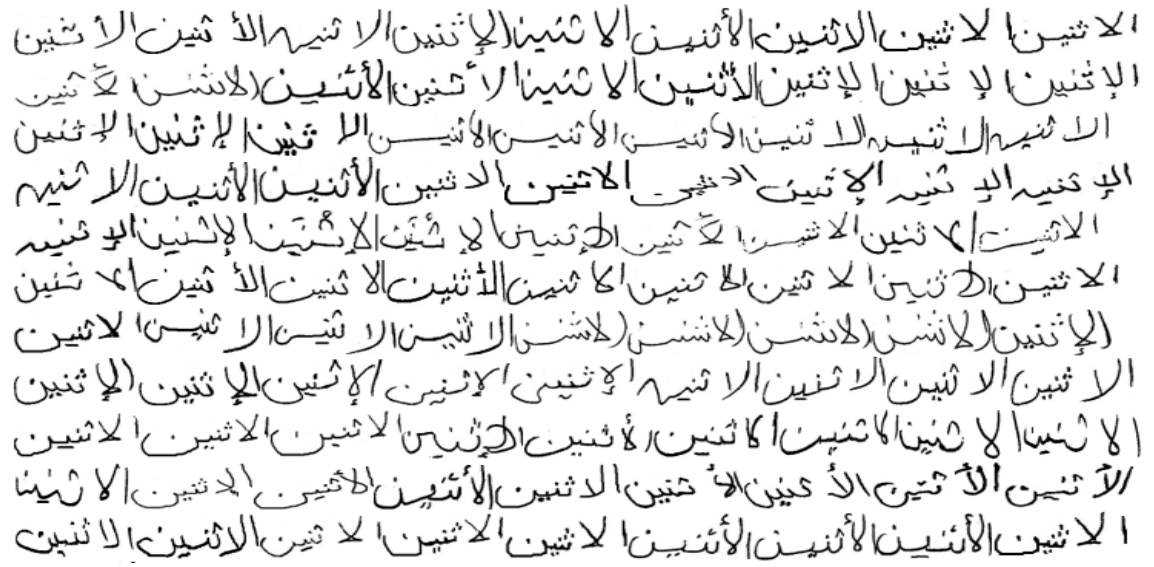


Figure 3. 6 Sample of preprocessed Arabic handwritten Monday after cropping.

In stage four seven folders were created and named as following, Saturday's folder, Sunday's folder, Monday's folder, Tuesday's folder, Wednesday's folder, Thursday's folder, and Friday's folder as shown in Figure 3. 7. The following work was to open each MS-word file which was created in stage 3 and cut each cropped preprocessed Arabic

handwritten weekday word and paste it back to MS-paint program and then to save it later as .JPG in the specified weekday folder, such as Saturday's folder or Monday's folder with range of 32 x 64 pixel to 38 x 78 pixel as shown in Figure 3. 8 and Figure 3. 9, all same weekday words grouped in one folder. The purpose of the size range is to avoid any time wasting in fixed images. there is a way to fix the image during feeding the image as an input into our model, the size will be resized to 32 x 64 pixel as a power of 2.

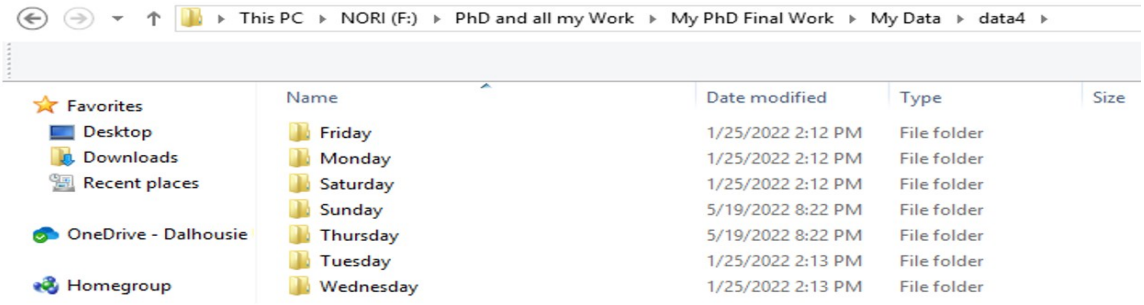


Figure 3. 7 Seven folders of AHWD

3.1.2 Organization of AHWD

To fit the model with the AHWD dataset, the AHWD dataset was organized as follows:

[(0: Saturday 3048 السبت_samples), (1: Sunday 3014 الاحد_samples), (2: Monday الاثنين 3080_samples), (3: Tuesday 3059 الثلاثاء_samples), (4: Wednesday 3017 الاربعاء_samples), (5: Thursday 3059 الخميس samples), (6: Friday 3077 الجمعة_samples)]. Each weekdays folder contained several samples, these samples refer to the weekday names. For example, Saturday's folder contains 3048 Saturday's image, Monday's folder contains 3014 Monday's image, and so on, as shown in Figure 3. 8 and Figure 3. 9.

Figure 3. 8 Sample of Saturday's folder with .JPG format

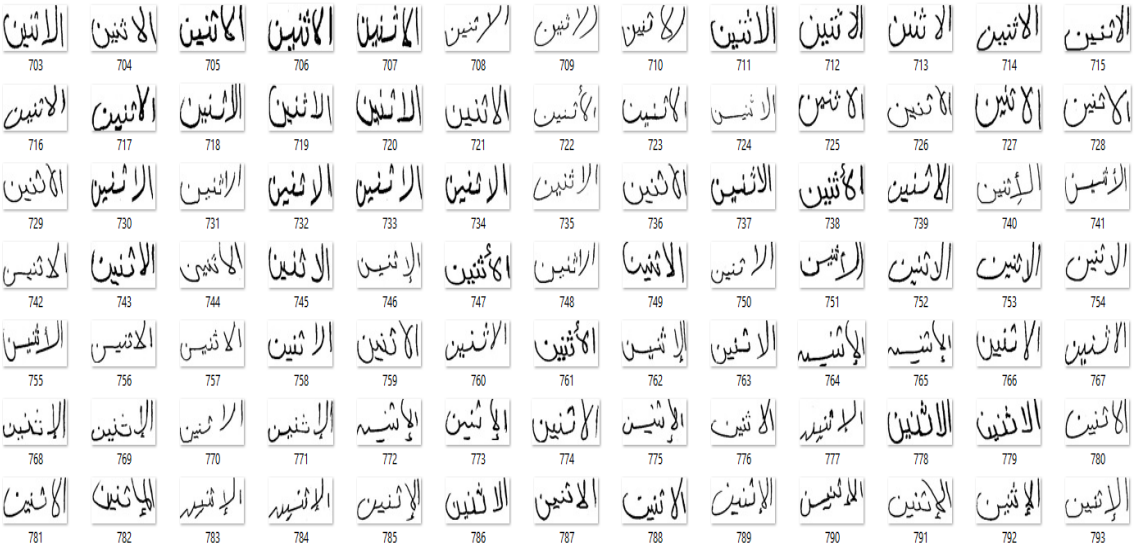


Figure 3. 9 Sample of Monday's folder with .JPG format

3.3 IFN/ENIT Dataset

In this section the IFN/ENIT dataset is described and reorganized to test the model and compare the results with AHWD and augmented AHWD. IFN/ENIT is a handwritten Arabic Tunisian town/village names dataset which was collected in Tunis for the purpose of education and research. 2,265 forms were filled out by 411 different writers to guarantee

the wide range of writing style. The dataset contains 26,459 handwritten Tunisian town/village names as shown in Table 3. 1. IFN/ENIT is a very well-known dataset, is being used in many Arabic machine learning research and is the most popular dataset.

Table 3. 1 Details of IFN/ENIT dataset [62].

Quantity of words in town names	Quantity of town name images	Quantity of PAW's	Quantity of characters
1	12992	40555	76827
2	10826	54722	98828
3	2599	20120	36004
4	42	188	552
Total	26459	115585	212211

3.3.1 Reorganization of IFN/ENIT Dataset

Some work had to be done to reorganize most of IFN/ENIT to fit the model without changing the core of the words. There was a need to edit the dataset to make it work with the model by creating 21 folders as shown in Figure 3. 10, to promote code organization and reusability. Each folder contains the resemble Arabic handwritten words.

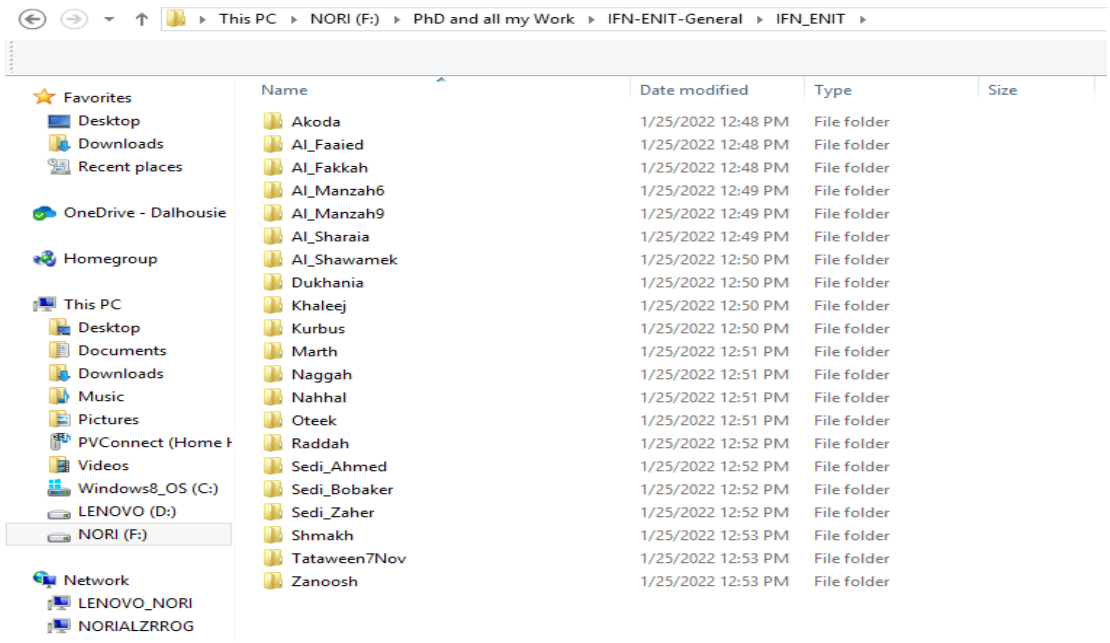


Figure 3. 10 Re-organization of IFN/ENIT to 21 folders

Each folder contains the same towns and villages names written by different writers as shown in Figure 3. 11, Figure 3. 12, and Figure 3. 13 as an example.

Figure 3. 11 Akoda village.

Figure 3. 12 Al_Shawamek town.

The final re-organization of the data to be fitted into the model is as follows:

[(0: Akoda 987 أكودة_samples), (1: Al_Faaied 987 الفايض_samples), (2: Al_Fakkah1012_ الفكة samples), (3: Al_Manzah6 920 6 :المنزة_samples), (4: Al_Manzah9 9965المنزة_samples), (5: Al_Sharaia 907 الشرايع_samples), (6: Al_Shawamek 1065 الشوامخ_samples), (7: Dukhania 1011 الدخانية_samples), (8: Khaleej 1037 الخليج_samples), (9: Kurbu 968 قربص_samples), (10: Marth 1018 مارث_samples), (11: Naggah 1009 نقه_samples), (12: Nahhal 1054 نحال_samples), (13: Oteek 1062 أوتيك_samples), (14: Raddah 1034 الرضاع_samples), (15: Sedi_Ahmed سيدي أحمد 995_samples), (16: Sedi_Bobaker 1024 سيدي بوبكر_samples), (17: Sedi_Zaher سيدي الظاهر 972_samples), (18: Shmakh 1022 شماخ_samples), (19: Tataween7Nov تطاوين 7 نوفمبر 1000_samples), (20: Zanoosh 1042 زنوش_samples)].

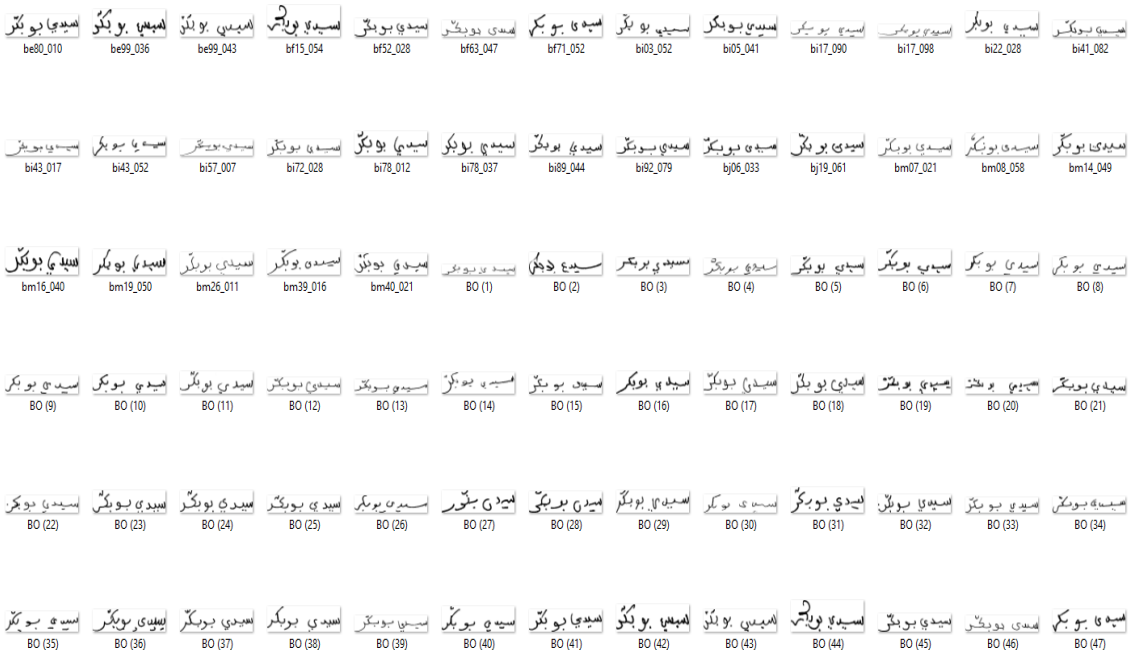


Figure 3. 13 Sedi_Bobaker village.

3.4 Augmented AHWD

The successful application of various deep learning models requires high-quality and plentiful data. Data augmentation is frequently employed in the context of deep learning since the volume and quality of the data are just as essential as the algorithm. Data augmentation is the process of applying one or more deformations on an available dataset to generate new, supplementary training data.

Therefore, several picture deformations were used in the data augmentation in our research such as artificial noise, noisy data, rotated data and shifted data. Each image was subjected to a random mix of the aforementioned deformations to generate different images. In this section the augmented AHWD is described as follows:

- Artificial noise.
- Noisy data.
- Rotated data.
- Shifted data.

Artificial noise, which was created by adding a random 100 black pixel to each image as shown in Figure 3.14

Figure 3.14 Artificial noise.

- Noisy data, which was created by adding gaussian noise with mean =0 and var=0.5 to each image as shown in Figure 3.15. The benefit of Gaussian noise is that the

distribution itself behaves well. It's named the normal distribution for a reason: it has useful features and is frequently employed in scientific and social sciences. It is frequently used to simulate random variables whose true distribution is uncertain. In other words, white Gaussian noise where the values are equally distributed and statistically independent at any two times (and hence uncorrelated). White noise has a zero mean, a constant variance, and is time independent. White noise, as the name indicates, has a power spectrum that is equally distributed throughout all allowed frequencies.

Figure 3.15 Noisy data

- Rotated data, which was created by rotating the image 5° counterclockwise as shown in Figure 3. 16. In this method of augmentation, additional life-like examples are introduced from which our model can learn. The images can be rotated by 0 to 360 degrees clockwise or counterclockwise. The purpose is to make the pixels of the image rotate in this method and change the position of the object. A rotating image rotates left or right along an axis while maintaining the same face toward you. When you flip an image, it rolls over, either vertically or horizontally, to become a mirror image. The choice of rotating by 5° is to save processing time because thousands of rotated images would fit in the model during the real time execution. The more degree of image rotation the more processing time needed to fit the data.

Figure 3. 16 Rotated data.

- Shifted data, which was created by shifting data 5 pixels to the right as shown in Figure 3. 17. The choice of shifting by 5 pixels is to save processing time because thousands of shifted images would fit in the model during the real time execution. The more pixels of image shifting the more processing time needed to fit the data.

Figure 3. 17 Shifted data.

Table 3.2 The size of AHWD before and after data augmentation explains in detail the size of AHWD and the size of augmented AHWD in terms of image's number and how many images each folder has.

Table 3.2 The size of AHWD before and after data augmentation

Number	Weekdays folder	Initial size	After augmentation
1	Saturdays	3048	10509
2	Sundays	3014	10430
3	Mondays	3080	10609
4	Tuesdays	3059	10535
5	Wednesdays	3017	10490
6	Thursdays	3059	10660
7	Fridays	3077	10482

CHAPTER 4 OPTIMIZATION

In this chapter, firstly, background on gradient descent is presented. Secondly, optimizer algorithms are described. Thirdly, the hyperparameters are optimized. Fourthly, how the optimal capacity can be reached using hyperparameters is analyzed.

4.1 Background

In deep machine learning, optimization refers to the process of getting the lowest error function (cost function, loss function) which will be used to fit the machine learning algorithm. During optimization, the algorithm goes through all potential variants of its parameter combinations to find the optimum one that permits the accurate mapping of characteristics and classes during training. A mapping function from inputs to outputs is learned by deep learning neural networks. This is accomplished by updating the network's weights in response to the errors the model makes on the training dataset. Updates are done to continuously lower this error until a suitable model is discovered. Optimization function (optimizer) is used to alter or change the neural network's attributes such as learning rate and weights to minimize the error (loss). Where weight is a model parameter and learning rate and batch size are model hyperparameters. Model parameter value starts with a random value and then updates itself during the execution process. Whereas the hyperparameter value is set prior to training and remains constant during training session. So, finding the best value for hyperparameters is called hyperparameter optimization (tuning). Size normalization is commonly used to reduce size variation and adjust the character or word sizes to an identical size, in the proposed model the size is normalized by 64x32 pixels [62] [63].

4.2 Gradient Based Optimization Algorithms

Gradient descent is an optimization process that locates the minimum of an objective function by following the negative gradient of the function. Gradient descent has the drawback of bouncing around the search space (search through a landscape) on optimization problems with many curvature or noisy gradients, as well as being trapped in flat regions of the search landscape with no gradient, where the landscape is referred to as an error surface. Through this landscape, the optimization algorithm iteratively moves

around, adjusting the weights and looking for good or low-elevation regions. Finding the bottom of a landscape with basic optimization issues is straightforward; in fact, it is so simple that extremely effective algorithms may be created to discover the optimal answer. The landscape has the shape of a large bowl. Convex is a mathematical term used to describe these kinds of optimization problems as shown in Figure 4. 1 [64] [65].

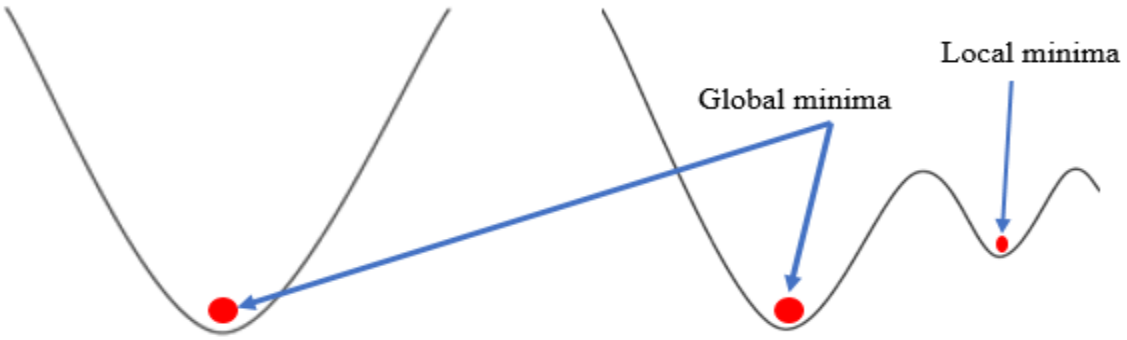


Figure 4. 1 Local and global minima

When optimizing the weights of a neural network while the navigated error surface is not shaped like a bowl, this means there are several hills and valleys in the landscape. Non-convex is a mathematical term used to describe these kinds of optimization problems as shown in Figure 4. 1. There is no method that can identify the best set of weights for a neural network in polynomial time. In mathematics, the optimization issue can be solved by neural network training, known as NP-complete. NP-complete is an abbreviation for a nondeterministic polynomial-time complete and defined as the complexity class of decision problems with a high degree of complexity for which solutions may be verified for correctness by using an algorithm whose execution time scales polynomially with the amount of the input [63] [64] [66]. The optimization of neural network weights is a challenging task because of several factors, including local minima and Saddle point (flat regions).

Local minima or local optima refer to the many locations of the error landscape (valleys) where the loss is small as shown in Figure 4. 1. The valley has a high elevation when looking at the entire landscape and better alternatives could be available. It is best practice to start the optimization process with a lot of noise so that the landscape may be sampled broadly before choosing a valley to fall into since it might be difficult to tell if the

optimization algorithm is in a local minimum or not [63] [64] [66]. Weights are updated based on the lowest error in the network.

The global minimum is the location where the landscape is at its lowest level and leads to the lowest error that is needed as shown in Figure 4. 1. The difficulty is that the distinction between the local and global minima may not be incredibly significant in neural networks, which may have one or more global minima. This has the implication that getting a good enough set of weights is frequently more feasible and, hence, more acceptable than finding a global optimum or best set of weights [63] [64] [67]. Global minima are the key solution to find the lowest error and in turn to find the generalization error.

Saddle point or flat regions as shown in Figure 4. 2 is a place on the landscape where there is no gradient (zero value). These can be discovered at the base of valleys or in the spaces between hills. A zero gradient is problematic since it shows that the optimization algorithm is unclear about the optimum path to take to improve the model, the solution for this problem is to add the momentum hyperparameter to avoid the zero gradient [63] [64] [66].

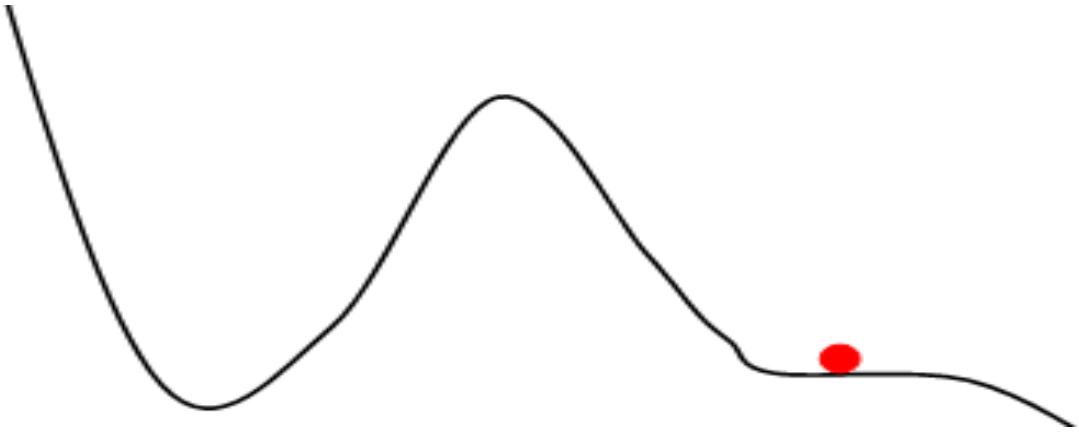


Figure 4. 2 Saddle point

4.2.1 Stochastic Gradient Descent algorithm (SGD)

The Stochastic Gradient Descent Algorithm is a fast optimization method (optimizer) and is an algorithm that trains the deep neural network by estimating the gradience of the error momently in each state and updates the weights of the model by using backpropagation algorithm. All weights of the neural network are calculated by empirical optimization approach not by analytical approach. Using mini-batch sizes can

approximate a good gradient descent using limited data samples, and they typically are powers of 2, such as 16, 32, 64, 128, 256, 512, 1024, and so on. The reason for using powers of 2 is to help mathematically facilitate the resources efficiently such as Graphic Processor Units (GPU) [64].

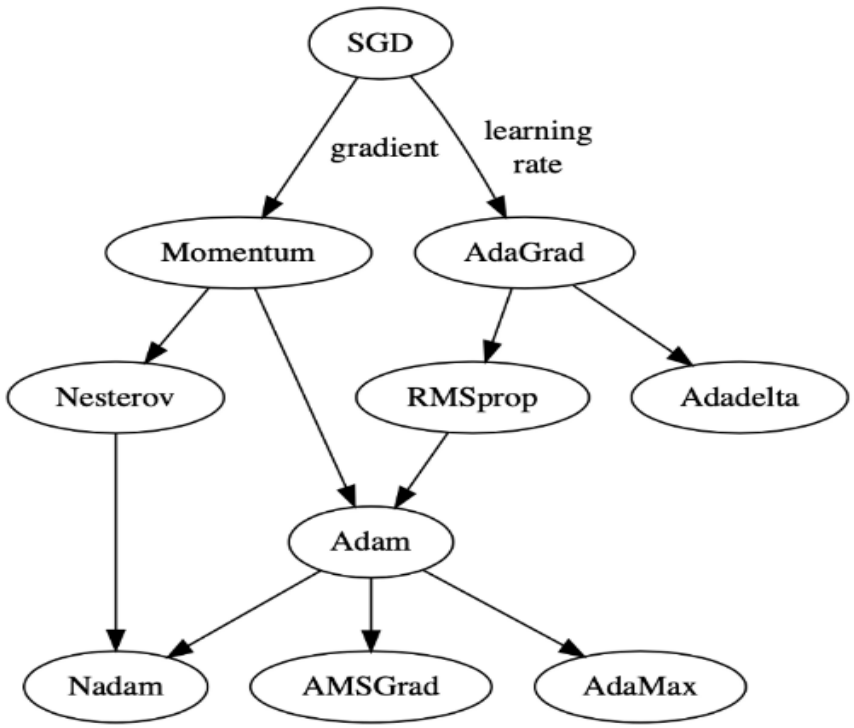


Figure 4. 3 Evolutionary map of optimizers [65].

4.2.2 Adaptive Gradient Algorithm (AdaGrad)

Adaptive gradient algorithm (AdaGrad) adjusts the parameters that are appropriate for the learning rate, to make large updates for discrete parameters and small updates for frequent parameters. By conducting training, it is suitable for processing scattered data, but the problem lies in some cases where the learning rate will decrease due to the accumulation of gradients from the beginning of the training. In addition, there is a point that the model will not learn again because the learning rate is almost zero, Adam's algorithm worked to solve it by making the learning rate go towards stability [63] [64] [65].

4.2.3 Root Mean Square Propagation Algorithm (RMSprop)

The Root Mean Square Propagation algorithm (RMSprop) is a derivative of the adaptive gradient algorithm. The learning of each coefficient depends on it (i.e., its overall learning rate is constant), but it computes the slope with an exponential mean regression instead of the sum of the scores of it. As a result, it automatically adjusts and responds to changing specific learning rates to prevent the overall learning rate of the model from drifting out of bounds and backtracking, the algorithm has excellent performance in unstable problems [63] [64] [65].

4.2.4 Momentum Algorithm

Momentum is an addition to the gradient descent optimization process that enables the search to develop inertia in a direction in the search space and get around noisy gradient oscillations and cruise over flat areas of the search space. Momentum is the process of moving to a new location in the search space by introducing an extra hyperparameter that regulates the quantity of history (momentum) to incorporate in the update equation. The hyperparameter's value is described as falling between 0.0 and 1.0, and it frequently has a value of 0.8, 0.9, or 0.99, which is near to 1.0. Gradient descent with no momentum is equivalent to a momentum of 0 [63] [64] [65].

4.2.4 Adam Algorithm

Adam algorithm is a method for stochastic optimization. It is the most popular optimizer in classification of deep learning neural network and is used to update the network weights iteratively based on training data. Adam is a new optimization method that takes the place of the old stochastic gradient descent approach. Adam optimizer uses both Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam algorithm is called first order optimization algorithm because it uses the first derivative of the function. Figure 4. 3 shows the developments of gradient descent algorithms, from this figure, Adam optimizer is originated of RMSprop optimizer, AdaGrad optimizer and Momentum optimizer algorithms [63] [64] [65].

4.3 Optimization of Hyperparameters

Most deep learning algorithms accompany numerous hyperparameters that control many parts of the algorithm's way of behaving to get the optimum values and reach the optimization. A portion of these hyperparameters influence the time and memory cost of running the algorithm. Some of these hyperparameters influence the nature of the model capacity and the model quality by the training process and the ability to get the predicted results when a new dataset is used. There are two ways to search for these hyperparameters: Automatic hyperparameter search and Manual hyperparameter search.

4.3.1 Automatic Hyperparameter Search

- Random search

  Describe a search space as a bounded domain of hyperparameter values and randomly select points within that domain.

- Grid search

  Creates a grid of hyperparameter values to represent a search space, and then analyses each point in the grid. An estimator using grid search must do an exhaustive search across the provided hyperparameter values. Grid search is a challenge since it needs significant computer resources to examine more parameters.

- Bayesian search

  Create a probability model for the objective function, then use it to choose the most promising hyperparameters to test against the real goal function.

4.3.2 Manual Hyperparameter Search

Utilizing Manual search requires knowledge, experience and understanding of what hyperparameters, and their relationships really do and how machine learning algorithms accomplish a satisfied generalization. Understanding the relationship between hyperparameters such as learning rate and batch size, training error, validation error, test error, generalization error, and available computing resources such as GPU, CPU, memory and run time setting up a strong establishment on the basic thoughts concerning the successful capacity of a learning algorithm. The benefit of manual search is to find the

proper hyperparameters for the learning algorithm to reach the optimal capacity or effective capacity. Optimal capacity can be reached by increasing the representational capacity of the model by adding more neurons in the hidden layers to accommodate more complicated functions, and by minimizing the cost function of the model to reach the generalization error.

4.4 Regularization Methods

Regularization is a family of techniques that provide more information to an ill-posed issue to change it into a more stable, well-posed problem for optimization. The most basic and possibly most often used regularization strategy is to apply a penalty to the loss function according to the amount of the weights in the model. The most common regularization methods are regularization L1, L2 are used with weight decay, early stop, dropout, and data augmentation. Dropout was used in the model by 20% after trying other percentages and got the best results in terms of low error and high accuracy. Data augmentation was created using AHWD dataset and used in the model. The more training the model the best results are produced. So, the augmented data was used and got the best results of the model as explained in chapter 7.

4.4 Model Capacity

The capacity of deep neural network model determines the range of mapping functions that it can learn. There are two characteristics of a model that can influence the neural network's capacity: number of nodes and number of layers. By expanding the model's capability, underfitting problem can be solved. When a model has greater capacity, it can perform a wider range of functions for mapping inputs to outputs. Capacity describes the model's ability to perform a range of tasks by altering the model's structure, and by including extra layers and/or nodes. It is more typical to have an overfit model since an underfit model is so readily rectified. Monitoring the model's performance throughout training by assessing it on both a training dataset and a holdout validation dataset makes it simple to identify an overfitting model. By plotting graphs, the effectiveness of the capacity during training would be noticeably clear, these graphs are called learning curves [68] [69].

The model learns from existing samples and generalizes from those existing examples to upcoming examples. To measure the model's capacity for generalization, techniques like train/test split and k-fold cross-validation are employed. It is challenging to learn new things and apply them to new situations. The model will perform badly on both training dataset and on fresh data if there is insufficient learning and if the model learns too much, it will perform well on the training examples but badly on new data, over-analyzing the problem. So, the model has no generalization in either situation. There are three situations where the model can go [64] [68] [69].

- Underfitting model: a model that is unable to adequately learn the problem, performs badly on a training dataset, and is unsatisfactory on a holdout sample.
- Overfitting model: a model that performs well on the training dataset but poorly on a holdout sample because it learns the training dataset too well.
- Good fitting model: a model that correctly notices the training dataset and applies effectively to the testing dataset.

Having a lot of layers can frequently boost the model's capability, functioning as a computational and learning shortcut to modelling a problem. A model with one hidden layer of twenty nodes, for example, is not equal to a model with two hidden levels of ten nodes each, the latter has much more capacity. The concern is that a model with more capacity (too many nodes,) than needed may overfit the training data. Similarly, to a model with too many layers, it will be unable to learn the training dataset, thereby becoming lost or stuck during the optimization phase. In general, models with a higher number of parameters are said to have high capacity, and they require a bigger amount of data to obtain generalization power to unknown test data [68] [69].

According to [68] [69], the complexity of models is a fundamental issue in deep learning. They did a thorough review of the most recent papers on deep learning model complexity. Deep learning model complexity may be divided into expressive capacity and effective model complexity. They examine previous research on those two categories in terms of four key factors: model framework, model size, optimization technique, and data complexity. Furthermore, they reviewed current studies on effective complexity from two perspectives: broad measurements of effective complexity and the high-capacity low-reality issue. There was a discussion over the use of deep learning model complexity,

particularly in generalization capabilities, optimization, model selection, and design. Deep learning model complexity is still in its infancy. There are several intriguing difficulties for future works. They also explore deep learning model complexity applications such as comprehending model generalization, model optimization, and model selection and design.

CHAPTER 5 EXPERIMENTAL SETUP

This chapter outlines the processing platforms and the proposed DCNN model implementation,

5.1 Processing Platform and Frameworks

This section explains how the work was done in terms of training and testing the model with standalone machine and online server. The comparison between the two platforms is important in terms of the resources needed in machine learning. The standalone machine was limited in executing the model whereas the online server was good in executing the model.

5.1.1 Portable computer

Initially, a Lenovo portable computer with the following specifications was used:

- Processor Intel Core^{TM} i5-3230M CPU @ 2.60 GHz,
- Installed memory (RAM) 8.00 GB,
- System type: 64-bit Windows 8.1 Operating System with x64 based processor,
- and Hard disc capacity 500 Gb.

Then Anaconda Navigator (anaconda3) was installed and the Jupyter notebook was used for to provide the interface. This system did not satisfy the required amount of resources to run the DCNN and an alternative environment was sought.

5.1.2 Online Server

Due to the limited resources a final decision was made to subscribe with Google Colab pro+ with unlimited resources and quick processing, and access to GPU. The work was much easier and more accurate when all three datasets were used.

5.1.3 Frameworks

Keras and Tensorflow were used as the framework. Keras is the high-level Application Programming Interface (API) of TensorFlow 2.

5.2 Activation Functions

The activation function is a transfer function. It is a mathematical method used by each neuron in the neural network to activate. It fires the output of the neuron to the next neuron or to the network output if it satisfies the condition. By using an activation function, a non-linearity input transformation in the Neural Network is there. Activation functions must be well chosen; each model has many appropriate activation functions that are used to get accurate output and make the model simple and function well [67] [70]. There are many activation functions; such as sigmoid function, tangents hyperbolicus function (tanh), Rectified Linear Unit (ReLU) and Softplus.

5.2.1 Sigmoid Activation Function

The sigmoid function is a nonlinear activation function and transforms the values between 0 and 1. The formula of the sigmoid function is represented as:

$$f(x) = \frac{1}{1 + e^{-1}}$$

There are two problems with using sigmoid function. First, the upper and lower slope tail are almost equal to one and zero, respectively. This is called sigmoid function saturability due to an exceptionally large or an exceedingly small input values which in turn makes the value of the gradient almost zero. Second, the output of sigmoid functions is not zero centered; this means if all the data entering a neuron always has positive value, then the gradient value would be all negative values or all positive values during the training stage (backpropagation). So, it is not suitable for our model [71].

5.2.2 Tangents Hyperbolicus Activation Function (TanH)

The tanh function is a nonlinear activation function and due to the drawbacks with the sigmoid function as explained in the previous section, there was too much work to overcome these problems. The tanh function is considered an enhanced version of sigmoid function, overcame the problem of non-zero-centered output to become symmetric output (zero centered). The formula of the tanh function is represented as:

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The range of the tanh function values is between -1 and 1, so the inputs to the layers would be negative or positive. The slope of the tanh function finds the differentiation value of that point. The gradients are free to move in a different direction, which tells that the gradient diffusion problem still around.

5.2.3 Rectified Linear Unit Activation Function (ReLU)

The ReLU function is a nonlinear activation function and unsaturated because the issue of the gradient diffusion is resolved. The output value of ReLU function is zero if the input is negative or zero, and the output value would be the same if otherwise. The formula of ReLU function is represented as:

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{Otherwise} \end{cases} \quad \text{Or} \quad f(x) = \max(o, x)$$

The ReLU function makes the model learn amazingly fast and perform much better than sigmoid and tanh functions due to the resolved gradient problem. ReLU function is considered as a default activation function in multilayered neural networks and Convolutional Neural Networks (CNN). Based on all the previous advantages, ReLU was chosen in our model.

5.2.4 Softplus Activation Function

The softplus function is a nonlinear activation function and unsaturated because the issue of the gradient diffusion is resolved. It functions as ReLU activation function. The formula of the softplus function is represented as:

$$f(x) = \ln(1 + e^{x})$$

5.3 Deep Convolutional Neural Network model (DCNN)

This section defines the Deep Convolutional Neural Network model (DCNN) used for Arabic handwritten words pattern recognition and outlines how the model was implemented, and which layers, functions and methods were used to improve the performance on the three datasets: AHWD, IFN/ENIT, AND augmented AHWD.

The proposed model was finalized with an input layer, three hidden layers and a fully connected layers after building many others models with four hidden layers, five hidden layers, and seven hidden layers. After traing and testing all the models, the conclusion was to have less number of layers with many filters in terms of Deep Convolutional Neural Network. The proposed model was chosen because it has given the best results of all measures that were used in the model such as accuracy rate and testing error rate (Generalization Error Rate).

5.3.1 Input Layer

In Keras, the input layer is a tensor rather than a layer, the beginning tensor is transmitted to the first hidden layer. For simplicity it is called an input layer. The input layer is a pixel image with size H x W x D, where H is the height, W is the width, and D is the depth. Since the image is the main core here, there is a consideration whether it is colored or grayscale, which can be done by adding 3 as colored RGB image or by adding 1 as grayscale image. It is critical to understand that input images are represented as arrays of hundreds, thousands, or millions of pixels. Each pixel is represented by a single point and may differ in color from its neighbors. Grayscale images are used in our model with the size of (32 x 64 x 1) pixels in a two-dimensional array format analogous to a matrix in grayscale. Specifically, the computer records the values that describe each image's pixel. Keras will require the input shape in the first layer because it is the only one that must be defined when implementing the model. So, our input_shape = (32,64,1), this is the representation of the grayscale input images.

5.3.2 Hidden Layers

In this section, the main components of hidden layers in CNN are outlined and how do they work with each other in the proposed model. Hidden layers are composed of convolutional layers, Pooling layers, and fully connected layers.

Convolutional layer is an essential part of CNN components, and its main function is to extract the features from the input image (input matrix, raw image, or raw matrix H x W x D) by using weighted filters (kernels, or feature detector). These features could be edges, dots, endpoints, corners, ascending and descending letters, lower and higher

diacritic dots, diacritic marks, and letter loops. All the information of these features are saved from loss by applying padding in all convolutional layers.

The filter size is identified by (N x N x R) where N is the height, and the width and R is the number of channels. Our model is composed of three convolutional layers. The first convolutional layer is loaded by an input matrix as an input shape (32 x 64 x 1), this is only done when the new image is loaded. After loading the raw image, the size and the number of filters to be used are determined , the start is to use 32 filters with a size of (3 x 3), then a multiplication process would start by multiplying each element in the kernel (3 x 3 x 32) with each element in the input matrix (binary image input) and sum each result of multiplication and save it in the feature (activation) map, then repeat the same procedure by shifting one pixel right each time and doing the multiplication with the filter until the last pixel then save the result in the feature map.

Next, one pixel is shifted down (if stride = 1) from the beginning of the input matrix and do the multiplication and save the result in the feature map as shown in Figure 5. 1 and Figure 5. 2. this way is continued until the end of the input matrix. The feature map with (32 x 64 x 32) has been created after applying a ReLU activation function.

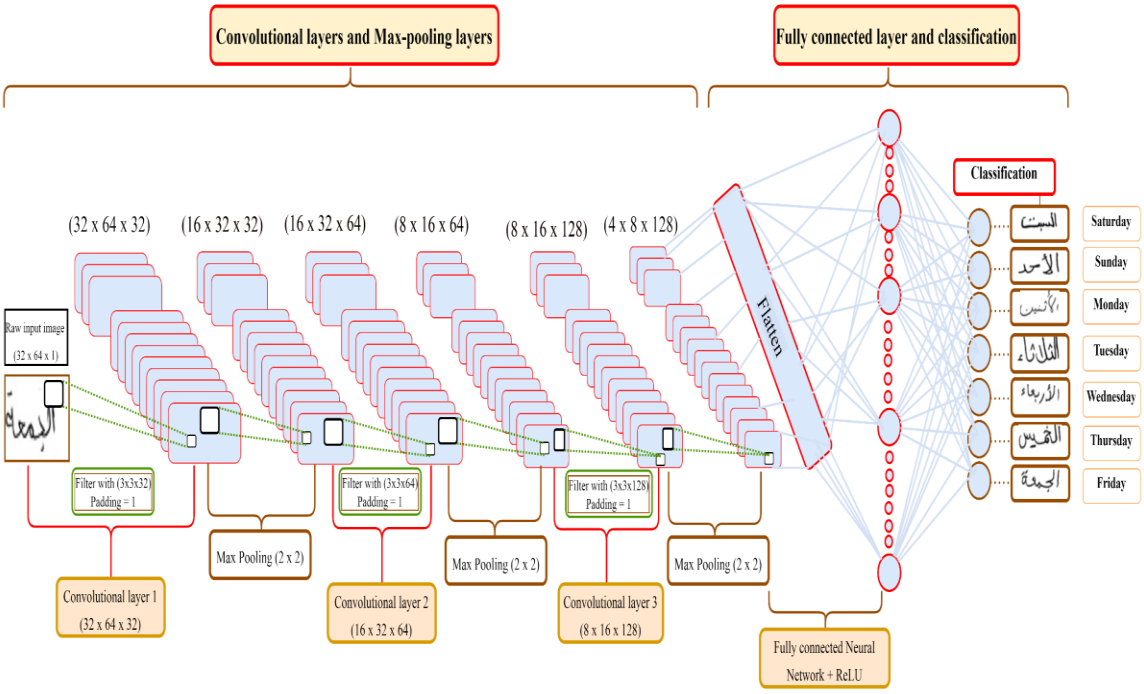


Figure 5. 1 DCNN of our model using AHWD.

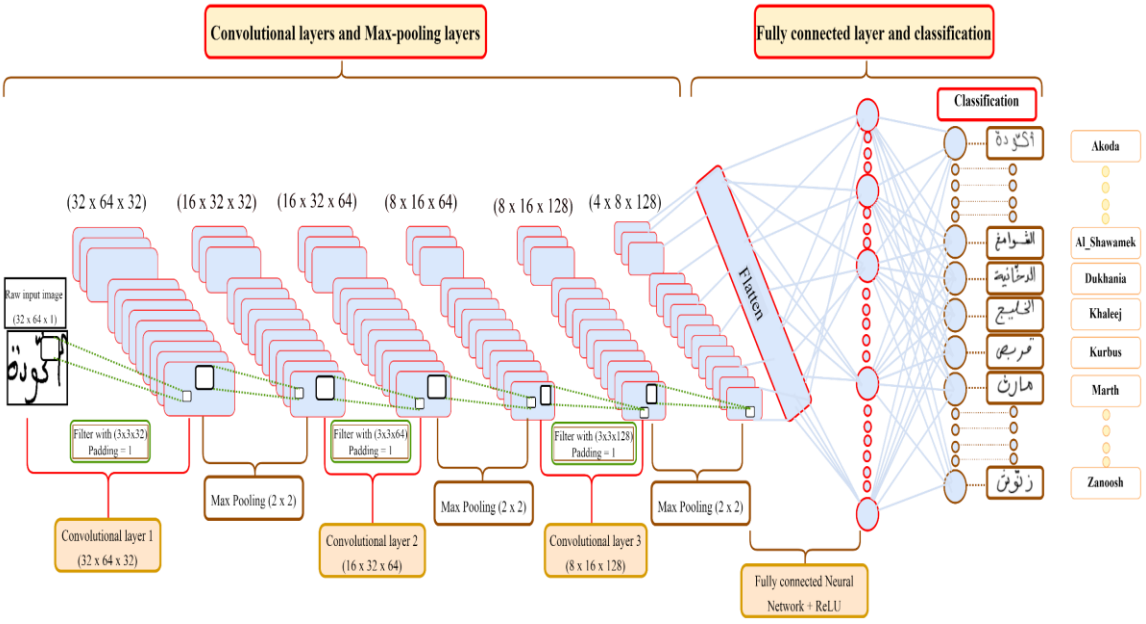


Figure 5. 2 DCNN of our model using IFN/ENIT.

The pooling layer comes after the feature map and watch the features in the perception scope and extracts the dominant features in the area to lower the number of hyper-parameters such as filter size, padding, and Pooling method and in turn reduce the inner dimensionality of the feature map (subsampling).

After getting the feature map with (32 x 64 x 32), max-pooling with pool-size = (2 x 2) is applied to get another feature map with (16 x 32 x 32). In the second convolutional layer, the input would be the feature map with (16 x 32 x 32) and by applying 64 filters with the size of (3 x 3) and ReLU activation function a new feature map with (16 x 32 x 64) is produced, and by applying max-pooling with pool-size = (2 x 2) a new activation map with (8 x 16 x 64) is produced. In the third (final) convolutional layer, the input would be the feature map with (8 x 16 x 64) and by applying 128 filters with the size of (3 x 3) and ReLU activation function a new feature map with (8 x 16 x 128) is created, and by applying max-pooling with pool-size = (2 x 2) a new activation map with (4 x 8 x 128) is created.

5.3.3 Fully Connected Layer.

After all the calculation in terms of summing learned features weights in the previous layers (convolutional layer and pooling layer), there is a third layer called a fully connected layer with ReLU activation function which is constructed of many neurons. Each neuron is connected to all other neurons as a fully connection. The output from the final convolutional layer would be flattened and converted into one dimensional array of vectors and is passed to fully connected layer where each input with trainable weight is hooked up with an appropriate output as shown in Figure 5. 1 and Figure 5. 2. Flattening is considered as an input layer for the Artificial Neural Network (ANN).

5.3.4 Output Layers

In the proposed model, each dataset has its own number of classes, when using AHWD and augmented AHWD the output layer is 7 class softmax layer. On the other hand, when using IFN/ENIT dataset the output layer is 21 class softmax layer. The final stage is the classification where the softmax works as classifier to classify all the features to its labeled class.

5.4 Hyperparameters

Manual hyperparameter search was used to find the best values of hyperparameters and their relationships with each other to reach the optimal model by observing the results and adjusting the values of the hyperparameters. In this section, a complete study would be done in the hyperparameters used in the proposed model.

5.4.1 Learning Rate

The learning rate is a hyperparameter that specifies how much the model should change in response to the predicted error each time the model weights are updated. In the proposed model, trial and error approach is used with five different values of learning rate (10^{-3} to 10^{-7}), each one is associated with seven batch sizes (16, 32, 64, 128, 256, 512, 1024), and all the experiments are done using the three datasets: AHWD, IFN/ENIT, and augmented AHWD. Results are analyzed in the next chapter 6.

5.4.2 Batch Size

Seven batch sizes (16, 32, 64, 128, 256, 512, 1024) were used in the proposed model. Since the manual hyperparameter is used, it is noticeably clear to report how the model performs in each batch size. Batch normalization was not used in the proposed model because Batch Normalization has traditionally performed badly when the batch size is too small [72].

5.4.3 Random_state=42

Random_state is set to 42 as a default value. However, any integer can be used if we want the system to be deterministic. Moreover, if the system runs every time without specifying the value of random_state, the results would be different each time, the system is not deterministic.

5.4.4 Validation_split=0.2

Since Keras is used in this proposed model, 20% of training datasets are set into a validation dataset and test the performance of the proposed model on that validation dataset in each epoch to tune the hyperparameters. It might be useful to visualize the effect of a single hyperparameter on the training and validation scores to see whether the estimator is overfitting or underfitting for certain hyperparameter values.

5.4.5 Epochs

In machine learning, an epoch is defined as one full iteration of the training dataset through the algorithm. The number of epochs in the proposed model is equal to 3000 epochs as a standard number to avoid any variation in the results. These results are evaluated by the accuracy rate, error rate, the convergence between the training accuracy curve and the validation accuracy curve in the learning curve plot, the convergence between the training loss curve and the validation loss curve in the learning curve plot, and confusion matrix.

5.4.6 Relationships between Learning Rate, Batch Size, and Epoch

Increasing the learning rate accelerates the model's learning but risks exceeding its minimal loss. By reducing batch size, the model utilizes less data to calculate the loss in

each training process. When learning rate is low, batch size is small, and high epoch number, the system would learn in a slow manner. If the learning algorithm has fine-tuned learning rate and fine-tuned batch size but with a small number of epochs, the system may not perform well and has a bad generalization. Since the manual hyperparameters are used in this proposed model, the results are checked and compared with the previous results.

Chapter 6 Result Analysis

This chapter discusses the performance of the DCNN algorithm for three experimental phases for each dataset: AHWD, IFN/ENIT, and augmented AHWD. All results are analysed and discussed to choose the best accuracy rate and the lowest error rate or the Generalization Error Rate (GER). A comparison is also made with current state-of-the-art.

6.1 AHWD Experimental Phase

As mentioned in Chapter 5, the model is evaluated by using the testing dataset. On AHWD, different batch sizes equal to 16, 32, 64, 128, 256, 512, and 1024 were tested. Also, learning rates of 10^{-3}, $10^{-4}, 10^{-5}, 10^{-6}$, and 10^{-7} were evaluated with epochs equal to 3000. With each learning rate, the model was trained with 7 different batch sizes. Five tables of results were produced, and each table consists of one learning rate and 7 batch sizes with results (training iteration number, testing iteration number, accuracy rate, loss rate, and confusion matrix errors) where the loss rate here is considered as the GER. The results between the five tables are evaluated based on the following criteria:

- Learning rate values (10^{-3} to 10^{-7}) and batch sizes.
- A low loss error rate or Generalization Error Rate (GER) has the highest priority.
- High accuracy rate.
- Confusion matrix errors.
- Model response speed by looking at the learning curve graph and determine in which epoch number the accuracy curve starts rising and in which epoch number does the loss curve start coming down.

After the evaluation by using these criteria, the best performance from each table has been collected.

First, when learning rate = 10^{-3} and batch sizes (16, 32, 64, 128, 256, 512, 1024) are used, the results would be as shown in Table 6.1.

By looking at Table 6.1, the best accuracy rate is 99.86%, the lowest error rate is 0.5773, and the lowest confusion matrix error number is 9.

Table 6.1 AHWD dataset with learning rate = 10^{-3}.

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	716	441	0.9973	658.0825	12
32	3000	358	221	0.9970	249.3682	11
64	3000	179	111	0.9980	158.4381	11
128	3000	90	56	0.9983	51.5822	8
256	3000	45	28	0.9986	5.9887	10
512	3000	23	14	0.9982	3.4738	10
1024	3000	12	7	0.9986	0.5773	9

This conclusion is acceptable in this training and testing condition but cannot be generalized because Figure 6.1 shows an overfitting after epoch 2000; that is, the validation loss curve starts to diverge up off the training loss curve. In conclusion, the model generalization cannot be achieved for the AHWD dataset with a learning rate = 10^{-3} and for batch sizes 1024.

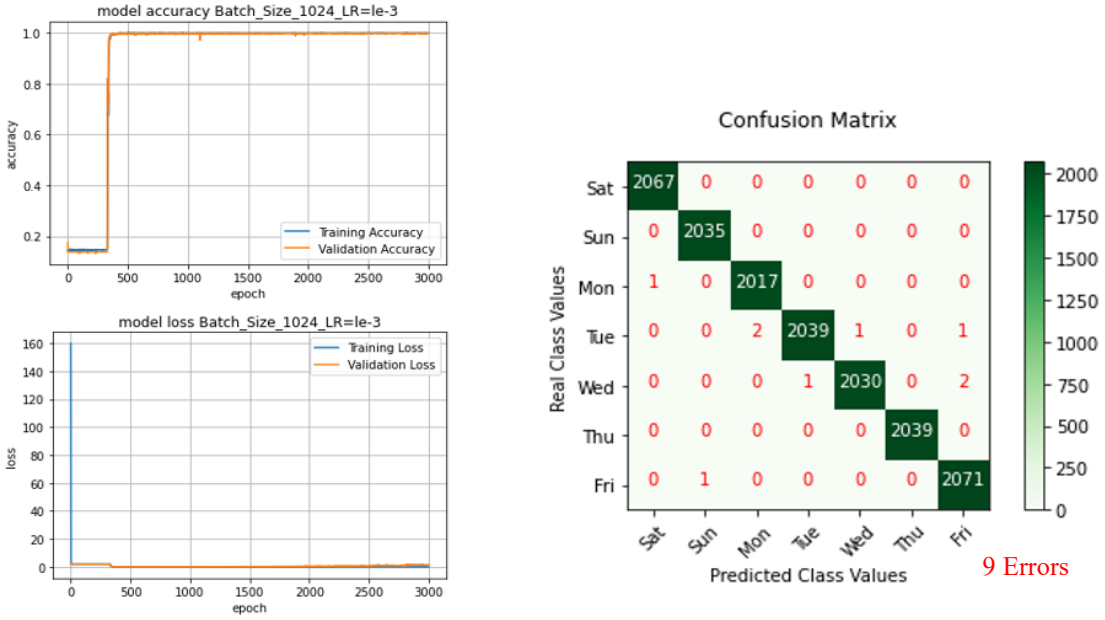


Figure 6.1 LC and CM with LR = 10^{-3} and BS = 1024 AHWD.

All other results in Table 6.1 have very high error rates, and this leads to overfitting and cannot be generalized as shown in Figure 6.2 to Figure 6.7.

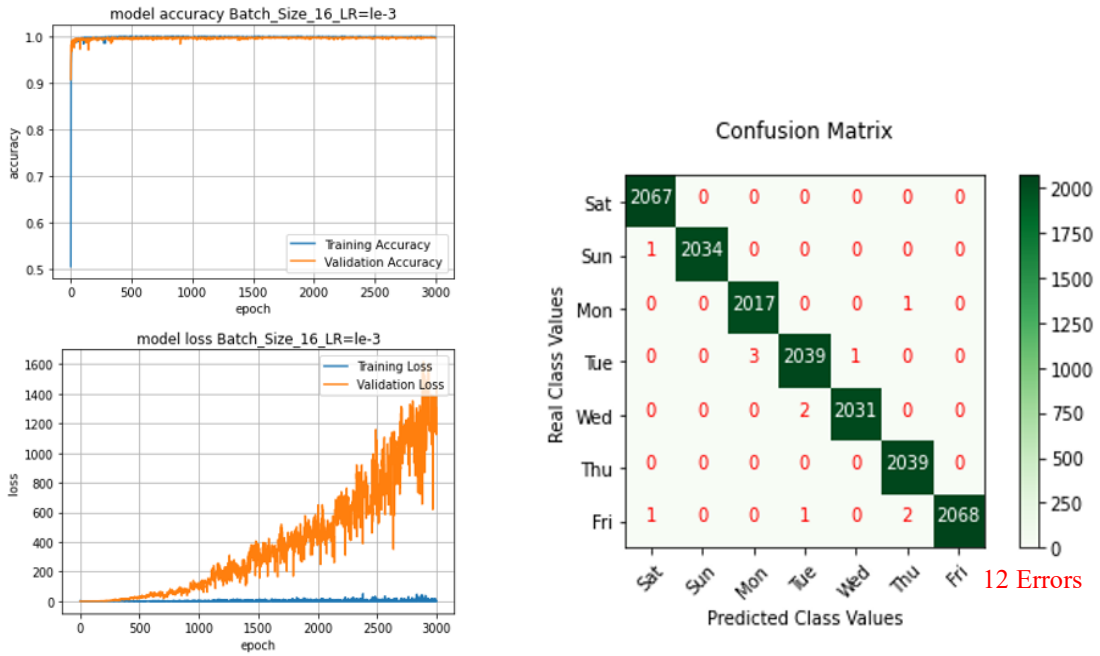


Figure 6.2 LC and CM with LR = 10^{-3} and BS = 16 AHWD.

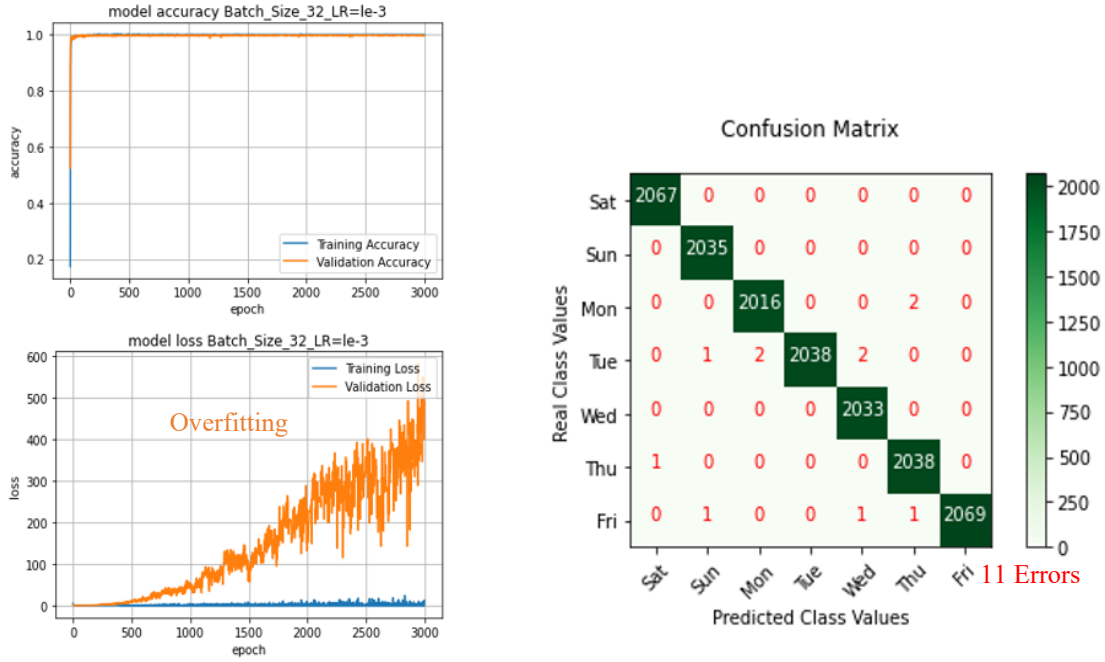


Figure 6.3 LC and CM with LR = $10^{-3}$and BS = 32 AHWD.

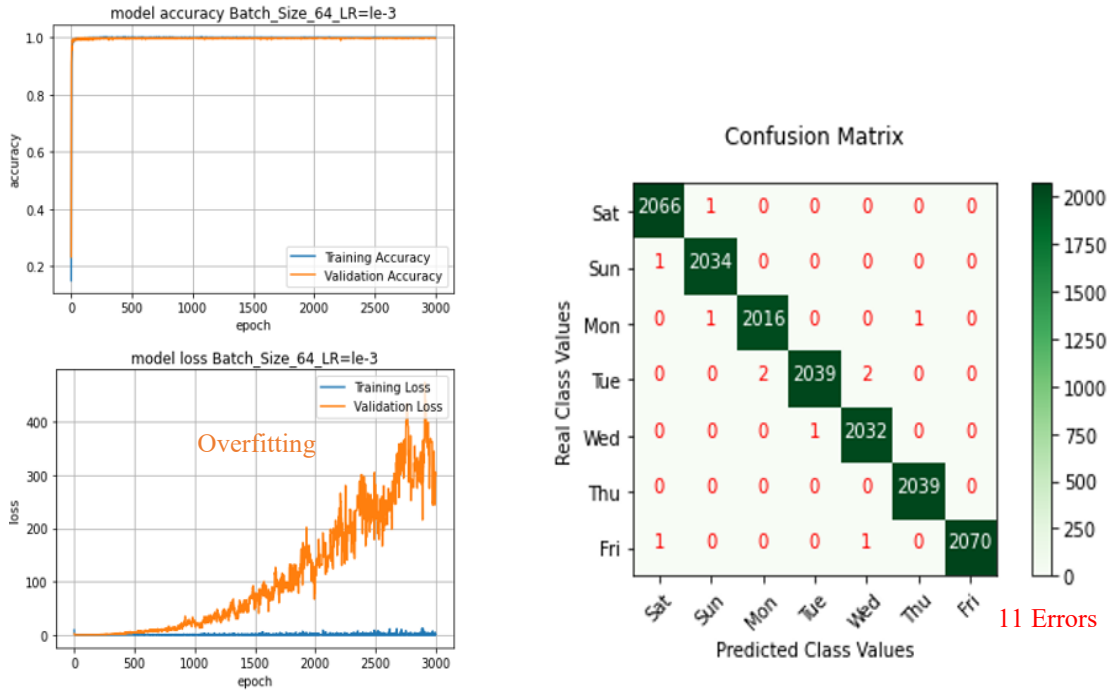


Figure 6.4 LC and CM with LR =10^{-3} and BS = 64 AHWD.

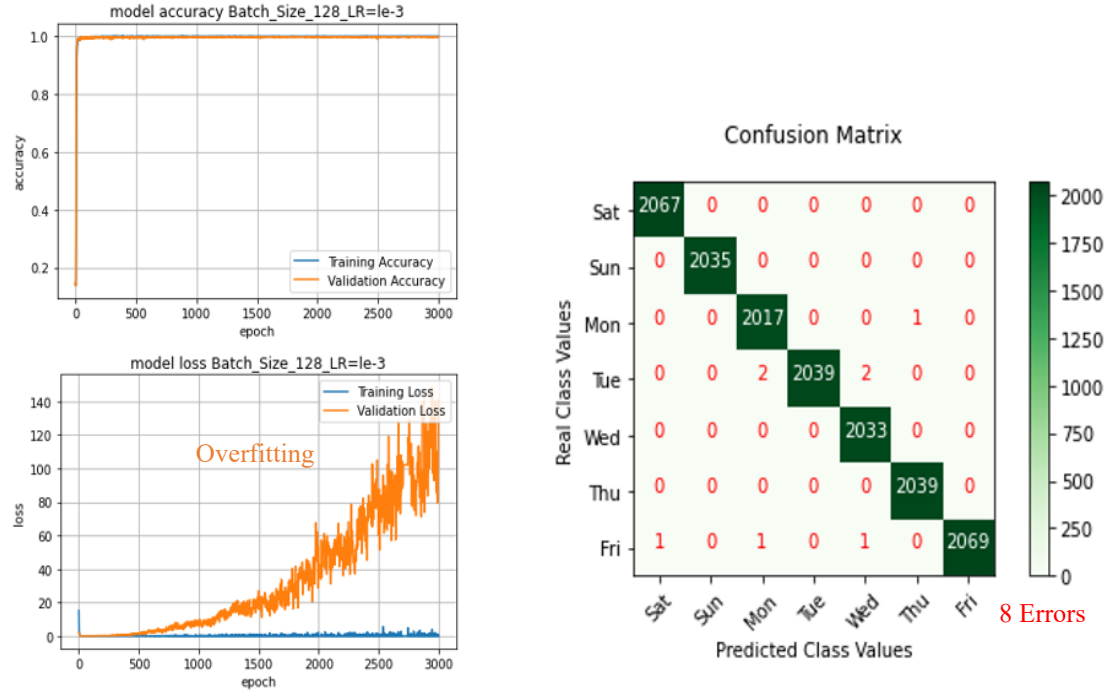


Figure 6.5 LC and CM with LR = 10^{-3} and BS = 128 AHWD.

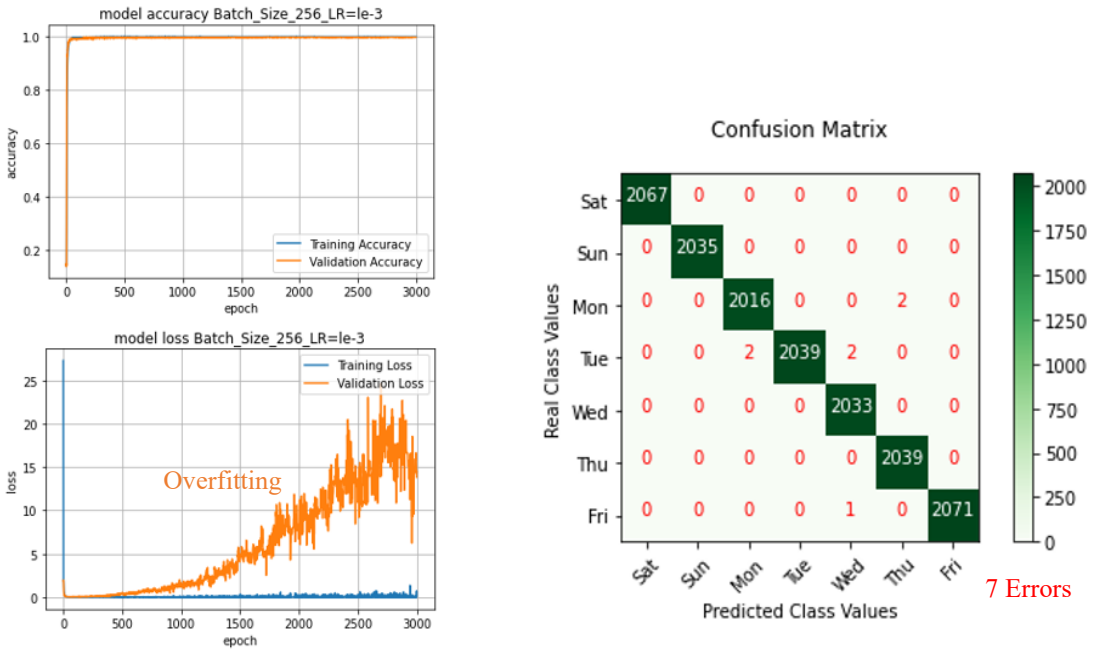


Figure 6.6 LC and CM with LR = 10^{-3} and BS = 256 AHWD.

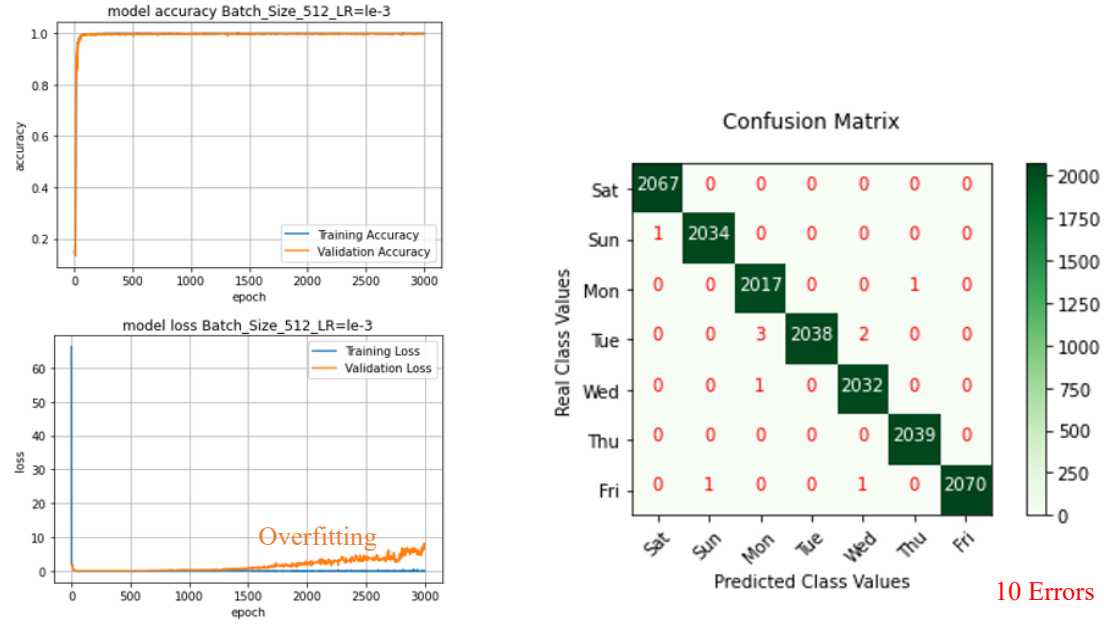


Figure 6.7 LC and CM with LR = 10^{-3} and BS = 512 AHWD.

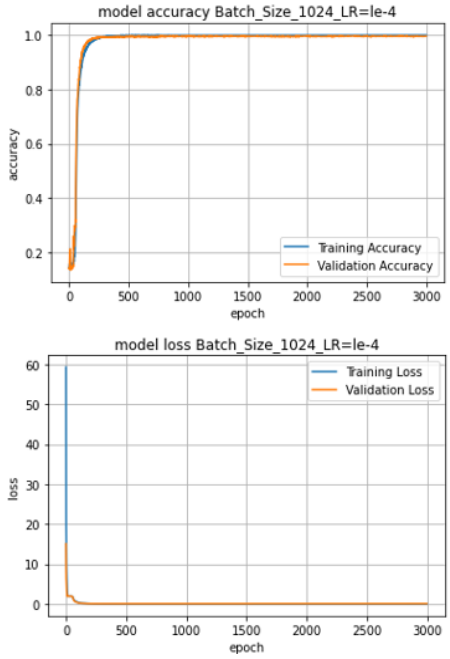
Second, when learning rate = 10^{-4} and batch sizes (16, 32, 64, 1024) are used, the results are shown in Table 6.2.

By looking at Table 6.2 and by rounding up, all the accuracy rates are the same as 99.70%. So, by choosing the lowest error rate (GER) = 0.0349, a low confusion matrix error number = 9 is obtained, and the lowest iteration is for (training =12 and testing = 7) which in turn takes less computing time. This conclusion is the best and is accepted in this training and testing session with batch size = 1024 as shown in Figure 6.8 where the convergence between the training accuracy curve and validation accuracy curve starts approximately at epoch = 300 and continue all the way straight. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 100 where the convergence started to be steady, and the error is the lowest.

Table 6.2 AHWD dataset with learning rate = 10^{-4}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	716	441	0.9973	0.6848	9
32	3000	358	221	0.9977	0.3508	15
64	3000	179	111	0.9966	0.2413	11
128	3000	90	56	0.9965	0.1557	10
256	3000	45	28	0.9969	0.0926	8
512	3000	23	14	0.9970	0.0520	11
1024	3000	12	7	0.9965	0.0349	9

In Table 6.2, there are some low error rates 0.0926 and 0.0520 with no overfitting as shown in Figure 6.10 and Figure 6.10 respectively. However, there are error rates 0.6848, 0.3508, 0.2413 and 0.1557 with overfitting as shown in Figure 6.11 to Figure 6.14 respectively. None of these results were chosen because they are not the best.



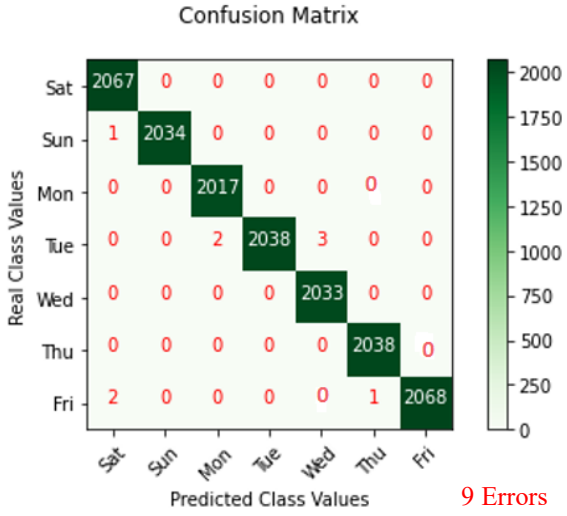
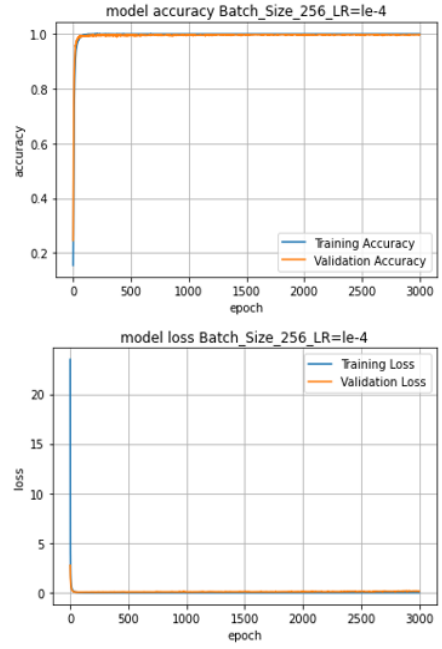


Figure 6.8 LC and CM with LR = 10^{-4} and BS = 1024 AHWD.



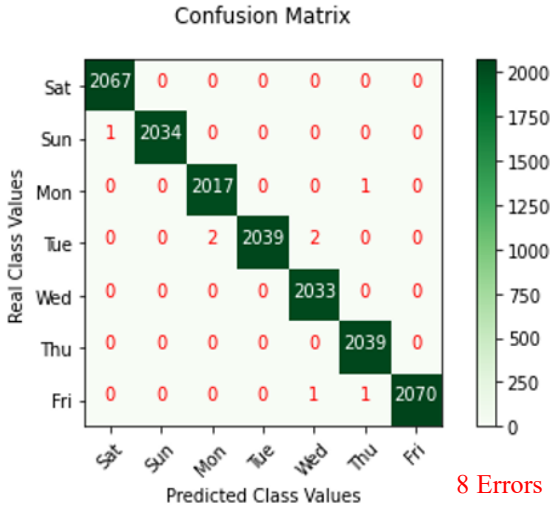


Figure 6.9 LC and CM with LR = 10^{-4} and BS = 256 AHWD.

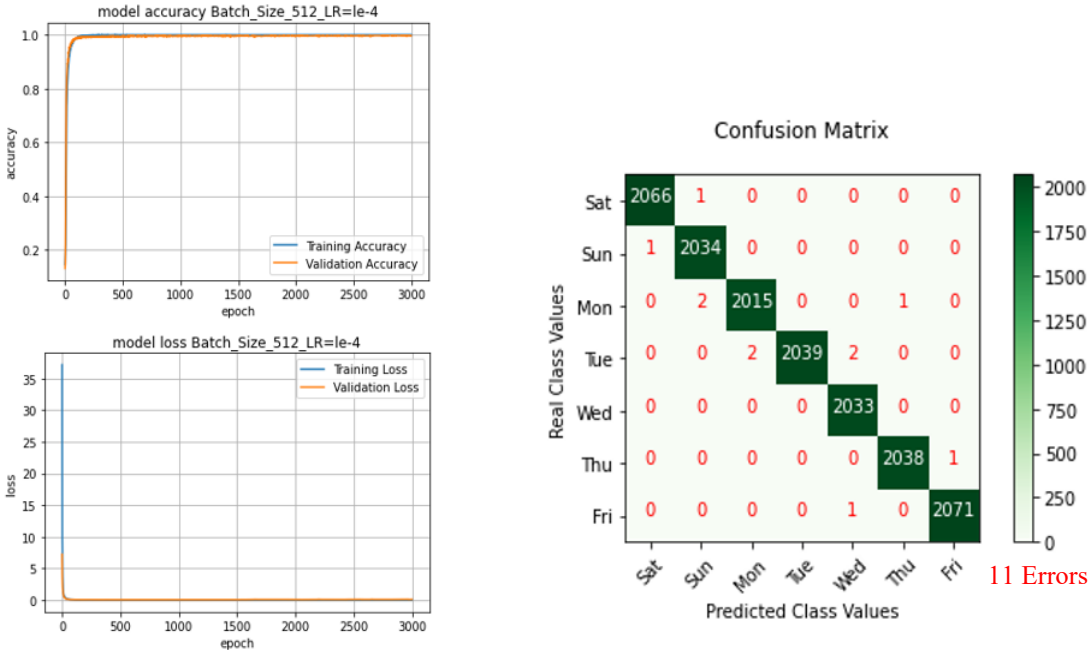


Figure 6.10 LC and CM with LR = 10^{-4} and BS = 512 AHWD.

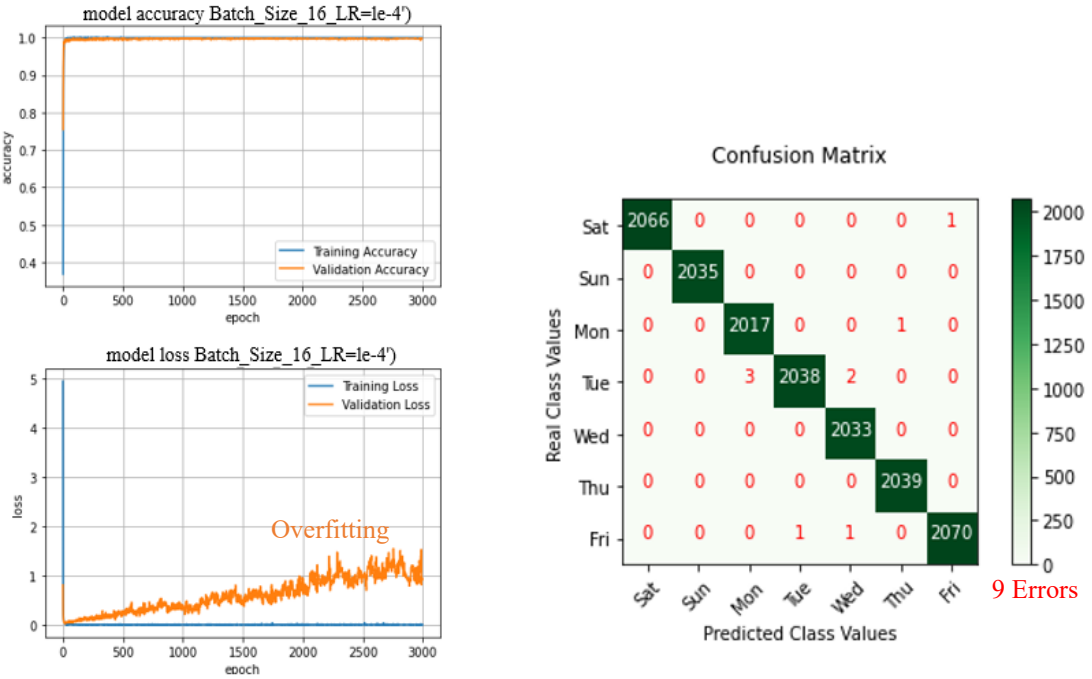


Figure 6.11 LC and CM with LR = 10^{-4} and BS = 16 AHWD.

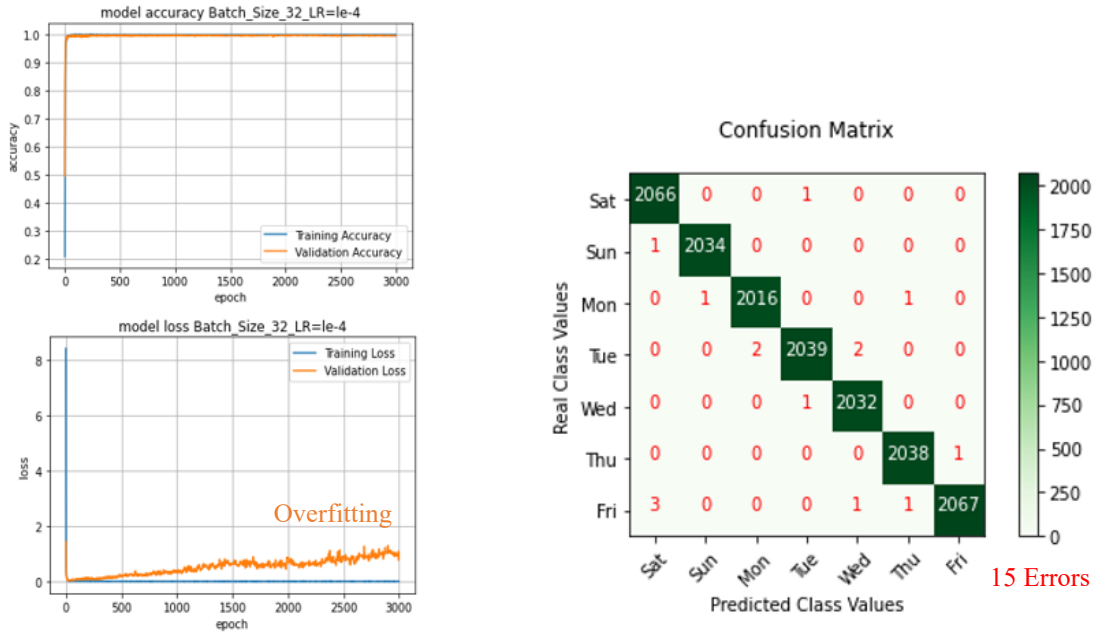


Figure 6.12 LC and CM with LR = 10^{-4} and BS = 32 AHWD.

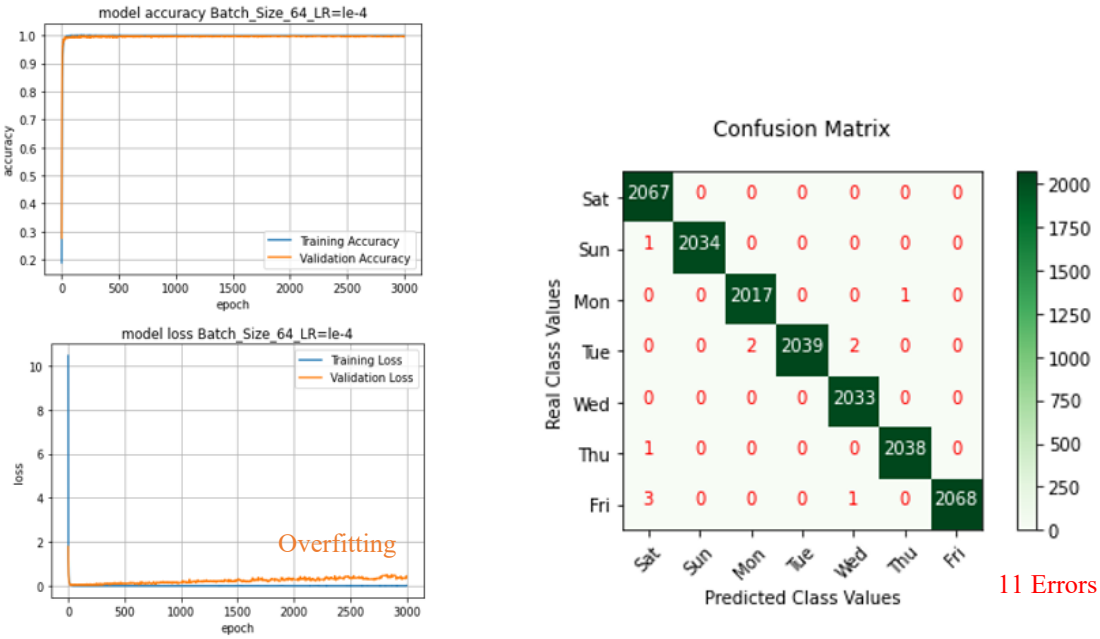


Figure 6.13 LC and CM with LR = 10^{-4} and BS = 64 AHWD.

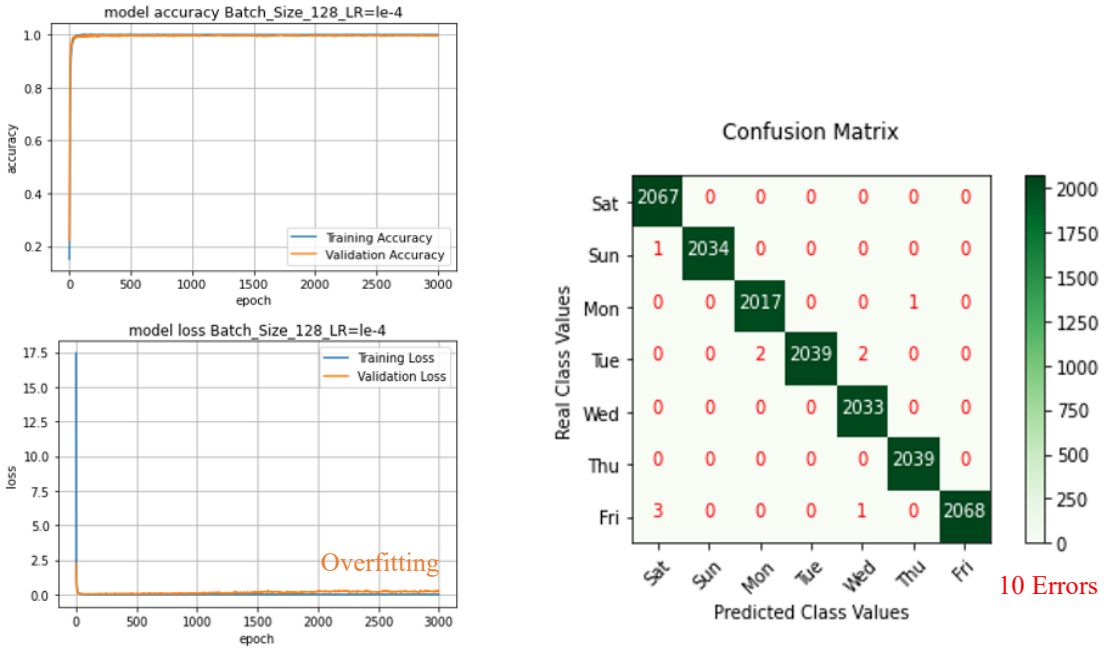


Figure 6.14 LC and CM with LR = 10^{-4} and BS = 128 AHWD.

Third, for learning rate = 10^{-5} and batch sizes (16, 32, 64, 1024), the results are shown in Table 6.3.

By looking at Table 6.3, the highest accuracy rates are 99.76%, with low error rate (GER) = 0.0230, low confusion matrix error number = 11 errors, and low iteration (training =23 and testing = 14) which in turn takes less computing time. This conclusion is the best performance and is accepted in this training and testing session with batch size = 512 as shown in Figure 6.15, where the convergence between the training accuracy curve and validation accuracy curve starts approximately at epoch = 400 and continue all the way straight. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 200 where the convergence started to be steady, and the error is the lowest.

Table 6.3 AHWD dataset with learning rate = 10^{-5}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	716	441	0.9957	0.0555	11
32	3000	358	221	0.9965	0.0522	16
64	3000	179	111	0.9972	0.0342	16
128	3000	90	56	0.9963	0.0299	14
256	3000	45	28	0.9963	0.0280	15
512	3000	23	14	0.9976	0.0230	11
1024	3000	12	7	0.9965	0.0194	14

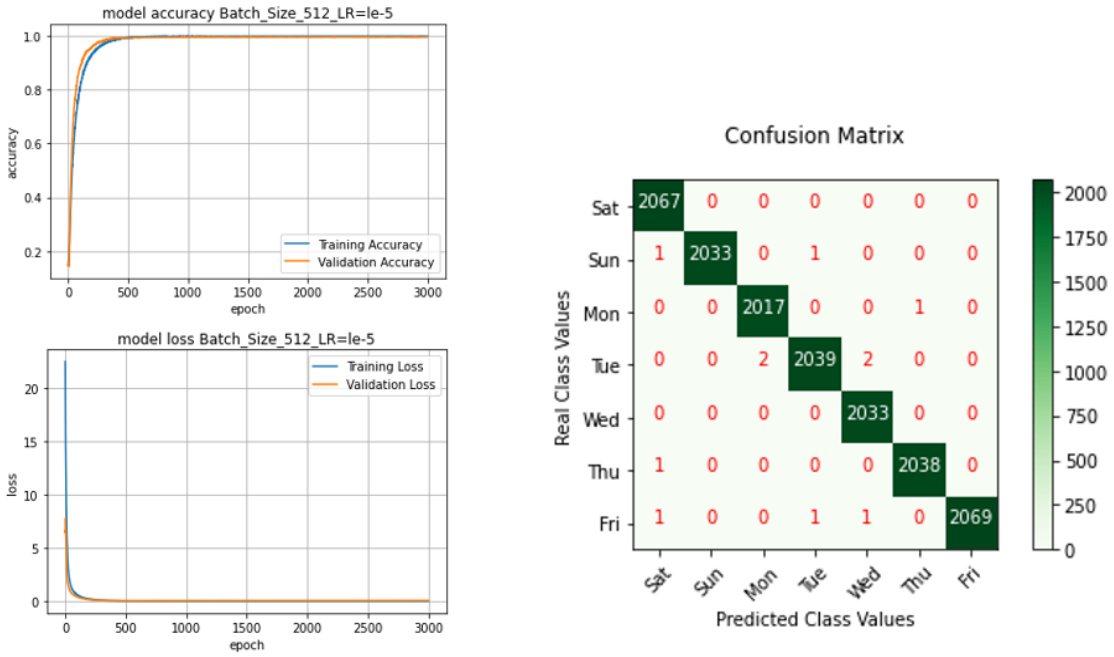


Figure 6.15 LC and CM with LR = 10^{-5} and BS = 512 AHWD.

In Table 6.3, all error rates are considered low, and all accuracy rates are high, too. The error rate 0.0191 with batch size 1024 and accuracy rate is the lowest but has been excluded because of the slowing performance in model response speed as shown in Figure 6.16, where the training accuracy curve and validation accuracy curve start raising up in

convergence fashion at epoch 350 until approximately epoch 1,000 where the convergence goes all the way long, whereas in Figure 6.15 the convergence the convergence between the training accuracy curve and validation accuracy curve starts approximately at epoch = 400 and continue all the way straight. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 200 where the convergence started to be steady, and the error is the lowest.

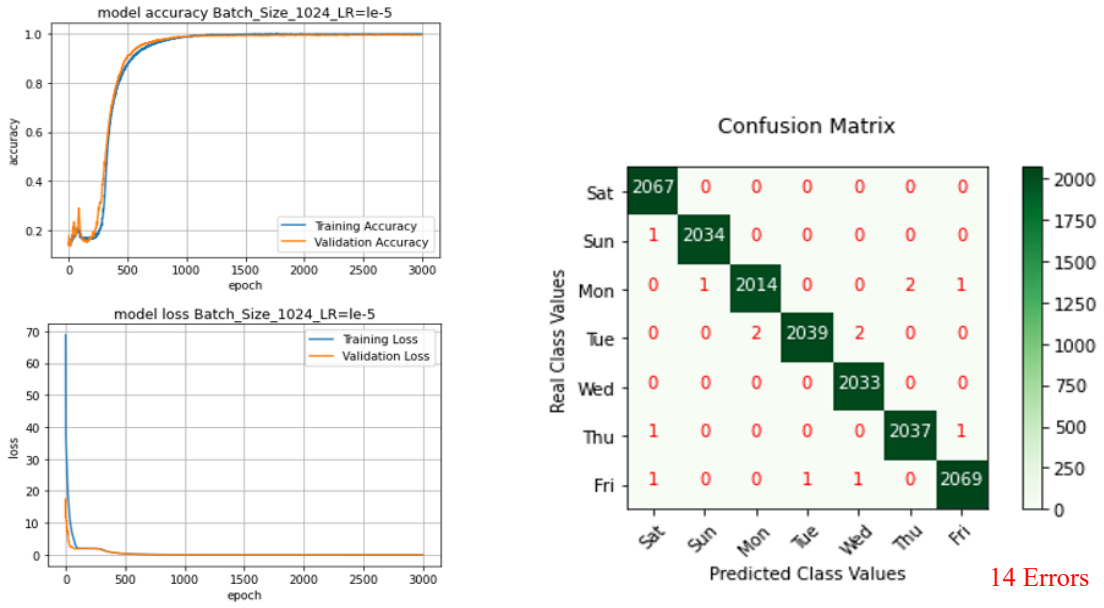


Figure 6.16 LC and CM with LR = 10^{-5} and BS = 1024 AHWD.

Fourth, for a learning rate = 10^{-6} and batch sizes (16, 32, 64, 128, 256, 512, 1024), the results are shown in Table 6.4.

By looking at Table 6.4, the highest accuracy rate is 99.72%, with lowest error rate (GER) = 0.0177, and low confusion matrix error number = 16 errors. This conclusion is the best performance and is accepted in this training and testing session with batch size 64 as shown in Figure 6.17, where the training accuracy curve and validation accuracy curve start raising up in convergence fashion from the beginning until approximately epoch 1000 where the convergence goes all the way long, Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the

beginning until it reached epoch 400 where the convergence started to be steady, and the error is the lowest.

Table 6.4 AHWD dataset with learning rate = 10^{-6}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	716	441	0.9957	0.0264	15
32	3000	358	221	0.9955	0.0288	16
64	3000	179	111	0.9972	0.0177	16
128	3000	90	56	0.9953	0.0231	18
256	3000	45	28	0.9934	0.0219	20
512	3000	23	14	0.9872	0.0473	111
1024	3000	12	7	0.9725	0.0948	360

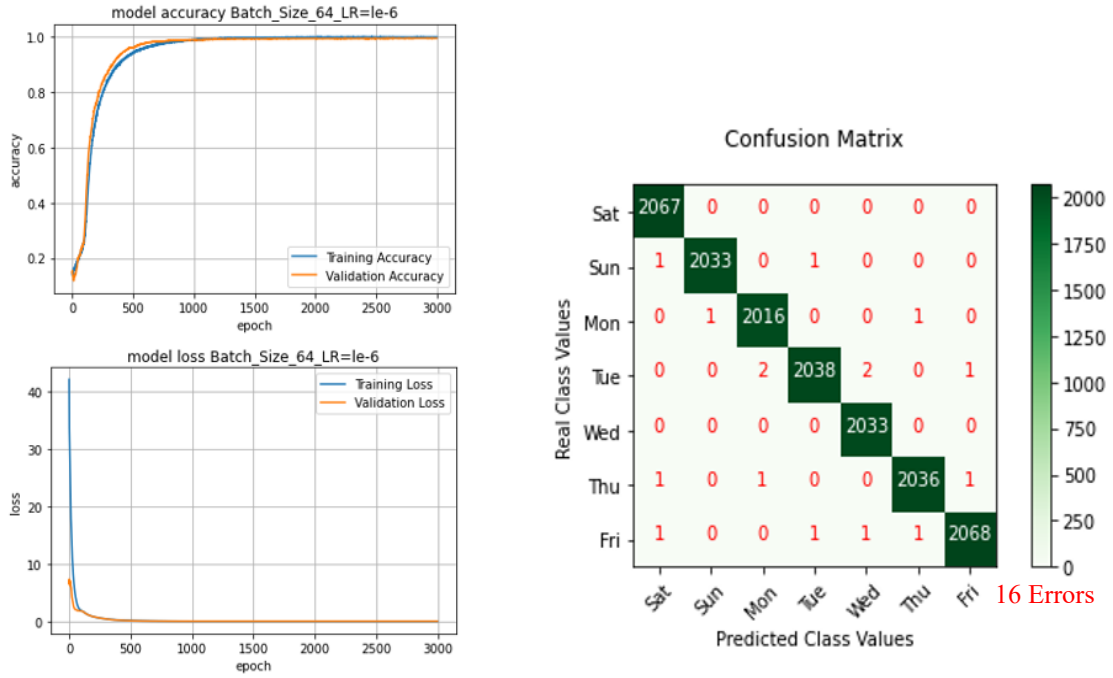


Figure 6.17 LC and CM with LR = 10^{-6} and BS = 64 AHWD.

In Table 6.4, some results in training and testing sessions with batch size 16 with accuracy rate 99.57%, and with loss rate (GER) 0.0264; and batch size 32 with accuracy rate 99.55%, and with loss rate (GER) 0.0288; and batch size 128 with accuracy rate 99.53%, and with error rate 0.0231; and batch size 256 with accuracy rate 99.34%, and with 0.0219 are acceptable but not chosen because the error rates are high comparing to the chosen error rate 0.0177 with accuracy rate 99.72%.

However, for a batch size 512 with an accuracy rate 98.72%, and with error rate 0.0473; and batch size 1024 with accuracy rate 97.25%, and with 0.0948 are not accepted in this training and testing session because the error rates are high, the confusion matrix errors numbers are high, and the model response speed is low in both as shown in Figure 6.18 and Figure 6.19, the conclusion with the last two results is that the system is unstable and the batch sizes 512 and 1024 are not suitable here.

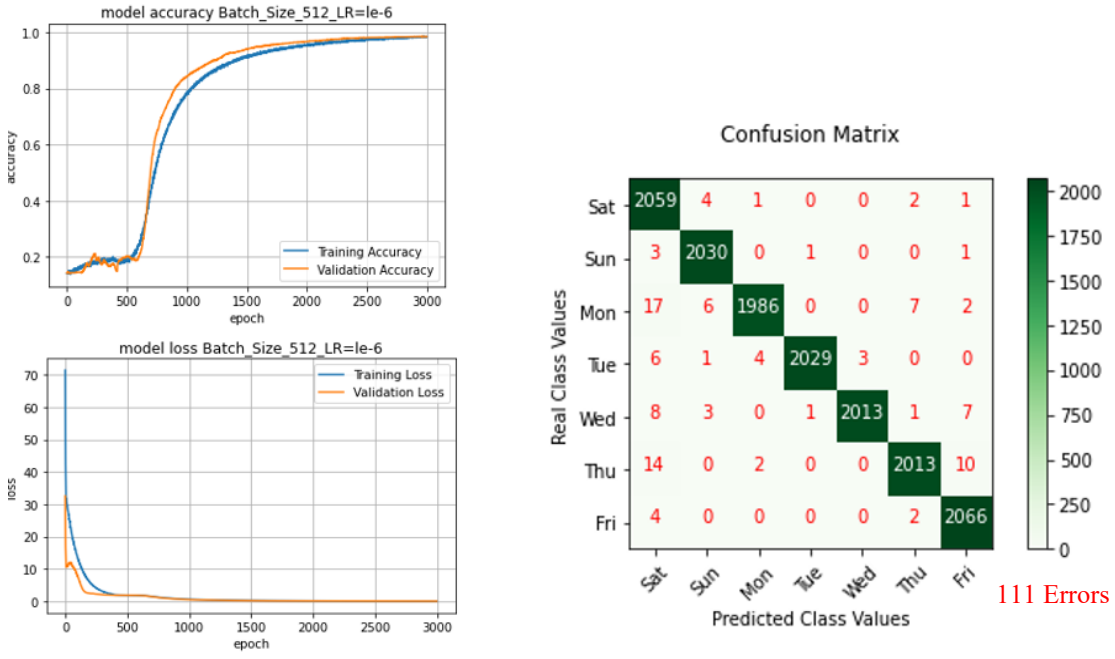


Figure 6.18 LC and CM with LR = 10^{-6} and BS = 512 AHWD.

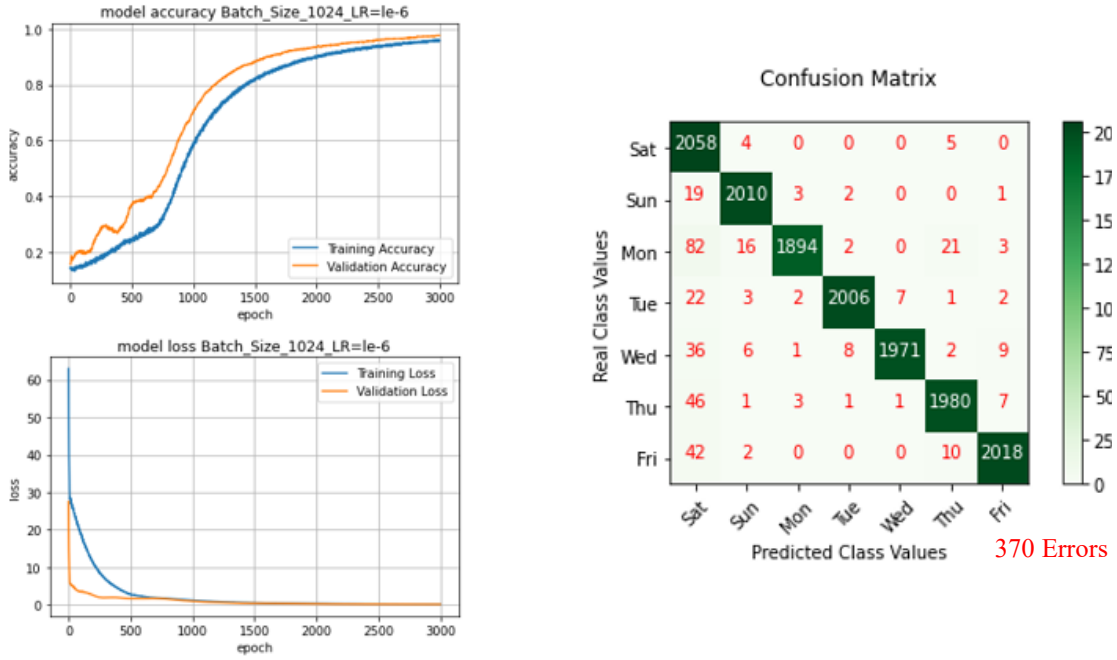


Figure 6.19 LC and CM with LR = 10^{-6} and BS = 1024 AHWD.

Fifth, when the learning rate = 10^{-7} and for batch sizes (16, 32, 64, 128, 256, 512, 1024), the results are as shown in Table 6.5.

By looking at Table 6.5, the best accuracy rate is 96.10%, error rate (GER) = 0.1530, and the confusion matrix error = 663. This conclusion is not accepted in this training and testing session with batch size = 32 because the error rate is high, and the confusion matrix error number is remarkably high, and the model is unstable as shown in Figure 6.20

In more detail, using learning rate = $10^{-7}$is not suitable for the proposed model since all the training and testing session using 10^{-7} with batch sizes are (16, 32, 64, 128, 256, 512, 1024) make the proposed model unstable and unrobust as shown in Figure 6.20 to Figure 6.26.

Table 6.5 AHWD dataset with learning rate = 10^{-7}

AHWD dataset LR = 0.0000001 = le-7						
Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	716	441	0.9296	0.2223	1065
32	3000	358	221	0.9610	0.1530	663
64	3000	179	111	0.7995	0.6671	3849
128	3000	90	56	0.7705	0.8354	> 6000
256	3000	45	28	0.6556	1.0785	> 6000
512	3500	23	14	0.2225	1.9288	No diagonal
1024	3500	12	7	0.2256	2.0243	No diagonal

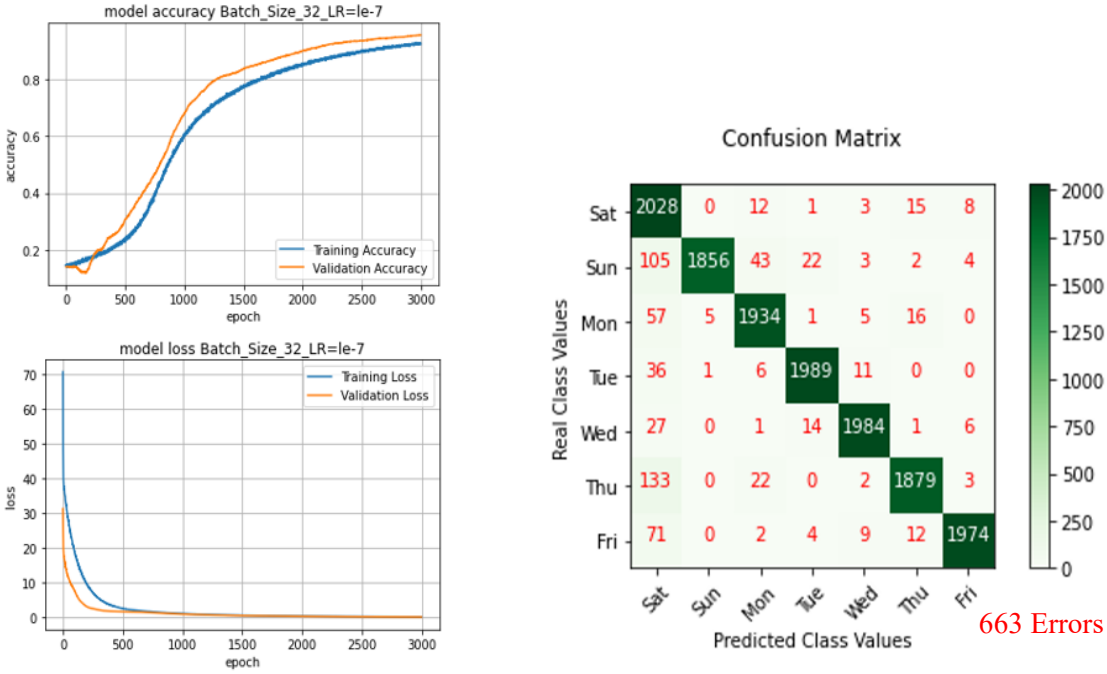


Figure 6.20 LC and CM with LR = 10^{-7} and BS = 32 AHWD.

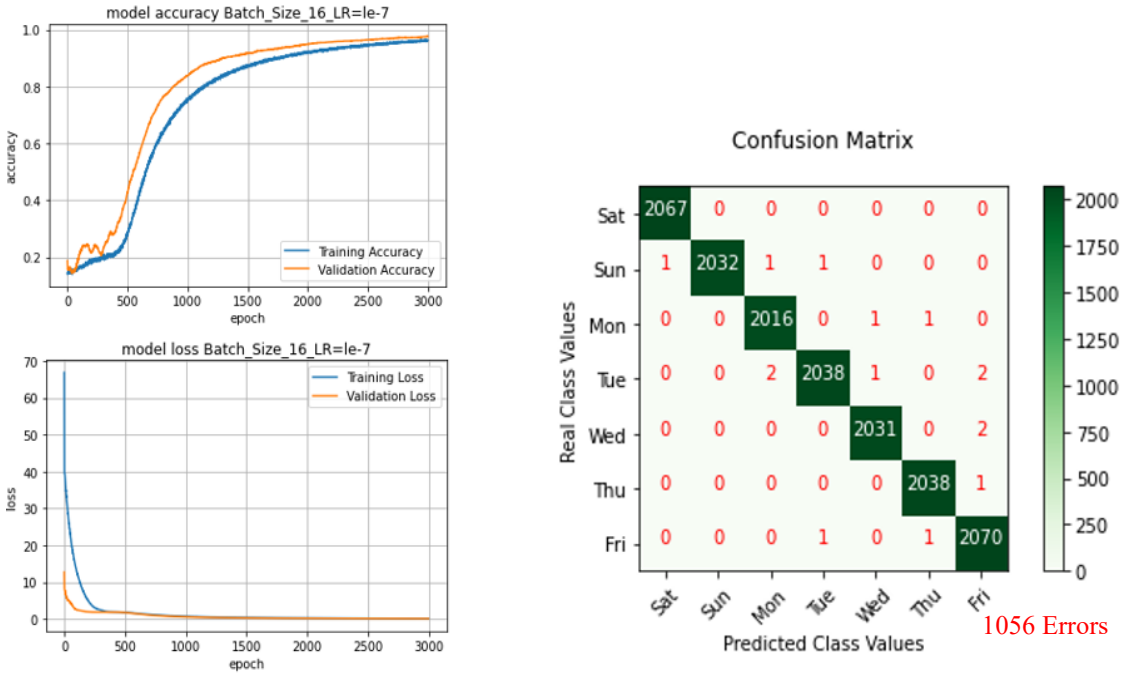


Figure 6.21 LC and CM with LR = 10^{-7} and BS = 16 AHWD.

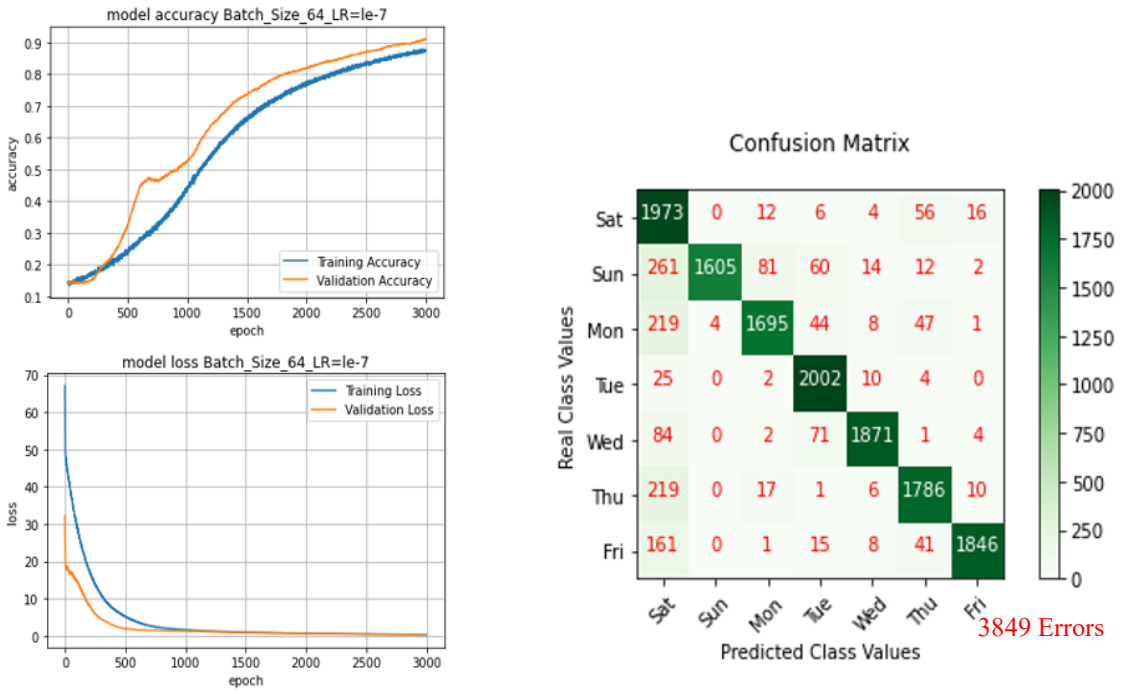


Figure 6.22 LC and CM with LR = 10^{-7} and BS= 64 AHWD.

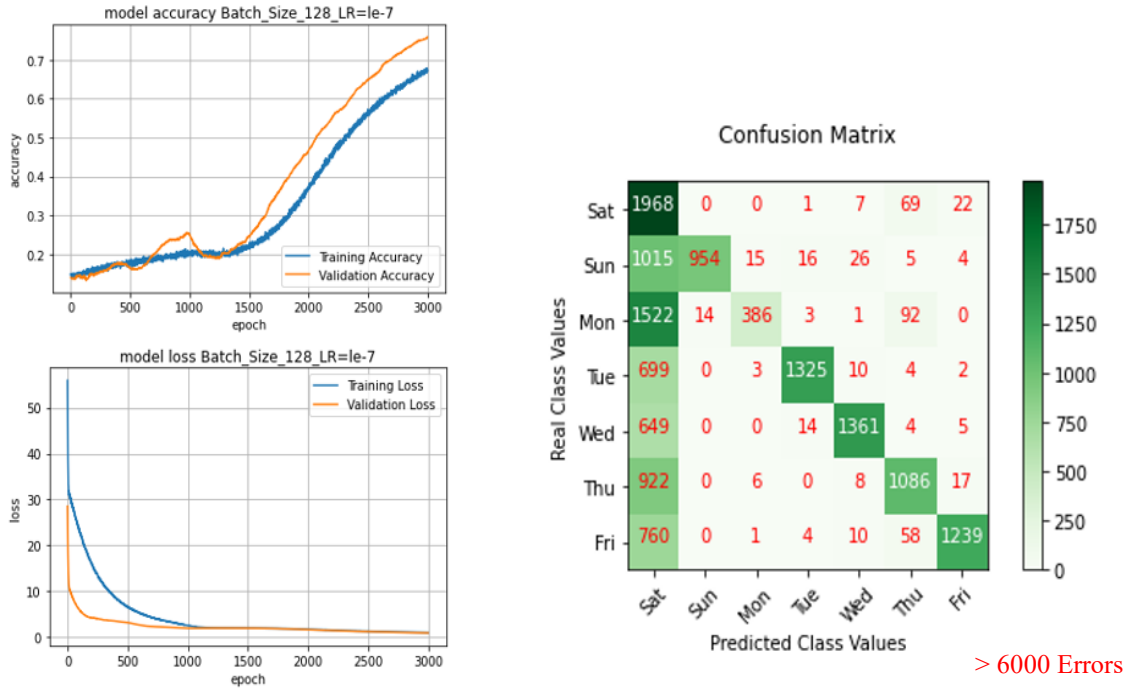


Figure 6 23 LC and CM with LR = 10^{-7} and BS = 128 AHWD.

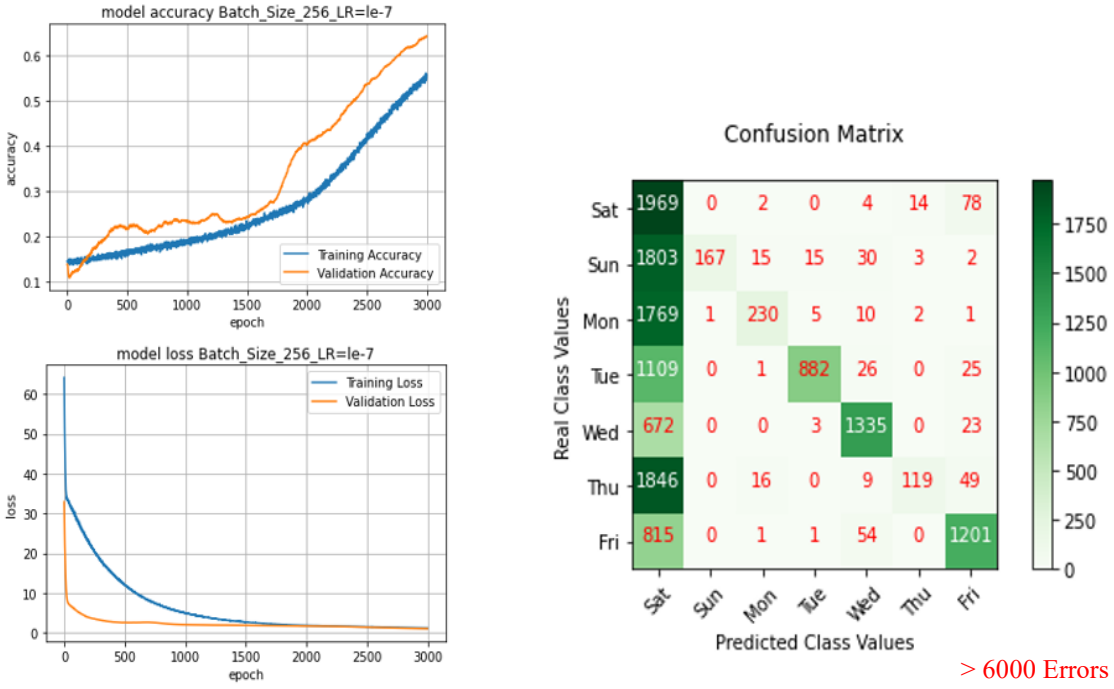


Figure 6.24 LC and CM with LR = 10^{-7} and BS = 256 AHWD.

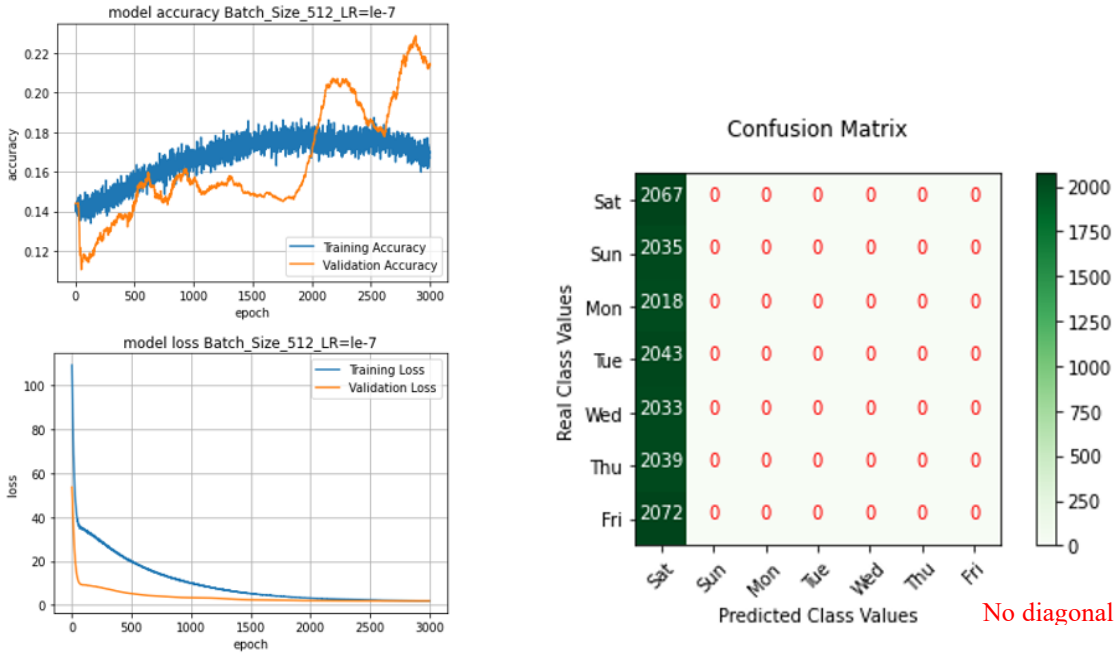


Figure 6.25 LC and CM with LR = 10^{-7} and BS = 512 using AHWD.

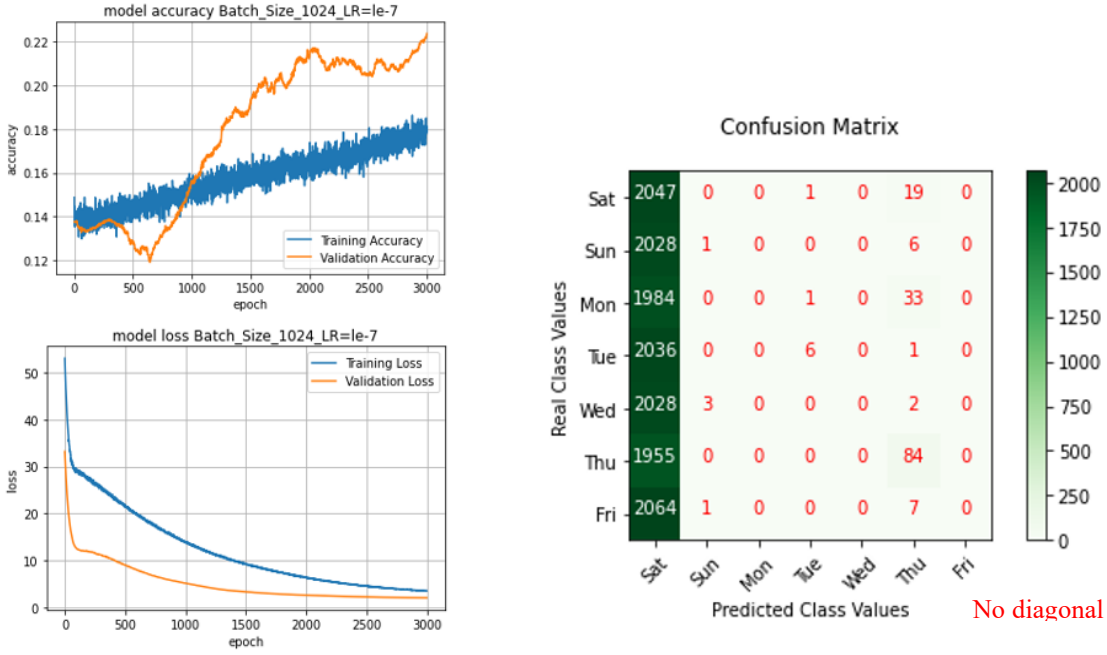


Figure 6.26 LC and CM with LR = 10^{-7} and BS = 1024 using AHWD.

By looking at Figure 6.20 to Figure 6.26, there is instability in the proposed model when using a learning rate = 10^{-7}. The learning rate here is very small makes the gradient erratic and the convergence between the training accuracy curve and validation accuracy cannot be realized. Moreover, as the batch size increases:

- Both the training and validation loss curve diverge, and this divergence is clear in Figure 6.26.
- The number of confusion matrix errors increases, the classification accuracy rate decrease, and the error rate increases, as well.

The Adam optimizer is used to evaluate the performance when applying AHWD using the proposed model. For the loss function, the categorical cross-entropy loss is used. After training the DCNN model to calculate the probability of each image over the classes, the model is evaluated by using the testing dataset. On AHWD, different batch sizes equal to 16, 32, 64, 128, 256, 512, and 1024 were tested. Also, learning rates of 10^{-3}, $10^{-4}, 10^{-5}, 10^{-6}$, and 10^{-7} were evaluated with epochs equal to 3000. With each learning rate, the model was trained with 7 different batch sizes. So, five tables of results were produced, and each table consists of one learning rate and 7 batch sizes with results (accuracy rate, loss rate, and confusion matrix errors) where the loss rate here is considered as the GER. The results between the five tables are evaluated based on the following criteria:

- Learning rate values (10^{-3} to 10^{-7}) and batch sizes.
- Low loss error rate (GER) has the highest priority.
- High accuracy rate.
- Confusion matrix errors.
- Model response speed by looking at the learning curve graph and determine in which epoch number the accuracy curve starts rising and in which epoch number does the loss curve start coming down.

After applying these criteria, a table summarizing the conditions to obtain the best results is compiled for each dataset. The result is shown in Table 6.6 for AHWD. After

applying the above criteria on Table 6.6, it was concluded that the best accuracy rate is 99.76% and error rate GER = 0.0230 for the following reasons:

- The learning rate (10^{-5}) is low which makes the model train at a reasonable speed and allows the gradient descent to produce a smooth output.
- There is a reasonable number of epochs (3000 epochs with 23 training iterations and 14 testing iterations for each epoch) required and batch size is = 512. These conditions allow relatively fast computing time.

Table 6.6 Best of the five tables of AHWD dataset

Learning rate	Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
			Train	Test			
10^{-3}	1024	3000	12	7	0.9986	0.5773	9
10^{-4}	1024	3000	12	7	0.9965	0.0349	9
10^{-5}	512	3000	23	14	0.9976	0.0230	11
10^{-6}	64	3000	179	111	0.9972	0.0177	16
10^{-7}	32	3000	358	221	0.9610	0.1530	663

- The lowest error rate 0.0177 with high accuracy rate was not chosen because by looking at Figure 6.17, the training accuracy curve and validation accuracy curve start raising up in convergence fashion from the beginning until approximately epoch 1,000 where the convergence goes all the way long, Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 400 where the convergence started to be steady, and the error is the lowest. Whereas Figure 6.15 shows that the convergence between the training accuracy curve and validation accuracy curve starts approximately at epoch = 400 and continue all the way straight. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 200 where the convergence started to be steady, and the error is the lowest which is 0.0230. Now, by comparing Figure 6.15 with error rate 0.0230 and

Figure 6.17 with error rate 0.0177 in terms of model response speed in the accuracy curve and the loss curve. The conclusion is that Figure 6.15 with error rate 0.0230 shows faster response than Figure 6.17 with an error rate of 0.0177.

- By looking at Table 6.6 a setting converging to (GER = 0.0230) is selected even though there are lower values of GER in the table. This choice was made because the accuracy rate (99.76%) is the highest on the table. Also, as can be observed from Table 6.6 the number of errors in the confusion matrix for the chosen setting is low compared to other results.

6.2 IFN/ENIT Experimental Phase

As mentioned in Chapter 5, The model is evaluated by using the testing dataset. On IFN/ENIT, different batch sizes equal to 16, 32, 64, 128, 256, 512, and 1024 were tested. Also, learning rates of 10^{-3}, $10^{-4}, 10^{-5}, 10^{-6}$, and 10^{-7} were evaluated with epochs equal to 3000. With each learning rate, the model was trained with 7 different batch sizes. So, five tables of results were produced, and each table consists of one learning rate and 7 batch sizes with results (training iteration number, testing iteration number, accuracy rate, loss rate, and confusion matrix errors) where the loss rate here is considered as the GER. The results between the five tables are evaluated based on the following criteria:

- Learning rate values (10^{-3} to 10^{-7}) and batch sizes.
- A low loss error rate or Generalization Error Rate (GER) has the highest priority.
- High accuracy rate.
- Confusion matrix errors.
- Model response speed by looking at the learning curve graph and determine in which epoch number the accuracy curve starts rising and in which epoch number does the loss curve starts coming down.

After the evaluation by using these criteria, the best performance from each table has been collected.

- **First**, when learning rate = 10^{-3} and batch sizes (16, 32, 64, 128, 256, 512, 1024) the results would be as shown in Table 6. 7.

Table 6. 7 IFN/ENIT dataset with learning rate = 10^{-3}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	707	436	0.9983	637.5007	16
32	3000	353	218	0.9981	403.5989	20
64	3000	177	109	0.9977	107.2415	18
128	3000	89	55	0.9977	37.1814	20
256	3000	45	28	0.9983	6.6885	16
512	3000	23	14	0.9973	2.5058	18
1024	3000	12	7	0.9991	0.5749	9

By looking at Table 6. 7, the best accuracy rate is 99.91%, the lowest error rate (GER) = 0.5749, and the lower confusion matrix error = 9. This is acceptable in this training and testing session but cannot be generalized because Figure 6.27 shows an overfitting after epoch 1800; that is, the validation loss curve starts to diverge up off the training loss curve. In the conclusion, the proposed model cannot generalized on IFN/ENIT dataset with learning rate = 10^{-3} and batch sizes 1024.

All other results in Table 6. 7 have a remarkably high error rates, and this would lead to overfitting and cannot be generalized as shown in Figure 6.28 to Figure 6.33.

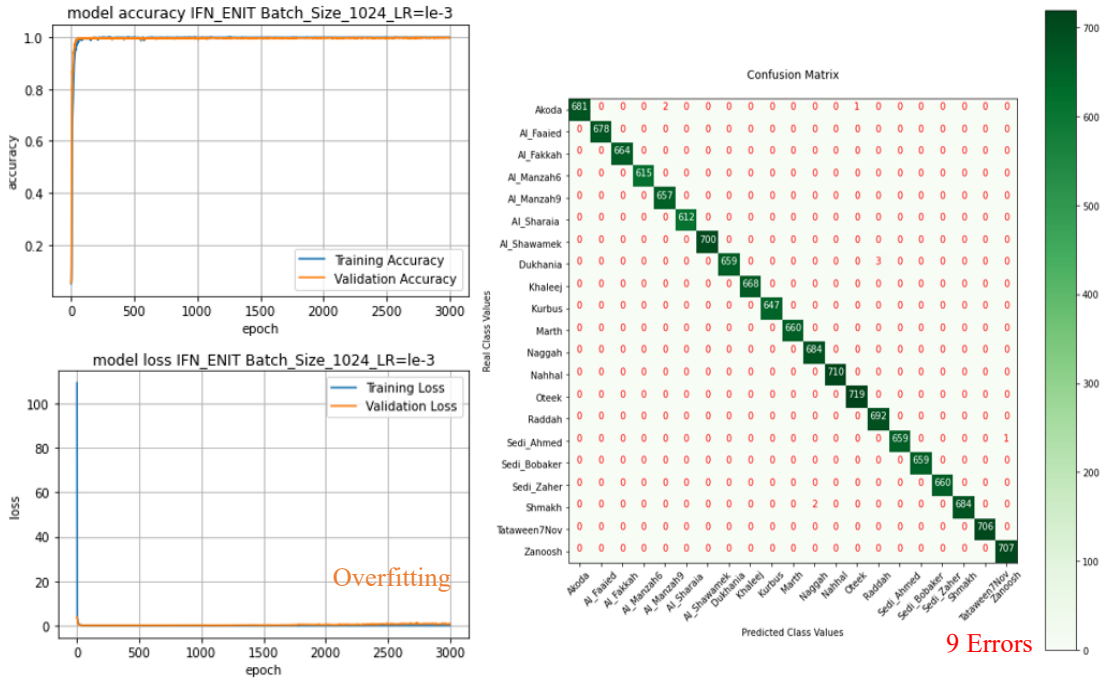


Figure 6.27 LC and CM with LR = 10^{-3} and BS = 1024 using IFN/ENIT dataset.

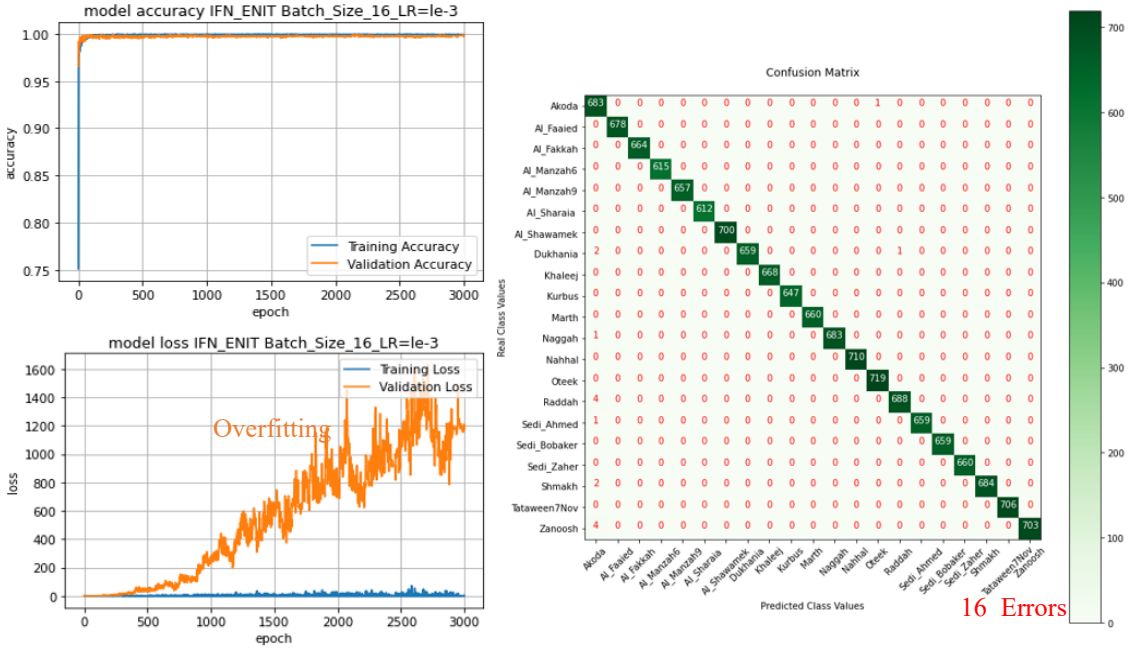


Figure 6.28 LC and CM with LR = 10^{-3} and BS = 16 using IFN/ENIT dataset.

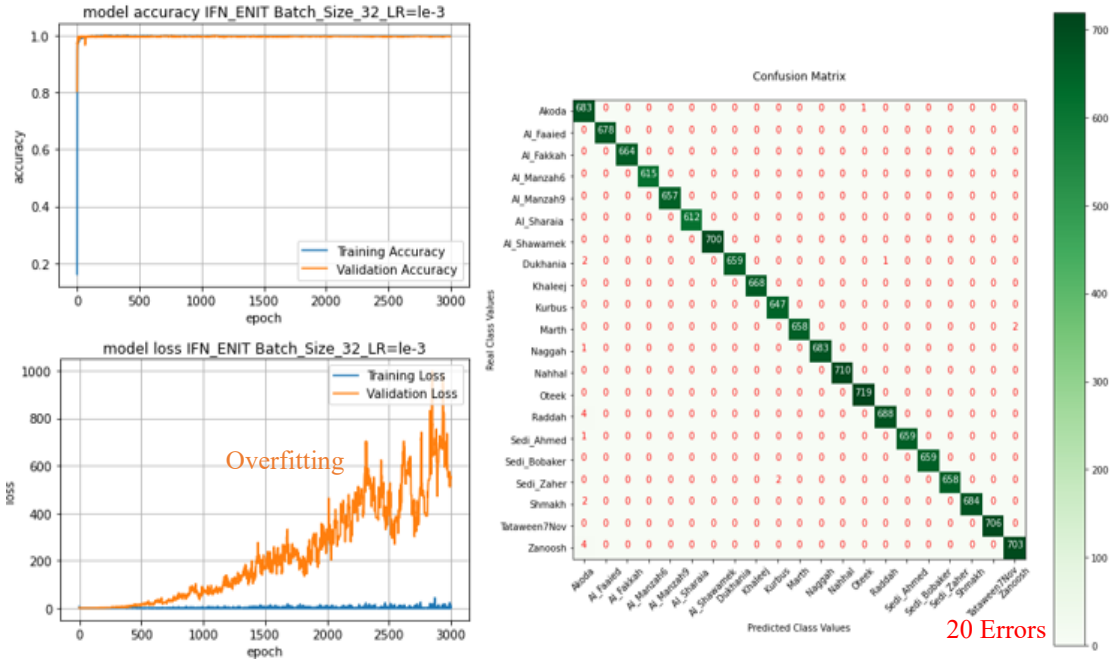


Figure 6.29 LC and CM with LR = 10^{-3} and BS = 32 using IFN/ENIT dataset.

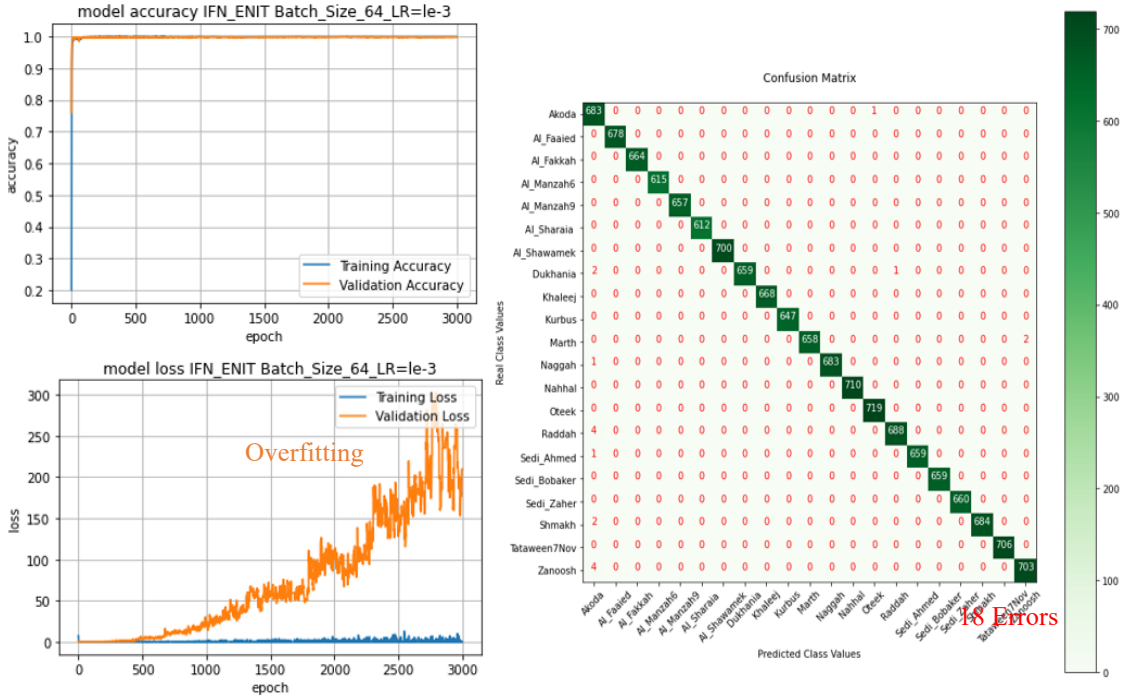


Figure 6.30 LC and CM with LR = 10^{-3} and BS = 64 using IFN/ENIT dataset.

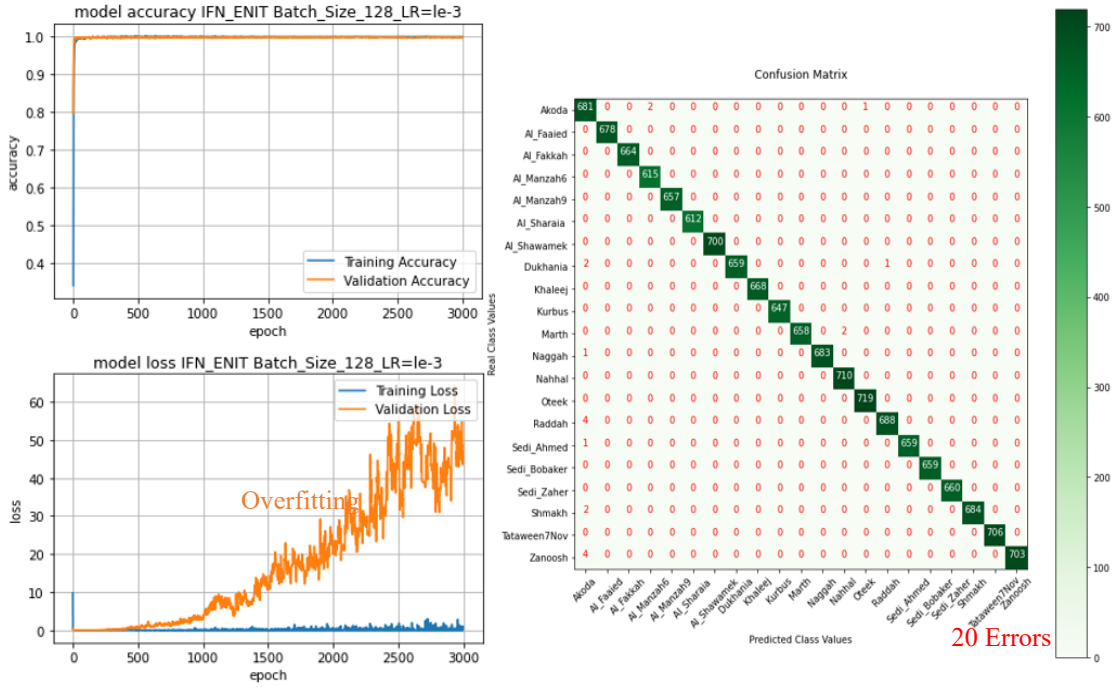


Figure 6.31 LC and CM with LR = 10^{-3} and BS = 128 using IFN/ENIT dataset.

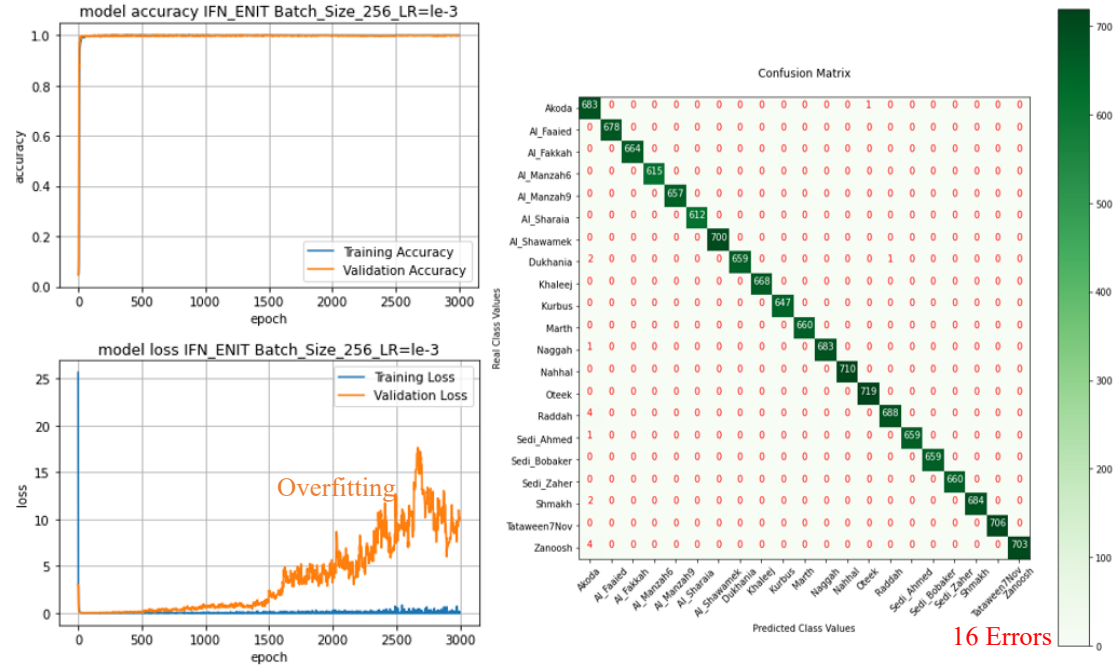


Figure 6.32 LC and CM with LR = 10^{-3} and BS = 256 using IFN/ENIT dataset.

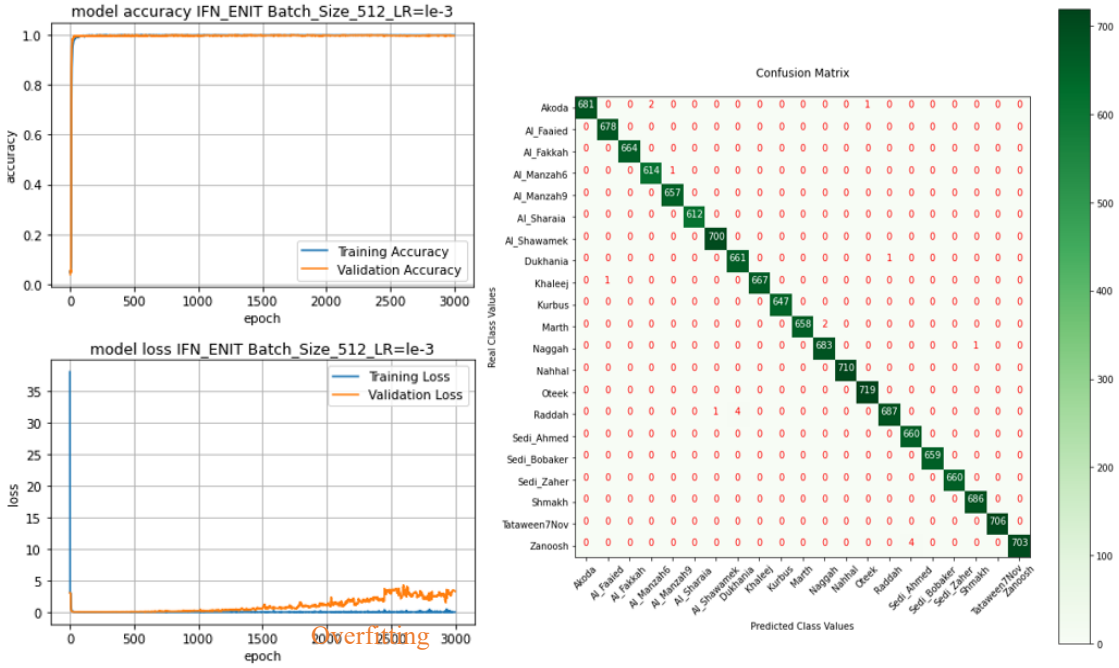


Figure 6.33 LC and CM with LR = 10^{-3} and BS = 512 using IFN/ENIT dataset.

- **Second**, when learning rate = 10^{-4} and batch sizes (16, 32, 64, 128, 256, 512, 1024) the results would be as shown in Table 6.8.

Table 6.8 IFN/ENIT dataset with learning rate = 10^{-4}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	707	436	0.9976	0.0917	10
32	3000	353	218	0.9980	0.0806	14
64	3000	177	109	0.9986	0.0610	14
128	3000	89	55	0.9973	0.0677	9
256	3000	45	28	0.9984	0.0266	9
512	3000	23	14	0.9987	0.0181	8
1024	3000	12	7	0.9971	0.0255	13

Table 6.8 depicts that the best accuracy rate is 99.87%, the lowest error rate (GER) = 0.0181, and the lowest confusion matrix error number = 8. This conclusion is the best performance and is acceptable in this training and testing session with batch size = 512

because it has lower error rate with highest accuracy rate, and the lowest confusion matrix error number.

Figure 6.34 depicts that the training accuracy curve and validation accuracy curve start up in convergence fashion from the beginning until approximately epoch 100 where the convergence goes all the way along. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 50 where the convergence started to be steady, and the error is the lowest. The curves never touch each other, the gap between them is called a generalization gap.

The rest of the results in Table 6.8 are acceptable except the training and testing sessions with batch size 16 and batch size 32, and with error rate 0.0917 and error rate 0.0806 consecutively because by looking at Figure 6.35 and Figure 6.36 there is data overfitting, and the conclusion would be that the proposed model cannot be generalized utilizing learning rate 10^{-4} hyperparameter with batch size 16 and 32.

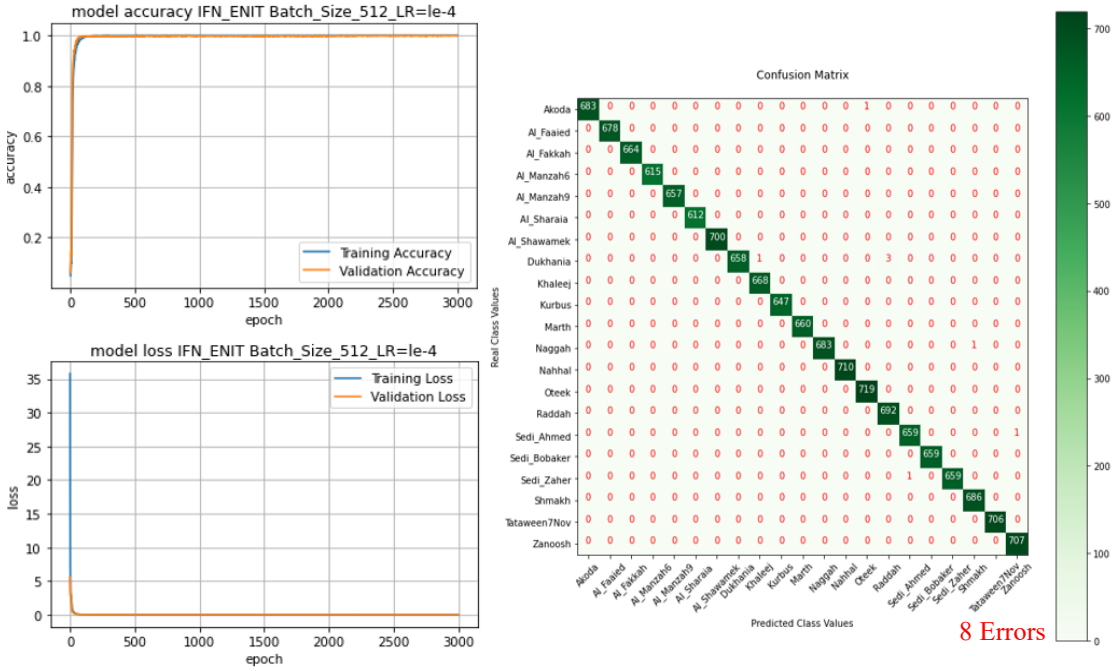


Figure 6.34 LC and CM with LR = 10^{-4} and BS = 512 using IFN/ENIT dataset

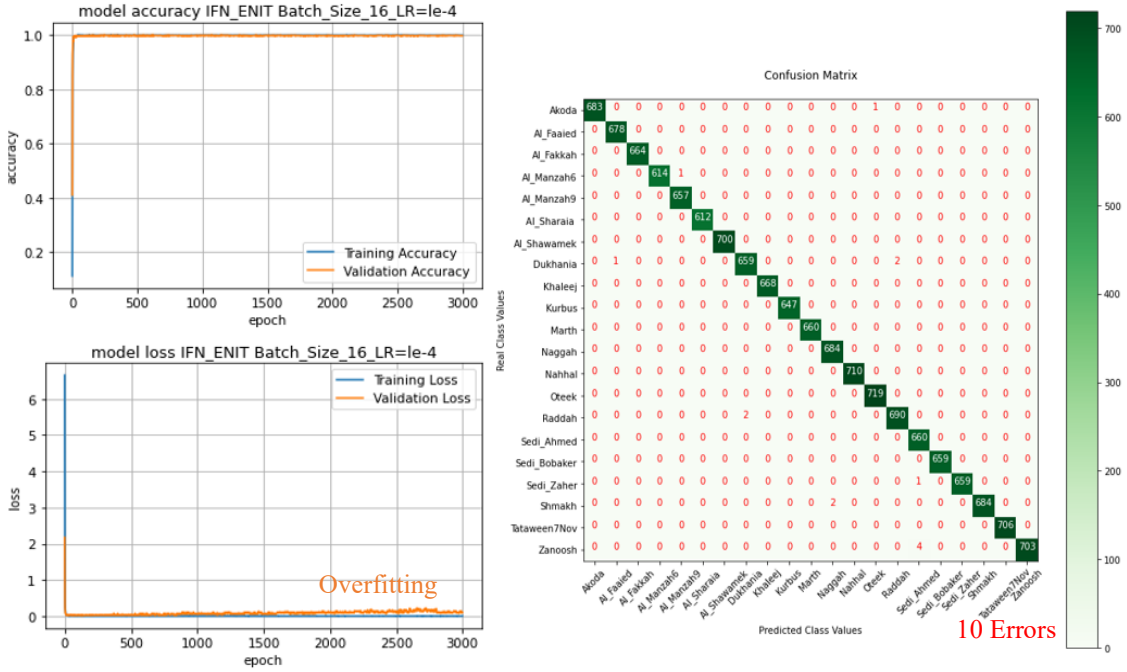


Figure 6.35 LC and CM with LR = 10^{-4} and BS = 16 using IFN/ENIT dataset

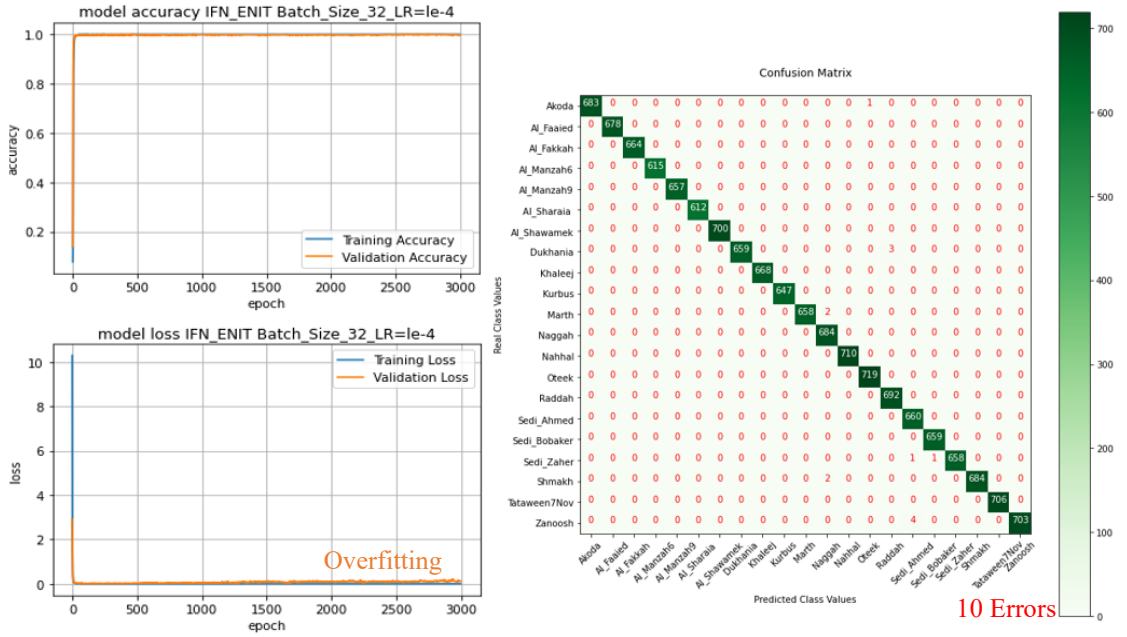


Figure 6.36 LC and CM with LR = 10^{-4} and BS = 32 using IFN/ENIT dataset

- **Third**, when learning rate = 10^{-5} and batch sizes (16, 32, 64, 128, 256, 512, 1024) the results would be as shown in Table 6.9.

Table 6.9 IFN/ENIT dataset with learning rate = 10^{-5}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	707	436	0.9961	0.0200	9
32	3000	353	218	0.9961	0.0210	9
64	3000	177	109	0.9967	0.0215	16
128	3000	89	55	0.9967	0.0178	13
256	3000	45	28	0.9966	0.0187	15
512	3000	23	14	0.9966	0.0178	15
1024	3000	12	7	0.9974	0.0242	13

Table 6.9 shows that all results are acceptable. However, the proposed model is measured to find the best performance, there are some equivalent results in Table 6.9. To find the best performance, there must be a complete analysis of the results. Analysis includes, lowest error rate, highest accuracy rate, model response speed using the associated learning curve graphs and number of confusion matrix errors.

By rounding up most of the values of the classification accuracy rates in Table 6.9, a valid value for the accuracy rates is equal to 99.70% except the first two values with batch size 16 and batch size 32 are 99.60%. The lowest error rate is 0.0178 and is associated with two accuracy rates 99.67% and 99.66% with batch size 128 and 512 respectively. By looking at Figure 6.37 and Figure 6.38 and see the model response speed in terms of the accuracy curve and the loss curve the analysis is as follows:

Figure 6.37 depicts that the training accuracy curve and validation accuracy curve start raising up in convergence fashion from the beginning until approximately epoch 200 where the convergence goes all the way along. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 100 where the convergence started to be steady, and the error is the lowest. The curves never touch each other.

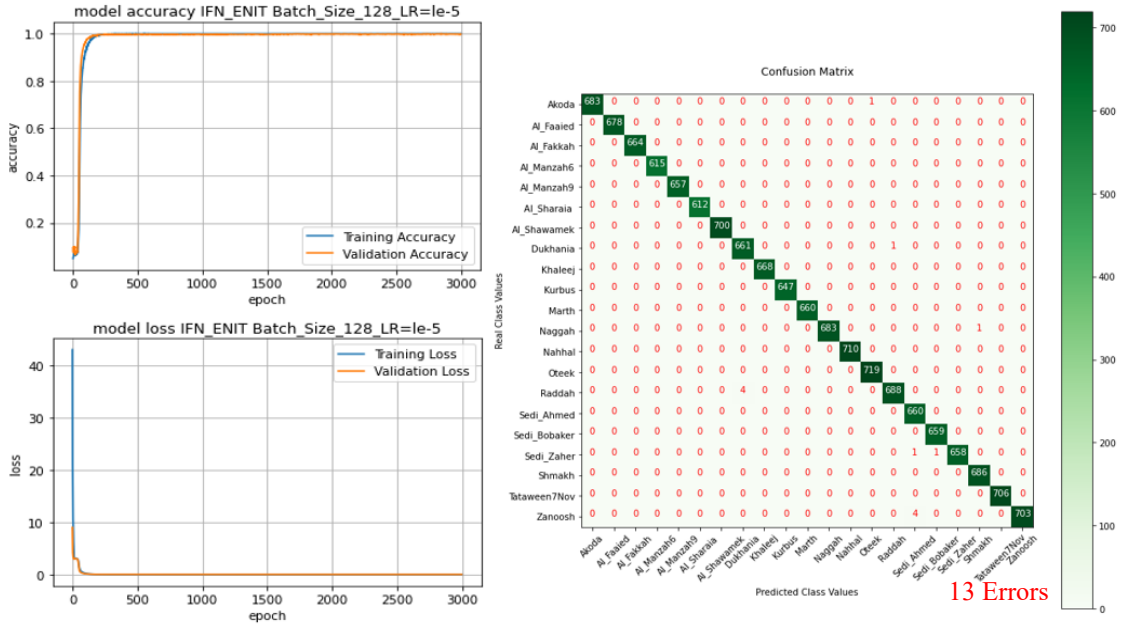


Figure 6.37 LC and CM with LR = 10^{-5} and BS = 128 using IFN/ENIT dataset

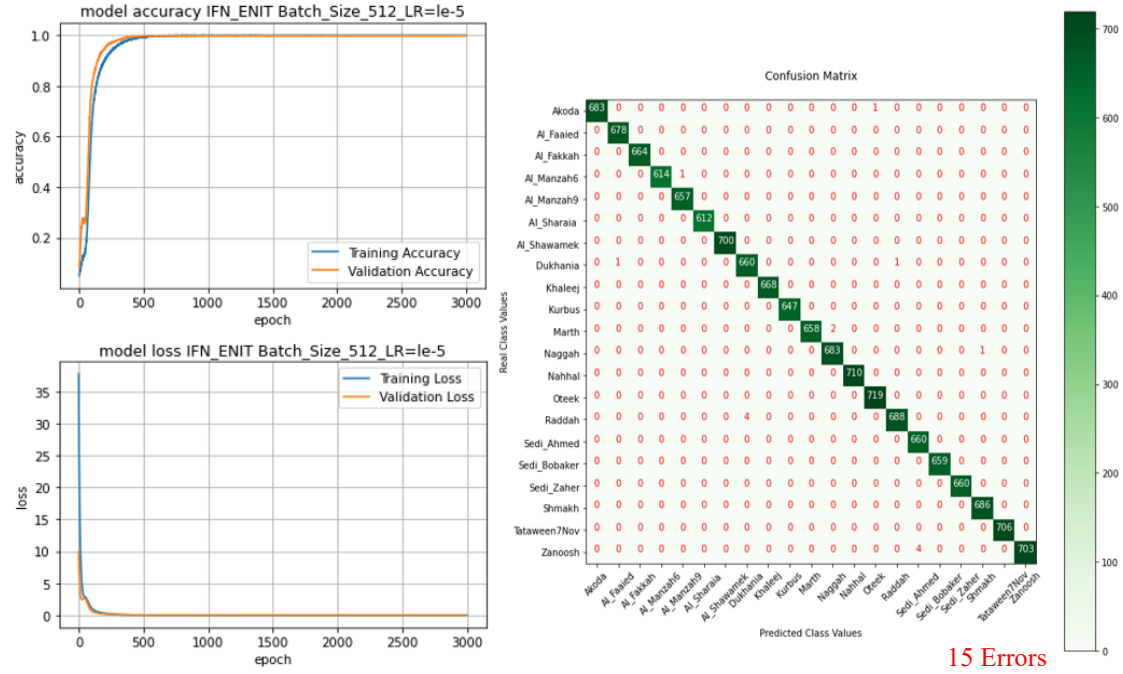


Figure 6.38 LC and CM with LR = 10^{-5} and BS = 512 using IFN/ENIT dataset

Figure 6.38 depicts that the training accuracy curve and validation accuracy curve start up in convergence fashion from the beginning until approximately epoch 500 where

the convergence goes all the way along. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 200 where the convergence started to be steady, and the error is the lowest. Both curves never touch each other.

Analysis showed that the best accuracy rate is 99.87%, the lowest error rate (GER) is 0.0178 with epoch 128 and confusion matrix error number = 13. This conclusion is the best performance and is acceptable in this training and testing session.

- **Fourth**, when the learning rate = 10^{-6} and batch sizes (16, 32, 64, 128, 256, 512, 1024) the results would be as shown in Table 6.10.

Table 6.10 depicts that the highest accuracy rates are 99.68%, with lowest error rate (GER) = 0.0195, and low confusion matrix error number = 18 errors. This conclusion is the best performance and is accepted in this training and testing session with batch size = 32 as shown in Figure 6.39. where the training accuracy curve and validation accuracy curve

Table 6.10 IFN/ENIT dataset with learning rate = 10^{-6}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	707	436	0.9961	0.0244	15
32	3000	353	218	0.9968	0.0195	18
64	3000	177	109	0.9963	0.0241	18
128	3000	89	55	0.9947	0.0302	18
256	3000	45	28	0.9964	0.0216	18
512	3000	23	14	0.9955	0.0282	39
1024	3000	12	7	0.4215	1.9947	No Diagonal

start raising up in convergence fashion from epoch 100 until approximately epoch 750 where the convergence goes all the way long, Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from epoch 200 until it reaches epoch 300 where the convergence starts to be steady, and the error is the lowest. So, the proposed model can be generalized using batch size 32.

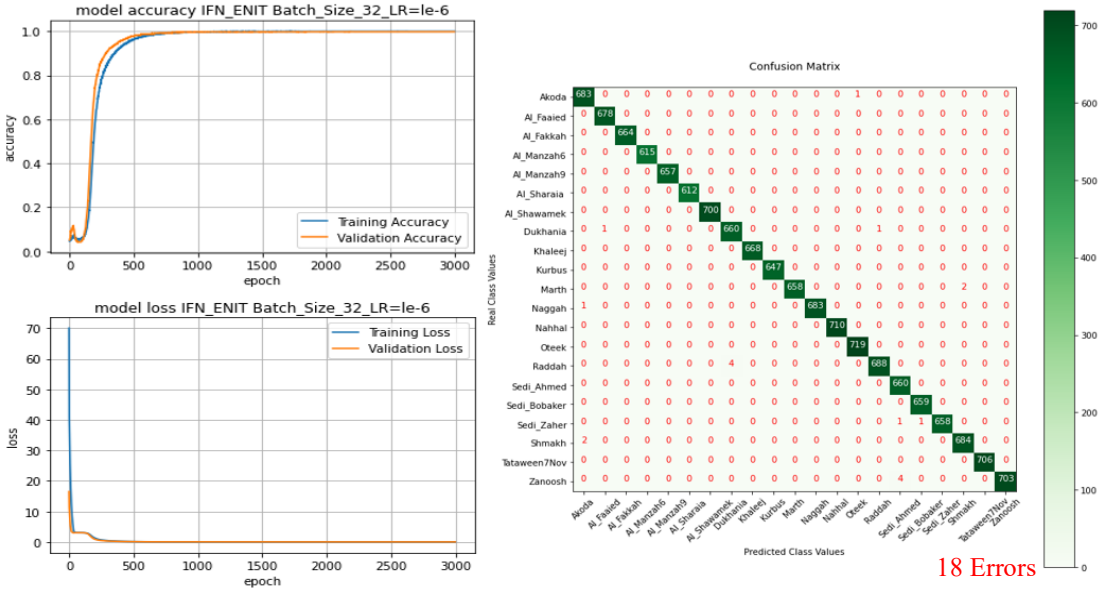


Figure 6.39 LC and CM with LR = 10^{-6} and BS = 32 using IFN/ENIT dataset

Table 6.10 shows that as the batch size increases the error rate (loss rate) and the confusion matrix error number increases, and the accuracy rate decreases. This conclusion leads to model instability using batch sizes 512 and 1024 as shown in Figure 6.40 and Figure 6.41, and the proposed model cannot be generalized with batch sizes 512, and 1024. However, the proposed system is acceptable with the batch size hyperparameters 16, 64, 128, and256 as shown in Figure 6.42 to Figure 6.45.

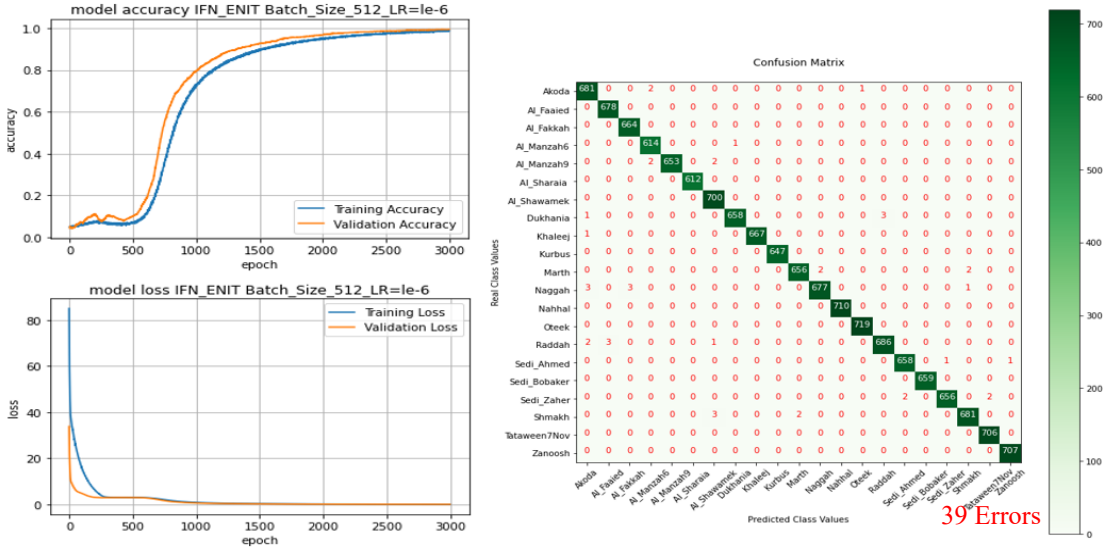


Figure 6.40 LC and CM with LR = 10^{-6} and BS = 512 using IFN/ENIT dataset

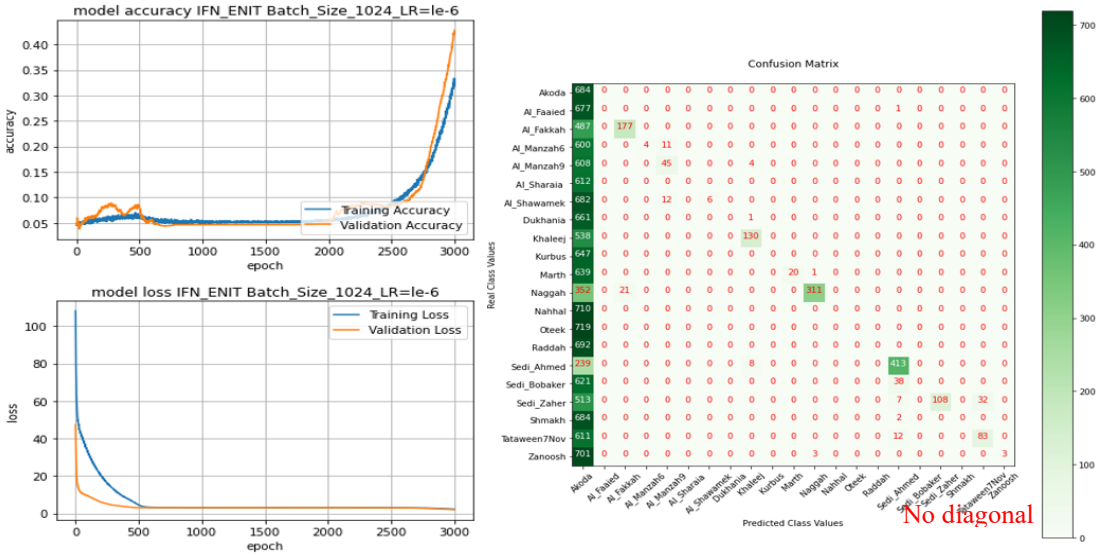


Figure 6.41 LC and CM with LR = 10^{-6} and BS = 1024 using IFN/ENIT dataset

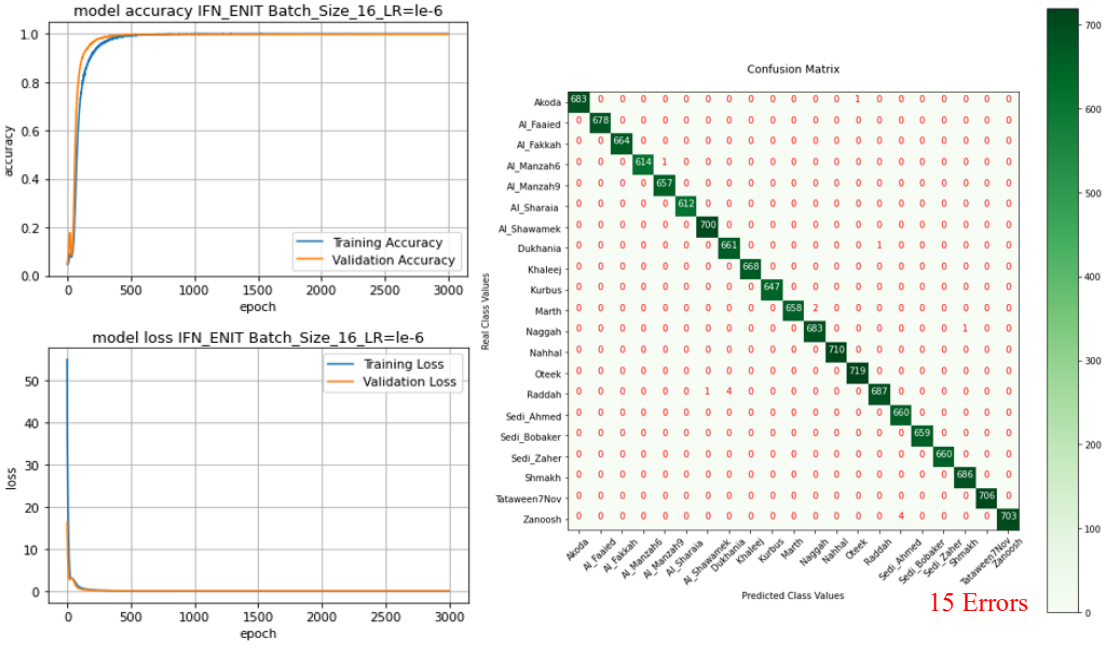


Figure 6.42 LC and CM with LR = 10^{-6} and BS = 16 using IFN/ENIT dataset

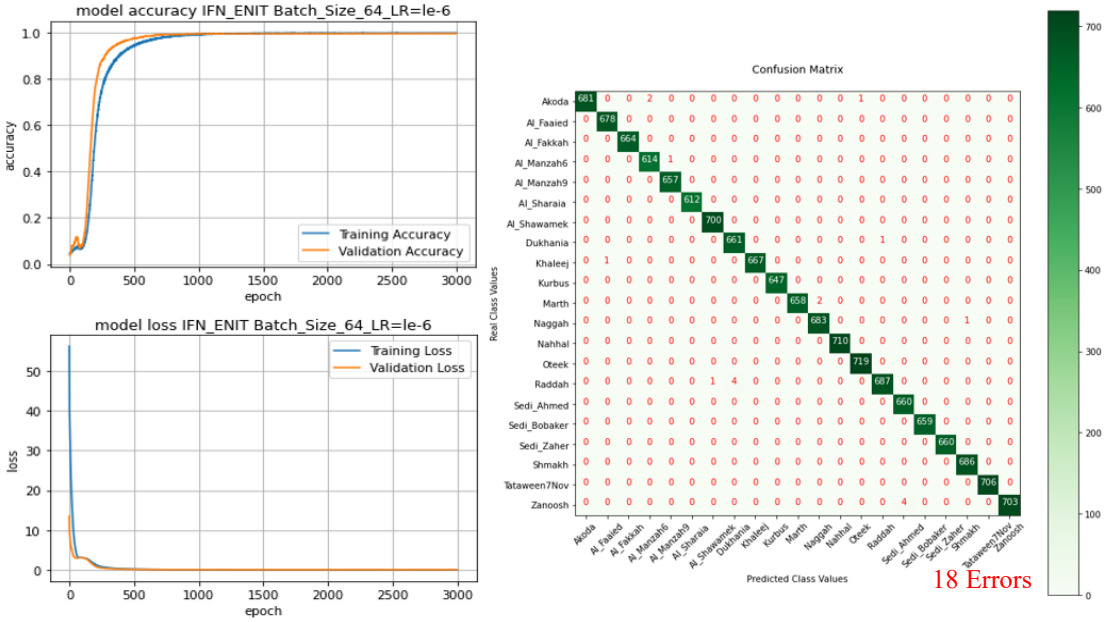


Figure 6.43 LC and CM with LR = 10^{-6} and BS = 64 using IFN/ENIT dataset

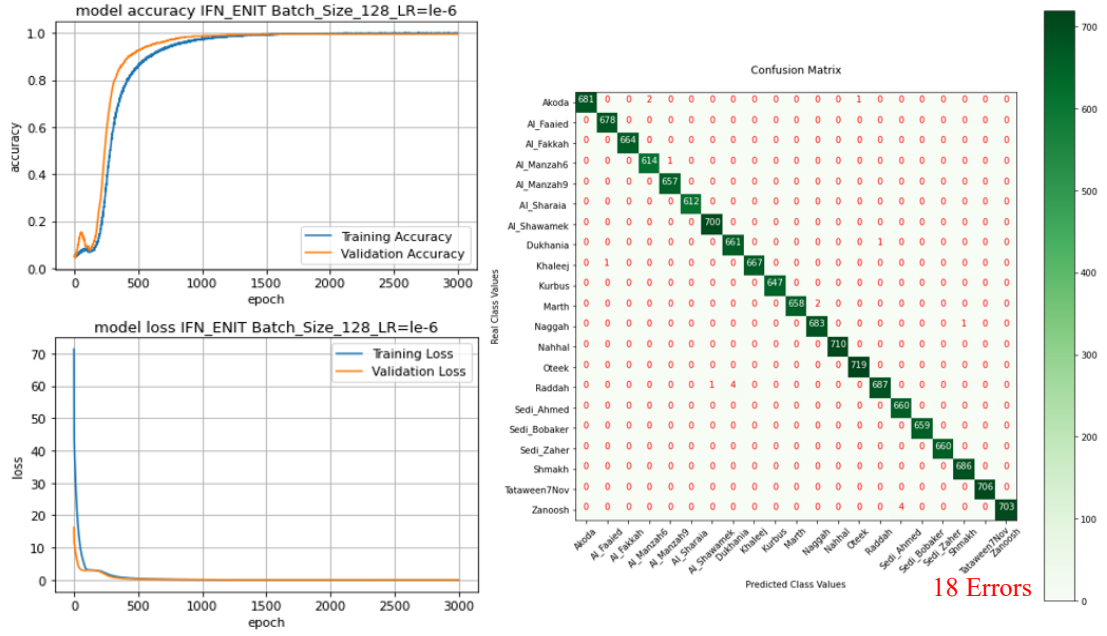


Figure 6.44 LC and CM with LR = 10^{-6} and BS = 128 using IFN/ENIT dataset

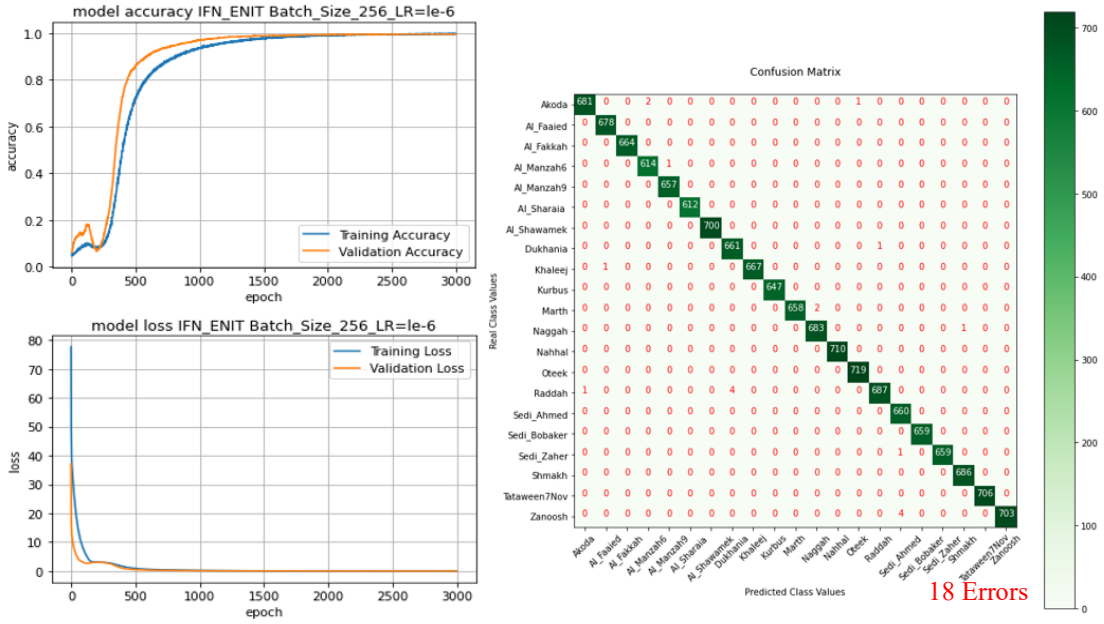


Figure 6.45 LC and CM with LR = 10^{-6} and BS = 256 using IFN/ENIT dataset

- **Fifth**, when learning rate = 10^{-7} and batch sizes (16, 32, 64, 128, 256, 512, 1024) we the results would be as shown in Table 6.11.

Table 6.11 shows that the training and testing sessions with batch sizes 16, 32, and 64 have acceptable accuracy rates 97.73%, 97.93, and 95.29% and error rate (loss rate) 0.0966, 0.0853, and 0.1723, respectively. However, Table 6.11 depicts that, as the batch size hyperparameter increases the error rate and the confusion matrix error number increases, and the accuracy rate decreases.

Table 6.11 IFN/ENIT dataset with learning rate = 10^{-7}

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	3000	707	436	0.9773	0.0966	263
32	3000	353	218	0.9793	0.0853	226
64	3000	177	109	0.9529	0.1723	779
128	3000	89	55	0.2090	2.7252	No Diagonal
256	3000	45	28	0.7555	0.9723	No Diagonal
512	3000	23	14	0.0714	3.0376	No Diagonal
1024	3000	12	7	0.1721	2.7306	No Diagonal

The conclusion is that using learning rate = 10^{-7} cannot be generalized and is not suitable for the proposed model on IFN/ENIT dataset. All the training and testing sessions using learning rate 10^{-7} with batch sizes (16, 32, 64, 128, 256, 512, 1024) make the proposed model unstable, unrobust, and convergence between the training accuracy curve and the validation accuracy curve would not be stable in all training and testing sessions and at some points the two curves would cross each as shown in Figure 6.46 to Figure 6.52.

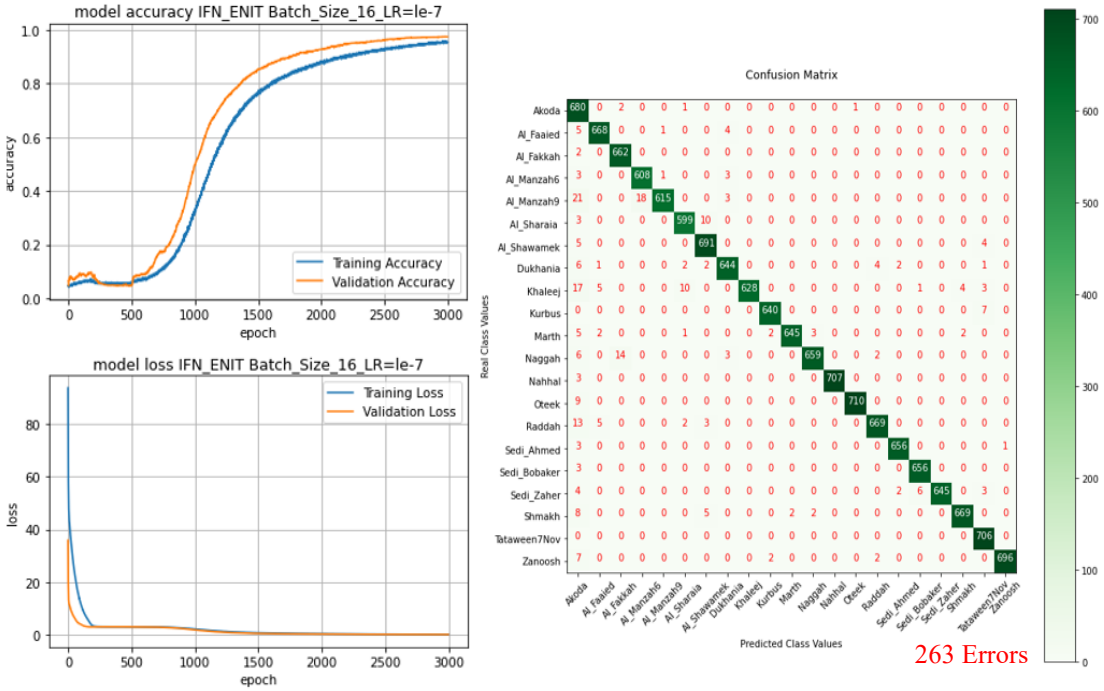


Figure 6.46 LC and CM with LR = 10^{-7} and BS = 16 using IFN/ENIT dataset

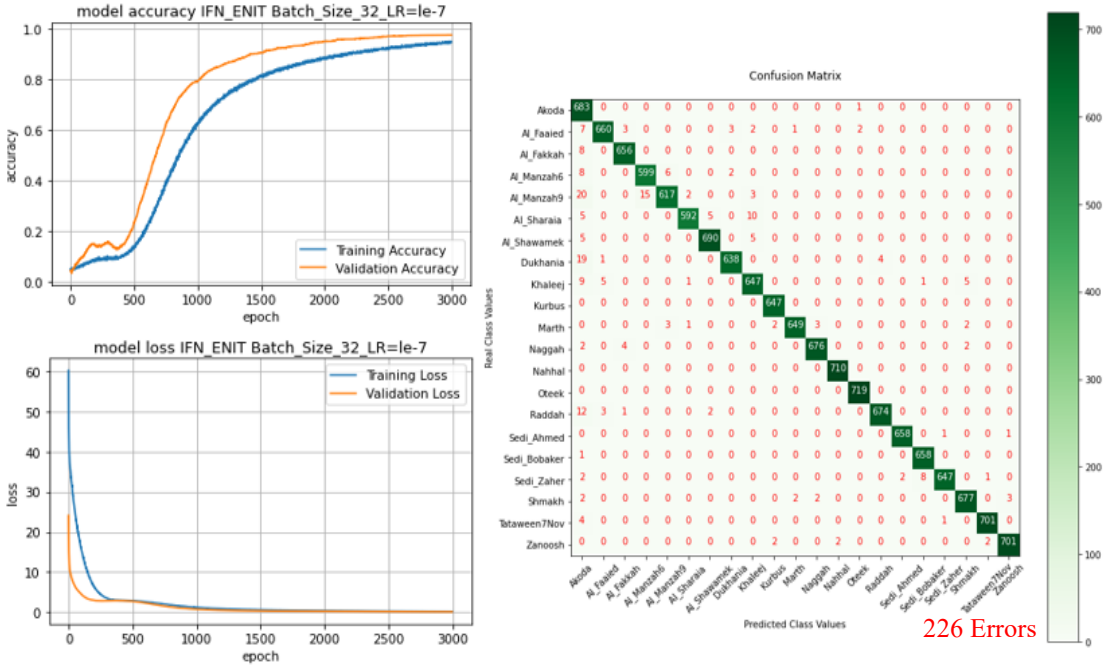


Figure 6.47 LC and CM with LR = 10^{-7} and BS = 32 using IFN/ENIT dataset

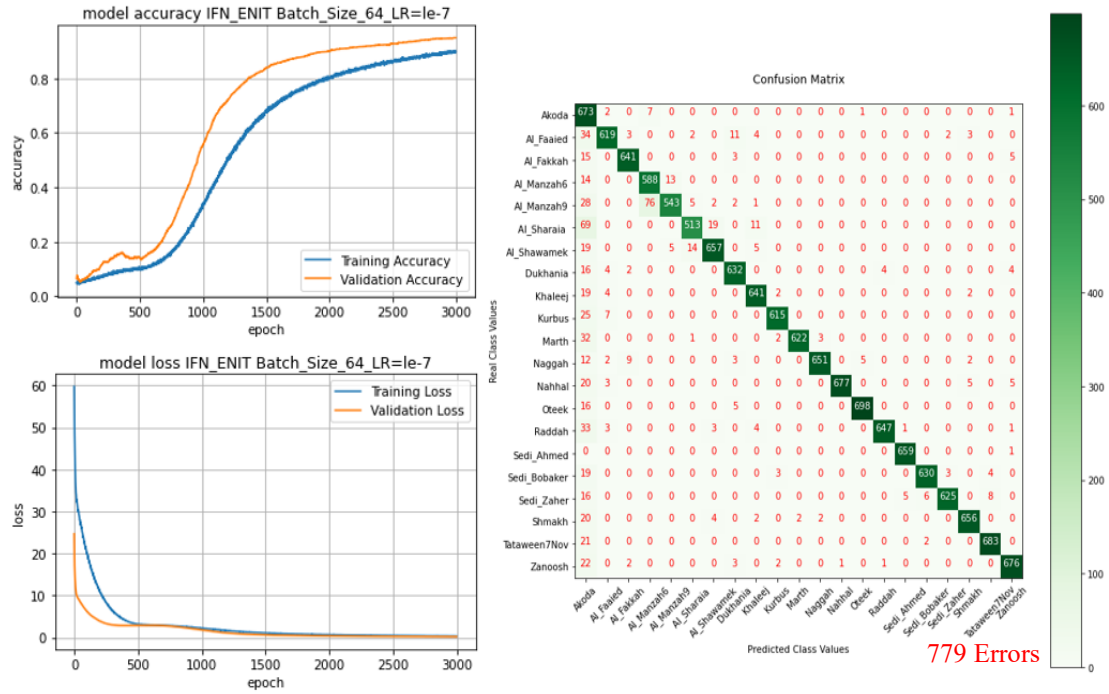


Figure 6.48 LC and CM with LR = 10^{-7} and BS = 64 using IFN/ENIT dataset

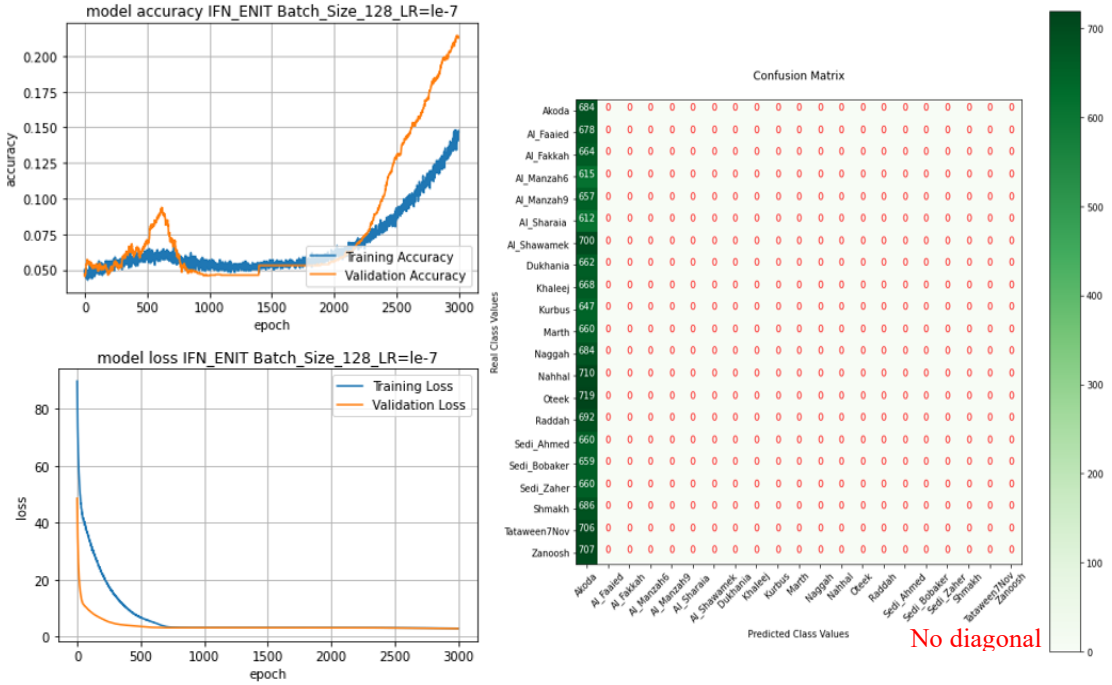


Figure 6.49 LC and CM with LR = 10^{-7} and BS = 128 using IFN/ENIT dataset

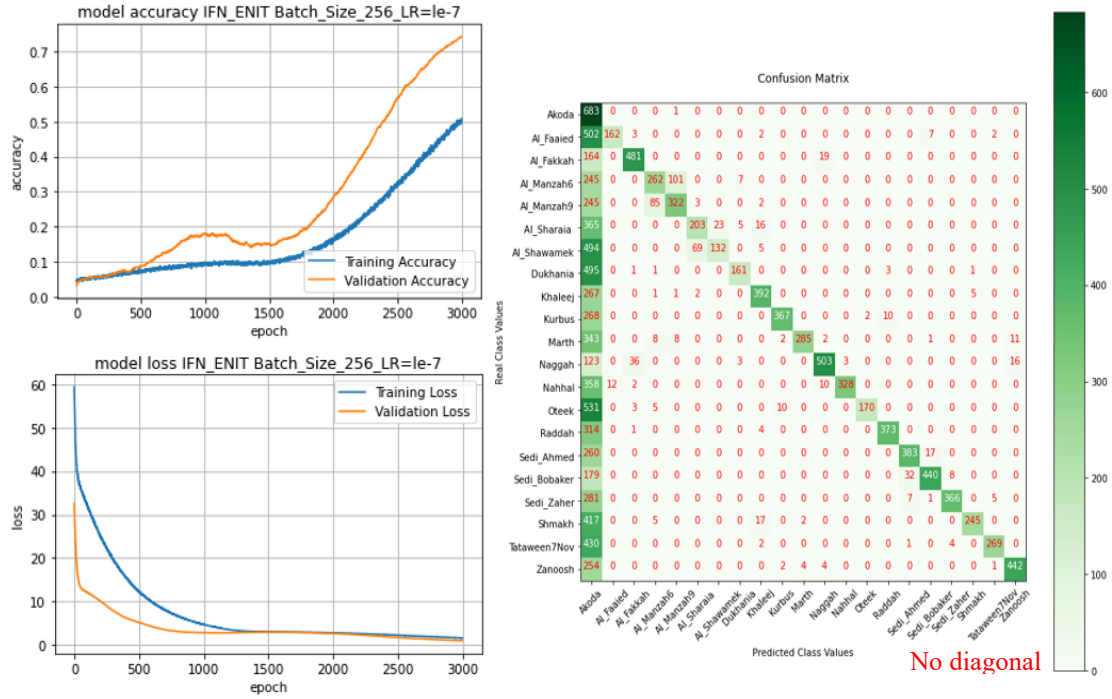


Figure 650 LC and CM with LR = 10^{-7} and BS = 256 using IFN/ENIT dataset

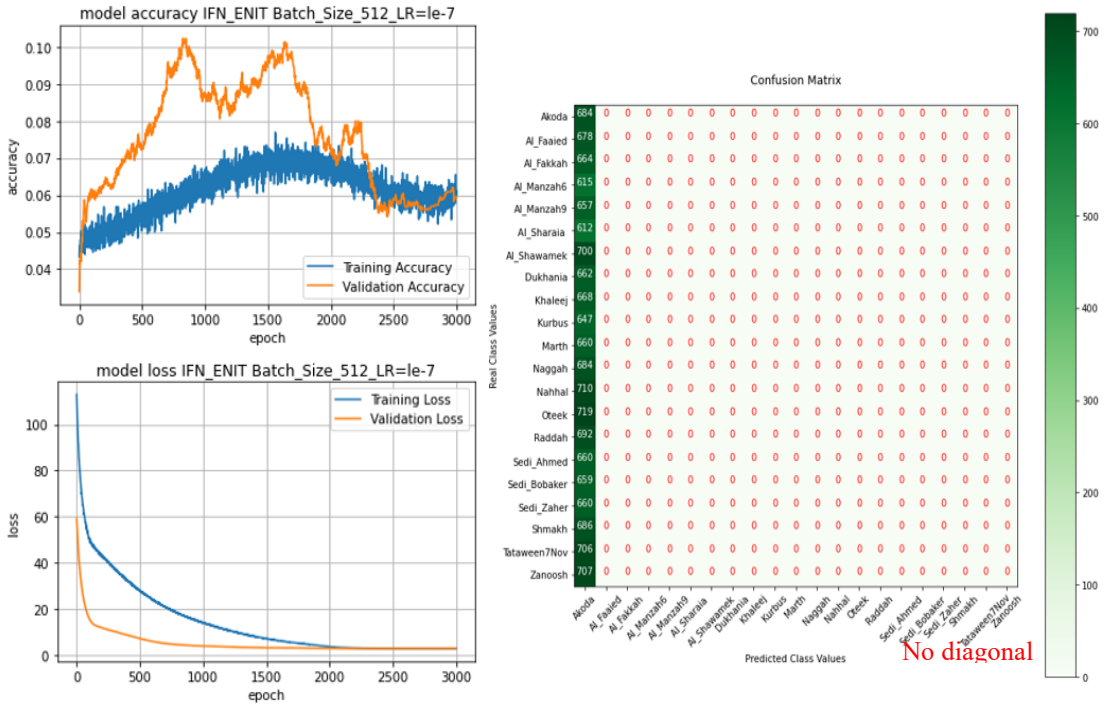


Figure 6.51 LC and CM with LR = 10^{-7} and BS = 512 using IFN/ENIT dataset

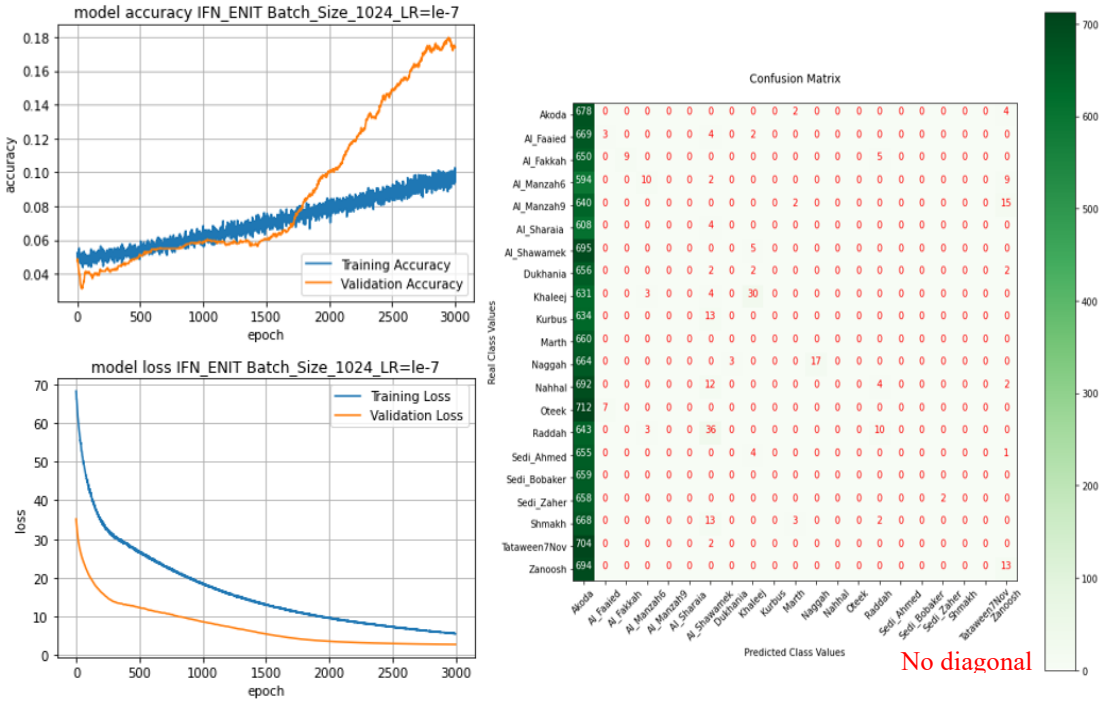


Figure 6.52 LC and CM with LR = 10^{-7} and BS = 1024 using IFN/ENIT dataset

By looking at Figure 6.46 to Figure 6.52, there is instability in the proposed model on IFN/ENIT dataset when using learning rate = 10^{-7}. The learning rate here is very small which would make the gradient moves uneven and the convergence between the training accuracy curve and validation accuracy curve inaccurate. Moreover, as the batch size increases.

- Both the training and validation loss curve diverge, and this divergence is clear in Figure 6.Figure 6.52
- The number of confusion matrix errors increases and reaches to the No diagonal state, and the classification accuracy rate will decrease.

The conclusion is, the hyperparameters such as learning rate = 10^{-7} with batch sizes (16, 32, 64, 128, 256, 512, 1024) when applying IFN/ENIT is not suitable, and the system is unstable.

The Adam optimizer is used to evaluate the performance when applying IFN/ENIT using the proposed model. For the loss function, the categorical cross-entropy loss is used. After training the DCNN model to calculate the probability of each image over the classes, the model is evaluated by using the testing dataset. On IFN/ENT, different batch sizes equal to 16, 32, 64, 128, 256, 512, and 1024 were tested. Also, learning rates of 10^{-3}, $10^{-4}, 10^{-5}, 10^{-6}$, and 10^{-7} were evaluated with epochs equal to 3000. With each learning rate, the model was trained with 7 different batch sizes. So, five tables of results were produced, and each table consists of one learning rate and 7 batch sizes with results (accuracy rate, loss rate, and confusion matrix errors) where the loss rate here is considered as the GER. The results between the five tables are evaluated based on the following criteria:

- Learning rate values (10^{-3} to 10^{-7}) and batch sizes.
- Low loss error rate (GER).
- High accuracy rate.
- Confusion matrix errors.

- Model response speed by looking at the learning curve graph and determine in which epoch number the accuracy curve starts rising and in which epoch number does the loss curve start coming down.

After applying these criteria, a table summarizing the conditions to obtain the best results is compiled for each dataset. The result is shown in Table 6.12 for IFN/ENIT dataset. After applying the above criteria on Table 6.6, it was concluded that the best accuracy rate is 99.87% and the error rate GER = 0.0181 for the following reasons:

- The learning rate (10^{-4}) is low which makes the model train at a reasonable speed and allows the gradient descent to produce a smooth output.
- There is a reasonable number of epochs (3000 epochs with 23 training iterations and 14 testing iterations for each epoch) required and batch size is = 512. These conditions allow relatively fast computing time.

Table 6.12 Best of the five tables of IFN/ENIT dataset

Learning rate	Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
			Train	Test			
10^{-3}	1024	3000	12	7	0.9991	0.5749	9
10^{-4}	512	3000	23	14	0.9987	0.0181	8
10^{-5}	128	3000	89	55	0.9967	0.0178	13
10^{-6}	32	3000	353	218	0.9968	0.0195	18
10^{-7}	32	3000	353	218	0.9793	0.0853	226

- The lowest error rate 0.0178 with high accuracy rate was not chosen because by looking at Figure 6.37, the training accuracy curve and validation accuracy curve start raising up in convergence fashion from the beginning until approximately epoch 200 where the convergence goes all the way along. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 100 where the convergence started to be steady, and the error is the lowest. Whereas Figure 6.34, shows that the convergence between the

training accuracy curve and validation accuracy curve starts approximately at epoch = 100 and continue all the way as a straight line. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 20 where the convergence started to be steady, and the error is the lowest which is 0.0181. Now, by comparing Figure 6.34 with error rate 0.0181 and Figure 6.37 with error rate 0.0178 in terms of model response speed in the accuracy curve and the loss curve, the conclusion is that Figure 6.34 with error rate 0.0181 shows faster response than Figure 6.37 with an error rate of 0.0178.

- By looking at Table 6.6 a setting converging to (GER = 0.0181) is selected even though there are lower values of GER in the Table 6.12. This choice was made because the accuracy rate (99.87%) is high and the number of errors in the confusion matrix for the chosen setting is the lowest compared to other results.

6.3 Augmented AHWD Experimental Phase

As mentioned in Chapter 5, The model is evaluated by using the testing dataset. On augmented AHWD, different batch sizes equal to 16, 32, 64, 128, 256, 512, and 1024 were tested. Also, learning rates of 10^{-3}, $10^{-4}, 10^{-5}, 10^{-6}$, and 10^{-7} were evaluated with epochs equal to 1000. With each learning rate, the model was trained with 7 different batch sizes. So, five tables of results were produced, and each table consists of one learning rate and 7 batch sizes with results (training iteration number, testing iteration number, accuracy rate, loss rate, and confusion matrix errors) where the loss rate here is considered as the GER. The results between the five tables are evaluated based on the following criteria:

- Learning rate values (10^{-3} to 10^{-7}) and batch sizes.
- A low loss error rate or Generalization Error Rate (GER) has the highest priority.
- High accuracy rate.
- Confusion matrix errors.

- Model response speed by looking at the learning curve graph and determine in which epoch number the accuracy curve starts rising and in which epoch number does the loss curve start coming down.

After the evaluation by using these criteria, the best performance from each table has been collected.

- **First**, when learning rate = 10^{-3} and batch sizes (16, 32, 64, 128, 256, 512, 1024) are used, the results would be as shown in Table 6.13.

Table 6.13 Augmented AHWD dataset with learning rate = 10^{-3}.

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	1000	2470	1521	0.1427	1.9461	No Diagonal
32	1000	1235	761	0.8086	0.4965	>5000
64	1000	618	381	0.7665	0.8864	>5000
128	1000	309	191	0.9938	0.9282	251
256	1000	155	96	0.9984	2.0016	21
512	1000	78	48	0.9990	0.6597	21
1024	1000	39	24	0.9984	0.1148	23

By looking at Table 6.13, the lowest error rate (GER) is 0.1148, with accuracy rate 99.84%, and lower confusion matrix error = 23. This is acceptable in this training and testing session but cannot be generalized because Figure 6.53 shows an overfitting starts to rise after epoch 900; that is, the validation loss curve starts to diverge up off the training loss curve. In the conclusion, the proposed model cannot generalized on augmented AHWD with learning rate = 10^{-3} and batch sizes 1024.

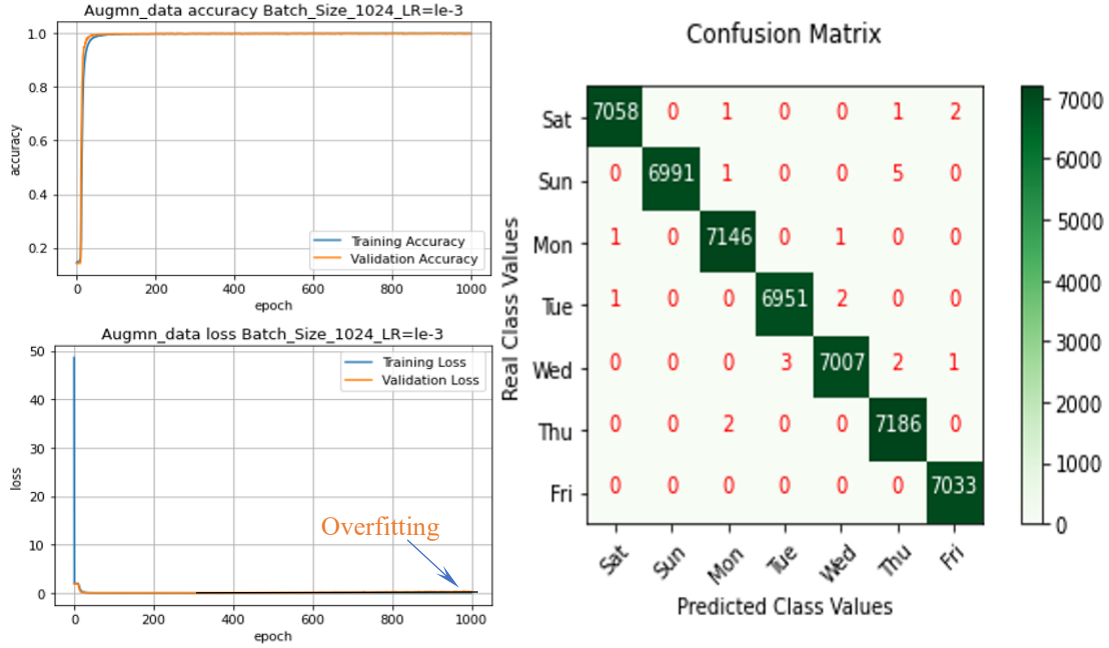


Figure 6.53 LC and CM with LR = 10^{-3} and BS = 1024 using augmented AHWD

All other results in Table 6.13 have a high error rate, and this would lead to overfitting or instability in the system, then cannot be generalized as shown in Figure 6.54 to Figure 6.59

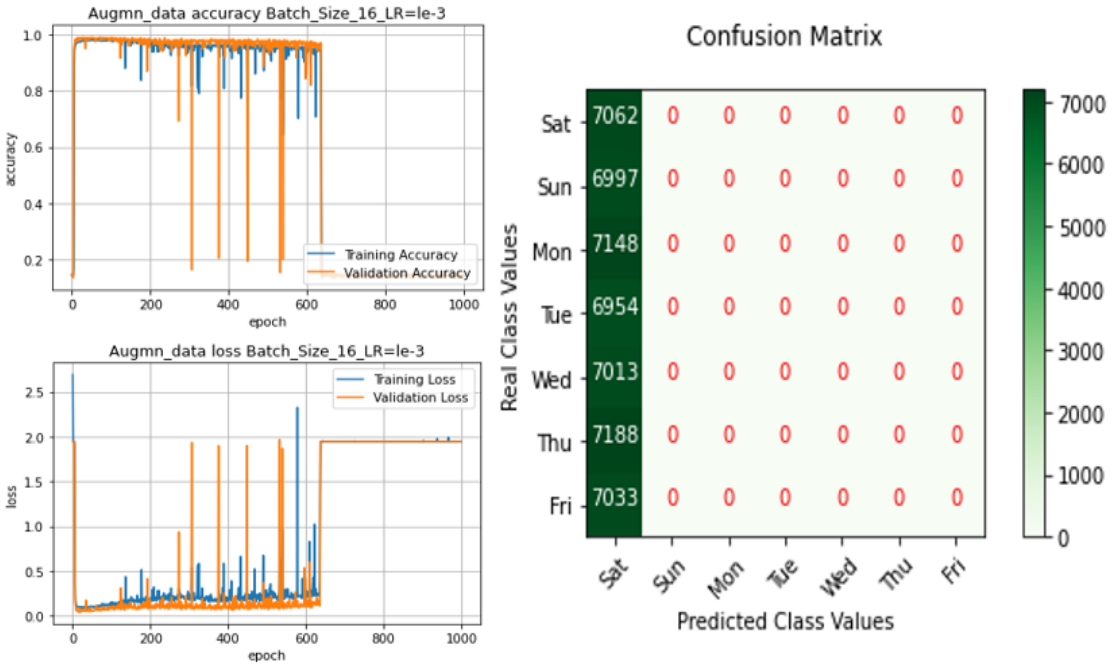


Figure 6.54 LC and CM with LR = 10^{-3} and BS = 16 using augmented AHWD

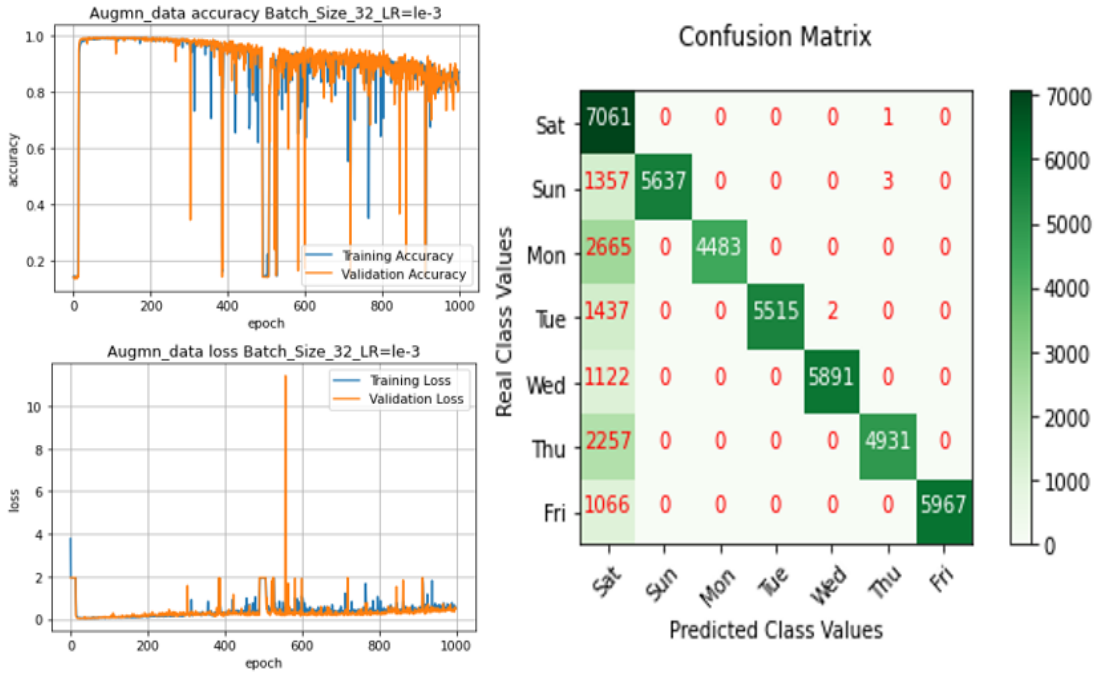


Figure 6.55 LC and CM with LR = 10^{-3} and BS = 32 using augmented AHWD

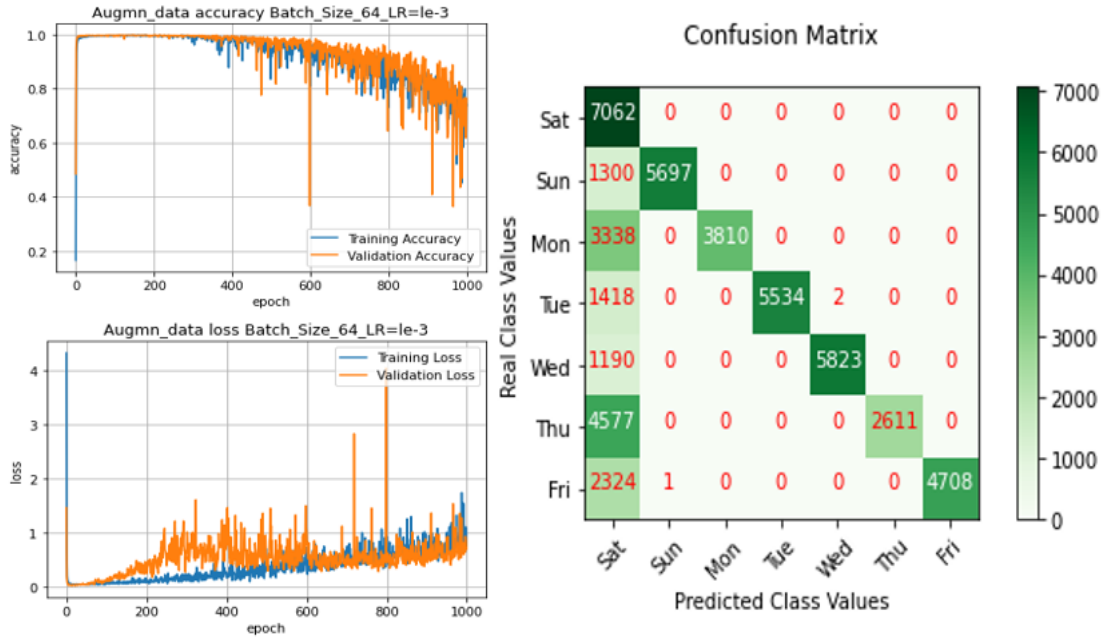


Figure 6.56 LC and CM with LR = 10^{-3} and BS = 64 using augmented AHWD

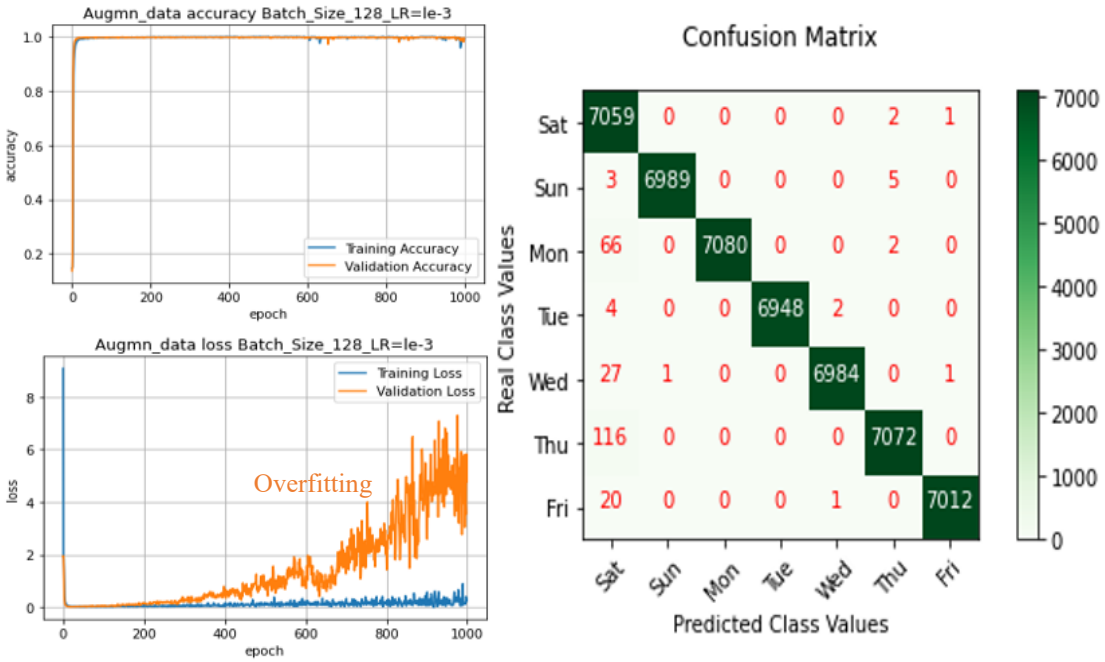


Figure 6.57 LC and CM with LR = 10^{-3} and BS = 128 using augmented AHWD

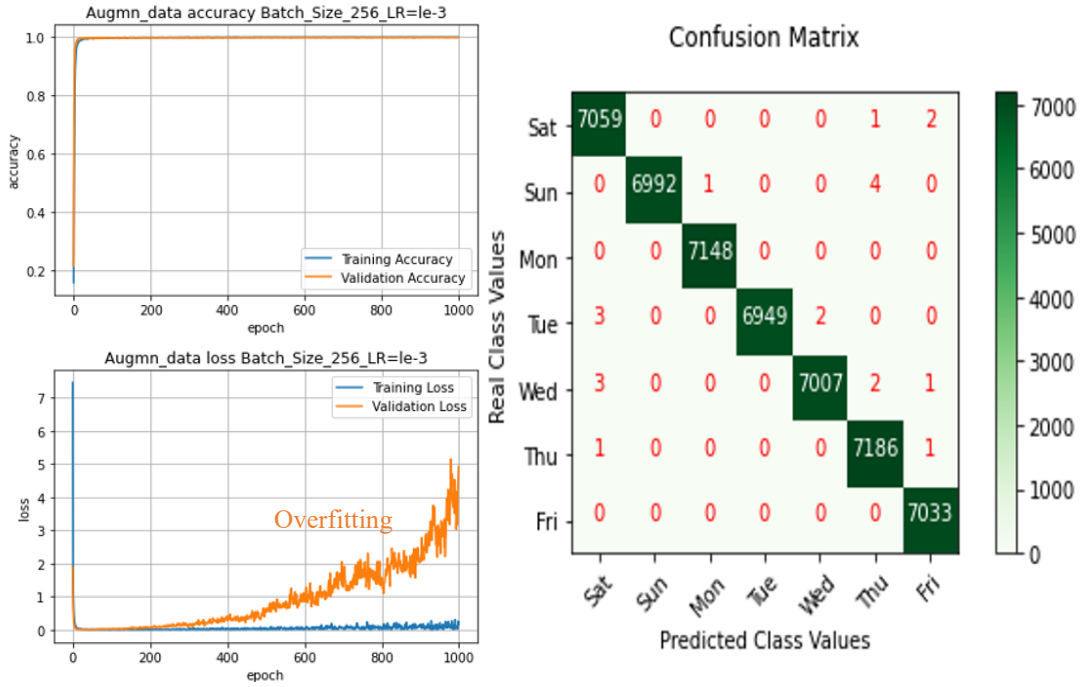


Figure 6.58 LC and CM with LR = 10^{-3} and BS = 256 using augmented AHWD.

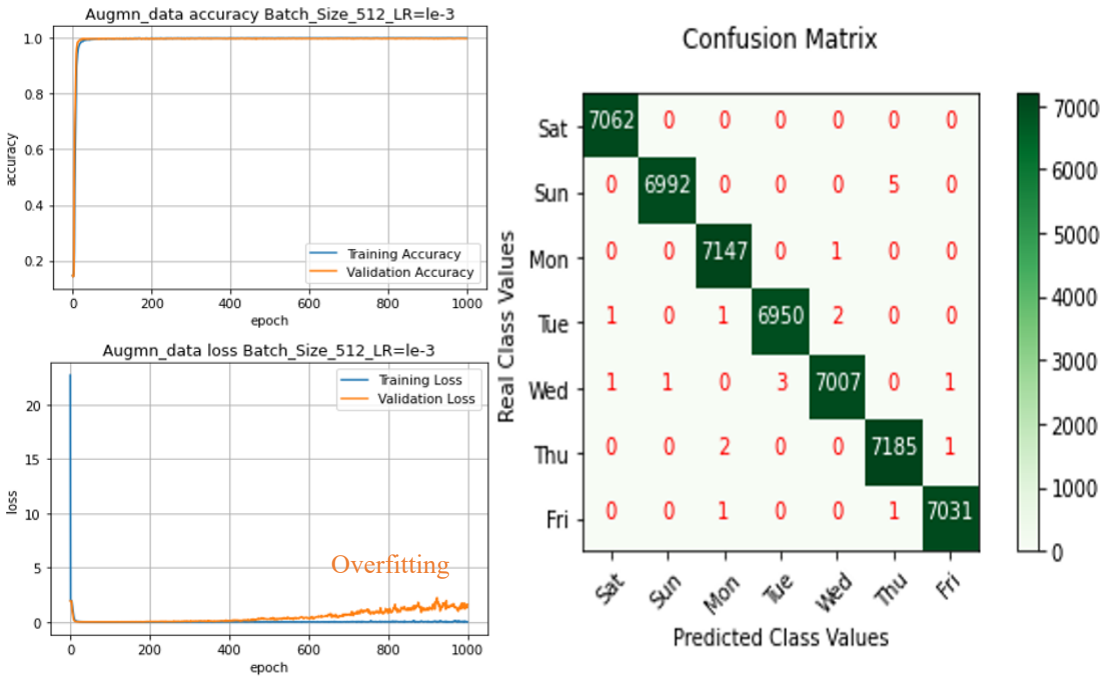


Figure 6.59 LC and CM with LR = 10^{-3} and BS = 512 using augmented AHWD.

- **Second**, when learning rate hyperparameter = 10^{-4} and batch sizes hyperparameters (16, 32, 64, 128, 256, 512, 1024) are used, the results would be as shown in Table 6.14.

Table 6.14 Augmented AHWD dataset with learning rate = 10^{-4}.

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	1000	2470	1521	0.9991	0.2076	20
32	1000	1235	761	0.9989	0.1037	17
64	1000	618	381	0.9991	0.0554	18
128	1000	309	191	0.9989	0.0365	15
256	1000	155	96	0.9988	0.0182	22
512	1000	78	48	0.9988	0.0149	20
1024	1000	39	24	0.9990	0.0074	22

Table 6.14 depicts that the best accuracy rate is 99.90%, the lowest error rate (GER) = 0.0074, and low confusion matrix error number = 22. This conclusion is the best

performance, can be generalized, and is acceptable in this training and testing session with batch size hyperparameter = 1024 because it has lowest error rate with the highest accuracy rate, and the lowest processing time. Figure 6.60 depicts that the training accuracy curve

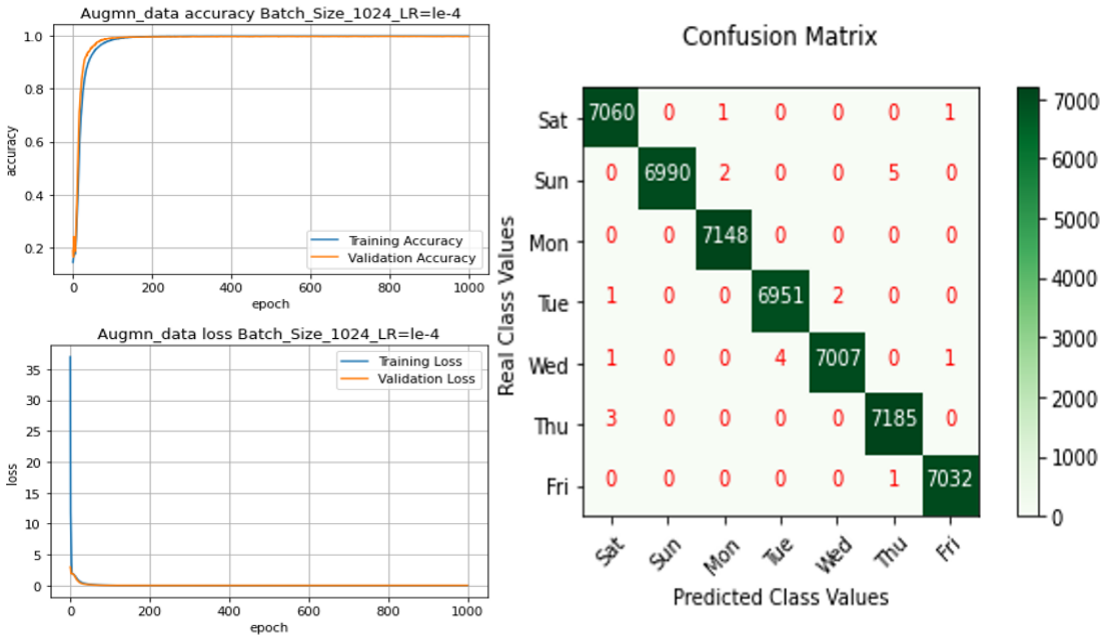


Figure 6.60 LC and CM with LR = 10^{-4} and BS = 1024 using augmented AHWD an

approximately epoch 125 where the convergence goes all the way long. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 50 where the convergence started to be steady, and the error is the lowest. Both curves never touch each other, there is a gap between them called a generalization gap.

The results in Table 6.14 with batch sizes hyperparameters 256 and 512 are acceptable and no overfitting is produced as shown in Figure 6.61 and Figure 6.62. However, the results in Table 6.14 with batch sizes hyperparameters 16, 32, 64, and 128 are not acceptable because of the overfitting as shown in Figure 6.63 to Figure 6.66. The conclusion is that the proposed model cannot be generalized utilizing learning rate hyperparameter 10^{-4} with batch sizes hyperparameters 16, 32, 64, and 128. So, it is unstable and not robust.

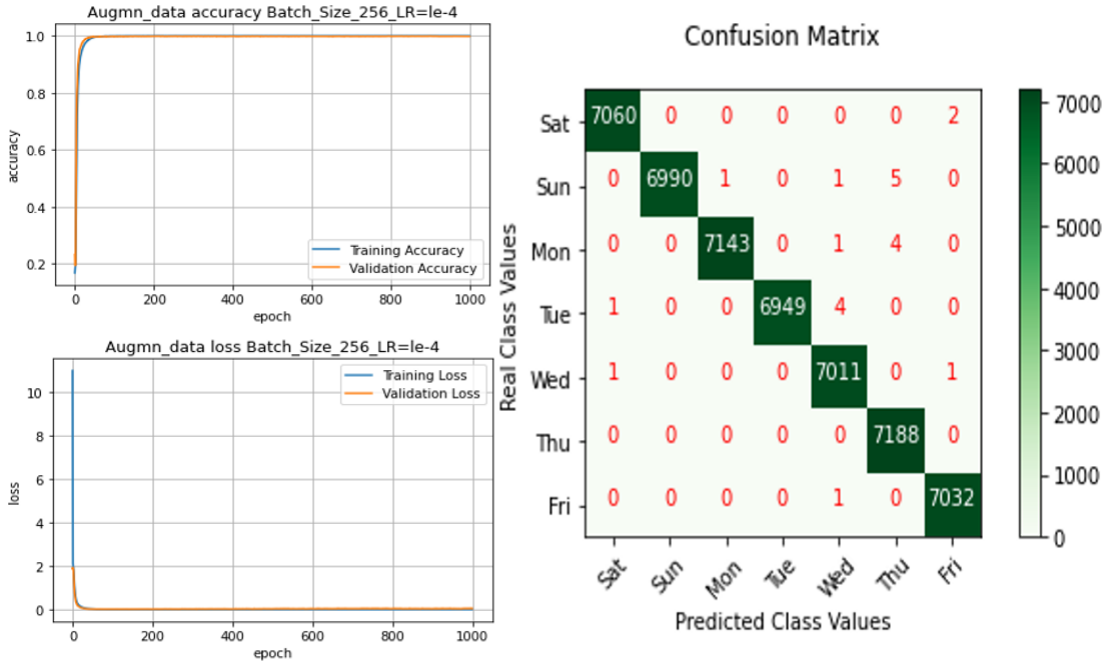


Figure 6.61 LC and CM with LR = 10^{-4} and BS = 256 using augmented AHWD

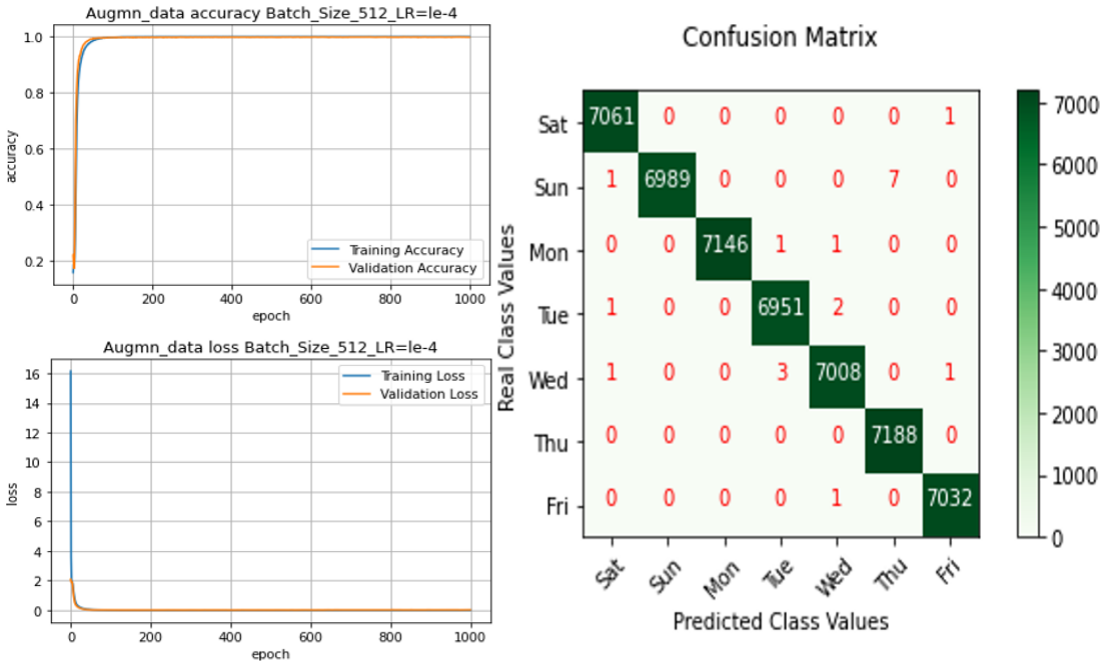


Figure 6.62 LC and CM with LR = 10^{-4} and BS = 512 using augmented AHWD

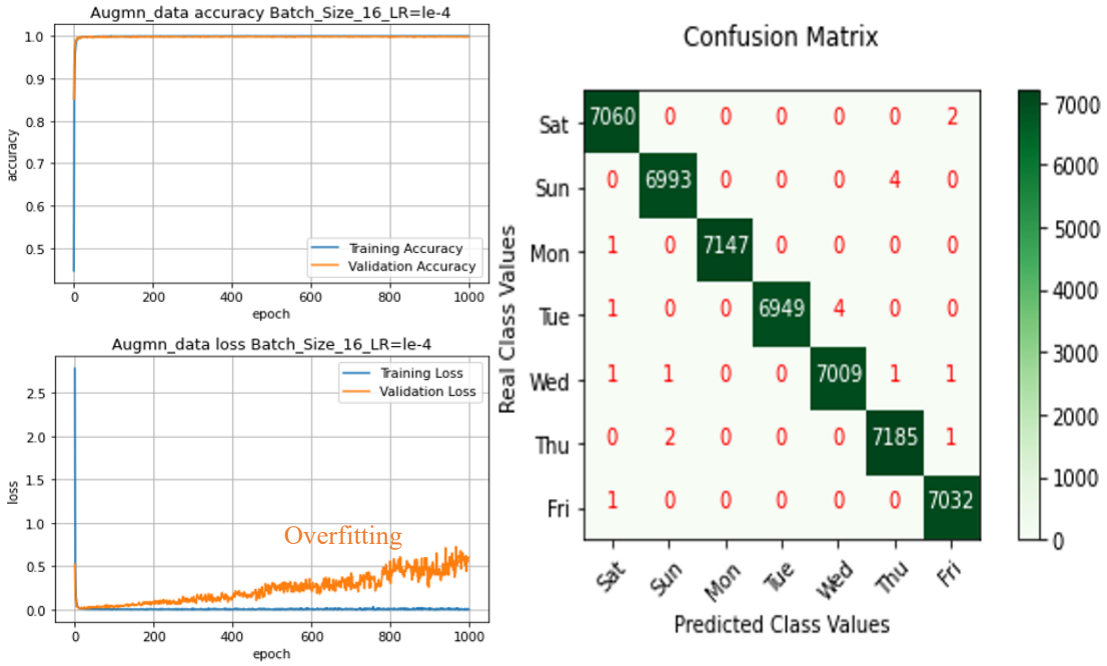


Figure 6.63 LC and CM with LR = 10^{-4} and BS = 16 using augmented AHWD

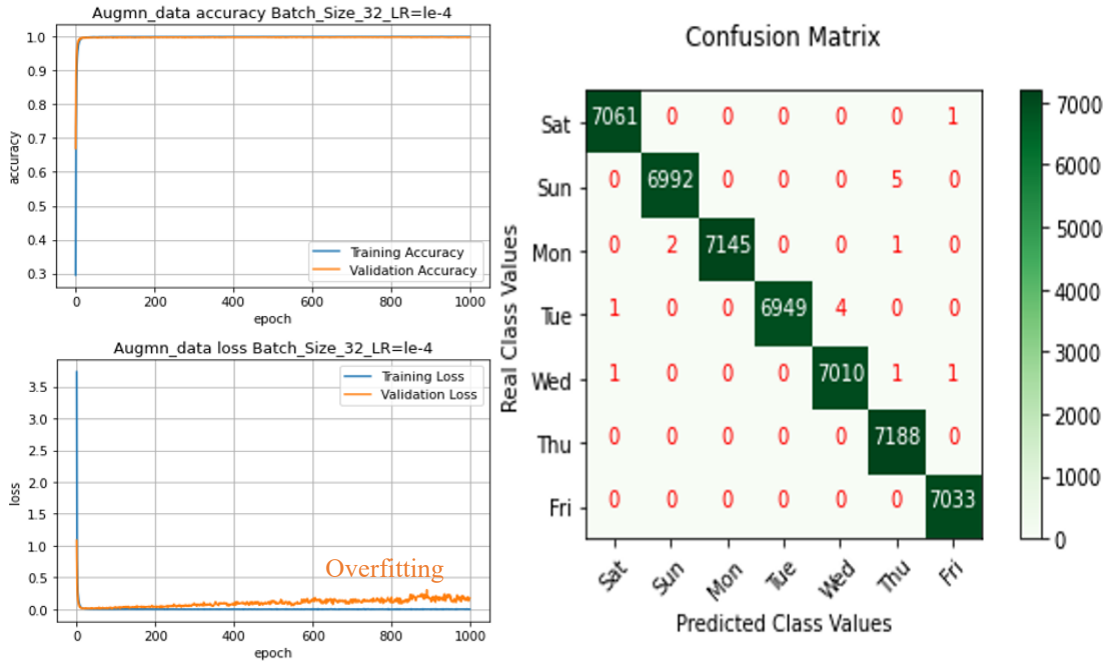


Figure 6.64 LC and CM with LR = 10^{-4} and BS = 32 using augmented AHWD

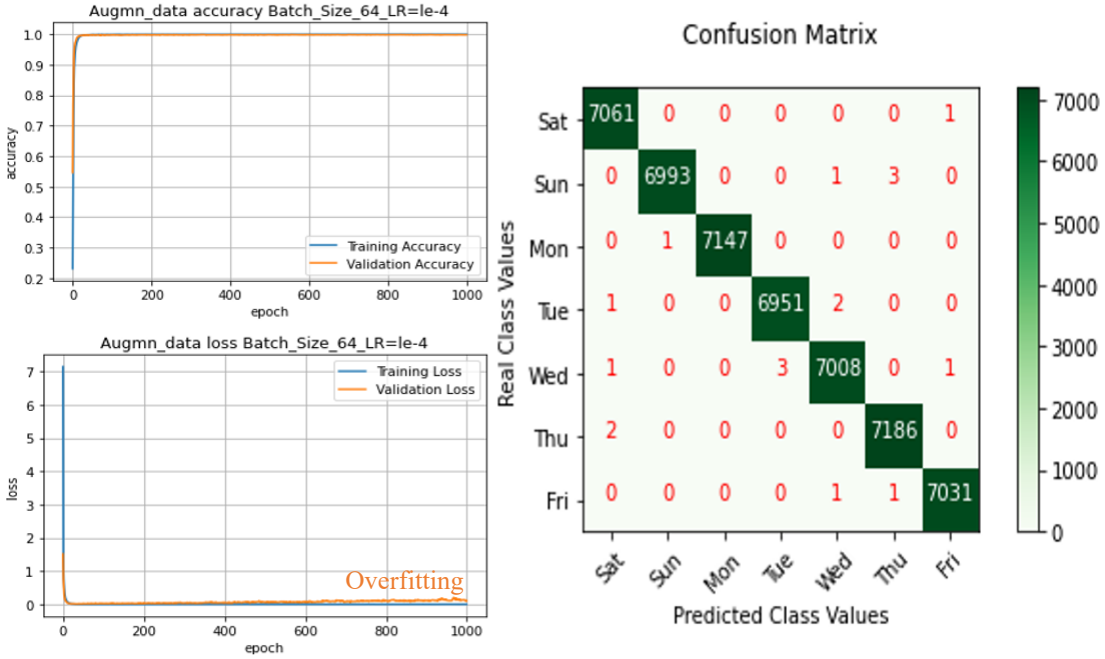


Figure 6.65 LC and CM with LR = 10^{-4} and BS = 64 using augmented AHWD

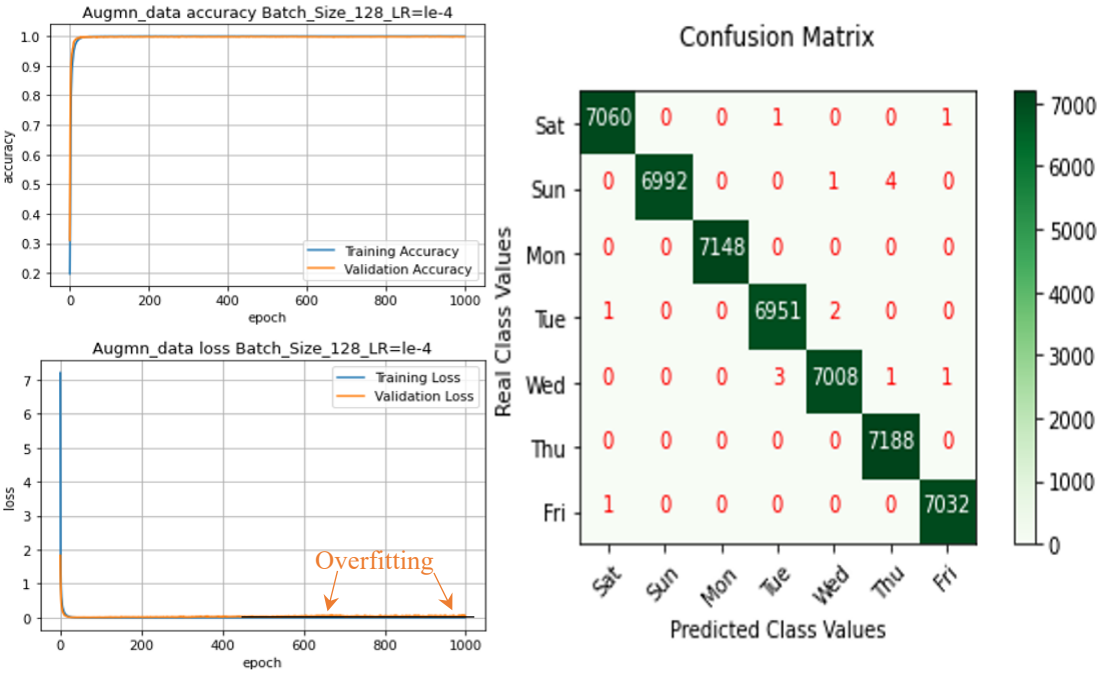


Figure 6.66 LC and CM with LR = 10^{-4} and BS = 128 using augmented AHWD

- **Third**, when learning rate hyperparameter = 10^{-5} and batch sizes hyperparameters (16, 32, 64, 128, 256, 512, 1024) are applied, the results would be as shown in Table 6.15.

All results in Table 6.15 are acceptable and the accuracy rates are 99.90%, 99.90%, 99.86%, 99.86%, 99.88%, 99.81%, and 99.77% when using batch sizes 16, 32, 64, 128, 256, 512, and 1024 with error rates (GER) are 0.0116, 0.0091, 0.0111, 0.0070, 0.0086, 0.0094, and 0.0100 respectively, and the number of confusion matrix error is 20, 17, 27, 22, 24, 27, and 22, respectively. There is no overfitting in all the training sessions.

Table 6.15 Augmented AHWD dataset with learning rate = 10^{-5}.

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	1000	2470	1521	0.9990	0.0116	20
32	1000	1235	761	0.9990	0.0091	17
64	1000	618	381	0.9986	0.0111	27
128	1000	309	191	0.9986	0.0070	22
256	1000	155	96	0.9988	0.0086	24
512	1000	78	48	0.9981	0.0094	27
1024	1000	39	24	0.9977	0.0100	22

By rounding up the accuracy rate values, most of the values would be 99.90%. However, Table 6.15 depicts that the lowest error rate (GER) = 0.0070, with high accuracy rate 99.86% (99.90% rounded up), and low confusion matrix error number = 22. This conclusion is the best performance, can be generalized, and is acceptable in this training and testing session with batch size hyperparameter = 128 as shown in Figure 6.67.

Figure 6.67 depicts that the training accuracy curve and validation accuracy curve start up in convergence fashion from the beginning until approximately epoch 100 where the convergence goes all the way along. Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 50 where the convergence started to be steady, and the error is the lowest.

Both curves never touch each other, there is a gap between them called a generalization gap.

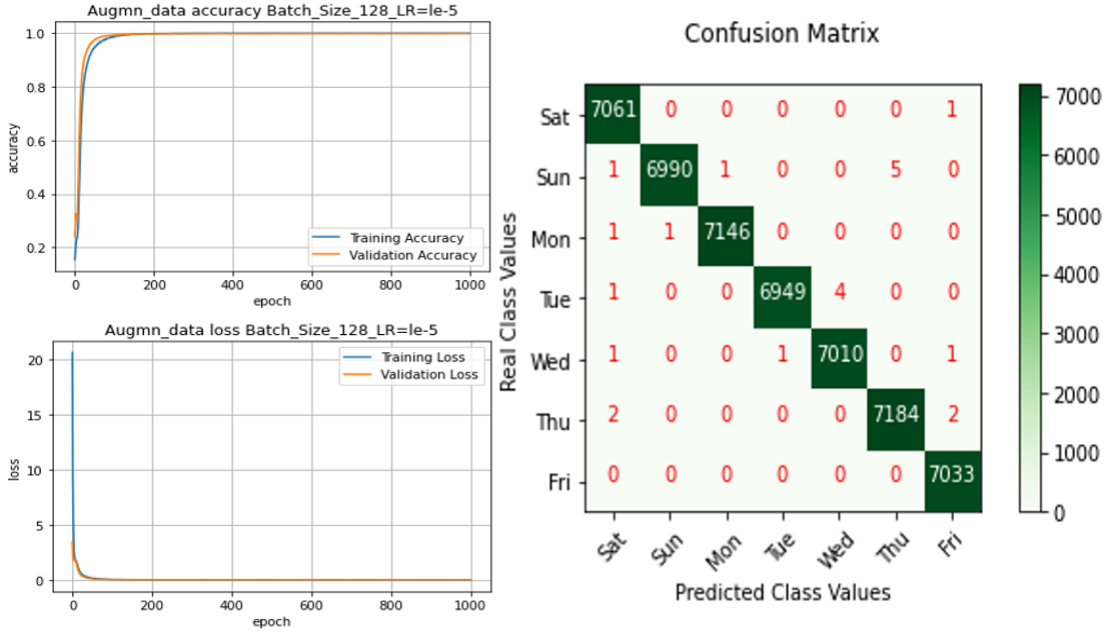


Figure 6.67 LC and CM with LR = 10^{-5} and BS = 128 using augmented AHWD

- **Fourth**, when learning rate hyperparameter = 10^{-6} and batch sizes hyperparameters (16, 32, 64, 128, 256, 512, 1024) are applied, the results would be as shown in Table 6.16.

All results in Table 6.16 are acceptable, but not all of them can be generalized, the lowest error rate (GER) = 0.0090, with the highest accuracy rate 99.81%, and the lowest confusion matrix error number = 25. This conclusion is the best performance, can be generalized, and is acceptable in this training and testing session with batch size hyperparameter = 16 as shown in Figure 6.68.

Table 6.16 Augmented AHWD dataset with learning rate = 10^{-6}.

Batch size	Epoch	Iteration		Acc rate	Loss rate	Confusion Matrix Errors
		Train	Test			
16	1000	2470	1521	0.9981	0.0090	25
32	1000	1235	761	0.9976	0.0094	27
64	1000	618	381	0.9966	0.0129	46
128	1000	309	191	0.9926	0.0222	92
256	1000	155	96	0.9910	0.0322	215
512	1000	78	48	0.9808	0.0730	891
1024	1000	39	24	0.9682	0.1204	1842

Figure 6.68 depicts that the training accuracy curve and validation accuracy curve start raising up in convergence fashion from the beginning until approximately epoch 300 where the convergence goes all the way long, Moreover, the convergence between the training loss curve and the validation loss curve starts to come down right from the beginning until it reached epoch 100 where the convergence started to be steady, and the error is the lowest. The curves never touch each other, there is a gap between them called a generalization gap.

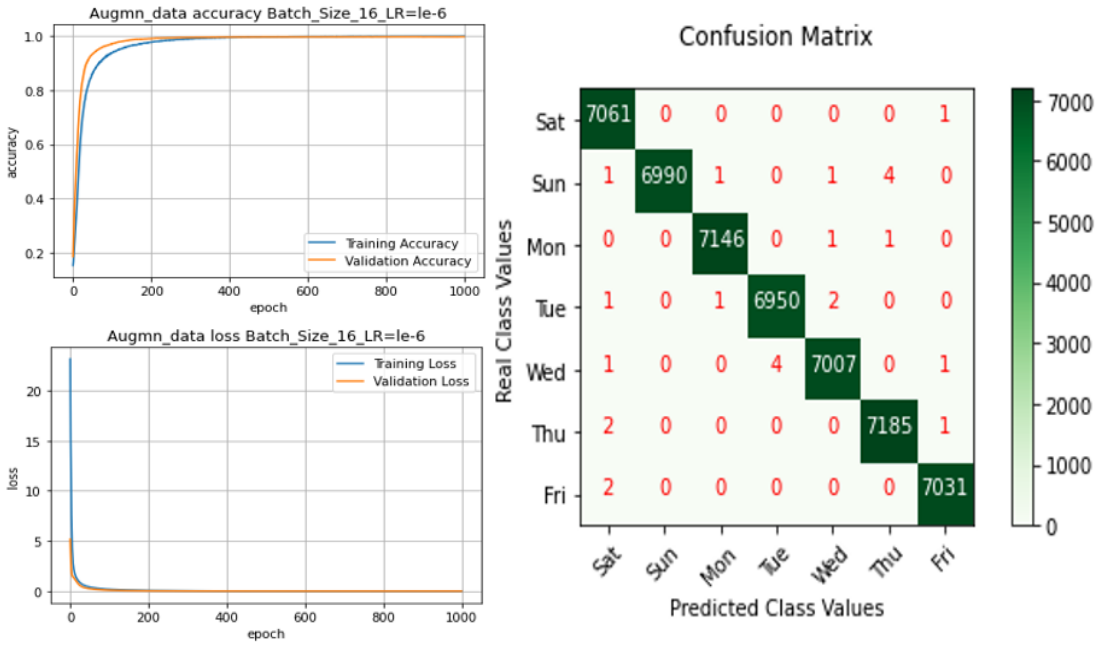


Figure 6.68 LC and CM with LR = 10^{-6} and BS = 16 using augmented AHWD

In this training and testing session using learning rate = 10^{-6}, as the batch size increases as the convergence between training accuracy curve and validation accuracy curve widens, the error rate (GER) increases, and the confusion matrix errors number increases. This conclusion shows the instability of the proposed model, and it is not robust as shown in Figure 6.69 to Figure 6.71.

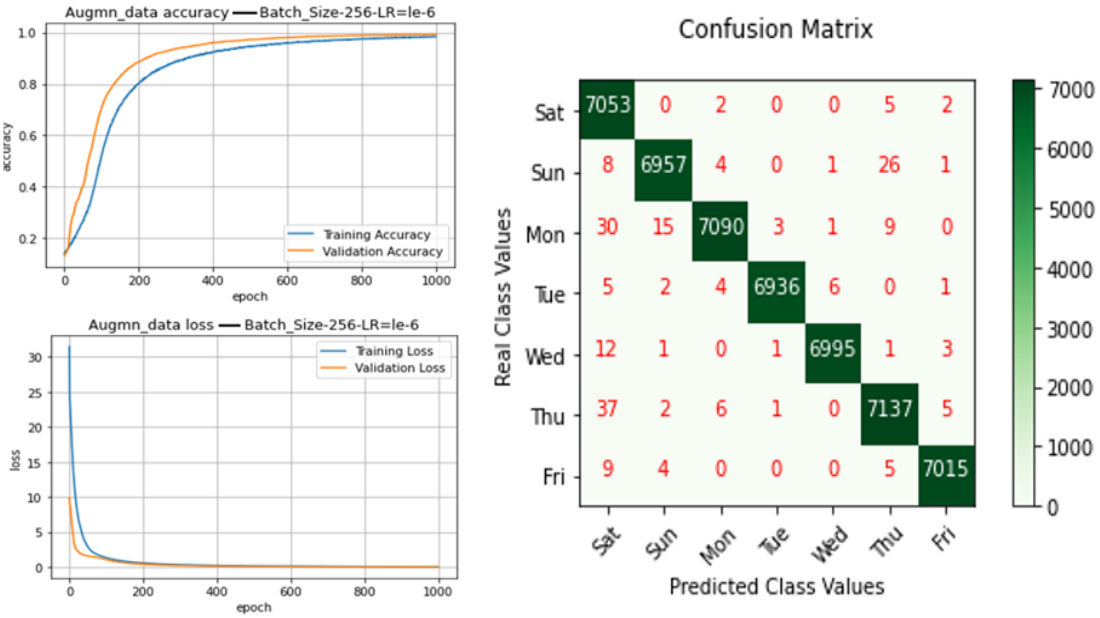


Figure 6.69 LC and CM with LR = 10^{-6} and BS = 256 using augmented AHWD

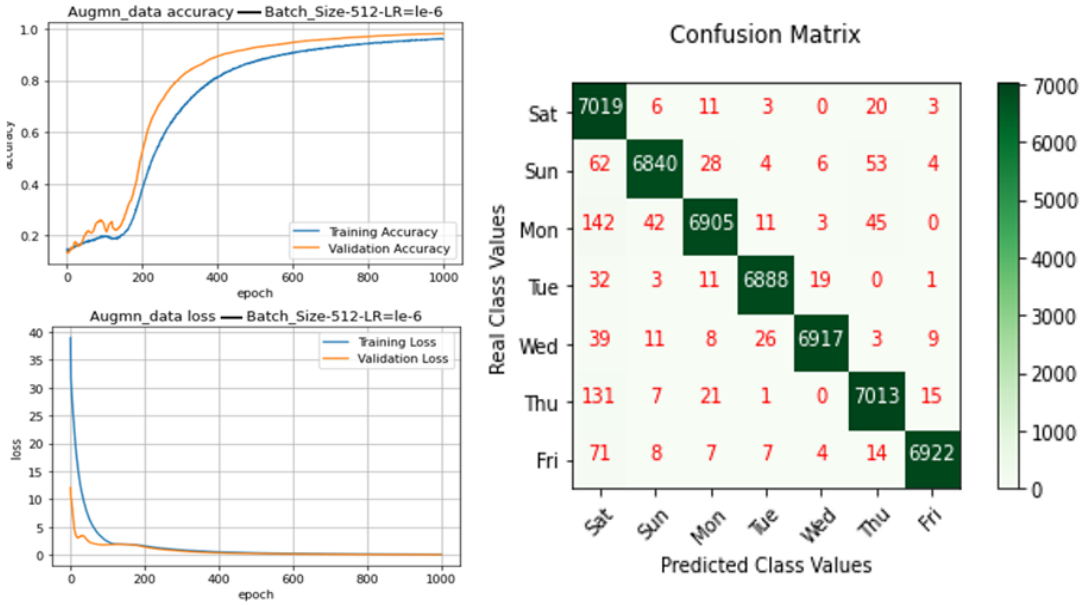


Figure 6.70 LC and CM with LR = 10^{-6} and BS = 512 using augmented AHWD

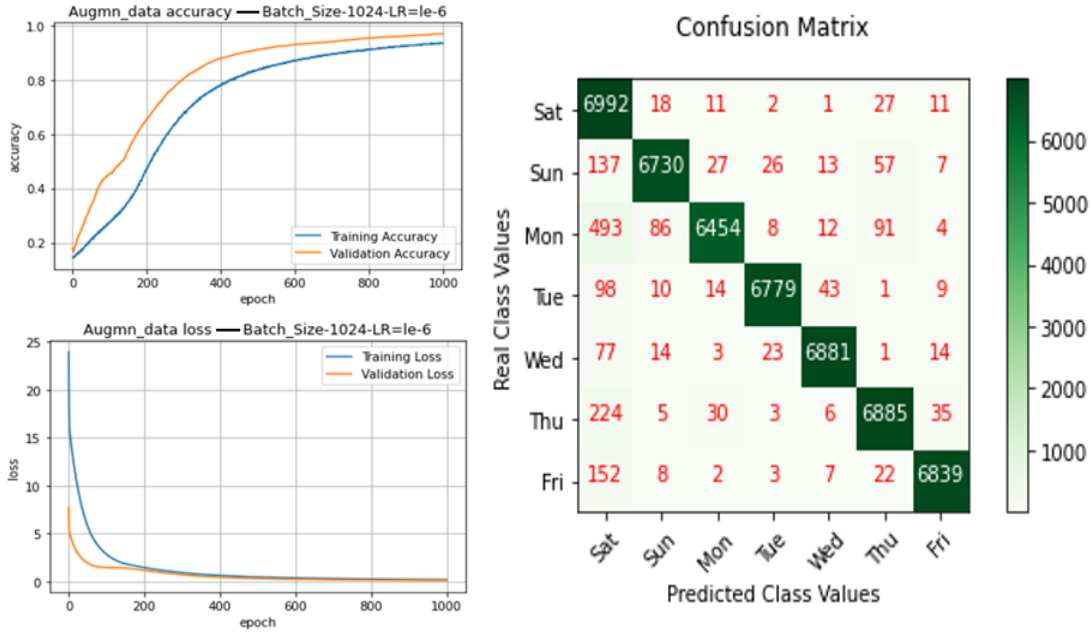


Figure 6.71 LC and CM with LR = 10^{-6} and BS = 1024 using augmented AHWD

- **Fifth,** when learning rate = 10^{-7} and batch sizes are (16, 32, 64, 128, 256, 512, 1024) are used, the results would be as shown in Table 6.17.

Table 6.17 Augmented AHWD dataset with learning rate = 10^{-7}.

Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
		Train	Test			
16	1000	2470	1521	0.9672	0.1240	1868
32	1000	1235	761	0.8695	0.4728	>5000
64	1000	618	381	0.9224	0.2907	>5000
128	1000	309	191	0.3497	1.7497	No Diagonal
256	1000	155	96	0.7278	1.0409	No Diagonal
512	1000	78	48	0.1789	1.9063	No Diagonal
1024	1000	39	24	0.2402	1.8649	No Diagonal

By looking at Table 6.17, best accuracy rate is 96.72%, error rate (GER) = 0.1240, and the confusion matrix error = 1868. This conclusion is somewhat acceptable in this training and testing session with batch size = 16. However, Table 6.17 shows that the error rate is high, the confusion matrix error number is high, and the processing time is high in terms of higher training iteration, higher testing iteration for each epoch.

So, as shown in Figure 6.72, the convergence between the training accuracy curve and the validation accuracy curve is indeterminate and very wide. From Table 6.17, as the batch size increases the classification accuracy rate decreases, the error rate (GER) increases, and the confusion matrix error increases. In more detail, the conclusion is that using learning rate = $10^{-7}$is not suitable for the proposed model on augmented AHWD since all the training and testing session using learning rate 10^{-7} with batch sizes (16, 32, 64, 128, 256, 512, 1024) make the proposed model unstable and unrobust as shown in Figure 6.72 to Figure 6.78.

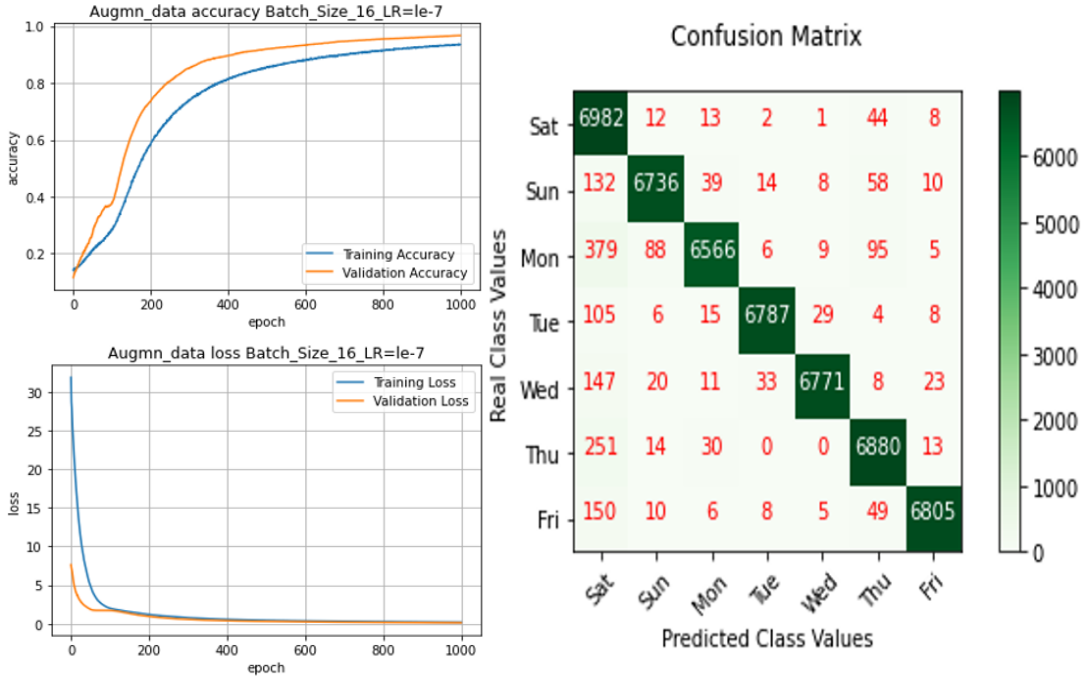


Figure 6.72 LC and CM with LR = 10^{-7} and BS = 16 using augmented AHWD

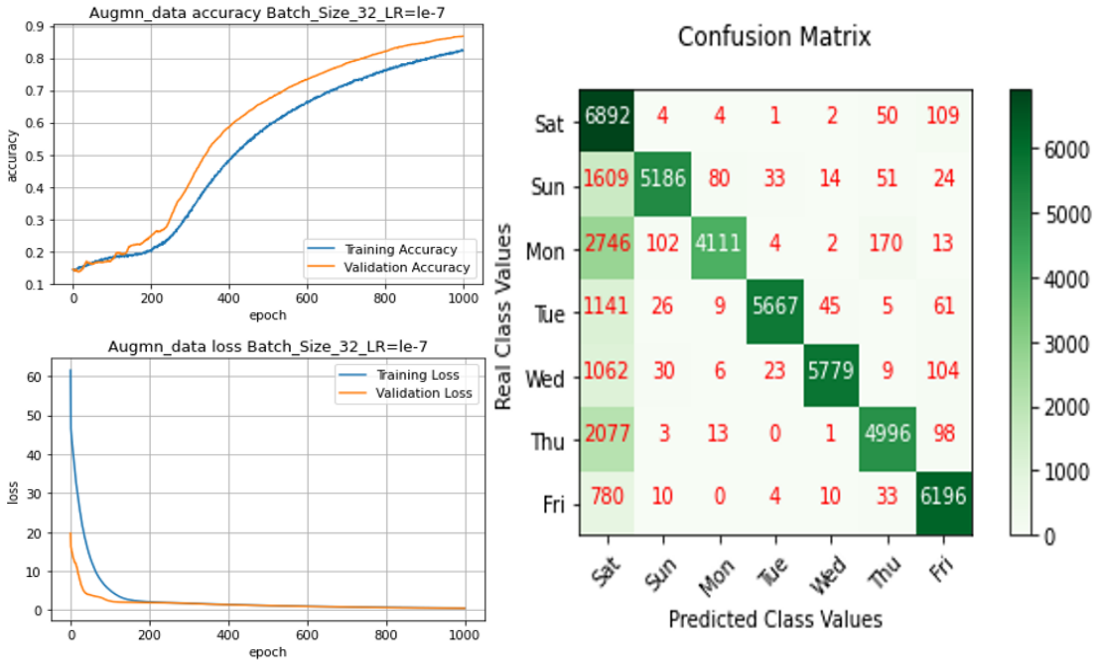


Figure 6.73 LC and CM with LR = 10^{-7} and BS = 32 using augmented AHWD

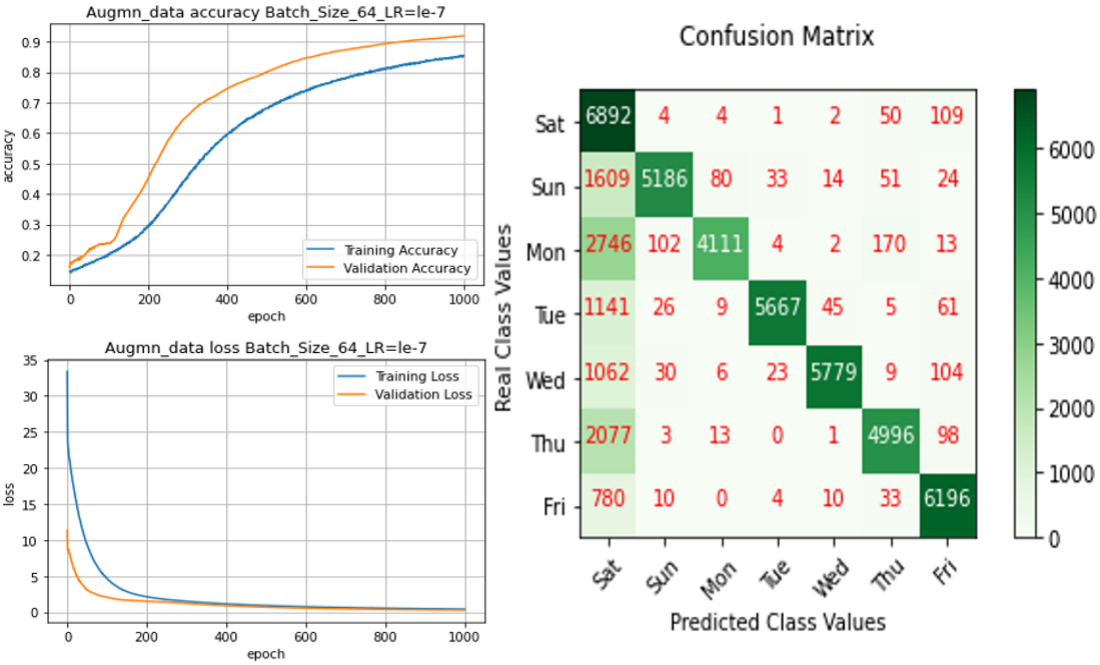


Figure 6.74 LC and CM with LR= 10^{-7} and BS = 64 using augmented AHWD

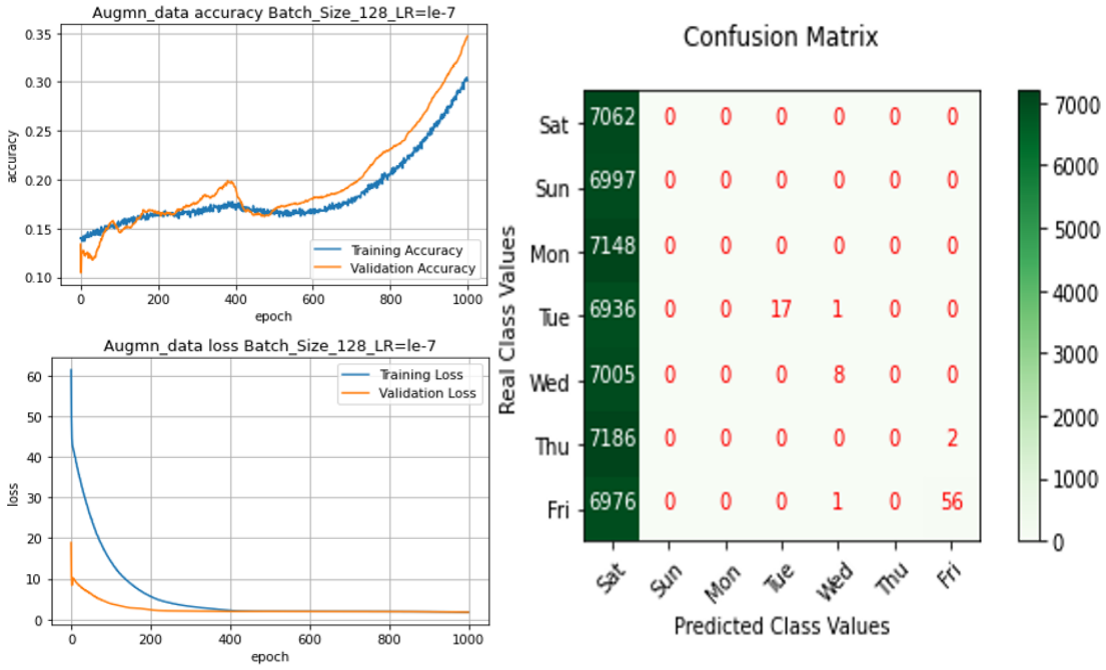


Figure 6.75 LC and CM with LR = 10^{-7} and BS = 128 using augmented AHWD

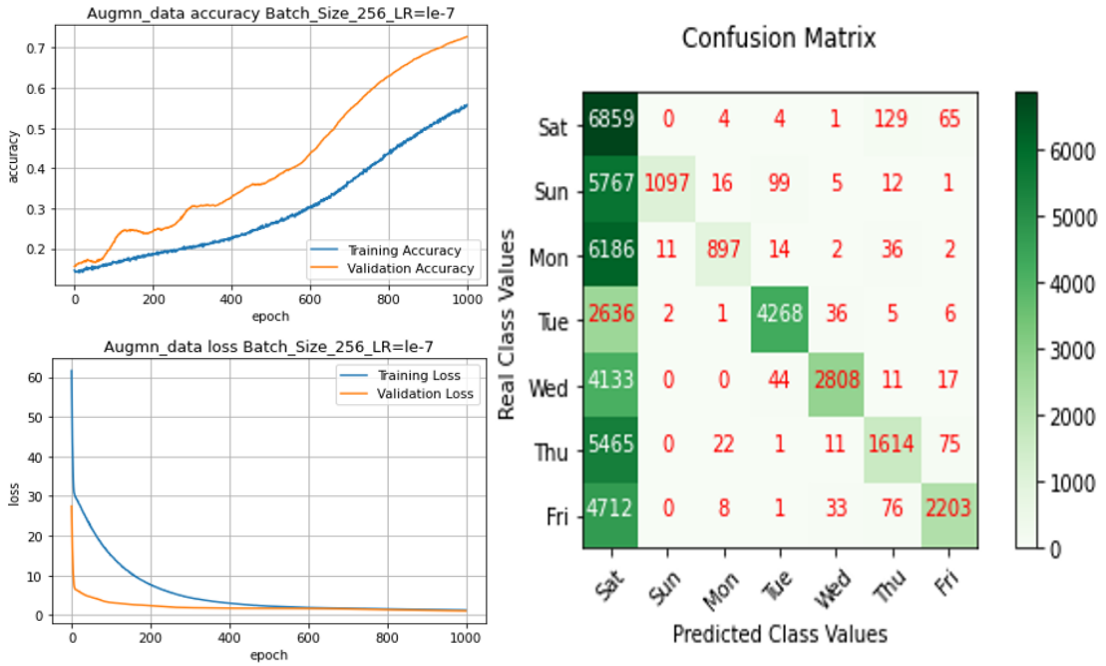


Figure 6.76 LC and CM with LR = 10^{-7} and BS = 256 using augmented AHWD

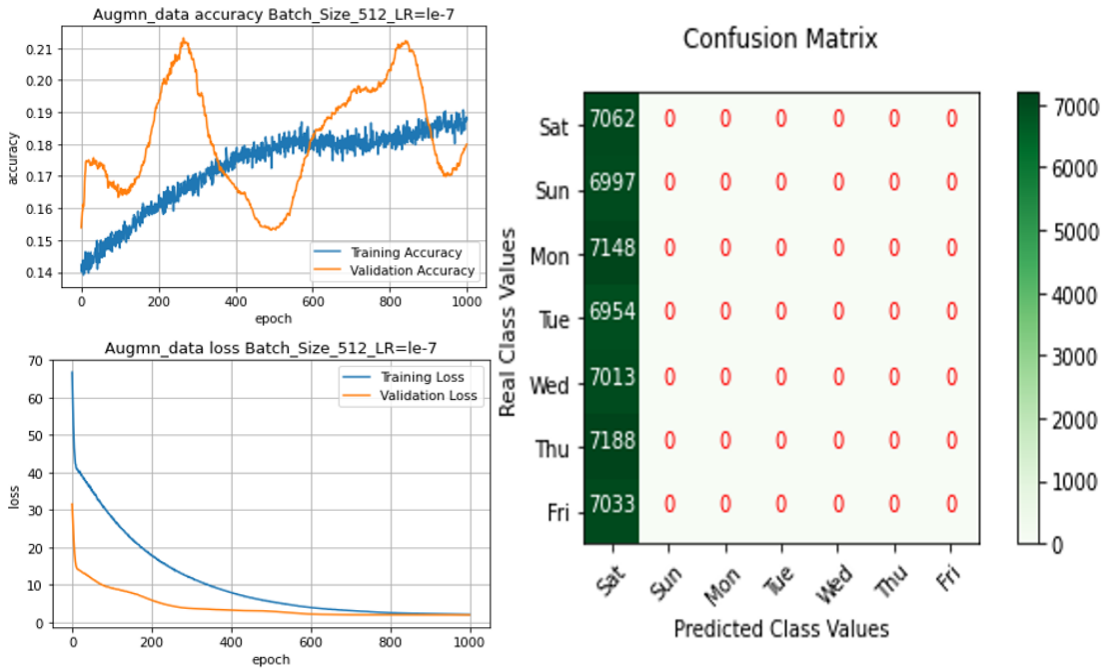


Figure 6.77 LC and CM with LR = 10^{-7} and BS = 512 using augmented AHWD

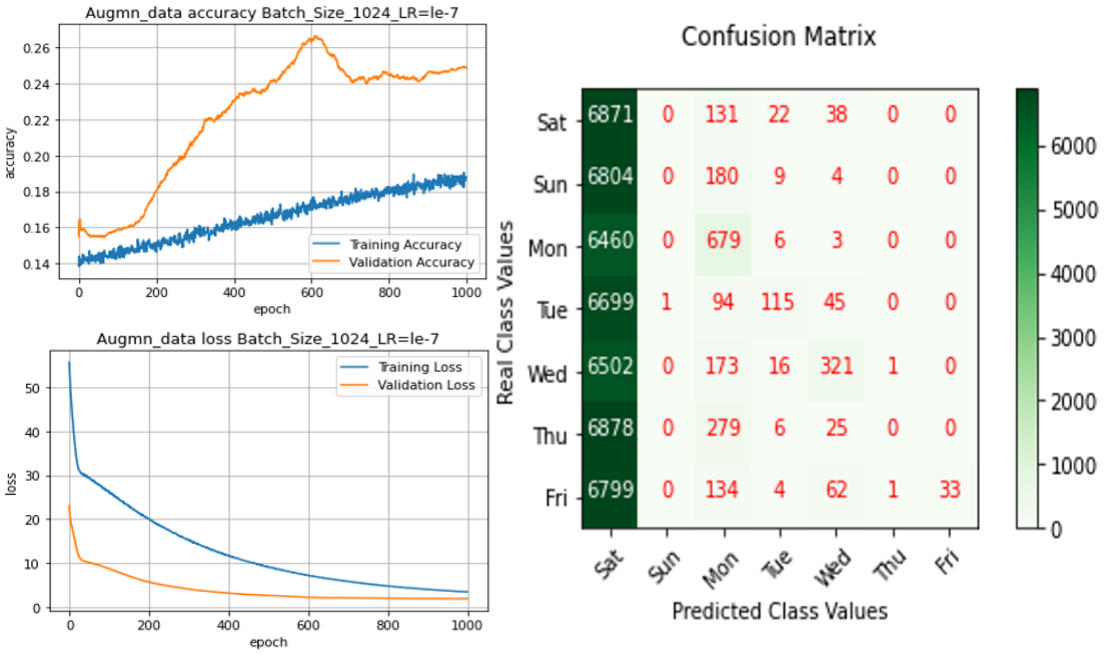


Figure 6.78 LC and CM with LR = 10^{-7} BS = 1024 using augmented AHWD

By looking at Figure 6.72 to Figure 6.78, the conclusion that there is instability in the proposed model when using learning rate = 10^{-7}. The learning rate here is very small which would make the gradient moves uneven and the convergence between the training accuracy curve and validation accuracy curve improbable. Moreover, as the batch size increases:

- Both the training and validation loss curve diverge, and this divergence is clear in Figure 6.Figure 6.78.
- The number of confusion matrix errors increases and reaches to the No diagonal state, and the classification accuracy rate will decrease.

The learning rate hyperparameter with value 10^{-7}, with batch size hyperparameters (16, 32, 64, 128, 256, 512, 1024) when applying augmented AHWD are not suitable, and the system is unstable.

Adam optimizer is used to evaluate the performance when applying augmented AHWD using the proposed model. For the loss function, the categorical cross-entropy loss is used. After training the DCNN model to calculate the probability of each image over the classes, the model is evaluated by using the testing dataset. On augmented AHWD, different batch sizes equal to 16, 32, 64, 128, 256, 512, and 1024 were tested. Also, learning rates of 10^{-3}, $10^{-4}, 10^{-5}, 10^{-6}$, and 10^{-7} were evaluated with epochs 1000. With each learning rate, the model was trained with 7 different batch sizes. So, five tables of results were produced, and each table consists of one learning rate and 7 batch sizes with results (accuracy rate, loss rate, and confusion matrix errors) where the loss rate here is considered as the GER. The results between the five tables are evaluated based on the following criteria:

- Learning rate values (10^{-3} to 10^{-7}) and batch sizes.
- Low loss error rate (GER) has the highest priority.
- High accuracy rate.
- Confusion matrix errors.
- Model response speed by looking at the learning curve graph and determine in which epoch number the accuracy curve starts rising and in which epoch number does the loss curve start coming down.

After applying these criteria, a table summarizing the conditions to obtain the best results is compiled for each dataset. The result is shown in Table 6.18 for augmented AHWD.

Table 6.18 Best of the five tables of augmented AHWD dataset

Learning rate	Batch size	Epoch	Iteration		Acc rate	Loss rate GER	Confusion Matrix Errors
			Train	Test			
10^{-3}	1024	1000	39	24	0.9984	0.1148	23
10^{-4}	1024	1000	39	24	0.9990	0.0074	22
10^{-5}	128	1000	309	191	0.9986	0.0070	22
10^{-6}	64	1000	618	381	0.9966	0.0129	46
10^{-7}	16	1000	2470	1521	0.9672	0.1240	1868

By applying the above criteria in Table 6.18, it is concluded that the best accuracy rates are 99.90% and 99.86%, with error rates GER = 0.0074 and 0.0070, with confusion matrix errors 22 and 22 using batch sizes 1024 and 128 using learning rates 10^{-4} and 10^{-5}respectively. The classification accuracy rate = 99.90% with lower error rate = 0.0074 were chosen as the best hyperparameters for the proposed model for the following reasons:

- The learning rate (10^{-4}) is appropriate which makes the model train at a reasonable speed and allows the gradient descent to produce a smooth output.
- 1000 epochs with 39 training iterations and 24 testing iterations for each epoch and batch size is = 1024. These conditions allow relatively less processing time and reliable computing in terms of any overfitting and guide the proposed model to be robust.
- By observing Table 6.18, the loss rate 0.0074 with learning rate 10^{-4} is chosen even though there is less value of loss rate in the table which is 0.0070 with learning rate 10^{-5}. This choice was made because 99.90% is the highest accuracy rate and has the lowest processing time comparing to the one using learning rate 10^{-5}.
- The model response speed has not been considered because both training sessions with learning rate 10^{-4} with batch size 1024 and 10^{-4} with batch size 128 has

almost the same model response speed in accuracy curve and in loss curve as shown in Figure 6.79and Figure 6.80.

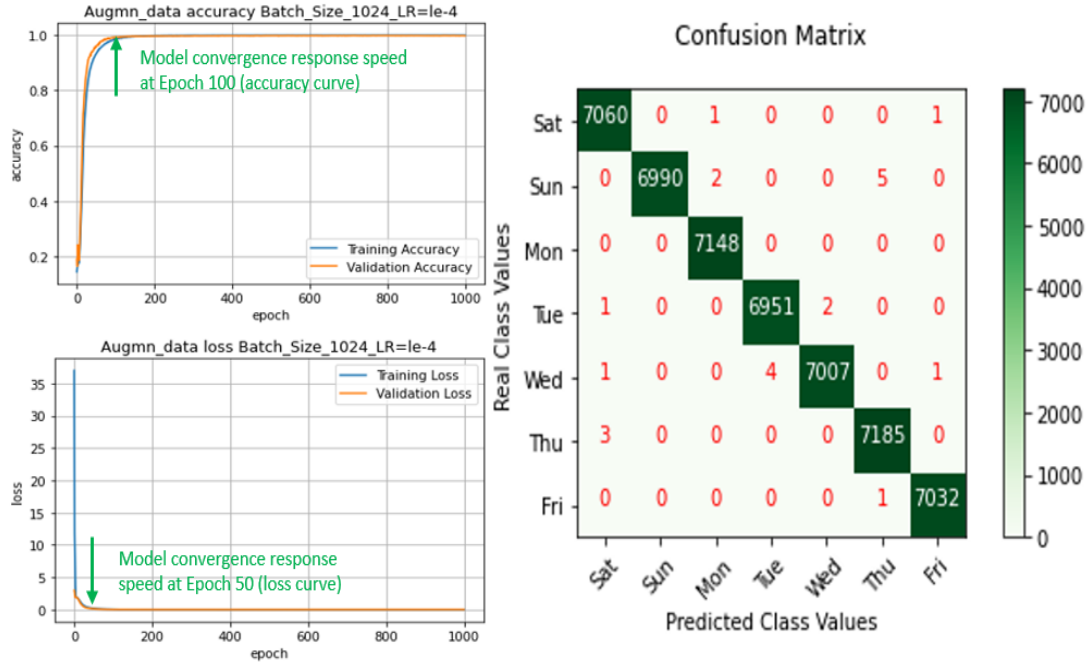


Figure 6.79 LC and CM with LR = 10^{-4} and BS = 1024 using augmented AHWD

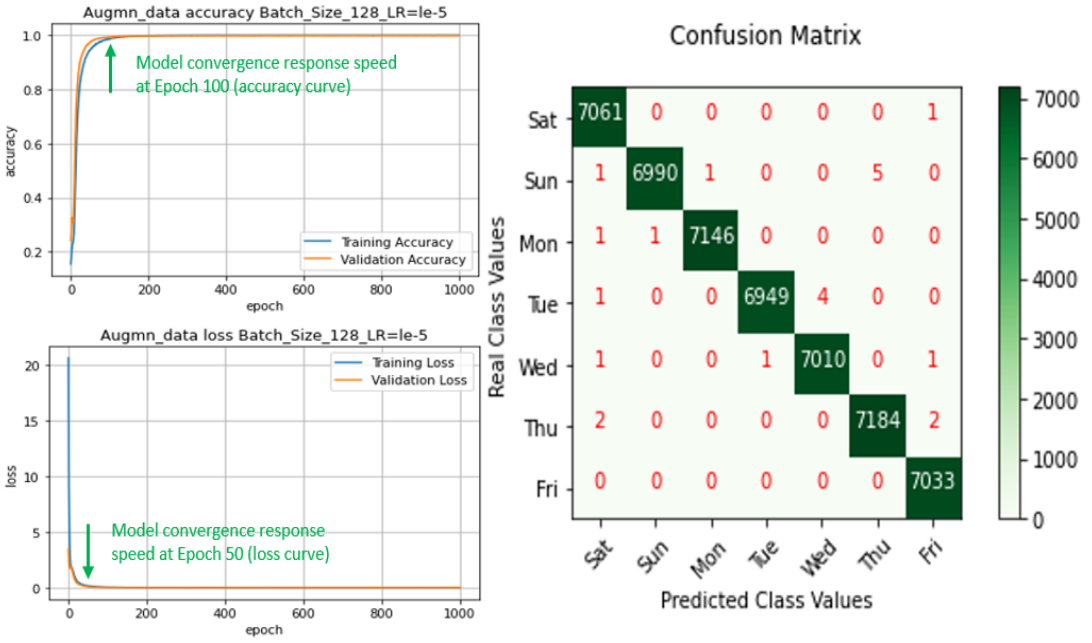


Figure 6.80 LC and CM with LR = 10^{-5} and BS = 128 using augmented AHWD

6.4 Comparison with Neural Networks-based Systems

Table 6.19 shows a comparison between our achieved results based on Neural Network using IFN/ENIT dataset, AHWD, and augmented AHWD and other system's results.

Table 6.19 Comparative results of Neural Networks -based systems.

Classifier	Feature extraction	Datasets	Evaluation metric		References
			Accuracy rate	Error rate	
CNN	CNN	IFN/ENIT (abcd-e) IFN/ENIT (abcde-f) IFN/ENIT (abcde-s) IFN/ENIT (abc-d)	97.07% 96.76% 94.09% 99.29%	2.93 WER 3.24 WER 5.91 WER 0.71 WER	(Poznanski et al., 2016) [74]
SVM	CNN	IFN/ENIT		7.05 CER	(Elleuch et al., 2016) [75]
CNN	CNN	IFN/ENIT		8.5 CER	(Almodfer et al., 2017) [76]
CNN CNN	CNN(AlexNet+ReLU) CNN(AlexNet+TanH)	IFN/ENIT IFN/ENIT	92.13% 92.55%		(Almodfer et al., 2018) [77]
HMM HMM	CNN CNN	IFN/ENIT (abcd-e) IFN/ENIT (abc-d)	89.23% 88.95%	- -	(Amrouch et al., 2018) [78]
AlexNet DBN	CNN DBN	IFN/ENIT IFN/ENIT (abcd-e)	95.6% 94.99%	- 6.5 CER	(Ghanim et al., 2020) [7]
CNN	**CNN**	**IFN/ENIT**	**99.87%**	**0.0181 GER**	**Proposed model.**
CNN	**CNN**	**AHWD**	**99.76%**	**0.0230 GER**	**Proposed model.**
CNN	**CNN**	**Augmented AHWD**	**99.90%**	**0.0074 GER**	**Proposed model.**

The CNN was applied as feature extraction and as classifier on IFN/ENIT dataset using some sets as training and one as testing as explained in [74]. Specifically, when using sets, a, b, c, and d for training and set e for testing they achieved 97.07% classification accuracy rate and 2.93 WER. When using sets, a, b, and c for training and set d for testing they achieved a 99.29% classification accuracy rate and 0.71 WER. In another study described in [77] , the CNN was used as feature extraction and as a classifier on IFN/ENIT dataset using sets, a, b, c, and d for training and set e for testing they achieved 92.13% classification accuracy rate using ReLU activation function, and 92.55% classification accuracy rate using TanH activation function.

According to [78] CNN was applied as a feature extraction and HMM as a classifier on IFN/ENIT dataset and achieved 89.23% classification accuracy rate when using sets, a, b, c, and d as training sets and set e as testing set. They achieved 88.95% classification accuracy rate when using sets, a, b, and c as training sets and set d as testing set.

Ghanim et al., [7] achieved 95.60% classification accuracy rate when they used CNN as a feature extraction and as a classifier using AlexNet on IFN/ENIT.

For the proposed model, the CNN was applied as a feature extraction and as a classifier on AHWD, an accuracy rate of 99.76% and a Generalization Error Rate of 0.0230 were achieved. When CNN was applied as a feature extraction technique and as a classifier on IFN/ENIT dataset, an accuracy rate of 99.87% and Generalization Error Rate of 0.0181 were achieved. When CNN was applied as a feature extraction technique and as a classifier on Augmented AHWD, an accuracy rate of 99.90% and Generalization Error Rate of 0.0074 were achieved.

In comparison to other systems, our proposed system achieved the highest classification accuracy rate with a very low GER on IFN/ENIT dataset, on AHWD, and on augmented AHWD.

CHAPTER 7 CONCLUSIONS AND FUTURE WORK

This chapter summarizes the contributions of the thesis and discusses issues related to the developed techniques. Limitations in the proposed methods and the used data are also discussed. Finally, future research directions relevant to analysis and recognition of Arabic Handwritten Words (AHW) are discussed.

7.1 Summary of Thesis Contributions

So far, research on AHW recognition and analysis has focused on the extraction of features from text lines and from image documents. Very few researchers have investigated deep learning for AHW documents. Moreover, many recent works focus on word isolation and extraction of global and/or local features of the word. The Fourier descriptor, the hue moment, histograms, zarniac moments and other structural features are examples of features. There is existing literature using features with neural classifier such as Support Vector Machine (SVM), K- Nearest Neighbor (KNN), Radial Basis Function (RBF) and Multilayer perception.

AHW identification remains a challenging application. It is carried out as a pattern recognition problem to allocate and identify images of handwritten samples/patterns to one class. Therefore, the process of image identification can be defined as an algorithm to assign a handwriting sample to one of the classes. While several AHW identification systems have been developed for various applications including document analysis and image classification, it is still receiving significant interest by the research community, because many issues are still unresolved such as insufficiency of datasets and handwriting material in different languages. The main aim of this work is to develop an accurate handwritten recognition system by investigating new techniques based on deep learning for the classification and analysis of AHW depending on different Convolution Neural Networks (CNN).

Another aim is to create a large set of AHW to support further studies in text conversion. Therefore, the general objectives of this thesis were to plan, analyze, design, build, and test novel classification algorithms and tools to support automatic recognition of AHW. In this work, some automatic AHW recognition approaches using advanced machine learning techniques have been investigated and the obtained results led to the following specific contributions to knowledge:

- This thesis presented analyses, design, building, and testing of learning algorithms of enhanced DCNN structure for classification. It also investigated the use of an end-to-end

open-source platform for machine learning namely Keras and Tensorflow with parallel processing.

- Thorough experimental tests and validation of the algorithm have been carried out using different datasets and the results obtained suggest that the proposed technique yields attractive results when compared to similar algorithms.
- A large data set was created and then augmented with different word variations created for system training and testing.
- The system was also tested on another data set known as IFN/ENIT and the results achieved excellent accuracy.
- A new datasets called the Arabic Handwritten Weekdays Dataset (AHWD) and augmented AHWD have been presented. The proposed model applied on AHWD, IFN/ENIT dataset, and augmented AHWD, produced respectively an accuracy rate of 99.76% with error rate 0.0230, an accuracy rate 99.87% with error rate 0.0181, and an accuracy rate 99.90% with error rate 0.0074. These results are excellent and compare favorably against previous work.

This work can be applied to the datasets where it should be horizontally extended to include more words to cover all the Arabic characters. It would be impossible to include all words in the dataset, that the system could predict outside of its domain by analyzing the word at the character level. The system could also be used for signature classification and fraud detection of signatures.

BIBLIOGRAPHY

[1] G. F. Simons, D. Charles and F. (eds.), "Ethnologue: Languages of the World, Twenty-first edition.," in SIL International Online Version: http://www.ethnoloque.com, Dallas, Texas, 2018.

[2] L. M. Lorigo and V. Govindaraju, "Offline Arabic handwriting recognition: a survey," in IEEE transactions on pattern analysis and machine intelligence. Vol. 28, no. 5, pp.712- 724, 2006.

[3] B. Al-Badr and S. A. Mahmoud, "Survey and bibliography of Arabic optical text recognition," in Signall Processing, vol. 41, pp. 49-77, Amsterdam: Elsevier B.V, 1995.

[4] M. T. Parvez and S. A. Mahmoud, "Offline Arabic handwritten text recognition: survey," in ACM Comput. Survey, vol. 45, no. 2, p. 23, 2013.

[5] R. Lippmann, "An introduction to computing with neural nets," in IEEE Assp magazine, 4(2), 4-22, 1987.

[6] K. Adam, S. Al-Maadeed and A. Bouridane, "based classification of Arabic scripts style in ancient Arabic manuscripts: Preliminary results," in IEEE 1st International Workshop on Arabic Script Analysis and Recognition (ASAR), pp. 95-98, 2017.

[7] T. M. Ghanim, M. I. Khalil and H. M. Abbas, "Comparative study on deep convolution neural networks DCNN-based offline Arabic handwriting recognition," in IEEE Access, vol. 8, pp. 95465–95482, 2020.

[8] R. Alaasam, B. Kurar, M. Kassis and J. El-Sana, "Experiment study on utilizing convolutional neural networks to recognize historical Arabic handwritten text," in 1st International Workshop on Arabic Script Analysis and Recognition (ASAR), pp. 124-128, 2017.

[9] D. Motawa, A. Amin and R. Sabourin, "Segmentation of Arabic cursive script," in In Proceedings of the fourth international conference on document analysis and recognition, Vol. 2, pp. 625–628, 1997.

[10] A. Alsaeedi, H. Al Mutawa, S. Snoussi, S. Natheer, K. Omri and W. Al Subhi, "Arabic words Recognition using CNN and TNN on a Smartphone," in In IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR), 2018, March.

[11] J. Smith and Z. Merali, "Optical character recognition: the technology and its application in information units and libraries," in Boston Spa, Wetherby, West Yorkshire: The British Library, LS23 7BQ,UK, 1985.

[12] F. Alkhateeb, I. A. Doush and A. Albsoul , "Arabic optical character recognition software: A review," in Pattern Recognition and Image Analysis., 27(4), pp.763-776, 2017.

[13] A. Nazif, "A system for the recognition of the printed Arabic characters," in M.Sc. Thesis. Cairo University, Cairo,Eygpt, 1975.

[14] A. Amin, "Handwritten Arabic Character Recognition by the IRAC system," in In 5th international conference on pattern recognition, pp. 729-731, Miami, Florida, USA, 1980.

[15] A. Amin and G. Masini, "Machine recognition of multifont printed Arabic texts," in In Proc. 8th Int. Conf. on Pattern Recognition, pp. 392-295, 1986, October.

[16] A. Nouh , A. Sultan and R. Tolba, "An Approach for Arabic Characters Recognition," in J. Eng. Science, vol. 6, pp. 185-191, 1980.

[17] K. Badi and M. Shimura, "Machine recognition of Arabic cursive scripts," in Trans. Inst. Electron. Commun. Eng, 65(2), pp. 107-114, 1982.

[18] R. M. Bozinovic and S. N. Srihari , "Off-line cursive script word recognition," in IEEE Transactions on pattern analysis and machine intelligence, 11(1), pp. 68-83., 1989.

[19] H. Almuallim and S. Yamaguchi, "A method of recognition of Arabic cursive handwriting," in IEEE transactions on pattern analysis and machine intelligence, (5), pp. 715-722., 1987.

[20] D. McClelland, "OCR: teaching your Mac to read," in Macworld November, pp. 169-178., 1991.

[21] E. M. Welch, "Can you read this? OCR software," in MacUser, Vol. 9, No. 8, pp. 169-178., 1993.

[22] T. S. El-Sheikh and S. G. El-Taweel, "Real-time Arabic handwritten character recognition," in Pattern recognition, 23(12), pp. 1323-1332, 1990.

[23] H. Al-Yousefi and S. S. Upda , "Recognition of Arabic characters.," in IEEE Transactions on Pattern Analysis & Machine Intelligence, (8), pp. 853-857, 1992.

[24] G. Olivier, H. Miled, K. Romeo and Y. Lecourtier, "Segmentation and coding of Arabic handwritten words," in In Proceedings of 13th International Conference on Pattern Recognition (Vol. 3, pp. 264-268). IEEE, 1996, August.

[25] B. Al-Badr and R. M. Haralick , "A segmentation-free approach to text recognition with application to Arabic text," in International Journal on Document Analysis and Recognition 1(3), pp. 147-166., 1998.

[26] B. Al-Badr and R. M. Haralick, "Segmentation-free word recognition with application to Arabic," in In Proceedings of 3rd International Conference on Document Analysis and Recognition (Vol. 1, pp. 355-359). IEEE, 1995, August.

[27] M. Y. Chen, a. Kundu and J. Zhou, "Off-line handwritten word recognition using a hidden Markov model type stochastic network," in IEEE transactions on Pattern analysis and Machine Intelligence, 16(5), pp. 481- 496, 1994.

[28] A. Emam , "Designing a reader machine for the blind," in Ph.D. Thesis. University of Alexandria, Alexandria, Egypt, 1995.

[29] K. Mostafa and A. M. Darwish, "Robust baseline-independent algorithms for segmentation and reconstruction of Arabic handwritten cursive script," in In Document Recognition and Retrieval VI (Vol. 3651, pp. 73-83). International Society for Optics and Photonics., 1999, January.

[30] T. Kanungo, G. A. Marton and O. Bulbul, "OmniPage vs. Sakhr: Paired model evaluation of two Arabic OCR products," in In Document Recognition and Retrieval VI (Vol. 3651, pp. 109-120). International Society for Optics and Photonics, 1999, January.

[31] J. Makhoul, R. Schwartz, C. Lapre and I. Bazzi, "A script-independent methodology for optical character recognition," in Pattern Recognition, 31(9), 1285-1294., 1998.

[32] P. Natarajan, R. Schwartz, M. Decerbo and T. Keller, "Porting the BBN BYBLOS OCR system to new languages," in In Symposium on Document Image Understanding Technologies (pp. 47-52)., 2003, April.

[33] J. Trenkle, A. Gillies, E. Erlandson, S. Schlosser and S. Cavin, "Advances in Arabic text recognition," in In Proc. Symp. Document Image Understanding Technology., 2001, April.

[34] L. Hamami-Mitiche, "Segmentation of an Arab Text Paragraph Printed in Characters.," In Proceedings of the 8th International Conference on Computer Theory and Applications, ICCTA'98, Alexandria, Egypt (pp. 15-17)., 1998, September.

[35] L. Hamami and D. Berkani, "Recognition system for printed multi-font and multi-size Arabic characters," in Arabian Journal for Science and Engineering, 27(1), 57-72., 2002.

[36] M. Pechwitz and V. Maergner, "HMM based approach for handwritten Arabic word recognition using the IFN/ENIT-database," in In Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings. (pp. 890-894). IEEE, 2003, August.

[37] A. Amin, "Recognition of hand-printed characters based on structural description and inductive logic programming," in Pattern recognition letters, 24(16), 3187-3196, 2003.

[38] I. A. Jannoud, "Automatic Arabic handwritten text recognition system," in American Journal of Applied Sciences, 4(11), 857-864., ISSN 1546-9239, 2007.

[39] H. Althobaiti and C. Lu, "A survey on Arabic Optical Character Recognition and an isolated handwritten Arabic Character Recognition algorithm using encoded freeman chain code," in In 2017 51st Annual Conference on Information Sciences and System, 2017, March.

[40] T. M. Rath, T. M. and R. Manmatha, "Word spotting for historical documents," in International Journal of Document Analysis and Recognition (IJDAR), 9(2-4), pp. 139-152., 2007.

[41] J. L. Rothfeder, S. Feng and T. M. Rath, "Using corner feature correspondences to rank word images by similarity," in In 2003 Conference on Computer Vision and Pattern Recognition Workshop (Vol. 3, pp. 30-30). IEEE., 2003, June.

[42] A. P. Giotis, G. Sfikas, B. Gatos and C. Nikou, "A survey of document image word spotting techniques," in Pattern Recognition, 68, pp. 310-332., 2017.

[43] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurons in the cat's striate cortex," in The Journal of physiology, 148(3), 574, 1959.

[44] R. A. Kirsch, "Talks about The Variable Shaped Pixel," in 2011 talk. https://vimeo.com/22179638, at ANKA Gallery in Portland, OR on Apr. 7th 2011.

[45] L. G. Roberts, "Machine perception of three-dimensional solids (Doctoral dissertation," in Massachusetts Institute of Technology, Dept. of Electrical Engineering), 1963.

[46] T. Wang, D. J. Wu, A. Coates and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in In Proceedings of the 21st international conference on pattern recognition (ICPR2012) (pp. 3304-3308). IEEE., 2012, November.

[47] R. F. &. ,. R. F. Moghaddam and M. Cheriet, "Application of multi-level classifiers and clustering for automatic word spotting in historical document images," in In 2009 10th International Conference on Document Analysis and Recognition (pp. 511-515). IEEE., 2009, July.

[48] M. Eltay, A. Zidouri and I. Ahmad, "Exploring Deep Learning Approaches to Recognize Handwritten Arabic Texts," in IEEE Access, 8, pp. 89882-89898, 2020.

[49] I. Ahmad and G. A. Fink, "Handwritten Arabic text recognition using multi-stage sub-core-shape HMMs," in International Journal on Document Analysis and Recognition (IJDAR), 22(3), PP. 329-349, 2019.

[50] T. Ahonen, A. Hadid and M. Pietikainen, "Face description with local binary patterns: Application to face recognition.," in IEEE transactions on pattern analysis and machine intelligence, 28(12), pp. 2037-2041., 2006.

[51] H. Zhang, A. Jolfaei and M. Alazab, "A face emotion recognition method using convolutional neural network and image edge computing," in IEEE Access, 7, pp. 159081-159089., 2019.

[52] S. Wshah, V. Govindaraju, Y. Cheng and H. Li, "A novel lexicon reduction method for Arabic handwriting recognition," in In 2010 20th International Conference on Pattern Recognition (pp. 2865-2868). IEEE., 2010, August.

[53] A. Krizhevsky , I. Sutskever and G. E. Hinton , "ImageNet classification with deep convolutional neural networks," in Communications of the ACM, 60(6),pp, 84-90, 2017.

[54] A. Mars and G. Antoniadis, "Arabic online handwriting recognition using neural network," in International Journal of Artificial Intelligence and Applications (IJAIA), 7(5), 2016.

[55] A. A. Alani, "Arabic handwritten digit recognition based on restricted Boltzmann machine and convolutional neural networks," in Information, 8(4), 142, 2017.

[56] A. Ashiquzzaman and A. K. Tushar, "Handwritten Arabic numeral recognition using deep learning neural networks," in In 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR) (pp. 1-4). IEEE, 2017, February.

[57] R. Almodfer, S. Xiong, M. Mudhsh and P. Duan, "Enhancing AlexNet for arabic handwritten words recognition using incremental dropout," in In 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 663-669), 2017, November.

[58] C. Neche, A. Belaid and A. Kacem-Echi, "Arabic handwritten documents segmentation into text-lines and words using deep learning," in In 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW) (Vol. 6, pp. 19-24, 2019, September.

[59] A. A. Almisreb, S. Turaev, M. A. Saleh and S. A. M. Al Junid, "Arabic Handwriting Classification using Deep Transfer Learning Techniques," in Pertanika Journal of Science & Technology, 30(1), 2022.

[60] G. J. Orchard, A. ayawant, G. K. CoheN and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," in Frontiers in neuroscience, 9, 437., 2015.

[61] G. Cohen , S. Afshar, J. Tapson and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in In 2017 international joint conference on neural networks (IJCNN) (pp. 2921-2926). IEEE, 2017, May.

[62] J. Bergstra and Y. Bengio, Random search for hyper-parameter optimization, : Journal of machine learning research, 13(2), 2012.

[63] C. C. Aggarwal, Aggarwal and Lagerstrom-Fife., Linear algebra and optimization for machine learning, Springer International Publishing, 2020.

[64] J. . Brownlee, Better deep learning: train faster, reduce overfitting, and make better predictions, Machine Learning Mastery, 2018.

[65] R. B. Karim. [Online]. Available: https://www.kdnuggets.com/2019/06/gradient-descent-algorithms-cheat-sheet.html.

[66] C. C. Aggarwal, Neural networks and deep learning, Springer; 10, 978-3, 2018.

[67] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems, O'Reilly Media, Inc, 2019.

[68] X. Hu, L. Chu, J. Pei, W. Liu and J. Bian, " Model complexity of deep learning: A survey," in Knowledge and Information Systems (2021) 63:2585–2619, https://link.springer.com/article/10.1007/s10115-021-01605-0, August 2021.

[69] X. Hu , W. Liu, J. Bian and J. Pei, "Measuring model complexity of neural networks with curve activation function," in In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery, 2020.

[70] W. Ballard, Hands-on deep learning for images with TensorFlow: build intelligent computer vision applications using TensorFlow and Keras, : Packt Publishing Ltd., 2018.

[71] Y. Wang, Y. Li, Y. Song and X. Rong, "The influence of the activation function in a convolution neural network model of facial expression recognition," in Applied Sciences, 10(5), 1897, 2020.

[72] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in In International conference on machine learning (pp. 448-456). PMLR., 2015, June.

[73] A. Melton, J. Ko and V. E. Guzik, "DATAQUEST," 3 January 2018. [Online]. Available: https://www.dataquest.io/blog/learning-curves-machine-learning/. [Accessed July 2022].

[74] A. Poznanski and L. Wolf, "Cnn-n-gram for handwriting word recognition," in in InProceedings of the IEEE conference on computer vision and pattern recognition (pp. 2305-2314), 2016.

[75] M. Elleuch, R. Maalej and M. Kherallah, "A new design based-SVM of the CNN classifier architecture with dropout for offline Arabic handwritten recognition," in in Proceding Computer Science, 80, 1712-1723, 2016.

[76] R. Almodfer, S. Xiong, M. Mudhsh and DuanP, "Multi-column deep neural network for offline Arabic handwriting recognition," in in In International Conference on Artificial Neural Networks (pp. 260-267). Springer, Cham, September, 2017.

[77] R. Almodfer, S. Xiong, M. Mudhsh and P. Duan, "Enhancing alexnet for Arabic handwritten words recognition using incrementaldropout," in in Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI, Jun. 2018, vol. 2017-November.

[78] M. Amrouch, M. Rabi and Y. Es-Saady, "Convolutional feature learning and CNN based HMM for Arabic handwriting recognition," in in In International conference on image and signal processing (pp. 265-274). Springer, Cham, July, 2018.

[79] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze and H. Amiri, "IFN/ENIT-database of handwritten Arabic words," in In Proc. of CIFED (Vol. 2, pp. 127-136). Citeseer., 2002, October.

[80] T. M. Ghanim, M. I. Khalil and H. M. Abbas, "Comparative study on deep convolution neural networks DCNN-based offline Arabic handwriting recognition," in IEEE Access, vol. 8, pp. 95465–95482, 2020.