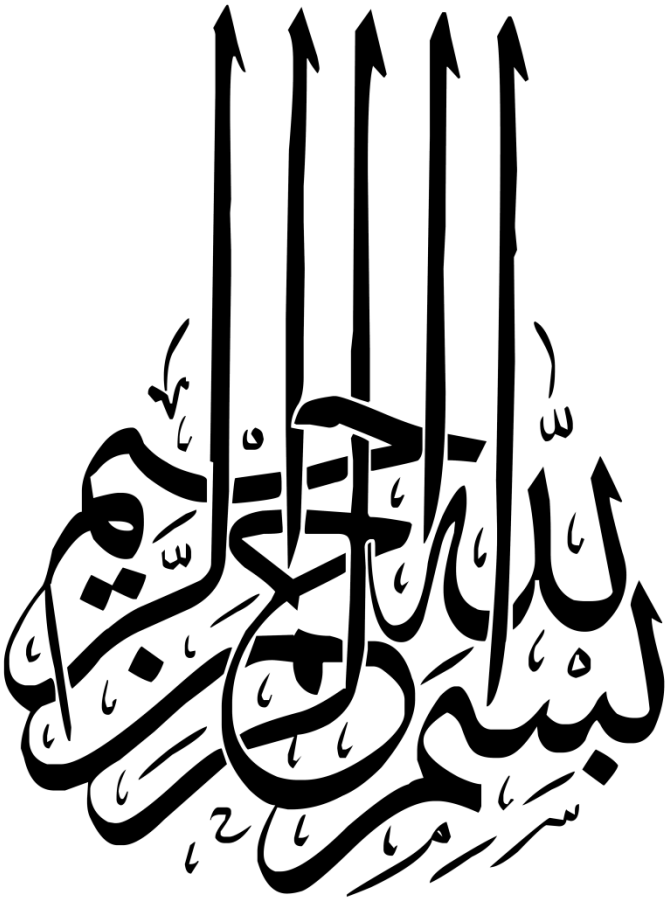
Arabic Manuscript Images Retrieval Using Deep Learning

By
Manal Mahmoud Ali Khayyat

A Thesis Submitted in Partial Fulfillment of the Requirements for the PhD Degree in Computer Science

Supervised By
Dr. Lamiaa Elrefaei

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY
JEDDAH – SAUDI ARABIA
Dhu'l Qi'dah 1441 H – July 2020 G


بسم الله الرحمن الرحيم

Arabic Manuscript Images Retrieval Using Deep Learning

By
Manal Mahmoud Ali Khayyat

This thesis has been approved and accepted in partial fulfillment of the requirements for the degree of PhD of Computer Science

EXAMINATION COMMITTEE

	Name	Rank	Field	Signature
Advisor and Rapporteur	Lamiaa Abdallah Elrefaei	Associate Professor	Computer Science	
Internal Examiner	Fathy Elbouraey Eassa	Professor	Computer Science	
External Examiner	Salah Ramadan Abdel-Mageid	Professor	Computer Engineering	

KING ABDULAZIZ UNIVERSITY
Dhu'l Qi'dah 1441 H – July 2020 G

DEDICATION

This work is dedicated to my Parents, Husband, and my Daughters

ACKNOWLEDGMENT

Upon completion of this dissertation, I would like to express all my gratefulness to Allah (GOD) for providing me the blessings to complete this work. I ask him to make it a useful science.

I would like to express my thanks and gratitude to my advisor **Dr. Lamiaa A. Elrefaei** for initiating this research topic and for her supervision, guidance, support and help in every step to complete this work.

My special gratitude goes to **my Parents and my Husband** for being patient, supportive and source of motivation and encouragement for my studies.

Manal Khayyat

Arabic Manuscript Images Retrieval Using Deep Learning

ABSTRACT

The ancient Arabic manuscripts are valuable pieces of historical information that reflect the education, culture, society, and tradition during specific time periods. Therefore, due to their significance and substantial role in enriching the valuable historical information, this study aims to collect the ancient Arabic manuscripts in a dataset. Then, classify its images to be able to retrieve the ranked top similar images to a user query image accurately and instantaneously. The retrieval system can be according to different search criteria. We satisfied the retrieval according to the manuscript, author, and the calligraphy.

The automatic extraction and classification according to the most distinguishable features, is a crucial step to detect the similarities among images successfully. Considering that, the historical Arabic manuscripts are text-based images. Thus, it is important to extract the text from the images and to retrieve them according to their textual features. This step is performed through developing an optimized bidirectional LSTM deep learning model including attention and batch normalization layers. Then,

the similarities among the textual contents measured using three different distance metrics named: Manhattan, Euclidean, and Cosine.

The manuscripts' images are also not purely textual. Instead, they include handwritten signatures, drawings, figures, tables, side-notes, …etc. Thereby, it is necessary to consider the non-textual parts within the images and to retrieve them according to their visual features. To accomplish this, we transferred learning from four pre-trained convolutional neural networks named: MobileNetV1, DenseNet201, ResNet50, and VGG19. The Siamese deep learning model along with the three distance metrics tested for measuring the similarities among the images and retrieve them.

The most accurate visual and textual deep learning models were fused at three different fusion-levels named: decision-level, features-level, and score-level. The score-level fusion model resulted in a considerable improvement of each model used individually. Moreover, we experimented an alternative solution for the images' retrieval utilizing an enhanced deep reinforcement learning technique. The model uses the features-level fusion of both the VGG19 and the BiLSTM deep learning models. Then, it hashes the fused features to reduce the dimensionalities among them.

Thus, the study introduces a novel approach that utilizes a fusion model of both the visual and the textual models to classify and retrieve the historical Arabic manuscripts' images successfully.

PUBLICATIONS

I. M. Khayyat, and L. Elrefaei, "Towards Author Recognition of Ancient Arabic Manuscripts Using Deep Learning: A Transfer Learning Approach," Accepted Manuscript for publication in the *International Journal of Computing and Digital Systems, (IJCDS),* vol. 9, no. 4, pp. 1-18. 2020. **[Scopus Indexed Journal, SJR2019=0.11]**

II. M. Khayyat, and L. Elrefaei, "A Deep Learning Based Prediction of Arabic Manuscripts Handwriting Style," Accepted Manuscript for publication in the *International Arab Journal of Information Technology, (IAJIT),* vol. 17, no. 5, 2020. **[ISI Indexed Journal, IF2019=0.467]**

III. M. Khayyat, and L. Elrefaei, "Manuscripts Image Retrieval Using Deep Learning Incorporating a Variety of Fusion Levels," *IEEE Access Journal*, vol. 8, pp. 136460 – 136486. 2020. **[ISI Journal, IF2019=3.745]**

AWARDS

I. First place prize in the *11th Scientific Forum for King Abdulaziz University,*2020. (in the Arabic Language section).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS AND TERMINOLOGY

AF	Average F-Score
AI	Artificial Intelligence
AP	Average Precision
AR	Average Recall
AUC	Area Under Curve
BiLSTM	Bidirectional Long Short-Term Memory
BN	Batch Normalization
BoW	Bag of Words
CBIR	Content Based Images Retrieval
CNN	Convolutional Neural Networks
CV2/ OpenCV	Open Source Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DRN	Deep Reinforcement Network
DRQN	Deep Reinforcement Q-learning Network
FBIR	Feature Based Image Retrieval
IR	Image Retrieval

KNN	K-Nearest Neighbor
LSH	Locality Sensitive Hashing
LSTM	Long Short-Term Memory
MAE	Mean Average Error
mAP	Mean Average Precision
MESR	Maximally Stable Extremal Regions
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
OCR	Optical Character Recognition
PCA	Principal Component Analysis
PRC	Precision-Recall Curve
RBIR	Region Based Image Retrieval
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TBIR	Text Based Image Retrieval
VA	Validation Accuracy
2D / 3D	Two Dimensional / Three Dimensional

Chapter I

Introduction

This chapter provides a general background about the image retrieval systems, as well as the stages employed in the retrieval process. It also discusses the deep learning technology and its efficiency in retrieving images. It highlights the image retrieval system of the Arabic manuscripts in particular. In addition, it explains the objective behind this study, the problem statement, the research methodology, the challenges to be addressed, and our contributions to the field.

1.1 Image Retrieval Systems

Due to the tremendously increasing amount of data being used and retrieved, there has been substantial research dedicated to image retrieval methods. Note that there are two main systems for image retrieval, which are the Text-Based Image Retrieval (TBIR) and the Content-Based Image Retrieval (CBIR). The CBIR is also known as the Feature-Based Image Retrieval (FBIR) that is because it mainly depends on the features extracted from the images to retrieve the output set [1].

Comparing the performance of the TBIR with the CBIR system, CBIR is more preferred due to its automatic extractions of low-level features from the images utilizing visual content descriptors. An example of low-level features could be the shape of the image, its color, texture, dimensions, etc. On the other hand, the TBIR

matches the titles of the images manually with the humans' labeling of the images. Using manual text annotation for the retrieval process is becoming tedious and time wastage. Therefore, more research is focusing on explaining the contents of the model using automatic text annotation [2]. In addition, Xia et al. [3] affirm that the automatic image retrieval system utilizing a direct visual view is crucial and that the manual classification usually tends to cause errors and lack to explain the visual contents of the images.

Furthermore, the authors in [1] admit that there is an extended version from CBIR called Region-Based Image Retrieval (RBIR). In this particular method, the image segmented into several regions and then, the features extracted from each region. Afterward, the similarity measure is computed between the query image and the feature vectors of the images in the dataset, which is very costly due to the many regions of the same single image. For this reason, RBIR is rarely used.

There is also another method used for image retrieval based on recognizing characters, and it is called the Optical Character Recognition (OCR). Tyagi [4] states that extracting low-level features from the images is easy to compute and is relatively compared to low-level humans' perceptions. On the other hand, there are other advanced techniques that can be better utilized for image retrieval, such as deep learning. That is because deep learning can automatically capture and extract higher-level features from the images, which is relatively compared to real human's perceptions.

Even though a number of techniques have been proposed for image retrieval, there still a semantic gap presented in extracting low-level features from the images as captured by electronic devices and between extracting high-level semantic concepts as viewed

by real-humans' brains. Therefore, there is a significant need to explore the ability of deep learning technology to retrieve images successfully.

1.2 Image Retrieval Stages and Utilized Methods in Each Stage

Using any image retrieval technique, the same stages remain similar, as illustrated in figure 1.1.

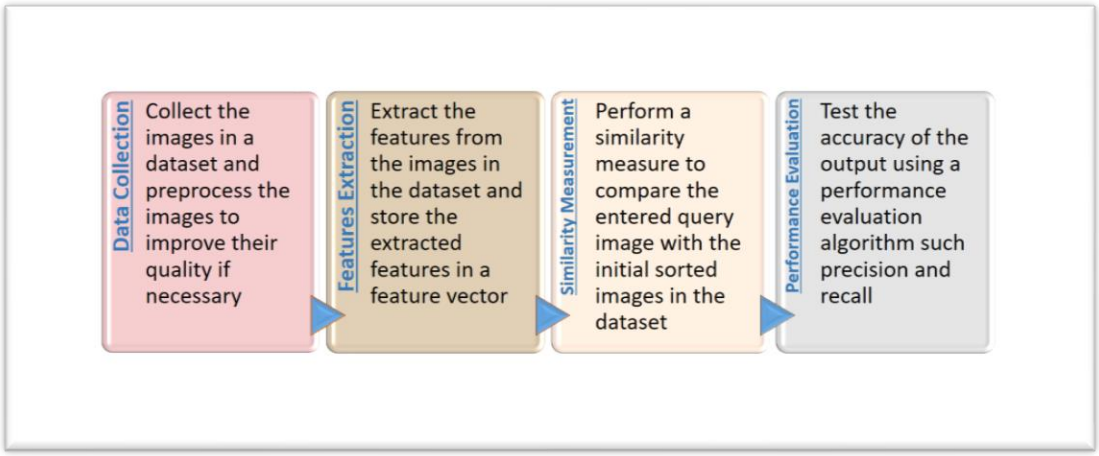


Figure 1.1: Image retrieval stages

First stage: includes the collection of the images dataset and preprocess the images to improve their quality if necessary, through resizing them, reducing the dimensions among the input data, transform images colors into greyscale, etc.

Second stage: includes extracting the features from the images in the dataset and store the extracted features in a feature vector as a knowledge-based database.

The authors in [1] claim that for the feature extraction, we might use color moments, ranklet transformation, moment invariants method, sketch-based local binary pattern (SBLBP), etc.

Third stage: includes performing a similarity measure to compare the entered query image with the initial sorted images in the dataset.

For the similarity measurement, we might compute the distance using the Chi-squared test (X^2 statistics), Manhattan distance (City block distance), Euclidean distance, Canberra distance, cosine measure, and histogram intersection [1]. The researchers admit that the Euclidean method to compute the distance metric for the similarity measurement is the best comparing it to other methods such as the Chi-square test, city block distance, and Canberra distance. That is because the Euclidean distance method recorded the accurate precision and recall among the others.

On the other hand, the author in [4] believes that we cannot generalize that the Euclidean method holds true for all types of applications, as it may not give the most accurate results in some images retrieval programs. Hence, he recommends considering the hierarchical indexing scheme based on the Self Organization Map (SOM) as an alternative solution.

Furthermore, the Bag of visual Words (BoW) approach is employed to index and search for similar images. Note that the BoW approach based on the concept of collecting extracted local features from the images dataset such as the SIFT features and then mapping them to visual words. Afterward, compare the allocated visual words with specified visual vocabularies. Beecks et al., [5] admit that even though the Bow approach reaches high retrieval performance, it is restricted due to the use of static visual vocabularies.

In addition, the authors in [6] performed a comparative performance study for several CBIR techniques and stated that there are eleven different distance methods that can be utilized for image retrieval purposes. The methods are linear, linear-2, standardized linear-2, normalized linear-2, city-block, minkowski, chebyshev, cosine similarity, pearson' s correlation, spearman' s rank correlation coefficient, and relative standard deviation. Leveraging a database of 9000 images, the researchers evaluated the

performance of the eleven different methods in retrieving images by calculating three statistical measures, which are the average images retrieval time, precision, and recall. Moreover, they had also evaluated the eleven CBIR methods on non-database images. They concluded that for the database images, the linear, city-block, and the relative standard deviation methods performed the best (i.e., reached the highest precision and recall but, with lower retrieval time). On the other hand, for the non-database images, the linear-2, minkowski, and the normalized linear-2 methods performed better.

Fourth stage: includes testing the accuracy of the output using a performance evaluation algorithm. For the performance evaluation, most of the studies compute the precision, recall, and F-score algorithms. Note that we can use only a single measure or combine multiple measures to obtain better output results. The precision is also known as the Positive Predictive Value (PPV). It returns the percentage of the correctly retrieved images among all the retrieved images where they were relevant or not. However, the recall, which is also known as the sensitivity value, returns the percentage of the correctly retrieved images among all the relevant images stored in the dataset [1]. In addition, the F-score measures the weighted harmonic mean of precision and recall. An alternative performance evaluation measure is the Normalized Discounted Cumulative Gain (NDCG), which calculates the effectiveness of applying multiple measuring methods to obtain better results [5].

1.3 From Artificial Intelligence to Deep Learning

Deep learning is a subfield of machine learning, which is itself a subfield of artificial intelligence. The concept of deep learning technology is based on simulating real human's perceptions in visualizing images. Hence, it aims to extract higher-level features such as activities or objects presented within images automatically, which is

more complex and requires more time. In contrast, machine learning requires humans' interventions to perform the features extraction task, as illustrated in figure 1.2.

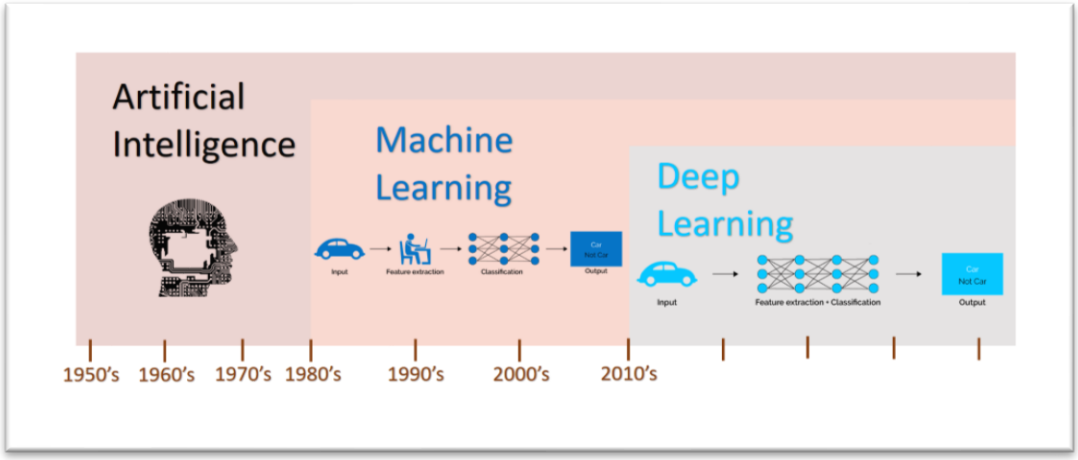


Figure 1.2: From AI to DL [7]

There are several types of machine learning techniques, including:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

The supervised learning is used in the regression and classification tasks. Regression problems include predictions and forecasting. In comparison, classification problems include classifying images, diagnostics, and detection. On the other hand, reinforcement learning is used in developing games, robots, learning tasks, and real-time decisions.

Al-Ayyoub, et al. [8] admit that deep learning mimics humans' brains through leveraging multiple complex algorithms to discover the right model for extracting the distinguishing features. It bases on an architecture consisting of one input layer, one

output layer, and many hidden layers in between them responsible for performing the computations and representations. However, deep neural networks proved their efficiency in resembling humans' brains by developing many non-linear transformations that create hierarchal abstraction layers that can intelligently learn complicated features and can retrieve accurate results [4].

Zhou and Jia [2] believe that utilizing deep neural networks as a learning method accomplished substantial success compared with other methods that depend on classical computational methods. That is because traditional approaches require domain experts, time-consuming, error-prone, and scalable to new problems. On the other hand, the deep learning approach is a computing model that learns from data, easy to extend, and is able to speed up with GPUs [7]. New developments in the field of deep learning showed innovative solutions in Natural Language Processing (NLP), speech recognition, and computer vision [8]. Note that deep neural networks can be trained on datasets and become able to select unique features using either supervised learning or unsupervised learning. Knowing that supervised learning is simpler than unsupervised learning because it initially trains the dataset using a group of training samples. In contrast, unsupervised learning lets the machine train itself by providing it with the information without any training samples, and based on the similarities or differences in the given information; the machine will sort and group the new dataset. The deep learning technique includes stacked convolutional neural layers that are able to learn and predict on their own after seeing a massive number of data. Thus, the deep learning learns from training then, expands the gained knowledge on new unseen data. In comparison, reinforcement learning is a cutting-edge technique in the machine learning world. It communicates agent with a continuous dynamic environment looking for optimal behavior. Thus, reinforcement learning learns from trial-and-error

actions taken by the agent [9]. Figure 1.3 illustrates the difference between the DL and the RL.

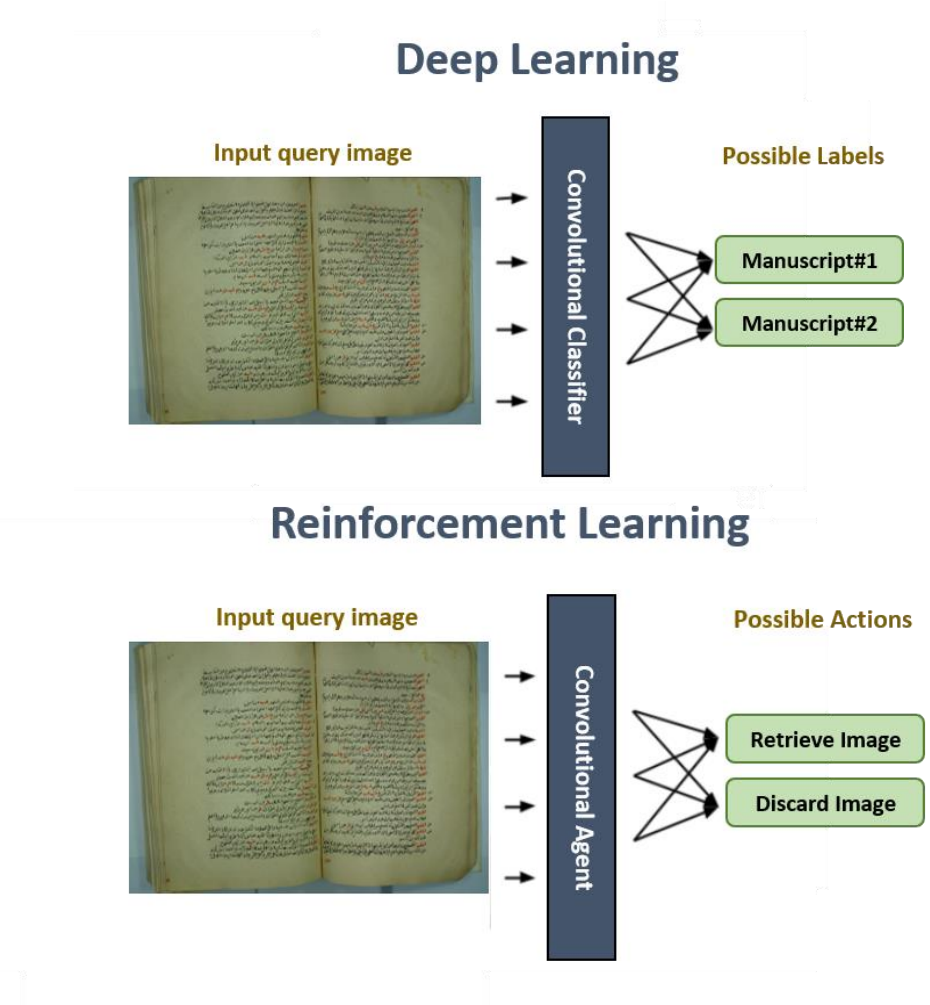


Figure 1.3: Difference btw DL and RL

There are five main tasks of deep learning as following:

- Detection
- Classification
- Segmentation
- Prediction
- Recommendation

The detection task includes detecting specific objects/regions within images. Ren et al. [10] proposed a fusion model of Region Proposal Network (RPN) with Fast R-CNN, which are deep convolutional networks that can detect regions of images accurately. While the classification task includes categorizing inputs into specific groups. For instance, we might need to classify research papers into a particular genre based on the used terminologies as accomplished in [11]. In addition, Chan et al. [12] proposed a deep learning image classification model based on texture features. The architecture utilizes cascaded Principal Component Analysis (PCA) to classify input images as either face, handwritten text, or digits. The authors concluded that leveraging the PCANet-2 deep neural network recorded between 83.74% and 86.66% accuracy of classifying texture images.

The segmentation task refers to dividing the images into separate rows or columns according to the desired goal. It is usually needed with text images to be able to read individual words and characters. For example, the author in [13] segmented historical Arabic manuscripts into lines and then segmented the lines further into connected components. The prediction task is used for predicting future phenomena such as predicting the effect of air pollutions on citizens' health. Hence, in all prediction cases, we should test the correlation between the two parameters. Lv et al., [14] predicted the traffic flow leveraging deep learning. They have examined the relationship between the spatial and temporal parameters to predict the traffic time in highways. The authors evaluated the accuracy with 15, 30, 45, and 60 minutes' traffic flow prediction in freeways with more than 450 vehicles and were able to record accuracy of more than 90%.

The recommendation task considers items ratings done by users as the origin of information to learn and eventually recommend specific ideas based on the desired goal.

Wang et al. [15] suggested a Collaborative Deep Learning (CDL) recommendation system. It is called collaborative because users collaborate by entering their ratings or feedback into the system to improve the learning process. The authors tested their proposed model using three datasets, two of them from "CiteULike," and the third dataset is from "Netflix". They evaluated the model in recommending movies to users based on other ratings. The authors concluded that their model reached a precision of 70%.

1.4 Various Types of Deep Neural Networks

There are many types of deep neural networks, such as the deep reinforcement networks, the convolutional neural networks, the recurrent neural networks, the long-short-term memory, etc.

1.4.1 Deep Reinforcement Networks (DRN)

The primary deep reinforcement network consists of an agent and an environment. The interaction begins from the environment; it sends the state to the agent. Then, the agent takes action based on the assigned state. Afterward, the environment rewards the agent according to the chosen action, as illustrated in figure 1.4.

The agent can learn from its historic feedback of the performed actions while trying to fine-tune the learning hyperparameters. Therefore, the final made decision by a reinforcement network is taken after enough iterations and through reinforcing the machine to learn from the successfully taken actions [16].

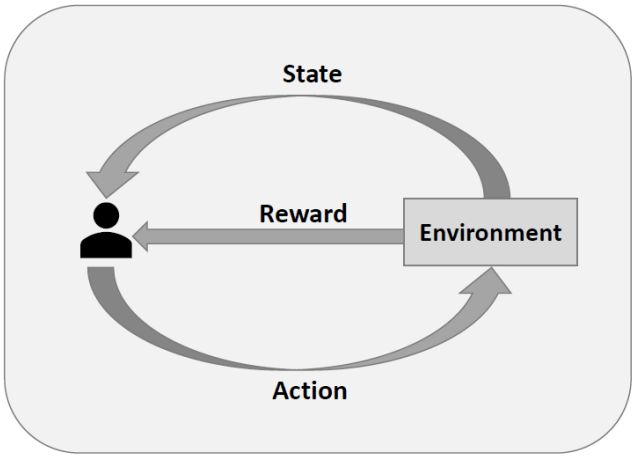


Figure 1.4: The basic architecture of the reinforcement network

The reinforcement network is entirely autonomous in its learning, and it requires fewer data to learn than the deep learning network. The reinforcement network has been widely adopted in electronic games such as the Atari, chess, and in developing robots [17]. It can be on-policy or off-policy. The main difference between these two types of reinforcement learning networks is that the on-policy reinforcement learning network learns the value of the policy being taken by the agent. In contrast, the off-policy reinforcement learning network doesn't consider the agents ' taken actions while learning the policy's optimal value. An example of on-policy learning is the R-Learning, while the Q-Learning is off-policy.

A classical reinforcement learning network is using hand-crafted states, such as temperature, speed, etc. [18]. However, the deep learning technology could be merged with the reinforcement learning network to reach a Deep Reinforcement Learning (DRL) model.

1.4.2 Convolutional Neural Networks (CNN)

The Convolutional Neural Networks, which abbreviated as CNNs or ConvNets, are composed of several numbers of neural layers comparable to humans' natural neurons. Liang et al. [19] admit that CNN is a multi-layer network, whereas each layer has its own neuron nodes. The nodes use weights and biases to be able to learn the relationships between input variables and output variables [20].

The classical CNN consists of three main layers. First, it is called the convolution layer. Second is called the pooling layer, and the last layer is the fully connected layer. Rawat and Wang [21] state that the convolutional layers are playing the role of features extractor. Thus, they learn and extract the features from the input images to organize the neurons located on the convolutional layers into feature maps. On the other hand, the role of the pooling layers is to minimize the spatial resolution exited in the previously arranged feature maps to reach spatial invariance. Usually, there are several numbers of stacked convolutional and pooling layers on top of each other to extract the features and reduce the distortion in the data. Finally, the fully connected layer responsible for computing the final loss function, such as the “Softmax” to solve classification problems.

Krizhevsky et al. [22] state that using deep CNN would improve the performance of image retrieval, especially when dealing with large and complex datasets. CNN can be trained on datasets and become able to select the best distinguishing features using either supervised learning or unsupervised learning.

1.4.3 Recurrent Neural Networks (RNN)

RNN is named recurrent because it executes sequential elements in an identical manner, whereas the generated output is depending on the preceding execution [23].

RNN has been used excessively in the Natural Language Processing (NLP) applications because it inspects sequential information. Thus, it can achieve excellent results in predicting the next words in a sentence. That is due to preserving a "state vector" in the network to hold a copy of all elements' previous history. RNN employs a backpropagation algorithm in their training, which made them rigid but more difficult to train since the back-propagated elements might get smaller or bigger in every step [19].

RNN differs from CNN in that there is no connection between the nodes of the layers. However, the layers are fully connected. Hence, we can imagine that each layer is presenting the network computations at a specific time slot. The information in RNN can persist as if they are being held in memory because RNN has loops within the network (repeating module). These loops create chain-like information that maintains the previous history of every element in the list [24].

According to Elmowafy [25], there are three main tasks of RNN as following:

1) Sequence Recognition
2) Sequence Reproduction
3) Temporal Association

Note that each task is dedicated to solving a specific type of problem. For instance, sequence recognition can recognize handwritten texts and sentiment analysis. While the sequence reproduction task is utilized in language models applications such as predicting the residual words in a sentence. On the other hand, the temporal association task can associate specific inputs to outputs such as translating from one language to another.

1.4.4 Long Short-Term Memory (LSTM)

LSTM presents a special type of RNN. They were initially proposed by the authors in [26] and later on, they were modified and enhanced by many other researchers. However, LSTM differs from RNN in that they can refer longer than RNN. Therefore, LSTM can connect large gaps and present their related information [23].

Colah [24] claim that both RNN and LSTM consist of chain-like repeating modules. However, the repeating module in RNN is much simpler than the repeating module of LSTM. That is because, in RNN, there is only one single neural network layer, such as the “Tanh” activation layer, for example. On the other hand, the repeating module in LSTM consists of four linked neural network layers. This unique structure of the connected neural layers in the LSTM enables it to contain a transporter belt that tight and pass-through all the layers, called “cell state”. The cell-state in LSTM simplifies the addition and deletion of information utilizing special “gates” while preserving the entire rest of information flowing through the chain to be maintained.

LSTM showed success in many prediction applications depends on long memorable information.

1.5 Convolutional Network Models

CNN has the ability to learn features automatically and capture visual similarity, which made them successful in many fields [3]. Note that leveraging the different transformation structures in deep learning, such as spatial transformation structure and the recurrent connection, would improve the accuracy of retrieved images remarkably. According to Karpathy [27], there are many different architectural models for ConvNets, such as LeNet, AlexNet, ZFNet, GoogLeNet, VGGNet, ResNet, etc. Note that, LeNet was the first implantation for the CNN in 1990 specialized in classifying

digits and Zip codes. It was designed to have only one single convolutional layer followed by also one single pooling layer. However, in 1998 LeNet design was modified to have three convolutional layers, two pooling layers, and one fully connected layer [28].

Afterward, in 2012, AlexNet was applied in a similar manner as LeNet but, with more layering depth to increase the output accuracy. A year later, in 2013, another architectural model for CNN was invented known as ZFNet. It made adjustments on the previous CNN by decreasing the size of the stride and filter that lays on the first input layer. Afterward, in 2014 a CNN from google was invented known as GoogLeNet. The improvements in this particular type of CNN was minimizing the number of useless parameters through replacing the fully connected layers by an average pooling. In the same 2014 year, another CNN architecture also occurs, known as VGGNet. It consists of 16 homogeneous convolutional layers, which gives it more depth, but it is harder to evaluate since it consumes more memory than other types of ConvNets models.

In 2015, a new CNN architectural model was designed that is known as ResNet. Note that, it is an abbreviation for Residual Network. It increased the depth of the layers to become 152 layers in order to achieve higher performance in the output results. This increased in layers' depth increased the execution process, too [29]. Following table 1.1 provides a summary of some of the most common CNN architectural models and illustrates the revolution of depth in the neural network layers.

Table 1.1: Some of the common ConvNets architectural models

Model Name	Invention Year	Researchers	No. of Depth Layers	No. of Parameters
LeNet5	1998	LeCun et al.	7	0.060
AlexNet	2012	Krizhevsky et al.	8	60
ZFNet	2013	Zeiler and Fergus	8	60
VGGNet	2014	Simonyan and Zisserman	19	138
GoogLeNet	2014	Szegedy et al.	22	4
ResNet	2015	He et al.	152	6.8
DelugeNet	2016	Kuen et al.	146	20.2
ResNexT	2017	Xie et al.	101	68.1
Convolutional Block Attention Module	2018	Woo et al.	101	48.96

According to the authors in [29], stacking more layers will not improve the learning behavior in the network unless each layer has a useful role. They prove this by developing an experiment with two convolutional neural networks different in the layering depth. The first network has only 20 dump layers, while the second network has 56 dump layers. Their experiment proves that the deeper 56 layers' network generated higher training errors, as well as, higher test errors than the 20 layers' network. However, the authors were able to develop the ResNet CNN model with deeper layers and lower training errors, as well as lower test errors. That is because they designed their model with increased dimensions through adding three conclusion layers in the middle that are having 1*1 (64), 3*3 (64), and 1*1 (256) convolutional layers instead of using all 3*3 (64) similar convolutional layers. Hence, they made deeper layers in their ResNet practical design undoubtedly better.

Khan et al., [30] admit that the current trend in developing a new CNN architectural model is to use blocks instead of the conventional layer structure. That is because using blocks to boost the learning process has proven higher performance than the conventional convolutional layering structure. Since blocks are taking advantage of the information provided by spatial and feature maps, making the whole architecture

simpler and easier to understand. Figure 1.5 summarizes some of the ConvNet architectural models.

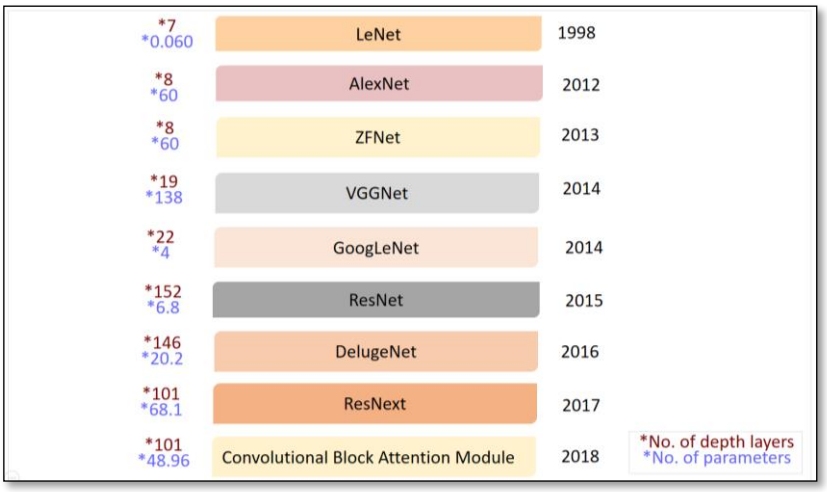


Figure 1.5: ConvNets architectural models

1.6 Arabic Manuscripts

The Arabic manuscripts are a significant source of historic knowledge presented in manual handwritten pages. The Arabic manuscripts image retrieval is to retrieve similar images from the same manuscript after matching the user's input query image with all other saved images in the dataset to retrieve the most similar images to the user query image, as illustrated in figure 1.6.

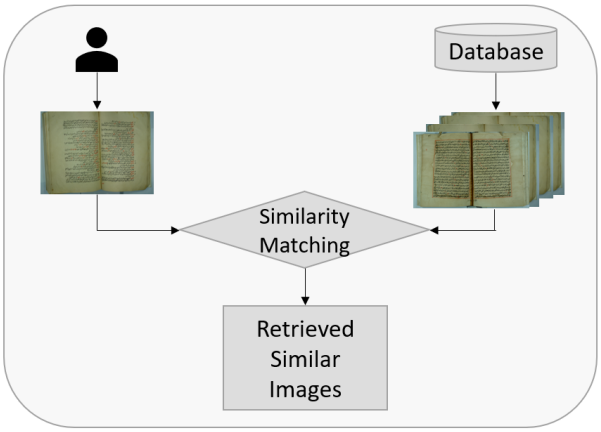


Figure 1.6: General image retrieval system

1.7 Challenges Associated with The Historical Arabic Text-Based Images

Many image retrieval systems employ handwritten text-images as their inputs. Thus, training the machine to recognize and process handwritten texts has been an essential concern for many researchers. They need to preprocess and segment the text into paragraphs, lines, sentences, words, or even characters. Because after the segmentation process, it is much easier to understand and to handle the segmented parts of the document for further analysis.

There are many traditional methods for text-based image retrieval, such as word spotting, text queries, segmentation, and recognizing optical characters. However, Alaei et al. [31] believe that the traditional optical character recognizer-based method is inadequate for document image retrieval due to its requirements to high computational cost, vulnerability to images resolution, and due to its dependency on the used language. Moreover, Yahia [13] admits that using the classical OCR for the retrieval is unsatisfactory, especially when retrieving the texts from historical Arabic manuscripts. That is due to the special characteristics in the ancient Arabic manuscripts that make the retrieval process more complicated. The historical Arabic manuscripts, in particular, posing additional challenges due to their degraded quality of blotched papers, faded inks, and lots of non-textual noise added within the main text. Al-Ayyoub et al. [8] state that the automatic manipulation and understanding of the Arabic language is challenging due to its distinct characteristics. In addition, Al-Jawfi [32] admits that the handwritten Arabic language has characteristics that make recognizing it harder than other languages. For instance, the fact that Arabic words are written from right to left and their dots might appear above or below the letters, also the dots are ranging from 0-3, makes the recognition of the Arabic language more challenging. Even though the Arabic alphabets are consisting of only 28 letters, most of them come

in four different forms depending on their position within the word, which increase the alphabet patterns from 28 to around 60. El Makhfi [33] admits that the old handwriting styles used for writing the Arabic manuscripts and saved in an image format are constituting main obstacles for text recognition and extraction.

1.8 Objectives of the Study

Our goal is to develop deep learning models that retrieve the ancient Arabic manuscripts' images instantaneously and accurately. Therefore, we investigated the use of deep learning technology fusion model that utilizes both visual and textual extracted features from the handwritten Arabic manuscripts for a precise image retrieval system. Hence, we can summarize our objectives in the following points:

1. Collect ancient Arabic manuscripts and store them in a dataset.
2. Analyze the algorithms used in both the computer vision and in the NLP fields to reach the most suitable algorithms.
3. Employ deep learning technology to automate the feature extraction phase.
4. Extract the images' visual and textual features. Then, integrate them into one fusion model for better image retrieval.

1.9 Problem Statement

There is a need to retrieve low-quality faded inks historic images, and not only the high-quality computer printed and recent images.

Moreover, there is a need to automate the feature extraction phase to overcome the low quality presented in the ancient manuscripts' images. Another problem that is an essential concern of this research is to train the machine to recognize and to

successfully extract the handwritten Arabic texts, which are saved in an image format. Because the historical Arabic manuscripts' images are not purely textual. Thus, it is necessary to consider the non-textual parts and retrieve the images according to their visual and textual features.

Therefore, an image retrieval system that is dedicated for the Arabic manuscripts is the solution to all these problems.

1.10 Research Methodology

This study aims to accurately retrieve historical Arabic manuscripts, which are very old, handwritten, and having noise. Therefore, we decided to investigate the ability of merging the computer vision methods with the natural language processing methods for successful retrieval of the ancient Arabic manuscript images. To accomplish our goal, we plan to do the work on three main phases:

1- Data collection

- Collect ancient Arabic manuscripts and store them in a dataset.
- Classify the dataset images according to three main criteria, which are the manuscript, author, and calligraphy.

2- Images Preprocessing and Segmentation

- Recognize the text written inside the images and extract it.
- Clean the extracted text.
- Augment the visual images to generalize the models on them.

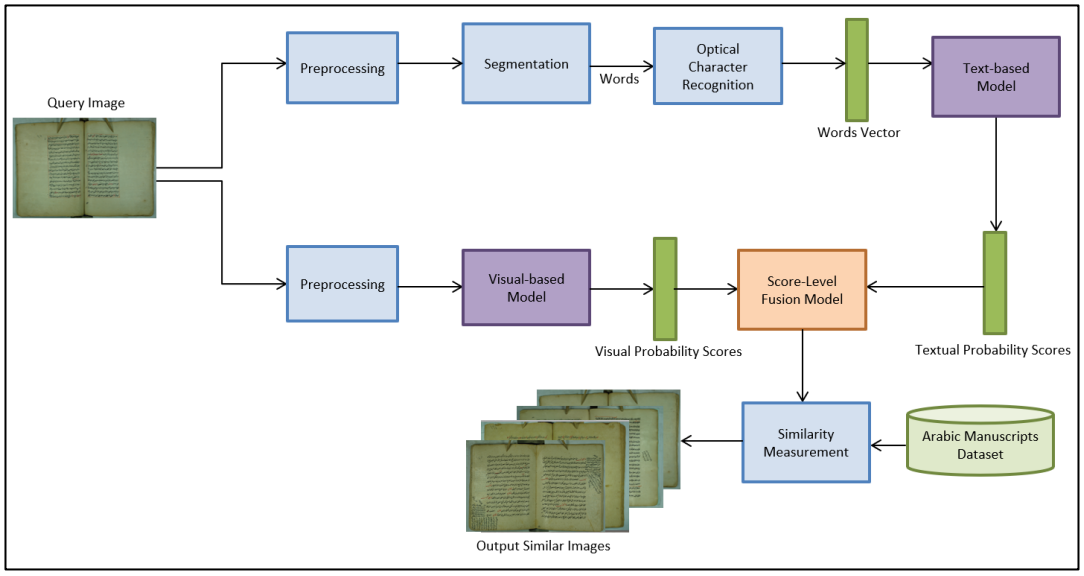
3- Deep Networks Development

- From the computer vision models, develop a deep CNN to extract the visual features from the images.

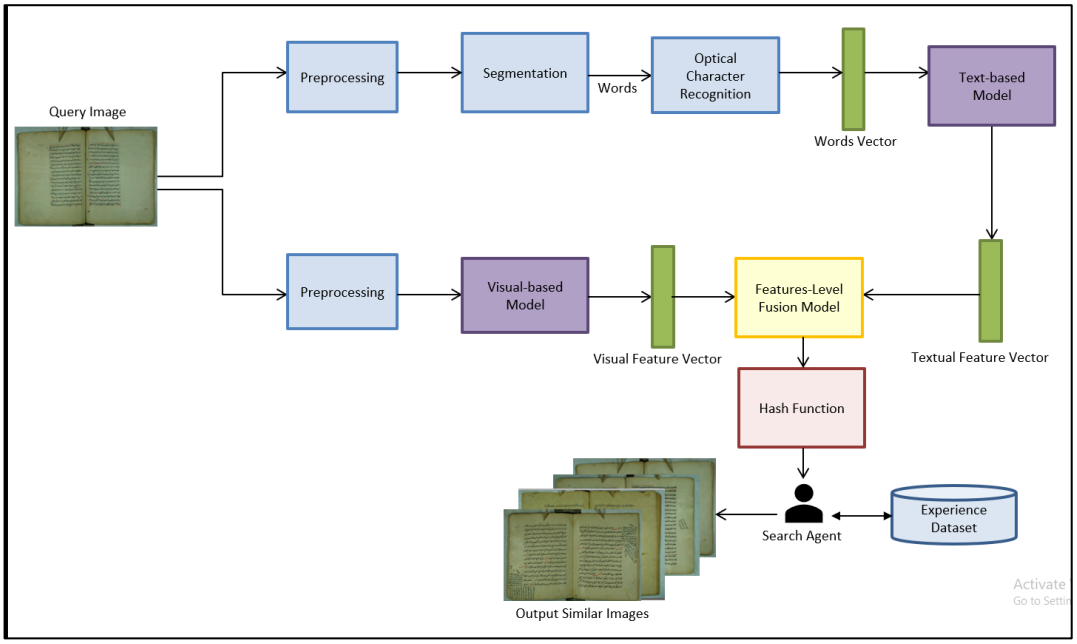
- From the NLP models, develop an attentional deep LSTM learning model to extract the textual features from the images.
- Fuse both the visual and the textual models into one model that more accurately characterize the images.
- Develop a deep reinforcement learning model to experiment its performance in retrieving the images of the Arabic manuscripts.

Figure 1.7 illustrates the high-level design of the proposed image retrieval system.

From figure 1.7, we notice that the proposed method for retrieving the Arabic manuscripts' images consists of two alternative solutions. The first solution (a) is to develop two supervised deep learning models. One to generate the textual probability scores, and the second to generate the visual probability scores. Both scores are fused using one score-level fusion model. Then, the similarity is measured between the manuscripts' images stored in the dataset and the user's query image to retrieve the most similar images.



(a)



(b)

Figure 1.7: The high-level design of the proposed Arabic manuscripts IR system. (a) Using supervised DL models, and (b) Using DRL model

On the other hand, the second alternative solution for the images' retrieval (b) is to develop two supervised deep learning models. One to generate the textual features vectors, and the second to generate the visual features vectors. Both vectors are fused using one features-level fusion model. Then, the fused features are hashed and sent to the search agent, which retrieves the most similar images to the user's query image.

1.11 Contributions to the Field

1. Data collection and labeling according to the manuscript, author, and calligraphy.
2. Develop visual DL model for Arabic manuscripts images retrieval with the following contributions:
    2.1 Investigate transferred learning from four pre-trained deep CNNs to classify and to extract the visual features from the images while

experimenting the best categorization for the dataset and modulating the learning hyperparameters to reach the most accurate classification.

2.2 Match and retrieve similar images using the Siamese deep neural network, which is the first work done on Arabic manuscripts. As well as, measuring the similarities using three different static distance metrics.

3. Develop a textual DL model for Arabic manuscripts images retrieval with the following contributions:

3.1 Recognize and extract the text using Google OCR then, preprocess the text utilizing the NLTK and convert it into feature vector using the AraVec word embedding tool.

3.2 Develop an optimized bidirectional LSTM deep learning model, including attention and batch normalization layers, to classify and to extract the textual features from the recognized text. The bidirectional layer is significant with the Arabic language in particular because the classical LSTM layer is unidirectional. Thus, it reads the text from left-to-right only, which is the default direction used with the English language. But the Arabic language has the opposite direction. Thus, reading the text from the left-to-right and from the right-to-left showed significant improvements with Arabic text classification. Moreover, the attention technique assists the model to focus on the important words, which simulates the humans' concentration on specific words only. The batch normalization layer updated the final weights, which enhanced the classification further.

3.3 Measure the similarity between the classified text using three different static distance metrics.

4. To the best of our knowledge, we are the first study that fuses both visual and textual deep learning models at different levels and using different rules to retrieve the Arabic manuscripts' images accurately. The decision-level fusion model is experimented with four merge layers named: concatenate, average, maximum, and multiply. Moreover, we experimented the features-level fusion using the concatenate merge layer. And we experimented the score-level fusion using three different rules named: Min rule, Max rule, and Sum rule.
5. Retrieve the top-k similar images to a user query image and compute the mean accuracy of the retrieved images according to the manuscript, author, and calligraphy.
6. To the best of our knowledge, we are the first work that develops an enhanced deep reinforcement learning model to retrieve the Arabic manuscript images. The model based on the features-level fusion of both the visual and textual models. The fused features are then hashed to reduces the dimensionalities among them and to make the environment more stable through removing the randomness existed in the search process.
7. Develop and deploy a web-based prototype for the Arabic manuscripts' image retrieval. We aim in the future to make it the largest database for the ancient Arabic manuscripts and the main search engine using the manuscripts' pictures.

1.12 Thesis Outline

The rest of the thesis is organized as follows:

- Chapter II discusses the literature review, including the currently existing techniques for image retrieval and the gap in previous works.

- Chapter III explains the used strategy for collecting and preparing the visual and textual data. As well as it explains the classification of the collected dataset and the challenges faced during its collection.
- Chapter IV introduces in detail the proposed Arabic manuscripts' image retrieval system along with its implementation.
- Chapter V evaluates the performance of the proposed Arabic manuscripts image retrieval system and discusses the experimental tests with obtained results.
- Chapter VI includes the conclusion, limitations, and future work.

Chapter II

Literature Review

With the rapid production of massive amounts of images and the necessity to use them in various applications such as medical, educational, business, tourism, and other types of different applications. There is a critical need to discover rapid systems that are able to retrieve accurate images successfully and instantaneously. Many studies implemented the image retrieval system, some of them based on extracting the visual features. While other studies based on extracting the textual features. There were also some efforts implemented to fuse multiple features into one fusion model.

The extracted features from the images could be local such as color, shape, texture, etc., which are known as the hand-crafted features. In contrast, there are global high-level features of images, such as their activities or objects, which are known as deep learning features.

The literature review chapter begins by reviewing the classification of image retrieval techniques by many researchers. Afterward, we divide the image retrieval techniques into three main subsections: visual-based features, textual-based features, and fusion-based image retrieval.

There were many efforts of classifying the image retrieval techniques into main categories. Figure 2.1 summarizes some of the previous researchers' classification for image retrieval techniques.

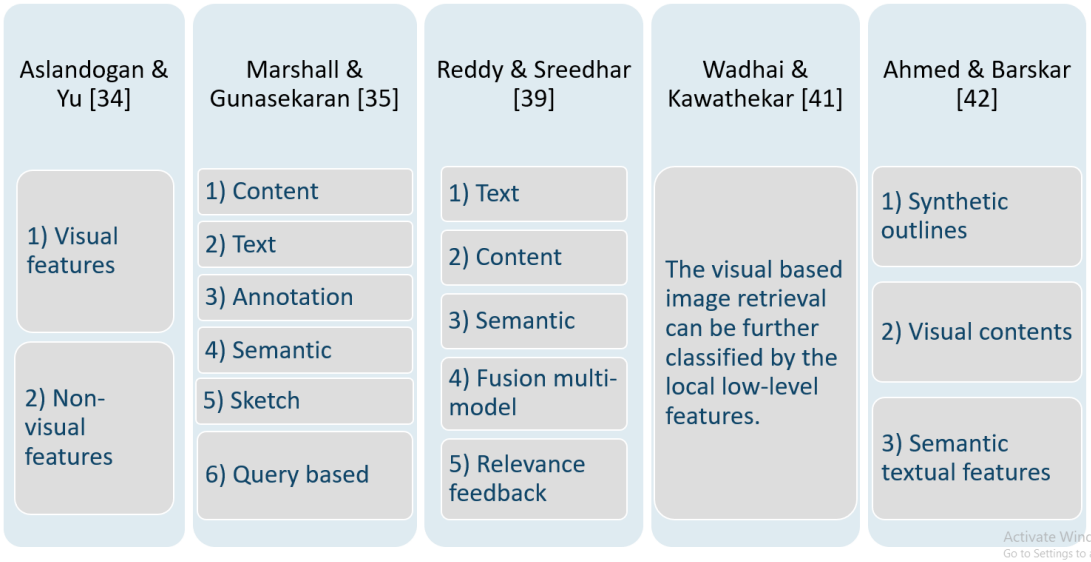


Figure 2.1 Researchers classification of the image retrieval techniques

Aslandogan and Yu [35] classified the image retrieval techniques into two broad categories as either visual features or non-visual features. However, Marshall and Gunasekaran [36] customized the techniques further by classifying them into six main categories as following: 1) content, 2) text, 3) annotation, 4) semantic, 5) sketch, and 6) query based image retrieval.

Considering that the content-based image retrieval techniques are methods that leverage the low-level visual features presented in the images. On the other hand, the text-based image retrieval is based on segmenting the images and extracting the text included in the images. Annotation-based image retrieval techniques are based on labeling the images to be able to search and match specific keywords with the required label of the annotated images to retrieve them. In contrast, the semantic-based image retrieval techniques tend to concentrate on the semantic knowledge presented in multimedia images such as their regions. The sketch-based image retrieval is the technique that captures the drawings to retrieve images.

In regard to the annotated image retrieval techniques, there were many studies to do the annotation automatically instead of manually. For example, Saleem et al. [37] used the Markovian Semantic Indexing (MSI) method to do the annotation automatically. Furthermore, Kumar et al. [38] employed the weighted nearest-neighbor, as well as the multi-class classification techniques, to annotate medical images automatically. Moreover, Hare et al. [39] utilized a semantic-space method using linear algebra to do the annotation softly through training.

Reddy and Sreedhar [40] classified the image retrieval techniques into five major categories as following: 1) text, 2) content, 3) semantic, 4) fusion multi-model and 5) relevance feedback image retrieval techniques. The authors admit that many efforts were trying to integrate more than one image retrieval approach to generate a fusion multi-model that is able to increase the accuracy of the output. Moreover, they state that there is another approach for image retrieval called relevance feedback, which is mainly base on the interventions and feedback from the interacted users with the retrieval system to amend and retouch the retrieved images.

Methods in relevance feedback technique include delta mean algorithm, standard deviation and variance, query point movement, Bayesian framework, support vector machine, biased discriminant analysis, and kullback-leibler distance.

Rui et al. [41], employed the relevance feedback method to retrieve the images successfully. They experimented their model on 384 texture images from “MIT media lab” dataset that is available online, and they were able to record an increased precision using the feedback case than without employing any feedback.

According to Wadhai and Kawathekar [42], visually based image retrieval techniques can be further classified by the local low-level features that are extracted from the

contents of the image. Hence, they have summarized the utilized techniques in each type of extracting the local features as following:

1- Color features
    a. Conventional Color Histogram (CCH)
    b. Fuzzy Color Histogram (FCH)
    c. Color correlogram
2- Texture features
    a. Steerable pyramid
    b. Contour let transform
    c. Complex Directional Filter Bank (CDFB)
3- Shape features
    a. Fourier descriptor
    b. Moment invariants
    c. Directional histograms

Ahmed and Barskar [43] had a different classification of image retrieval techniques. They believe that based on the state-of-the-are researches, the techniques are divided into three categories as following: 1) retrieval based on synthetic outlines, 2) retrieval based on visual contents, and 3) retrieval based on the semantic textual features included within the images.

Moreover, Dureja and Pahwa [44], classified image retrieval techniques into three other distinct categories. They believe that the techniques to image retrieval started with depending on the visual features only to retrieve images and then developed into using the distance metric learning; until it reached using deep learning technology to retrieve images. Note that they admit that the deep learning techniques that leverage the CNN layers to extract the images features automatically are currently the best

techniques for retrieving images successfully, especially when dealing with large datasets.

After reviewing current existing techniques for image retrieval, we classify the techniques into three main categories as:

1. Visual-based image retrieval
    - Handcrafted features
    - Deep learning features
2. Textual-based image retrieval
    - Handcrafted features
    - Deep learning features
3. Fusion-based image retrieval

Both the visual and textual-based image retrieval are classified further into handcrafted and deep learning features. While the fusion model could include a mix of handcrafted and deep learning features.

Figure 2.2 highlights our classification of image retrieval techniques according to our problem.

We reviewed the papers that worked on the image retrieval in general, as well as; we focused on the papers that worked on the Arabic image retrieval in particular, looking for the best techniques that we could utilize with the Arabic manuscripts' images.

2.1 Visual-based Image Retrieval

2.1.1 Handcrafted Features

Rana et al. [1] investigated integrating parametric and nonparametric features for a better content-based image retrieval process. To accomplish their goal, five different image datasets used. They are as following: Simplicity, Corel-10K, Corel-5K, Caltech-

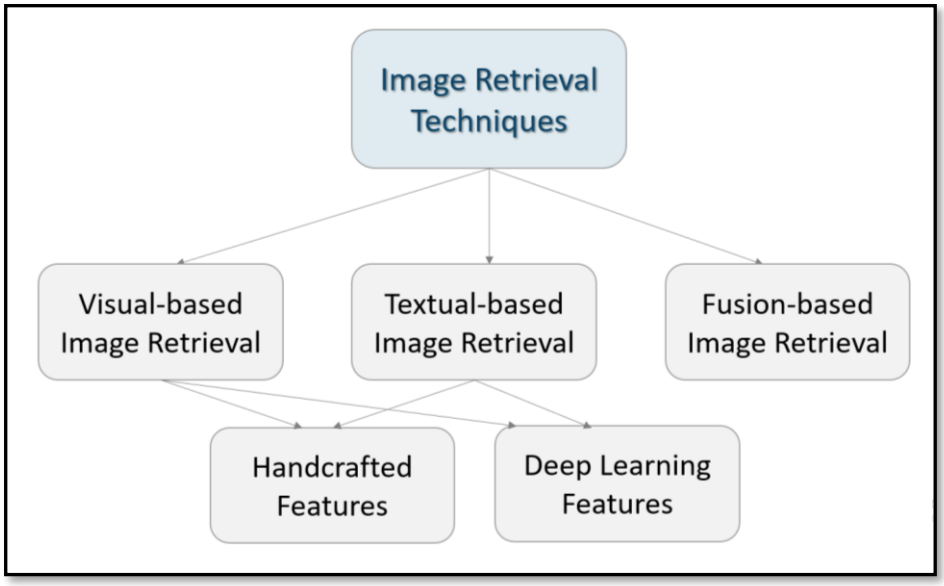


Figure 2.2: Our classification of the image retrieval techniques

101, and Microsoft Research (MSR). Considering that it is hard to use only one single feature to characterize document images for their retrieval purposes. Therefore, the authors recommended integrating parametric features such as color and shape with non-parametric features such as texture-based features to retrieve content-based images successfully. CRM method used to extract the features from both the offline images in the dataset. As well as to extract the features from the online query images. Note that, the "C" letter stands for Color moments, "R" letter stands for Ranklet transformation, and the "M" stands for Moment invariants. Each one of the three mentioned methods has its own algorithm and implementation. In regard to the parametric methods, the researches chose color moments and moment invariants for obtaining the color distribution and shapes of the images. However, in regard to the nonparametric features, the researchers chose the ranklet transformation method to narrate the nonparametric features. In order to find out the similarity between images, the authors used the Euclidean distance metric. Moreover, the authors evaluated their proposed technique with seven other recent techniques used for images retrieval

namely: Color Difference Histogram (CDH), Edge Histogram Descriptor (EHD), Multi-Texton Histogram (MTH), Color Auto Correlogram (CAC), Distribution of Color Ton (DCTon), Golomb-Rice (GR) coding based indexed histogram, and Local Binary Pattern (LBP) based method with the combination of Discrete Wavelet Transformation (DWT) and Legendre Moments (LM). Even though the authors admit that their proposed method would retrieve correct images, they also claim that there is a need to reduce the features vector length in their method, which is 247 as it is long compared with other techniques, and this extra length complicate the images retrieval time. Furthermore, the authors stated that there is a need for humans' intervention to filter-out the output results after retrieving images.

Zhou and Jia [2] proposed a new learning framework that utilizes the multilayer perceptrons technology to retrieve 3D images based on sketches and shapes entered datasets. To perform the study, the authors used the National Taiwan University dataset and a sketch dataset. They preprocessed their selected datasets in the pair-processing pipeline to be able to get the finest view of the inputs. They employed the Principal Component Analysis (PCA) based on the whitening algorithm to minimize the dimensions between the input features and to obtain a unit variance for each dimension. The authors employed the Sketch-Based Local Binary Pattern (SBLBP) hand-crafted algorithm to extract the features from the shapes manually. Afterward, they used the Support Vector Machine (SVM) classifier to take a vector of numbers and decide whether it belongs to a particular class or not. They concluded that their proposed method is better for image retrieval compared with other methods and that "SIFT" image retrieval method generates bad results specifically when using sketches as input to the model.

Beecks et al. [5] investigated the rigidness of utilizing signature-based similarity measurement to correctly and rapidly retrieve content-based images from large datasets. MIR Flickr database used to perform this study. However, the researchers used another three different image datasets to test their proposed model, which are Holidays, UKBench, and Copydays databases. They suggested feature signature representation of the original images to extract three local features from them, which are the color, position, and texture. Therefore, each individual image was quantized by its visual vocabulary represented through its assigned feature signature. The authors utilized various distance signature-based methods such as the Earth Mover's Distance (EMD), which measures the transformation cost from one feature signature into another. Moreover, the Perceptually Modified Hausdorff Distance (PMHD) is also used to measure the similarity between two feature signatures. Furthermore, the Signature Quadratic Form Distance (SQFD) is used to measure the correlation between two feature signatures, which is mainly measuring how similar the two feature signatures are. An alternative utilized correlation-based similarity measure is the Weighted Correlation Distance (WCD).

Eventually, the authors calculated the average precision stability to compute the stability of the signature-based measure putting-in-mind that there is a difference in the quality between the query images and the images in the database. The researchers used three different image datasets to test their proposed model, which are: Holidays, UKBench, and Copydays databases. They concluded that there is no direct correlation between the stability and the retrieval performance as the highest stability measure doesn't always have to match the highest image retrieval performance. That is because the highest Average Precision Stability (APS) was recorded by the Copydays dataset

as 0.763, while its retrieval performance was not that good comparing it to other datasets.

Alaei et al. [31] compared different texture-based features extraction methods for a better image retrieval process. Thus, they used three document images datasets, which are "MTDB", "ITESOFT", and "CLEF-IP". They preprocessed their datasets by performing (3 × 3) mean filtering method three times on the initial images to convert them into greyscale versions. Afterward, they applied a size normalization method to resize all the converted greyscale images into a similar size. They selected the smallest size in the MTDB dataset, which is (140 × 90) pixels.

After preprocessing the images, the authors computed features vectors through twenty-six texture-based features extraction methods from four groups of textures features as following: 1) statistical, 2) transform, 3) model and 4) structural-based approaches. Then, the extracted vectors were stored in a knowledge-based database. To measure the similarity between the extracted features from the training set and a given query, the authors employed the city-block distance measurement. Then, both the mean and standard deviation of the distance matrix were calculated to select the accepted retrieved images in a class. The equation is as follows:

$$t = \mu + \gamma * \sigma \tag{2.1}$$

Where t is the calculated threshold, μ is the mean of distances, γ is a selected fixed value, and σ is the standard deviation of distances.

The authors found out that the transform-based approach performed better in all trained datasets. It recorded 75.42%, which is the highest reached precision, 99.01% as the highest reached recall, and 85.62% as the highest reached F-score.

Bagasi and Elrefaei [45] proposed retrieving historical Arabic manuscripts using the visual local-based features. The dataset used is manually collected from 30 books. It contains 1670 images of historical Arabic manuscripts. The manuscripts are classified into 29 classes based on the author of the manuscript. They preprocessed their manually collected dataset by converting the colored images into greyscale and then resized them into 256*256 pixels. The last step in the preprocessing phase is to use the Otsu's method to binarize grey-scale images to be able to visualize their contents better. The authors recommended extracting the local visual features from the ancient Arabic manuscripts using two CBIR techniques, which are the Speeded-up Robust Feature (SURF) and the Binary Robust Invariant Scalable Keypoints (BRISK). They compared the performance of retrieving images based on each one of the recommended techniques to be able to reach the best technique for image retrieval. Afterward, the authors employed the Hamming Distance (HD) measurement to find-out the matching images for BRISK feature extraction technique. While they used the Sum of Square Differences (SSD) measurement for the SURF technique.

Since the authors' main goal is to compare two well-known CBIR techniques for image retrieval, which are SURF and BRISK. Hence, they used the Matlab 2017 to conduct their experiments and computed both precision and recall to reach the accuracy of their proposed system. They also used the confusion matrix to evaluate their system as a whole. The authors reached that the SURF technique extracts the local visual features

better than the BRISK technique. That is because SURF accomplishes 70% for both precision and recall, while BRISK accomplish 53% recall and 50% precision. Noting that the overall accuracy of the system is 61% using the SURF technique and 37% using the BRISK technique.

Chen et al. [46] proposed a bipartite graph-based matching approach to retrieve 3D objects utilizing multi-feature collaborative learning. Three different 3D object datasets were used in the study, which are: ETH-80, NTU, and SHREC15. They proposed a 3D object descriptor method to extract multiple features from 3D objects and then, they recommended fusing the extracted features through an interactive feature learning model, which harmonizes the contour descriptors with the interior region of 3D objects. Afterward, they measured the similarity through two main steps as following: First, a Greedy Search (GS) algorithm is used to measure the similarity between the query object and the 3D objects stored in the dataset. Second, three bipartite graph matching algorithms are implemented to reach the perfect match between each bipartite graph pairs. The authors concluded that there is an essential improvement in the retrieval of 3D objects in particular when we utilize the recommended bipartite graph matching and the feature concatenation. Furthermore, they believe that applying a ranking method on the proposed method would improve its retrieval performance. For example, the Manhattan distance would improve its performance up-to 21.5% if ranking method applied on ETH, 1% if applied on SHREC, and a very little change if ranking method applied on NTU.

Ahmed et al. [47] suggested combining many features extraction techniques to accomplish a better image retrieval rate. To perform their study, they used ten different image datasets as following: ImageNet, Caltech-256, Caltech-101, 102-Flower, Corel-

10,000, 17-Flower, Corel-1000, COIL, ALOT, and FTVL tropical fruits. They preprocessed their original images by converting them into gray-scale images. Then, the interest points were discovered utilizing the intensity region method. Afterward, they extracted the spatial color features from their dataset images using the color correlation histogram algorithm. In addition, they employed the Bag-of-Words (BoW) to index and searched the k-nearest images from the queried image. The authors concluded that their proposed method is able to retrieve correct images from complex datasets accurately and that the technique is achieving high performance comparing it with other existing techniques, as it reached 6.8% higher mean average precision than other methods.

Raju et al. [48] proposed improving the performance of retrieved images by utilizing more than one distance method for measuring the similarity between the query image and the stored images in the dataset. A dataset consisting of 20 different colored images was utilized to perform the study. They recommended using a content descriptor to extract the visual features from their collected dataset according to both images' colors and edges. Afterward, they utilized a mini-max method to obtain the most accurate similarity measurement with the query image. The method combines the measurements from three different histogram distance methods, which are: Euclidean, cosine, and histogram intersection. They reached that combining multiple features to find-out the similarities is always better than using only one feature. In addition, they concluded that the cosine distance measurement accomplished more accurate results than the cosine and histogram intersection distance measurements. That is because it accomplished an 89% accurate precision rate.

Table 2.1 summarizes the studies that addressed the visual handcrafted features. Note that the cells highlighted using the blue color in all incoming tables are for the studies that worked on the Arabic language.

The “SF” in the similarity measurement column stands for Static Formula, “ML” stands for Machine Learning, and “DL” stands for Deep Learning.

2.1.2 Deep Learning Features

Radenovic et al. [49] proposed using deep neural networks for better image retrieval. They used datasets that include 3D objects, such as Oxford Buildings, Paris, and Holidays datasets. Unlike many researchers who recommend pre-processing images offline before entering them into the network, the authors advocated for post-processing images after they complete fine-tune training using CNN. The post-processing steps include whitening the images and reducing their dimensionality. Hence, all trained images were resized to (362 × 362) pixels. Moreover, the researchers used "EXIF" metadata to rotate the images, so they are re-located on the upright position of the portray images.

After post-processing the images, the authors recommended using ConvNets for high-level features extraction from 3D images. Then, they employed the constructive loss function to compare their trained inputs with the queried image. Finally, the authors evaluated their proposed model with other state-of-art models by computing the mean average precision, and there are able to record 91.9% using Oxford5k and Paris6k datasets.

Table 2.1: Visual-based handcrafted features

Reference# (Year)	Problem Domain	Approach	Dataset	Similarity Measure	Results
[1] (2019)	General images retrieval	Moment invariants, color moments, and ranklet transformation	Simplicity, Corel-10K, Corel-5K, Caltech-101, and Microsoft Research	*SF: Euclidean distance	• There is a need to reduce the features vector length • There is a need for humans' intervention to filter-out the output results
[2] (2018)	Retrieve 3D sketch and shape images	SBLBP	National Taiwan University dataset and sketch dataset.	*ML: SVM	• 60% precision on NTU dataset • 39% Precision on the sketch dataset.
[5] (2014)	General images retrieval	Feature signature representation	MIR Flickr, Holidays, UKBench, and Copydays	*SF: Earth mover's, perceptually modified hausdorff, signature quadratic form, and weighted correlation	• Average precision stability: 76.3% recorded by the Copydays dataset
[31] (2018)	General images retrieval	Statistical, transform, model, and structural-based approaches	MTDB, ITESOFT, and CLEF-IP	*SF: City-block distance, mean, and standard deviation	• Recall: 99.01% • Precision: 75.42% • F-Score: 85.62%
** [45] (2019)	Arabic manuscripts' images retrieval	SURF and BRISK	Manually collected Arabic manuscripts from 30 books	*SF: Hamming distance and Sum of square disatnce	• Overall accuracy: 61% using SURF and 37% using BRISK • SURF is better
[46] (2019)	Retrieve 3D images	3D object descriptor	ETH-80, NTU, and SHREC15	*SF: Greedy search algorithm and three bipartite graph matching technique	• 21.5% performance improvement if ranking method applied on ETH • 1% if applied on SHREC
[47] (2019)	General images retrieval	Color correlation histogram	ImageNet, Caltech-256, Caltech-101, 102-Flower, Corel-10,000, 17-Flower, Corel-1000, COIL, ALOT and FTVL	*ML: Bag-of-Words (BoW) to index and search the k-nearest images	• mAP: 6.8%
[48] (2014)	General images retrieval	Content descriptor	20 different colored images dataset	*SF: Euclidean, cosine, and histogram intersection	• Precision: 89%

The ** written before the reference number is to highlight that the study addressed the retrieval of the Arabic manuscripts' images.

Seddati et al. [50] explored the rigidness of utilizing convolutional neural networks for extracting visual features from images to retrieve them instantly. They used three well-known images datasets to conduct their study. The datasets are IN-RIA Holidays, Oxford5k, and Paris6k datasets. The authors used ResNet101 CNN feature extractor to retrieve images successfully. Their proposed approach is based on Multi-Scale Regional Maximum Activation of Convolutions (MS-RMAC) descriptor, which utilizes the resulting fully-connected CNN to extract images features. The authors measured the similarity using the K-nearest neighbor algorithm to find-out the nearest four images to the original queried image. They evaluated their model and accomplished an accuracy of 72.3 using Oxford5k, 87.1 using Paris6k, and 94.0 using the INRIA Holidays benchmark dataset.

Koch et al. [51] developed a deep Siamese neural network to recognize the handwritten digits within the Omniglot Dataset. The authors measured the similarity among the query digit and all other digits stored in the dataset using the Manhattan distance metric, and they recorded a final accuracy of 93.42%.

Ge et al. [52] experimented the effect of changing the feature vector size on the performance of the image retrieval system. Hence, they used three different pre-trained convolutional neural networks named: VGGM, VGG16, and GoogLeNet; to transfer learning into PatternNet dataset. The Euclidean distance metric used to measure the distances among the images. The authors concluded that the highest achieved mean average precision was using the GoogLeNet pre-trained deep neural network with (832) features dimension since it reached 65.98%.

Ong et al. [53] proposed using the architecture of the pre-trained VGG16 deep learning model to develop a Siamese neural network for image retrieval. The authors employed two image datasets named Oxford and Paris to evaluate their model. They computed

the similarities between the images using Euclidean measure and recorded 81.5% and 82.5% mAP on Oxford and Paris datasets, respectively.

Qiu et al. [54] recommended using the ResNet pre-trained model to develop the Siamese deep neural network. They used the TUM dataset and created two customized datasets to show the model both positive and negative samples from the original dataset images. The researchers utilized the Euclidean distance method to measure the similarities and accomplished 87.7% precision.

Chaudhuri et al. [55] used the Siamese graph convolution network (SGCN) to extract the visual features from the images and retrieve similar images to a query image. They begin by segmenting the images of both the UC-Merced and the PatternNet datasets. Afterward, they entered the segmented graphs of the images into the Siamese model to measure the similarity among the images using the Euclidean distance method within the Siamese model. The authors reached 69.89% mAP using the UC-Merced dataset and, 81.79% mAP using the PatternNet dataset.

Wiggers et al. [56] proposed using the architecture of the AlexNet pre-trained deep learning model to build the Siamese model. They used the Tobacco800 dataset to experiment their model and reached 0.944 as the highest recorded mAP on the retrieved top-5 similar images to a query image using a feature vector of size (4096) combined with the Euclidean distance metric.

Ioffe and Szegedy [57] strengthened their employed deep learning model by adding the batch normalization layer before the final dense layer. The authors experimented classifying labels from the MNIST dataset using the Inception pre-trained deep learning model with and without the addition of the batch normalization layer. They concluded that without the batch normalization layer, the Inception classification model recorded (72.2%) validation accuracy. However, after adding the batch

normalization layer, the validation accuracy raised up-to (74.8%), in fewer time steps, which prove the effectiveness of using the batch normalization for both increasing the accuracy rate and for minimizing the training time.

Zhao et al. [58] designed and implemented a deep reinforcement learning model for classifying vehicle images as an attempt to make the transportation system intelligent. They used the deep VGG CNN with a visual attention layer to focus on the important parts of the images while ignoring the trivial parts of them. Hence, the role of the agent is to find the attentional parts of the images. Afterward, the hash code of the query input image, along with all the other images in the dataset are computed and then entered into a hamming distance to measure the similarities between them. The generated similarities from the hamming distance are ranked to retrieve the top similar images to the user query image. The authors used the surveillance-nature dataset to assess their model and recorded 96.41% classification accuracy.

Lin et al. [59] recommended using the Deep Q-learning Network for imbalanced data (DQNimb) to solve the image classification problem, which is formalized as a sequential decision-making problem. Because the used dataset is imbalanced, the authors employed the Imbalanced Classification Markov Decision Process (ICMDP) solution method. The basic concept of the method is to reward the classified images from the minor classes more than the images from the classes that including a large number of images, to overcome the imbalanced dataset problem. Regarding the used policy, it's an ϵ-greedy policy that utilizes a replay memory to store the real-time experiences and use them to simplify the agent learning process. Hence, a new dataset is built from the agent's experience. The learning hyperparameters are (50,000) replay memory size, (120,000) steps to represent the interaction between the agent and the environment, (Adam) optimizer, (γ=0.1) discount factor, and (0.00025) learning rate.

After setting the learning hyperparameters, the authors used the “IMDB”, “Cifar-10”, “Mnist” and “Fashion-Minist” datasets to assess their proposed DQNimb. The authors recorded the highest G-mean score using the Mnist dataset as 0.991% with 1% imbalance ratio.

Nie et al. [60] proposed using the Markov Decision Process (MDP) to predict the views of 3D images and then retrieve the images using a deep reinforcement learning-based model. The authors utilized the OpenGL tool to visualize twelve ordered pictures from each 3D image. Then, they used the extracted twelve pictures, as well as, the extracted visual features from the images using the deep FDNnet CNN as the environmental states of the deep reinforcement learning model. According to the received state from the environment, the developed model selects between three pre-designed actions, which all move the 3D image into a specific direction. The authors then computed the Euclidean distance between the query image and the rest of the images in the dataset to retrieve the most similar images. The authors evaluated their proposed method using the ModelNet40 dataset, which consists of 3D images. They reached the highest F-score using the views extraction technique as 31.12%.

Wang et al. [61] recommended using the pre-trained ResNet50 CNN with the deep reinforcement learning technique to solve the videos’ highlight recognition for better retrieval of videos according to users’ preferences. The authors used a set of images and keywords to match the videos with the users’ preferences. Hence, the videos are first segmented into a set of images. Then, the model is trained on the segmented images and keywords. Finally, the user enters a query input image to the model to retrieve the matching videos. The authors evaluated their model using four complete videos and recorded the mAP. They reached 57.24% mAP using the big bang theory,

56.93% mAP using the academy awards, 58.25% mAP using love actually, and 58.62% mAP using the BBC video.

Zhou and Agichtein [62] introduced a dynamic ranked search environment called (RLIRank). The authors proposed utilizing the deep reinforcement learning approach to retrieve the most relevant images to a user query. The retrieved documents get filtered according to the user's feedback. Three stacked LSTM layers, followed by a dense neural network, are used to define the environments states. Once the agent receives a query, then it will perform the search process. The action taken by the agent is to retrieve the ranked most relevant images to the query. The TREC 2016 and 2017 datasets used to evaluate the proposed model and recorded 79.27% and 64.99% accuracy after the 10th iteration using the TREC 2016 and TREC 2017 datasets, respectively.

Yao et al. [63] proposed enhancing the personalized search using the reinforcement learning technique. They called their model (RLPer). It bases on the Markov Decision Process (MDP) to sequentially filter the retrieved documents according to the user's preferences. The model is trained at the beginning according to the expert policy; then, the MDP calculated policy is more engaged to improve the training process. The relevance score between the query and the retrieved documents is computed using three main parts as following: 1) relevance with the query, 2) short-term user's interest, and 3) long-term user's interest. The authors evaluated their model using the public AOL search log dataset and recorded 59.81% mAP.

Table 2.2 summarizes the studies that addressed the image retrieval task utilizing the visual deep learning features.

From table 2.2, we notice that we reviewed papers working on the images' classification only or working on both the image classification and retrieval.

Table 2.2: Visual-based deep learning features

Reference# (Year)	Problem Domain	Approach	Dataset	Similarity Measure	Results
[49] (2018)	Retrieve 3D images	VGG with generalized mean pooling layer and Siamese model	Oxford5k and Paris6k	*SF: Euclidean	91.9% accuracy.
[50] (2018)	General images retrieval	MS-RMAC with ResNet101 CNN	Oxford5k, Paris6k, and INRIA Holidays datasets.	*ML: K-NN	72.3% mAP using Oxford5k, 87.1% mAP using Paris6k, and 94% mAP using INRIA Holidays.
[51] (2015)	Retrieval of images including handwritten digits	Siamese Neural Network	Omniglot dataset	*SF: Manhattan	93.42% accuracy.
[52] (2018)	General images retrieval	VGGM	PatternNet dataset	*SF: Euclidean	61.50% mAP.
		VGG16			62.92% mAP.
		GoogLeNet			65.98% mAP.
[53] (2017)	General images retrieval	Vgg16-Siamese model	Oxford and Paris images datasets	*SF: Euclidean	81.5% and 82.5% mAP on Oxford and Paris datasets, respectively.
[54] (2018)	General images retrieval	ResNet-Siamese model	TUM dataset	*SF: Euclidean	87.7% precision.
[55] (2019)	General images retrieval	SGCN	UC-Merced and the PatternNet datasets	*SF: Euclidean	69.89% mAP using the UC-Merced dataset and, 81.79% mAP using the PatternNet.
[56] (2018)	Retrieve logo images	AlexNet-Siamese model	Tobacco800 public dataset	*SF: Euclidean	94.4% mAP on the retrieved top-5 similar images.
[57] (2015)	Classification of images including handwritten digits	Inception pre-trained model with BN	MNIST dataset	*DL: Softmax	74.8% accuracy.
[58] (2017)	Classification of vehicle images	Visual attentional reinforcement learning	Surveillance-nature dataset	*SF: Hamming Distance	96.41% classification accuracy.
[59] (2019)	Classification of general images	DQNimb model	IMDB, Cifar-10, Mnist and Fashion-Minist	*DL: Softmax	99.1% highest recorded G-mean score with 1% imbalance ratio using the Mnist dataset.
[60] (2019)	Retrieve 3D images	Deep reinforcement learning	ModelNet40 dataset	*SF: Euclidean distance	31.12% F-score using the views extraction technique.
[61] (2020)	Retrieve video images	Deep reinforcement learning	The big bang theory, academy awards, love actually, and BBC	*SF: Earth Mover's Distance	58.62% highest recorded mAP using the BBC video.
[62] (2020)	Retrieve ranked search images	RLIRank ranking deep reinforcement search	TREC 2016 and 2017 datasets	* DL: Softmax	79.27% and 64.99% accuracy after the 10th iteration using the TREC 2016 and TREC 2017 datasets, respectively.
[63] (2020)	Retrieve ranked personalized images	RLPer personalized deep reinforcement search	public AOL search log dataset	* DL: Softmax	59.81% mAP

Moreover, we notice from table 2.2 that we focused more on the papers worked on the visual deep learning features and addressed both image classification and retrieval, while three studies addressed the image classification only.

2.2 Textual-based Image Retrieval

2.2.1 Handcrafted Features

Yahia [13] recommended integrating both the content-based image retrieval (CBIR) techniques with the latent semantic indexing (LSI) approach to facilitate the indexing of historical Arabic manuscripts and eventually retrieving them. The used dataset was only two pre-scanned ancient Arabic manuscripts named: "Sahih Al-Bukhari" (صحيح البخاري) and "Mawaqeet Al-Haj wa Al-Umra" (مواقيت الحج والعمرة). The author preprocessed the collected dataset through two main operations, which are: binarization and smoothing. Note that binarization involves converting colored images into greyscale images and then, into binary images, as illustrated in figure 2.3.

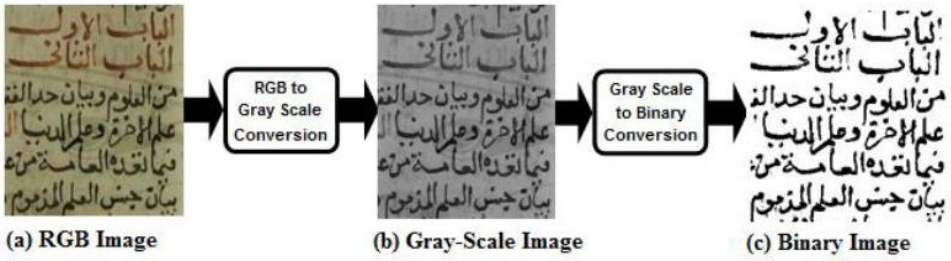


Figure 2.3: Image binarization steps, referenced from [13]

On the other hand, the smoothing algorithm involves removing any noise presented in the manuscripts and is not part of the main text. An example of a smoothing and noise removal algorithm is illustrated in figure 2.4.

Figure 2.4: Smoothing and noise removal, referenced from [13]

The author segmented preprocessed images into lines and segmented the lines further into words. Afterward, he constructed Latent Semantic Indexing (LSI) by computing the values of four local features as following: 1) concentric circle features, 2) angular line features, 3) rectangular region features, and 4) circular polar grid features. All of the four features were represented as numerical values to simplify implementing the LSI on them. Then, the author measured the similarities utilizing the Singular Value Decomposition (SVD) algorithm, which converts the document into a matrix to be able to compute the difference between the input document and the vector.

Finally, the author calculated the precision and recalled to evaluate the performance of his proposed model. For the testing purpose, the author implemented the system using Matlab and stored his information using Microsoft Excel. The same two pre-scanned Arabic manuscripts were tested through pre-processing them, segmenting them into individual words (5500 extracted words). The author used 20 words for the query process. Using the four features sets, the author concludes that the most accurate set is the circular polar grid with 78.8% recall.

Al-Jawfi [32] designed a neural network that utilizes LeNet, which is a convolutional neural network (CNN) specialized in recognizing handwritten and machine-printed characters. Therefore, the author employed the LeNet to identify the handwritten Arabic letters successfully. He accomplished his goal through two main stages. First, he worked on recognizing the general shapes of the letters, and second, he worked on recognizing the dots of the letters. To simplify the process of recognizing handwritten Arabic words, the word first segmented into individual letters. Afterward, the backpropagation algorithm used to feedforward the multi-layers' neural network. The researcher trained the network using some training data and utilizing the gradient descent function. Furthermore, the author computed the similarity between entered Arabic letters and the letters stored in the dataset using the error signal function. This function measures the difference between the actual output and the desired target output. Hence, it based on the initial neurons' weights that are assigned in each individual layer. Finally, he concluded that there is a lack of recognizing ancient handwritten Arabic manuscripts and that there is also a need to handle the other non-textual parts included within the manuscripts' pages and process them as images instead of text.

El-Makhfi [33] proposed recognizing the written words within the Arabic manuscripts’ images through extracting the handcrafted features. The authors started by preprocessing the input query image by converting it into a greyscale version then, segmenting it into lines and words. The textual features are then extracted from the Arabic manuscripts’ images using the Speeded Up Robust Features (SURF) method. The similarity is then measured between the user entered a query image, and all the other images saved in the database by comparing the points of interest and computing the distance using the Euclidean or the Mahalanobis metrics. The authors evaluated

their proposed method using the HADARA80P dataset and recorded 95.27% recognition accuracy.

Ezz et al. [34] classified and predicted only two Arabic handwriting styles, which are Naskh and Reqaa. The authors employed static SIFT and SURF algorithms to do the features extraction from 200 images. Then, they experimented four different machine learning classifiers as following: gaussian naive bayes, decision tree, random forest, and the K-nearest neighbor. They concluded that the best method for predicting the Arabic handwriting styles is utilizing the SIFT with the gaussian naive bayes classifier since it recorded 92% accuracy.

Adam et al. [64] used ancient Arabic manuscripts for the purpose of discovering and testing a good algorithm for manuscripts' age and authors detection. They utilized KERTAS dataset, which is illustrated in figure 2.5.

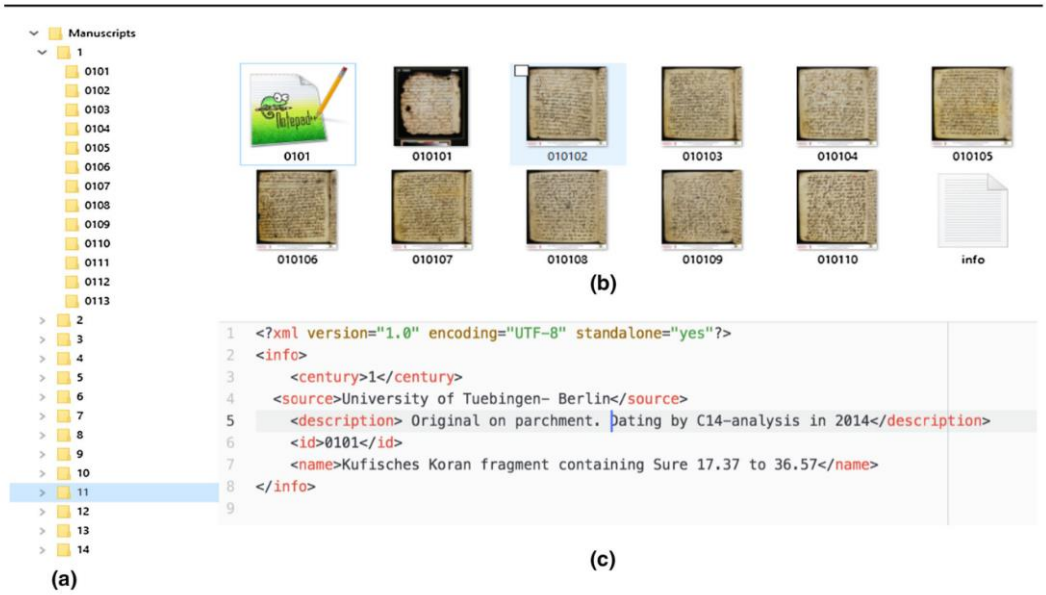


Figure 2.5: Sample of KERTAS dataset, referenced from [64]. (a) is directory structure, (b) is the content of 0101 manuscript, and (c) is the XML metadata file associated with the manuscript

The dataset consists of more than 2000 images of high-quality scanned ancient Arabic manuscripts. Note that, Qatar National Library is the primary ancestry of KERTAS dataset.

The authors used the complete images of KERTAS manuscripts without any cropping because they were interested in studying both the writing and the layout styles. However, they resized the images looking for the best size to discover manuscripts' features. Hence, they started with (12×12) pixels. Then, they increase the sizes to (25×25), (50×50), (100×100), (200×200) and till (250×250) pixels. The authors concluded that with reducing the size of the image, most of the features become unclear, which minimizes the chances to find the right matching manuscript successfully.

Similarly, increasing images size dramatically causes the same unclarity in visualizing the images' features. Eventually, they concluded that the most accurate size for visualizing images is (50×50) pixels. Afterward, the authors tackled the features extraction problem utilizing two techniques. First, is the sparse representation based technique, which uses normalization to choose the nearest sub-space of the manuscript being assisted. Second, is the handwriting style-based features. In order to measure the similarity between the queried image and the rest of the images in the database, the authors computed the K-nearest neighbor, with k=3. Eventually, they recorded the highest accuracy as 94.77% with predefined folds using the sparse representation-based manuscript detection algorithm and (50×50) images size.

Asi et al., [65] recommended a new algorithm for identifying the writer(s) of ancient Arabic manuscripts successfully. Their algorithm also includes determining the number of writers. Two datasets were used, which are: WAHD and KHATT. Note

that, the WAHD dataset consists of 353 manuscripts. While the KHATT dataset consists of around 1000 short manuscripts. The authors preprocessed the images of their utilized ancient Arabic manuscripts to improve their quality and, eventually, better visualize them for further analysis. Hence, the authors cropped the background of the scanner from original images and then segmented the main text, as illustrated in figure 2.6.

Figure 2.6: Preprocessed image from IHP dataset, referenced from [65]. (a) is the original image, (b) is the image after cropping background, and (c) is the image after segmenting main text

The authors recommended extracting the local features to be able to identify the manuscript writer successfully. Regarding the local features, which are the low-level features that represented through the curves and roundness of each manuscript hand-writings. They captured utilizing the "Modified Contour Based Feature (M-CBF)". Afterward, they used three classifications techniques. Then, based on the similarity measurement of the hand-writing style, the query image is classified into one of three

classifications techniques, which are: averaging, voting, or weighted voting. Noticing that the similarity measured using the cosine or the Chi-square distance metric. Finally, the authors evaluated their model using the "leave-one-out cross-validation" strategy. In this strategy, the authors randomly choose one manuscript for the query process. For the testing scenario, the authors tested all the manuscripts in KHATT dataset as it was relatively small. However, they also tested the IHP section from the WAHD dataset. Note that, the Islamic Heritage Project (IHP) section consists of 333 manuscripts written by 302 distinct writers. They eventually concluded that using the proposed algorithm would definitely reach accurate identification of manuscript writers recording 34.6% accuracy of top10 retrieved images using the M-CBF algorithm.

Dinges et al. [66] collected hand-written Arabic texts in a user-friendly system to use them for testing their new segmentation-based approach for handwritten Arabic text recognition. The authors collected around 50,000 famous Arabic words vocabularies in a dataset and called it "IESK-arDB SynWords". They preprocessed their manually collected dataset through segmenting it words into individual characters based on a set of rules and topological features. Afterward, they classified their segmented words into classes using either Support Vector Machines (SVMs) or using Active Shape Model (ASMs) classifiers to extract the features. Then, they used the levenstein distance for measuring the similarities between the query word and the stored vocabularies in the

synthesized dataset. The authors evaluated the correction of recognized words on two different levels. The word level and the character level. Note that for the word level, they inspected a collection of vocabularies for the tested word. On contrast for the character level, they computed the statistical distribution and correct errors using priori knowledge of the character. The authors used the same synthesized dataset to validate their system. They found out that their system can recognize Arabic words successfully even though, its operation is relatively slow. Thus, they recommend implementing the work using C/C++ instead of using Matlab functions. Moreover, they expect that implementing the system on GPU or FPGA would expedite the work. Eventually, they approximate from 0.1 to 0.2 seconds/word would process faster leveraging proposed implementation methods.

Al-Dmour and Fraij [67] suggested to segment the handwritten Arabic texts into lines using the Horizontal Projection Profile (HPP) technique and then segment the lines further into words using the Gap Metrics (GM) technique. They used Arabic Handwritten Data-Base (AHDB) to conduct their study. They preprocessed their AHDB dataset images through filtering the text images and binarizing them. Afterward, they investigated the best clustering algorithm to extract words correctly from the historical Arabic manuscripts. Hence, they used the word spotting technique utilizing the Fuzzy C-Means (FCM) algorithm, which is a distance based soft

clustering method. Then, they measured the similarity between extracted words by computing the spacing between them using the Euclidean distance metric.

The authors test four different clustering algorithms looking for the best method to extract words within historical Arabic manuscripts. The first two clustering algorithms are K-Means (KM) and Fuzzy C-Means (FCM), which are based on the distance. While, the third clustering algorithm is the Gaussian Mixture Model (GMM), which is based on the probability. The last tested clustering algorithm is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which is based on the density. The authors concluded that FCM is the best clustering algorithm because it recorded an 84.8% overall extraction rate.

Othman [68] proposed a model to recognize and analyze the Arabic text. He manually scanned and collected 120 ancient Arabic images to evaluate his model. He preprocessed the manually scanned Arabic images by converting the RGB-images into a greyscale version. Afterward, he converted the grey-scale images into binary-images using the adaptive binarization method. In addition, he removed the noise and all unrelated parts from the documents. In addition, he extracted the features from the ancient Arabic images utilizing the word spotting technique through the Bag of Word Fragments (BoWFs) method. Then, he measured the similarity among the query image and the rest of the images using two static formulas, which are the Histogram Intersection (HI) and the Earth Movers Distance (EMD). Finally, the author evaluated

his BoWFs word spotting technique through computing both the precision and recall of the resulted output images. He was able to reach 89.60% precision rate and 50% recall. However, the author said that the output results might be biased because the query image is being counted within the output results. While this weakness can be overcome through eliminating the query image from the output computation.

Snoussi et al. [69] recommended using the Outer Isothetic Cover (OIC) to segment the digital handwritten ancient Arabic manuscripts images into text lines. The dataset used is 100 handwritten Arabic images taken from form the KHATT Arabic text dataset. The (OIC) segmentation technique is based on Transparent Neural Network (TNN). Afterward, they evaluated their proposed method for segmenting handwritten Arabic text manually utilizing the "Test" subset within the KHATT dataset. They reached that their model recorded a 74% successful text extraction rate.

Al-Maadeed et al. [70] recommended a system for Arabic handwritten recognition that doesn't use the optical character recognition either the word spotting. Instead, they depended on an interface that performs a search using text inquiry to perform the retrieval process. Using the SQL server, the authors built their database of Arabic manuscripts. And for the testing purpose, the authors used the "Ibn Sina" database, which contains old Arabic manuscripts. They have also developed a web-based user-friendly interface using Microsoft Visual Studio and Asp.Net, as illustrated in figure 2.7.

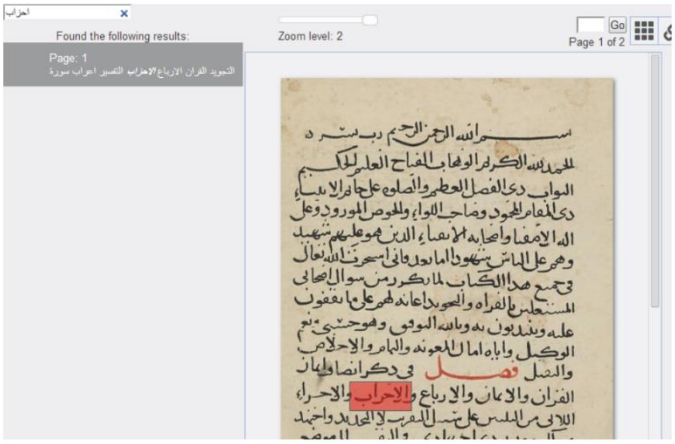


Figure 2.7: Manuscript retrieval using word search interface, referenced from [70]

Concerning the main purpose of the developed system, which is to search specific words in the ancient Arabic manuscripts, the authors utilized the "Solr" open-source platform from Apache, which simplifies searching text.

Finally, they believe that the segmentation phase in processing historical Arabic manuscripts is very challenging and that there is a significant need to convert manual written manuscripts into digital versions to simplify the process of searching and visualizing them.

Aghbari and Brook [71] introduced an approach for segmenting and classifying the ancient Arabic manuscripts. The authors used a hardcopy dataset called Historical Arabic Handwritten (HAH) manuscripts for their study by scanning them to convert them into digital copies that the Computer can process. Afterward, they preprocessed their used dataset through four steps as following: 1) binarization, 2) noise removal, 3) smoothing, and 4) thinning. These preprocessing steps improve the original poor-

quality presented in ancient Arabic manuscripts and simplify the rest of the classification stages. After preprocessing the manuscripts' images, the authors segmented them into words, and then, each word was segmented further into its connected parts. The features were then extracted from the connected parts by recognizing both the structural and statistical features and combine them in one textual feature vector. The generated feature vector entered as an input to a generalization Multi-Layer Perceptron (MLP) feed-forward neural network to extract the features from the segmented parts. Then, the authors measured the similarity utilizing the error signal function to compute the difference between the actual and the targeted output results. For the testing scenario, the authors used one historical Arabic manuscript named "كشف اللثام عن وجه الإسلام". There are 27 pages in the testing manuscript. The recorded average accomplished accuracy by the proposed MLP neural network is 89.3%.

Allaf and Al-Hmouz [72] predicted the Arabic calligraphy types utilizing an offline neural network recognition system. They used two different datasets for the classification and prediction task as following: local dataset that consists of three Arabic handwriting styles written by calligraphers and public dataset that consists of ten Arabic handwriting styles generated by the Computer. To pre-process the images of the dataset, the authors converted all the images into the binary version and removed the noise. Afterward, they extracted the visual features from the pre-processed images using the Genetic Algorithm. Finally, the authors reached a recognition error rate that equals 8.02% for the local dataset and 7.55% for the public dataset.

Bataineh et al. [73] proposed using a backpropagation neural network to predict the Arabic handwriting styles. Considering that the backpropagation neural network consists of one input layer, one hidden layer, and one output layer to perform the classification task. They started by pre-processing fourteen images, including seven handwriting styles, by converting them into binary versions and removing their edges and skews. Afterward, they extracted the features using Edges Direction Matrices (EDMS), which is a statistical algorithm for interpreting the texture features in images. The authors eventually accomplished 43.7% recognition accuracy.

Yu-Sheng et al. [74] classified and predicted five Chinese handwriting styles included within a dataset of 2000 images. Each image in the dataset includes only one Chinese character. The authors experimented four various machine learning algorithms looking for the algorithm that best classifies their dataset images. The experimented algorithms are Softmax regression, support vector machine, K-nearest neighbors, and random forests. Moreover, they tuned the learning hyperparameters using the k-fold cross-validation method. The authors reached that using the HOG descriptor with the Softmax regression algorithm outperforms other algorithms since it recorded 95.55% accuracy. Table 2.3 summarizes the papers worked on the image retrieval utilizing the textual handcrafted features. We notice that most of the cells in the table are highlighted using the blue color, which indicates our concentration on the studies used in the Arabic language.

2.2.2 Deep Learning Features

Peng et al. [75] proposed classifying images using RNN to have the ability to learn the current hashing function putting into account the error generated from previous hashing function. The authors used three common images datasets as follows:

Table 2.3: Textual-based handcrafted features

Reference# (Year)	Problem Domain	Approach	Dataset	Similarity Measure	Results
** [13] (2011)	Retrieve Arabic manuscripts' images	Word spotting through LSI	Sahih Al-Bukhari and Mawaqeet Al-Haj wa Al-Umra	*SF: Singular Value Decomposition algorithm	• Recall: 78.8% using the circular polar grid features set
[32](2009)	Recognize handwritten Arabic letters	LeNet CNN	758 segmented Arabic characters	*ML: Error signal function	There is a need to handle the non-textual parts within manuscripts as images not as text.
** [33] (2019)	Retrieve Arabic manuscripts' images	Arabic words spotting using SURF	HADARA80P dataset	* SF: Euclidean or the Mahalanobis distance metrics.	95.27% recognition accuracy
[34] (2019)	Predict two Arabic calligraphies	SIFT algorithm	Two historical Islamic Arabic books	*ML: Gaussian Naive Bayes (GNB)	92% recognition accuracy
** [64] (2018)	Retrieve Arabic manuscripts' images	Run length, edge direction, and edge hinge	KERTAS	*ML: K-nearest neighbor	94.77% accuracy with predefined folds and 42.31% accuracy with random train/test split using (50×50) size.
** [65] (2017)	Retrieve Arabic manuscripts' images	Modified contour-based feature and local key point descriptors	IHP and KHATT ancient Arabic manuscripts	*SF: Cosine or Chi-square distance metric	88.9% and 73% classification accuracy using KHATT and IHP datasets respectively. 34.6% retrieval accuracy of the top10 images using M-CBF.
[66] (2016)	Arabic texts recognition	SVMs or ASMs	IESK-arDB SynWords manuall collected dataset	*SF: Levenstein distance	• Recall: 65.1279% • Precision: 64.716%
[67] (2014)	Arabic texts extraction	Word spotting through FCM	AHDB database	*SF: Euclidean distance	• Extraction rate: 84.8%
** [68] (2015)	Retrieve Arabic manuscripts' images	Word spotting through BoWFs	Manually scan and collect 120 ancient Arabic images	*SF: Histogram intersection and Earth movers distance	• Recall: 50% • Precision: 89.60%

Table 2.3: Textual-based handcrafted features (Continued)

[69] (2016)	Arabic texts extraction	OIC transparent neural network	100 images from the KHATT Arabic dataset	*SF: Compute the upper and lower limits of diacritic point	• Extraction rate: 74%
** [70] (2017)	Retrieve Arabic manuscripts' images using text search	Optical Shape Recognition (OSR)	Ibn Sina" database	*SF: Index pages in XML file	NA
[71](2009)	Classify Arabic manuscripts' images	MLP feedforward neural network	HAH manuscripts	*ML: Error signal function	89.3% average accuracy.
[72] (2016)	Predict Arabic calligraphies	Genetic algorithm	Local dataset written by calligraphers	*ML: Gaussian Mixture Models (GMMs)	8.02% recognition error rate
			Public dataset generated by the Computer		7.55% recognition error rate
[73] (2011)	Classify Arabic manuscripts' images	Edge direction Matrices (EDMS)	Selected Arabic images	NA	43.7% recognition accuracy
[74] (no date)	Classify Arabic manuscripts' images	HOG descriptor	Single character images	*ML: Softmax Regression	95.55% recognition accuracy

All the papers are highlighted using the blue color because they addressed the Arabic manuscripts' images classification and/or retrieval.

The ** written before the reference number is to highlight that the study addressed the retrieval of the Arabic images.

CIFAR10, NUS-WIDE, and MIRFlickr. The datasets include 60000, 270000, and 25000 images, respectively. The authors advised implementing hash maooing functions to project images into binary codes utilizing RNN to be able to sequentially track previously recorded errors, as well as, to be able to map each image to its similar binary code. Finally, the authors utilized the Pytorch deep learning package to evaluate their framework. They computed the mean average precision for the three tested datasets. They reached that their model recorded 0.842 of mAP when tested on both CIFAR10 and NUS-WIDE datasets. While it recorded an average of 0.808 mAP when tested on the MIRFLICKR dataset.

Qian et al. [76] evaluated four deep learning models on two different datasets to identify authors. The used datasets are "Reuters_50_50" and "Gutenberg". While, the used deep learning models are "sentence-level GRU, article-level GRU, article-level LSTM, and article-level Siamese network". The authors measured the similarity among the articles using the Cosine static distance metric. Finally, they concluded that the best performing model was the article-level Siamese network because it recorded 99.8% accuracy on both Reuters and Gutenberg datasets.

You et al. [77] recommended using the bidirectional LSTM layer within their proposed tree-based AttentionXML deep learning model for text classification. They used six datasets for evaluating their model named: EUR-Lex, Wiki10-31K, AmazonCat-13K, Amazon-670K, Wiki-500K, and Amazon-3M. The authors measured the similarity among the classified text using the K- nearest neighbor algorithm. They computed the precision to evaluate their model, and the highest precision they reached equals 95.92% using the AmazonCat-13K dataset. Many studies handling the sequence classification of words to labels, tend to employ the "Attention" layer to improve the

performance of their classification models, such as the study done by Elnagar et al. [78]. The authors used two textual Arabic datasets named: SANAD and NADiA to experiment the effectiveness of deep learning models for Arabic text classification. They tested both the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN) models. They reached the highest classification accuracy as 96.94% when they added the attention layer after the RNN. Moreover, Liu and Guo [79] employed two attention layers after the main LSTM layer to attain the bidirectional approach. They concatenated both forward and backward representation of sentences to get the final comprehensive feature map. The authors evaluated their proposed model using seven different datasets named: MR, IMDB, SST-1, SST-2, Subj, RT-2K, and TREC. The highest accuracy they achieved equals 97.2%.

Du et al. [80] introduced a Convolutional Recurrent Attention Network (CRAN) for text classification. The authors used five datasets named: MR, SST-1, SST-2, Subj, and IMDB for evaluating their classification model. The highest recorded accuracy was 94.1% using the Subj dataset. Liu et al. [81] developed a bidirectional, RNN attention deep learning model for sentence classification. The final classification layer includes the Softmax activation function. The authors tested their model on eight benchmark datasets and recorded the highest accuracy as 94% using the TREC dataset. Yang et al. [82] proposed a Hierarchical Attention Network (HAN) for text classification. The author's model consists of two encoders, one for encoding the words and the second for encoding the sentences, as well as, two attention layers for focusing on both the words and the sentences. They experimented their model on six different datasets that were categorized as 80% for the training subset and 20% divided equally between both the testing and the validation subsets. The authors recorded the accuracy to evaluate their model and reached 75.8% using the Yahoo answer dataset. Gao et al.

[83] improved the text classification model that was initially proposed by [82] through adding a convolutional layer to it, so it became a Hierarchical Convolutional Attention Network (HCAN). The advantage of adding the convolutional layer is to generate the embeddings' matrices for updating the attention weights. The authors evaluated their model on four datasets named: Yelp Reviews 2016, Amazon Reviews Sentiment, Amazon Reviews Category, and Pubmed. They computed the accuracy, and the highest result they could achieve equals 89.9% using the Amazon Reviews Category dataset.

Saeed and Albakoor [84] used a neural network to segment lines, and words in both Arabic and English printed and handwritten texts. Their model uses a region growing based segmentation technique to recognize the words. The authors measured the similarity between the queried text and the retrieved output text using the Sigmoid function. Afterward, they evaluated their region growing based algorithm for segmenting text using the Matlab program and reached that their method performs better in handwritten Arabic text than in handwritten English text. That is because it recorded a 90% successful Arabic text recognition rate and an 80% successful English text recognition rate.

Bagnall [85] designed a multi-headed Recurrent Neural Network (RNN), which is a particular type of deep neural networks that execute sequential elements identically.

The author used the RNN to identify authors successfully utilizing the texts as inputs to his deep learning model. His task designed for the "PAN 2015" authors identification competition, and he was able to record higher than 80% average Area Under Curve (AUC).

He and Schomaker [86] experimented three methods for identifying the writers of images as following: baseline, linear adaptive, and deep adaptive learning methods.

They utilized images, including one single handwritten word. The images are taken from two freely available datasets named CVL and IAM. The researchers trained their convolutional neural network employing the “Tensorflow” deep learning library and NVIDIA GPU GTX 960. They concluded that the deep adaptive learning algorithm is the best algorithm for writers’ identification. That is because it recorded 78.6% top-1 and 93.7% top-5 recognition rates using the CVL dataset. In addition, it recorded 96.5% top-1 and 86.1% top-5 recognition rates using the IAM dataset.

Chen et al. [87] proposed recognizing and classifying multi-labeled images using the recurrent attentional reinforcement learning technique. The problem they are trying to solve is considered a sequential decision task since the images are including more than one semantic label. They started their work by localizing specific regions in the input images to the model using the deep VGG16 CNN. Then, they developed an attentional LSTM deep learning model that takes the extracted features from the images’ regions, as well as the preceding iterations’ hidden states as inputs to recognize the semantics of the images and use them in the classification process.

The entire models developed in a deep reinforcement fashion that considers the actions as classifying the localized attentional region of images. The states are chosen according to the preceding iterations ’details, and the features extracted from the images’ regions using the deep VGG16 CNN. The rewards are successfully classified as multi-labeled images. The authors used two datasets named: “PASCAL VOC” and “MSCOCO” to evaluate their proposed method. They computed the mean Average Precision (mAP) and recorded 92.0% using the PASCAL VOC dataset, and 71.1% F-score using the MS COCO dataset.

Table 2.4 summarizes the papers worked on the textual deep learning features.

Table 2.4: Textual-based deep learning features

Reference# (Year)	Problem Domain	Approach	Dataset	Similarity Measure	Results
[75] (2018)	Classify general images using their textual labels	RNN deep reinforcement learning model	CIFAR10, NUS-WIDE, and MIRFlickr	*SF: Euclidean	84% mAP using both CIFAR10 and NUS-WIDE. 81% mAP using MIRFLICKR dataset.
[76] (2017)	Authorship Identification	Article-level Siamese network	Reuters_50_50 and Gutenberg	*SF: Cosine	99.8% accuracy on both Reuters and Gutenberg datasets.
[77] (2019)	Classify English text	Tree-based AttentionXML deep learning model	EUR-Lex, Wiki10-31K, AmazonCat-13K, Amazon-670K, Wiki-500K, and Amazon-3M	*ML: K-NN	95.92% highest recorded precision using the AmazonCat-13K dataset.
[78] (2020)	Classify Arabic text	Attention-GRU	SANAD and NADiA Arabic datasets	*DL: Sigmoid + binary crossentropy	96.94% accuracy.
[79] (2019)	Classify English text	BiLSTM with attention and convolutional layers	MR, IMDB, SST-1, SST-2, Subj, RT-2K, and TREC	*DL: TanH	97.2% accuracy.
[80] (2018)	Classify English text	CRAN	MR, SST-1, SST-2, Subj, and IMDB	*DL: Softmax	94.1% highest accuracy using Subj dataset.
[81] (2018)	Classify English text	Recurrent networks with attention and convolutional networks	MR, Subj, SST-1, SST-2, IMDB, TREC, CR, and MPQA datasets	*DL: Softmax	94% highest accuracy using TREC dataset.
[82] (2016)	Classify English text	HAN	Yelp 2013-2015, IMDB, Yahoo answers, and Amazon	*DL: Softmax	75.8% highest accuracy using Yahoo answer dataset.

Table 2.4: Textual-based deep learning features (Continued)

[83] (2018)	Classify English text	HCAN	Yelp 2016, Amazon Sentiment, Amazon Reviews, and Pubmed	*DL: Softmax	89.9% highest accuracy using the Amazon Reviews dataset.
[84] (2009)	Arabic text recognition	Region growing based neural network	Random chosen Arabic text	*DL: Sigmoid function	Arabic text recognition rate: 90%
	English text recognition		Random chosen English text		English text recognition rate: 80%
[85] (2015)	Authorship Identification	Multi-headed Recurrent Neural Network (RNN)	PAN 2014	*DL: Softmax	Higher than 80% AUC.
[86] (2018)	Writer Identification based on single handwritten word images	The authors tested three methods: 1) baseline, 2) linear adaptive, and 3) deep adaptive learning	CVL and IAM datasets	*DL: Sigmoid function	The deep adaptive learning was the best method recording 78.6% and 69.5% top-1, as well as, 93.7% and 86.1% top-5 recognition rates using the CVL and IAM datasets respectively.
[87] (2017)	Classify general images using their textual labels	Recurrent attentional deep reinforcement learning	PASCAL VOC and MSCOCO	*DL: hybrid loss function	92.0% mAP using the PASCAL VOC dataset, and 71.1% F-score using the MSCOCO dataset.

The papers highlighted using the blue color addressed the Arabic text recognition and classification.

2.3 Fusion-based Image Retrieval

Xia et al. [3] recommended using a transformation-invariant deep hashing model for classifying and retrieving trademark images. They begin by augmenting the dataset's images. Then, they extracted the visual features from the images using two deep learning models named: Spatial Transformer Network (STN) and Recurrent Convolutional Network (RCN). Afterward, they fused the visual extracted features using a hashing layer to generate the binary codes of the images. The similarity between the query image and the rest of the images stored in the dataset computed utilizing the Hamming distance metric. The authors evaluated their model by calculating the mAP of the retrieved images, and they were able to record 0.449 mAP using the "NPU-TM" dataset, and 0.501 mAP using the "METU" dataset.

Potrus et al. [88] proposed a hybrid handwritten Arabic text recognition technique. They leveraged a manually collected dataset consisting of 4500 Arabic words (24,960 individual characters) to conduct the study. Moreover, they use another dataset that contains 7851 Arabic words known as (ADAB) dataset to evaluate their model. They preprocessed the written text by removing its noise and all unrelated non-textual parts included within the images. In addition, they utilized the Bezier cubic curve approximation algorithm for smoothing the characters' shape and filling the gaps between characters' points. After preprocessing the images, the authors segmented and recognized the Arabic letters online using an integrated Genetic Algorithm (GA) with a Harmony Search algorithm (HS). First, the dataset is segmented using a dominant point detection, and afterward, the segmented text are recognized using the GA-HS fused model. The authors use the harmony search character algorithm to be able to measure the similarity among the searched Arabic characters and the characters stored in the dataset.

The algorithm examines the directions of the characters looking for the best match with the queried one. It uses Pitch Adjustment Rate (PAR) value to decide wither to pick the characters on not. Finally, the authors evaluate their proposed GA-HS fusion model for online Arabic characters recognition using two datasets. The model recorded a 93.6% successful recognition rate when tested on the first dataset, which is collected manually. While the second dataset was called ADAB, the model recorded 94.68%-96.33% successful recognition rate when tested on the ADAB dataset. However, the authors claim that the search process takes very long to find-out the relevant characters because it uses the harmony search (HS) algorithm, which is based on the stochastic process.

Wang et al. [89] recommended a fusion between CNN and the RNN to accurately classify and retrieve multiple-label images. The use of RNN empowered the framework because of its ability to memorize the dependency occurrence of labels. They computed the similarity measurement using the greedy algorithm, but it didn't perform well due to its dependency on the initially predicted labels. Thus, they used the beam search algorithm to be able to predict the nearest labels in an image successfully. For the evaluation purpose, the authors used the Caffe deep learning framework to experiment with their CNN-RNN fusion model on three datasets. They calculated the mAP and reached 84%.

Sharif et al. [90] integrated both the Scale-Invariant Feature Transform (SIFT) local features descriptor with the (BRISK) features descriptor to retrieve similar images. They measured the similarities among retrieved images utilizing the Histogram

intersection distance metric and evaluated their fusion model by calculating the mAP on different image datasets. They reached 78.14% mAP on the “Corel-1.5K” dataset and 57.37% mAP on the “Corel-5K” dataset.

Guo et al. [91] proposed a CRAN hybrid CNN-RNN deep learning model with an attention layer for improved text classification. The model evaluated in both English and Chinese languages. In addition, it assessed on Chemistry, Physics, and Mathematics subjects. The highest F-Score the authors recorded equals 86.19% using the Chinese language dataset. Koesdwiady et al. [92] utilized the decision-level fusion model to investigate the relationship between traffic flow and weather conditions. They used the “San Francisco Bay Area” traffic and weather data to evaluate their fusion model. Initially, they extracted the weather features using the Deep Belief Network (DBN) to predict the traffic flow according to the extracted features. Then, they computed the Mean Average Error (MAE) and recorded 0.0487 using the traffic data only, 0.2192 using the weather data only, and 0.0405 using both fused predictions.

Li et al. [93] recommended increasing the accuracy of face recognition through a decision-level fusion model. The authors begin by extracting the features of faces’ images using the deep CNN and using the traditional 2-Dimensional Principal Component Analysis (2DPCA) method. Then they measured the similarity scores using the Euclidean distance metric for the 2DPCA and using the Mahalanobis distance metric for the CNN method and fused the similarity scores at the decision level. They used the “LFW” faces datasets to evaluate their fusion-model and reached 91.98% accuracy on rank-1.

Su et al. [94] employed two deep CNNs named LMCNet and MCNet to classify the environmental sounds. The developed two deep neural networks consist of four convolutional layers followed by the final classification dense layer. The authors fused

the “Softmax” layers at the decision-level and evaluated the classification accuracy of their models using the “UrbanSound8K” auditory dataset. The recorded accuracy using the LMCNet equals 95.2%, while the recorded accuracy using the MCNet equals 95.3%. On the other hand, the recorded accuracy using both fused models equal 97.2%, which is higher than the classification accuracy by each separate deep CNN.

Xie et al. [95] proposed fusing three extracted features at the decision-level to improve the classification accuracy of the lung nodules. They used the “LIDC-IDRI” dataset to experiment their proposed model. The authors begin by extracting the texture features from the dataset images using the Gray Level Co-occurrence Matrix (GLCM), as well as, extracting the shape features using the Fourier method, and finally extracting the visual features automatically from the lung medical images using the Le-Net-5 deep CNN. To assess the performance of the model, the authors computed the Area Under Curve (AUC) and accomplished 96.65%.

Liu et al. [96] developed an image retrieval system based on fusing both the low-level features extracted using the Dot-Diffused Block Truncation Coding (DDBTC) method, with the high-level features extracted using the GoogLeNet deep CNN. The authors computed the Average Precision Rate (APR) using ten different images’ datasets to evaluate the performance of their proposed features-level fusion model. They recorded 98.08% as the highest reached APR using the “USPTex” dataset.

Sudha and Ramakrishna [97] compared between the different features-fusion methods found in Matlab programming application and named: left-right, right-left, down-up, and up-down features fusion. The authors begin by pre-processing the iris images within the “CASIA” dataset. Afterward, they extracted the features from the pre-processed images using six different features extraction techniques called: Discrete Wavelet Transform (DWT), Local Binary Pattern (LBP), Gabor filters, Principal

Component Analysis (PCA), Support Vector Machine (SVM), and Fast Fourier Transform (FFT). Finally, the authors calculate the accuracy of each features-fusion method using various combinations from the six features extraction techniques. The authors concluded that the down-up features-level fusion method is outperforming the other methods recording 94.32% success rate using the features fused from the LBP-FFT and the LBP-DWT techniques.

Suk et al. [98] recommended fusing the features extracted from Alzheimer's disease images existed within the “ADNI” dataset to increase the accuracy of the disease diagnostic. Initially, the authors extracted the high-level features from both the Magnetic Resonance Imaging (MRI) and the Positron Emission Tomography (PET) images utilizing the Deep Boltzmann Machine (DBM). Then, they fused the extracted features from both images and computed the disease recognition accuracy. The highest accuracy they reached equals 95.35% successful recognition of Alzheimer's disease.

Zhang et al. [99] proposed a hybrid fusion model of both visual and auditory extracted features for improving the recognition of emotions in videos. The authors used the pre-trained AlexNet CNN to extract the visual features, and the 3D-CNN to extract the audio features from three different datasets named: RML, eNTERFACE05, and BAUM-1. Afterward, the two extracted features were fused using Deep Belief Networks (DBN). Then the two DBN networks were fused again using the SVM for classifying the emotions in the videos. The highest recorded accuracy by the hybrid proposed model equals 85.97% using the “eNTERFACE05” dataset.

Vishi and Mavroeidis [100] evaluated four different score-level fusion methods called: Minimum-Score (MinS), Maximum-Score (MaxS), Simple-Sum (SS), and User-Weighting (UW) for classifying the biometric features. The authors extracted the features using both the fingerprints and the finger-veins techniques from the

“SDUMLA-HMT” dataset. Then, they experimented the four score-level fusion methods combined with three various score normalization approaches named: Min-Max (MM), Z-Score (ZS), and Hyperbolic Tangent (TanH). Eventually, the authors concluded that the use of the SS score-level fusion method in conjunction with the TanH score normalization approach outperformed other methods recording 99.98% successful classification.

Lip and Ramli [101] experimented the fusion of the humans' visual images with their speech biometrics using the decision-level, features-level, and score-level fusion methods. The authors begin by extracting the visual features using the Region of Interest (ROI) and extracting the audio features using the Mel Frequency Ceptral Coefficient (MFCC) method. Afterward, they employed the SVM to classify the extracted features. The authors used the Audio-Visual digitized database, which includes the digital versions of both the speech and the images to evaluate their fusion methods. Eventually, they reached that the score-level fusion is outperforming the other two methods recording 99.95% accuracy, followed by the features-level fusion recording 99.75% accuracy, and finally, the decision-level fusion recording 99.66% accuracy.

Kaya et al. [102] fused the audio and visual features extracted from video inputs using a weighted score-level fusion model. The pre-trained VGG-Face deep learning model was used to extract the visual features, while the openSMILE model was used to extract the auditory features. The authors evaluated their fusion model using two videos datasets named: Extended Cohn‑Kanade (CK+) and MMI. They accomplished 98.47% accuracy using the “CK+” dataset, and 72.46% using the “MMI” dataset.

Kang et al. [103] proposed a score-level fusion model for face verification. The authors begin by extracting the visual features from the “LFW” datasets images using the

Multi-scale Convolution Layer Blocks (MCLBs). Afterward, they improved the recognition accuracy by using the high-dimensional LBP method for the extraction of the visual features as well. Then, both extracted features joint using the SVM classifier, and the scores fused at the score-level, reaching 99.08% overall accuracy.

Bhushan and Danti [104] recommended a score-level fusion model for text classification. The authors initially pre-processed the text through stemming it utilizing four English language textual datasets named: Vehicle Wikipedia, Google newsgroup, 20 Mini newsgroup, and 20 Newsgroup. Then, they fused the generated scores from both the interval-valued classifier and the deep network classifier. The authors experimented the score-level fusion after categorizing the dataset into 60% training and 40% testing and validation, as well as after categorizing the dataset into 40% training and 60% testing and validation. The highest recorded F-score was using the "Vehicle Wikipedia" textual dataset with the 60% training subset as 0.9773 successful text classification.

George and Routray [105] experimented identifying persons through extracting the statistical features from their eye movements. The used dataset consists of two main subsets. The first subset named (RAN), it contains white dot moving randomly on a black background. While the second subset named (TEX), it contains a white dot following a text written on the screen. Both RAN and TEX subsets were divided further into two versions: one version has only 30 minutes interval between the training and the testing data, denoted as (RAN_30 and TEX_30). While, the other version has a 1-year interval between the training and the testing data, denoted as (RAN_1yr and TEX_1yr). The authors begin by dividing the eye movement images into fixations and saccades. Afterward, they removed the noise from the eye movement images and extracted their features using the Gaussian Radial Basis Function Network (GRBFN).

The highest recorded accuracy was using the RAN_30 and TEX_30 on the evaluation subset as 98.69% and 98.04%, respectively.

Jhansi and Reddy [106] applied a score-level fusion model for retrieving sketched-based images. They used the "TU Berlin Sketch" dataset to assess their fusion model. Initially, they extracted the visual features from the images using the Histogram Of Gradient (HOG) descriptor. Then, they measured the distances among images using the Euclidean distance metric and fused the generated score distances from the HOG with the generated scores from the Gaussian Mixture Model (GMM). The authors computed the accuracy and reached 90% successful retrieval.

Jovic et al. [107] extracted the color, shape, and texture hand-crafted features from four image datasets. Afterward, they measured the distances among the query image's feature vector and all the dataset images' features vectors using the Euclidean distance metric to generate three similarity score lists. The final image retrieval is performed using one final fused and ranked similarity score list from all the three generated similarity scores lists. Three various methods tested to do the similarity scores fusion named: Inverse Rank Position (IRP), Borda Count (BC), and Leave Out (LO). The authors evaluated their fusion methods through computing Average Retrieval Precision (ARP). Finally, they concluded that the IRP is performing better than the other two similarity scores fusion methods reaching 49.04%, 39.13%, 73.89%, and 41.23% ARP using the C-1000-A, C-1000-B, B-1776, and V-668 datasets, respectively.

Xue et al. [108] designed a fusion model that concatenates the features extracted from medical laboratory reports. The reports are text-based images that contain Chinese and Latin characters with numbers and mathematical symbols. The fusion model is based

on a bidirectional LSTM model with CNN. The authors measured the distance among the extracted text using the Edit distance and recorded 98.6% precision.

Schaetti [109] participated in the "PAN 2017" competition and utilized a deep learning model that is using CNN for authors profiling. His model was confusion between deep learning and Term-Frequency-Inverse Document Frequency (TFIDF) model. The researcher experimented with the confusion model in many different languages; the Arabic language was one of them. The used CNN deep learning model leverages both "ReLU" and "Softmax" activation functions at the last two dense layers. To evaluate the author's model, he collected four tweet collections from the Twitter application. After assessing his model, he reached a final accuracy equals to 64% for Arabic language authors identification.

Al-Muzaini et al. [110] merged two deep learning models, which are CNN and RNN, using a feed-forward layer to caption the input images with the Arabic description. The authors used images from both MS COCO and Flickr8K datasets to train their model on general images. Then, they used existing Arabic captions from both ImageNet and Al-Jazeera news website to train the model to label the images with the used sentences. The Prague Arabic Dependency Treebank (PADT) algorithm used to connect the relevance words and form a complete caption for the input image. The authors evaluated their model using the Flickr616 dataset, and they computed the BLEU score, which recorded 46.2. The BLEU score stands for (bilingual evaluation understudy). This score is used to evaluate the performance of a model that employs translating one language to another different language.

Table 2.5 summarizes the papers worked on the fusion-based image retrieval.

Table 2.5: Fusion-based image retrieval

Reference# (Year)	Problem Domain	Approach	Dataset	Similarity Measure	Results
[3] (2019)	Classify and retrieve trademark images	STN and RCN deep learning models	NPU-TM and METU	*SF: Hamming Distance	44.9% and 50.1% mAP using NPU-TM and METU datasets, respectively.
[88] (2014)	Arabic characters recognition	GA-HS fused model	Manually collected dataset consisting of 4500 Arabic words and ADAB dataset	*SF: Harmony search character algorithm	• Recognition rate: 93.6% using the manual collected dataset. • Recognition rate: 94.68%-96.33% using ADAB Arabic dataset.
[89] (2016)	Classify and retrieve multiple-label images	CNN-RNN fusion model	NUS-WIDE, Microsoft COCO, and PASCALVOC2007	*ML: Beam search algorithm	84% mAP.
[90] (2018)	Retrieve general images	SIFT-BRISK fusion descriptor	Corel-1.5K and Corel-5K dataset	*SF: Histogram intersection	78.14% and 57.37% mAP using Corel-1.5K and Corel-5K, respectively.
[91] (2018)	Classify English text	Hybrid CNN-RNN attention-based model	Chemistry, Physics, and Mathematics subjects	*DL: Sigmoid	86.19% F-score.
[92] (2016)	Investigate the relationship between traffic flow and weather conditions	Deep Belief Network (DBN)	San Francisco Bay Area traffic and weather data	*ML: Spearman's rank correlation coefficient	0.0405 MAE.
[93] (2018)	Retrieve humans faces images	Deep CNN-2DPCA model	LFW datasets	*SF: Euclidean and Mahalanobis distance metric	91.98% accuracy on rank-1.
[94] (2019)	Classify auditory data	LMCNet and MCNet	UrbanSound8K auditory dataset	*DL: Softmax	97.2% accuracy.
[95] (2017)	Classify medical images	GLCM, Fourier, and Le-Net-5	LIDC-IDRI dataset	*ML: Fourier transformer	96.65% area under curve.
[96] (2017)	Retrieve general images	DDBTC and GoogLeNet	Corel 1 and 10, Brodatz, Vistex, ALOT, Holidays, USPTex, Outex, KTH-TIPS, and UKbench	*ML: K-NN	**98.08%** APR using the USPTex dataset.
[97] (2017)	Classify iris images	LBP-FFT and LBP-DWT features fusion	CASIA dataset	NA	94.32% accuracy.
[98] (2014)	Classify medical images	Deep Boltzmann Machine (DBM)	ADNI dataset	*DL: Softmax	95.35% accuracy.
[99] (2017)	Classify video images	AlexNet-3D-CNN deep belief networks	RML, eNTERFACE05, and BAUM-1	*DL: Softmax	85.97% highest accuracy using the eNTERFACE05 dataset.
[100] (2017)	Classify biometric images	SS score-level fusion with TanH score normalization	SDUMLA-HMT dataset	NA	**99.98%** accuracy.
[101] (2012)	Classify humans' images	MFCC-ROI features with SVM classifier	Audio-Visual digit database	NA	**99.95%, 99.75%, and 99.66%** accuracy using the score-level, features-level, and decision-level fusion method, respectively.

Table 2.5: Fusion-based image retrieval (Continued)

[102] (2017)	Classify video images	VGG-Face and openSMILE	Extended Cohn–Kanade (CK+) and MMI datasets	*DL: Softmax	**98.47%** accuracy using the CK+ dataset and 72.46% using the MMI dataset.
[103] (2017)	Classify humans faces images	MCLBs and high-dimensional LBP	LFW datasets	*DL: Softmax	**99.08%** accuracy.
[104] (2017)	Classify English text	SVSM and Unigram word representation fusion model	Vehicle Wikipedia, Google newsgroup, 20 Mini newsgroup, and 20 Newsgroup	*DL: Softmax	97.73%, 97.52%, 96.81%, and 96.48% F-score using Vehicle Wikipedia, Google newsgroup, 20 Mini newsgroup, and 20 Newsgroup, respectively.
[105] (2016)	Classify iris images	Score-level fusion of extracted fixations and saccades features	RAN and TEX subsets from BioEye 2015 database	*SF: Euclidean	**98.69%** and **98.04%** accuracy using the RAN_30 and TEX_30 subsets, respectively.
[106] (2015)	Retrieve sketch-based images	HOG and GMM score-level fusion model	TU Berlin sketch dataset	*SF: Euclidean	90% retrieval accuracy.
[107] (2006)	Retrieve general images	Similarity score fusion using the IRP, BC, and LO fusion methods	C-1000-A, C-1000-B, B-1776, and V-668 datasets	*SF: Euclidean	49.04%, 39.13%, 73.89%, and 41.23% ARP using the C-1000-A, C-1000-B, B-1776, and V-668 datasets, respectively.
[108] (2020)	Classify Chinese text-based images	CNN+BLSTM concatenation	Chinese Medical Documents Dataset (CMDD)	*SF: Edit distance	**98.6%** precision of text detection.
[109] (2017)	Authorship Identification	Confusion between deep learning and (TFIDF) model	Four tweet collections from Twitter	*DL: Softmax	64% Arabic authors identification accuracy
[110] (2018)	Retrieve the Arabic caption of the query image	CNN-RNN merge model	Flickr616 datasets	*DL: Softmax	46.2 for the BLEU-1 score

The papers highlighted using the blue color addressed the Arabic text recognition and classification.

2.4 Summary of the Literature

From the presented literature, we notice that the papers worked on the visual features used datasets, including general images such as animal pictures, persons' faces, flowers, fruits, cars, etc. On the other hand, the studies worked on the textual features included textual contents such as news, tweets, scientific books, medical reports, etc. Some of the studies worked on the textual features used the word spotting techniques. By analyzing the generated evaluation parameters, we found that the papers worked on the image retrieval system utilizing either the deep learning techniques or the reinforcement learning techniques recorded higher results than the papers that worked on the image retrieval system utilizing the handcrafted features. Moreover, we found that the fusion models included substantial details and more information about the fused data, which increased their accuracy than each individual used model. Therefore, there is a significant need to explore the ability of the fusion deep learning technology to retrieve images successfully.

Reviewing the papers worked on Arabic image retrieval, we notice that little attention given to the Arabic manuscripts' retrieval. While more efforts accomplished in recognizing and classifying the Arabic texts.

The input data to the Arabic image retrieval system could be a complete text-image, a sub-word image, or even images containing one Arabic character. In addition, we notice that there are various criteria for classifying the Arabic manuscripts' images, such as classify the images according to the title, author, genre, handwriting-style, content, etc. Thus, we can classify the ancient Arabic manuscripts using more than one criterion.

We found that the achievements in the deep reinforcement learning (DRL) technique are few, and none of the papers used the DRL solved the Arabic manuscript images

retrieval. As well as, none of the papers used the deep learning fusion technique to solve the Arabic manuscript images retrieval.

Thus, there is a lack in the image retrieval using the Arabic manuscripts in particular, which need more effort and considerations. There is a need to investigate more on the retrieval of the complete Arabic manuscripts' images according to different search criteria. Therefore, we plan to employ the deep learning features at both the supervised learning level and at the reinforcement learning level to retrieve the images of the Arabic manuscripts according to three different criteria, which are the manuscript, author, and calligraphy.

Chapter III

The Dataset Collection

This chapter discusses the dataset collection and classification. It highlights the significance of classifying the dataset according to different criteria. Then, it lists the classified images. Finally, it discusses the challenges encountered during the collection, along with the remedies that we followed to overcome them.

Many online libraries facilitated access to the historical Arabic manuscripts, such as the "wqf"[1] Islamic Arabic manuscripts site, "Al-Madinah Al-Munawarah Research and Studies Center"[2], "Al-Eskandareah Manuscripts Center"[3], "KAU Scientific Platform"[4], etc.

The libraries save the manuscripts through converting them from the manual handwritten form into a digital form that is easy to search and retrieve electronically. Thus, the text written inside the images are saved as pictures instead of textual contents, and this is considered a problem. In addition, there is no one digital library,

[1] https://wqf.me/

[2] http://www.mrsc.org.sa/public/doc_category/2

[3] https://www.bibalex.org/ar/center/details/manuscriptscenter?keywords=

[4] https://kaupp.sa/Browse/Subject/Books

including all the ancient Arabic manuscripts. Due to the lack of an existing and freely available historical Arabic manuscripts, and to be able to conduct the research study, we had to collect the dataset manually. The survey on data collection for machine learning [111] guided us through preparing our dataset. Thereby, we started by arbitrary collecting the required ancient Arabic manuscripts using the strategy illustrated in figure 3.1.

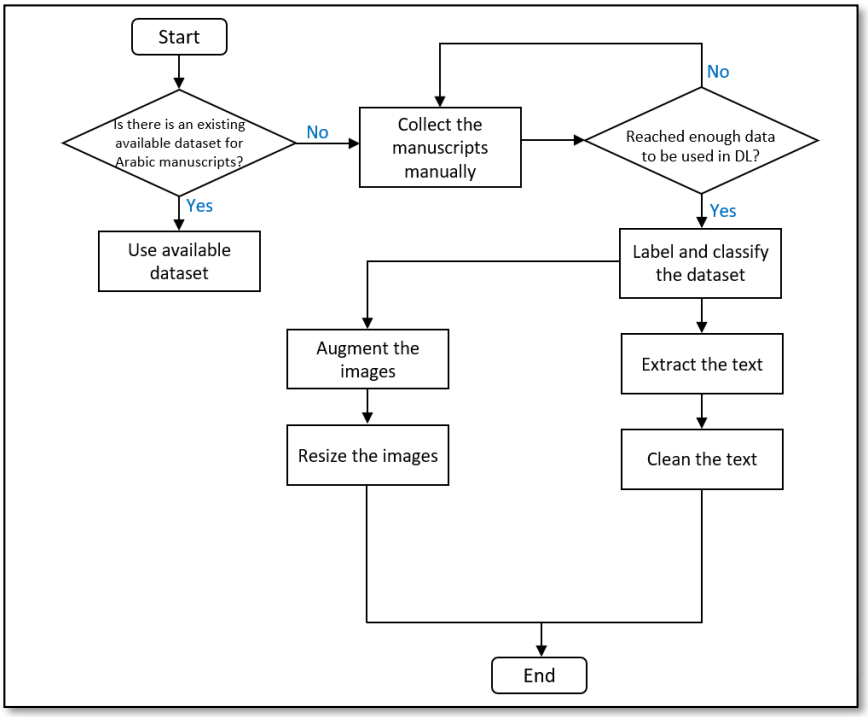


Figure 3.1: Flowchart of dataset collection and preparation

The dataset collected from the "wqf"[5] online website. We gathered a total of (8638) images. Then, we labeled them according to three search criteria, which are the manuscript, author, and calligraphy.

[5] http://wqf.me/?p=15619

The proposed image retrieval system accepts a complete ancient Arabic manuscript image as the query input and retrieve the ranked most similar images to the query image. This comparison between the images requires both the visual and the textual features. Thus, after collecting a sufficient number of the Arabic manuscripts' images, we prepared the textual contents through extracting and cleaning the text extracted from the images. The text recognition and preprocessing are explained in detail in the following chapter.

Moreover, we prepared the visual images by augmenting and resizing them. By analyzing the dataset images, we found that all of them are having (RGB) color representation. Furthermore, we found that most of the images are having (2160) pixels for the width dimension and (1440) for the height dimension, which is like a book dimension. However, there are many other images of different sizes. These images might be indices, appendices, or cover pages since they appear mostly either in the beginning or at the ending of the manuscript.

The number of images within each manuscript is ranging from (6) to (381) colored images, which means that the dataset is unbalanced.

3.1 Significance of the Retrieval Criteria

The automatic search and instantaneous retrieval of the most relevant images to a user query image is a crucial system in various domains. The retrieval can be according to different criteria. For instance, a user might be interested in retrieving the images from the same manuscript. Another query might be to retrieve images written by the same author or using the same calligraphy. This study satisfies all three retrieval criteria due to the importance of each one of them, as illustrated in the following sub-sections.

3.1.1 Retrieve Images from the Same Manuscript

The Arabic handwritten manuscripts are a valuable piece of historical information that reflect the education, culture, society, and traditions during specific time periods. They also highlight the developments in the language through time. Retrieving images from the same manuscript is a desirable application for electronic libraries that require retrieving specific book literature and knowledge.

3.1.2 Retrieve Images Written by the Same Author

The authors of the historical manuscripts made efforts to keep the text alive during an era that lacks technology. Thus, the retrieval of images according to a specific author represents the authors' thoughts and identities during that period of time. This retrieval can be used in the authors' identification or knowledge representation applications.

3.1.3 Retrieve Images Written Using the Same Calligraphy

Arabic calligraphies are significant social and political appliances in the Islamic world. The unique handwriting style used in writing the manuscript transfers its historical background, including the region that the manuscript was written at, the specific genre of the manuscript, and the calligraphers' style. It is also possible to identify the period of time that the manuscript written at, through analyzing its calligraphy.

Automating the process of predicting and retrieving the Arabic handwriting style eliminates the need for calligraphy experts to process and handle manuscripts [34] manually. Thus, the proposed solution can be the step-stone for calligraphy analysis, simulation, or generation applications.

3.2 The Dataset Classification

We classified the collected images into three main criteria, as illustrated in table 3.1.

Table 3.1: Collected dataset classification

Labeling Criteria	Manuscript	Author	Calligraphy
No. of labels	64	52	6
Total no. of images	8638		

From table 3.1, we notice that the dataset includes (64) ancient Arabic manuscripts, written by (52) authors, using (6) calligraphies.

3.2.1 The Dataset Classification According to the Manuscripts

Table 3.2 illustrates the dataset classification according to the (64) manuscripts.

From table 3.2, we notice that the collected manuscripts are historical because they were established a very long time ago. Some of the historical Arabic manuscripts written within the time periods between (1004) and (1384) "Hijri" date. While the time periods of other manuscripts in the dataset were unknown. They might be written before the "Hijra" of the Prophet "Mohammed," where the Islamic Hijri date gets started, or the writer might didn't indicate the date. Furthermore, we notice that the manuscripts are from different genres. For instance, most of the manuscripts are religious, but we can also find some linguistics and literature manuscripts.

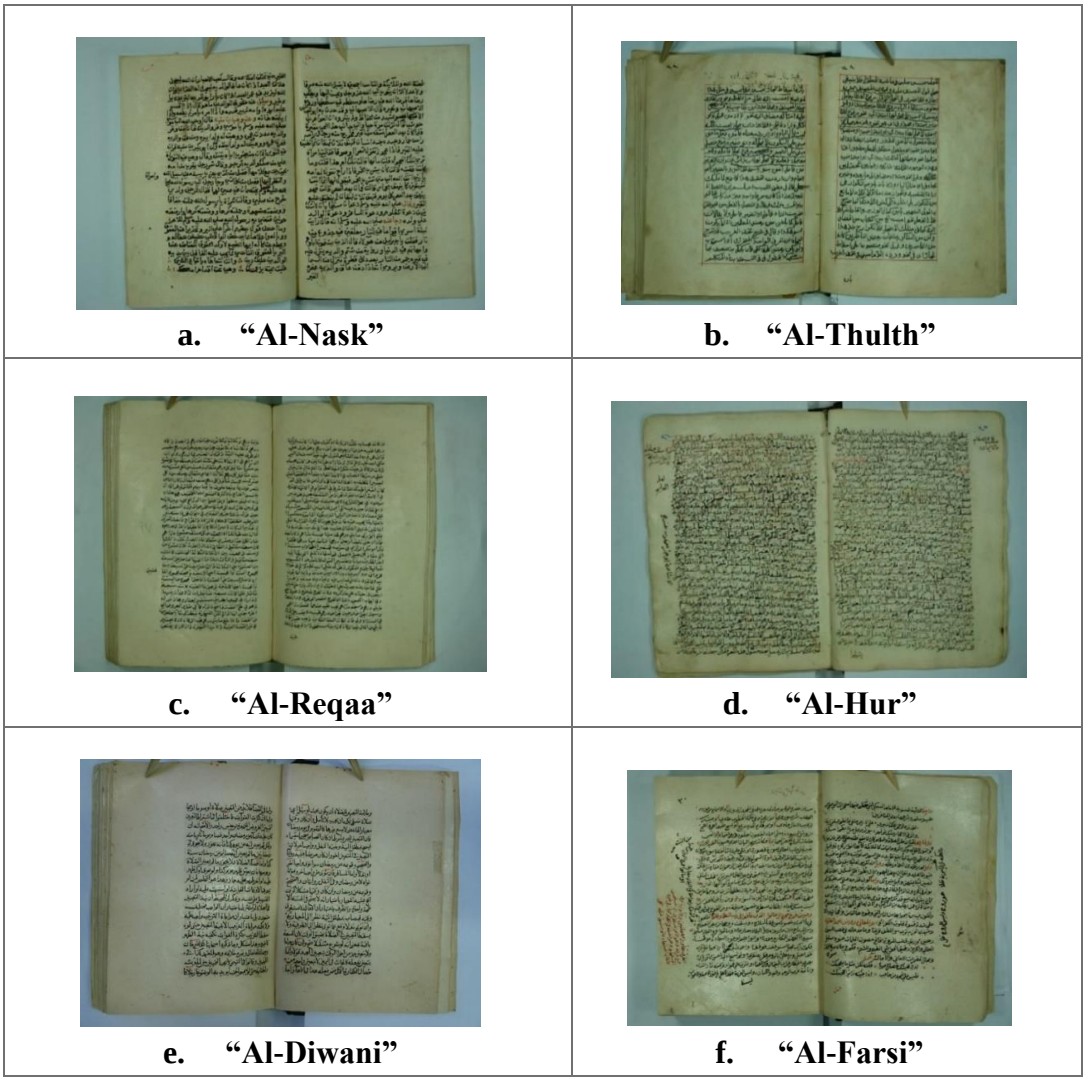
3.2.2 The Dataset Classification According to the Authors

Sometimes the same person establishes and writes the manuscript. While in other situations, one person creates the manuscript called "author"; and another different person writes it called "writer". There might also be an "editor" that reviews the written manuscript and modifies it. However, this study considers only the "authors" of the Arabic manuscripts. Table 3.3 illustrates the dataset classification according to the (52) authors.

3.2.3 The Dataset Classification According to the Calligraphies

Throughout generations, many Arabic calligraphies used, but we found in our dataset only six of them. Hence, the considered handwriting styles in this study are Al-Nask, Al-Thulth, Al-Reqaa, Al-Hur, Al-Diwani, and Al-Farsi. Table 3.4 illustrates the dataset classification according to the (6) Arabic calligraphies.

Figure 3.2 illustrates samples of the calligraphies presented in our dataset.



a. "Al-Nask" **b. "Al-Thulth"**

c. "Al-Reqaa" **d. "Al-Hur"**

e. "Al-Diwani" **f. "Al-Farsi"**

Figure 3.2: Manuscript images written using different Arabic calligraphies

Table 3.2 Dataset classification according to the manuscripts

Manuscript-ID	Time Period in Hijri	Title	Genre	Author	Calligraphy	No. of Images
1	1004	تيسير الوصول إلى جامع الأصول	حديث	محدث اليمن أبو الضياء عبد الرحمن بن علي بن محمد بن عمر بن الربيع الشيباني الشافعي	النسخ	191
2	0	شرح الجامع الصغير	حديث	المناوي	الحر	42
3	0	منتخب كنز العمال - نسخة أولى	حديث	الشيخ حسام الدين الشهير بالمتقي الهندي	الحر	293
4	0	قطعة من شرح معاني الآثار	حديث	أبو جعفر الطحاوي المصري	النسخ	80
5	1135	الأعمال الموجبة	حديث	محمد الشيبي	الثلث	12
6	1305	الهداية في علم الرواية	علم الرواية	شمس الدين محمد بن الجزري	الفارسي	16
7	0	مسند الإمام أحمد بن حنبل رواية ابنه عبد الله	حديث	الإمام أحمد بن حنبل	الحر	309
8	1180	مرقاة المفاتيح على مشكاة المفاتيح	حديث	سيدي على القارئ	الحر	313
9	1233	الجامع الصغير	حديث	عبد الرحمن بن أبي بكر السيوطي	الحر	277
10	1075	شرح صحيح مسلم بن الحجاج	حديث	أبو زكريا محي الدين النووي	النسخ	162
11	1033	مشكاة المصابيح - نسخة أولى	حديث	ولي الدين التبريزي	النسخ	260
12	1183	مشكاة المصابيح - نسخة ثانية	حديث	ولي الدين التبريزي	النسخ	264
13	0	رياض الصالحين	حديث	بو زكريا محي الدين النووي	الثلث	114
14	0	ذكر أسباب إصلاح البيوت	حديث	عقيل بن عمر	الرقعة	16
15	1232	قطعة من صحيح البخاري	حديث	محمد بن إسماعيل البخاري	الحر	114
16	1335	شرح الأربعين، المسمى الفتح المبين	حديث	ابن حجر الهيتمي	الحر	101
17	0	منتخب كنز العمال - نسخة ثانية	حديث	الشيخ حسام الدين الشهير بالمتقي الهندي	الحر	292
18	0	الزواجر في الكبائر	حديث	ابن حجر الهيتمي	النسخ	16
19	1307	ثبت الأمير	حديث	محمد بن محمد الأمير	الرقعة	30
20	1246	مسانيد	فقه	شهاب الدين أحمد ابن محمد بن محمد بن علي ابن حجر الهيتمي	النسخ	139

Table 3.2 Dataset classification according to the manuscripts (Continued)

27	النسخ	حسن الشرنيلالي	فقه	العقد الفريد لبيان الراجح في جواز التقليد	1384	**21**
44	الثلث	محمد أفندي عابدين	فقه	الرحيق المختوم شرح قلائد المنظوم	1305	**22**
10	الديواني	محمد مكي بن عزوز التونسي	فقه	الأجوبة المكية على الأسئلة الحفظية	0	**23**
49	النسخ	حكمة الهندي	فقه	خزانة الروايات	0	**24**
216	الثلث	حسن الشرنيلالي	فقه	نظم الفوايد شرح المقاصد	1096	**25**
158	الديواني	إبراهيم بن محمد الحلبي	فقه	ملتقى الأبحر	1064	**26**
6	الحر	عبد المعطي السملاوي	فقه	المربع في حكم العقد على المذاهب الأربع	1306	**27**
7	النسخ	حسن الشرنيلالي	فقه	تحفة التحرير	1064	**28**
25	الثلث	حضر بن أحمد	فقه	مقدمة عن الصلاة و شروطها	1284	**29**
104	النسخ	حافظ الدين النسفي	فقه	كنز الدقائق	0	**30**
65	الحر	محب الدين الحموي	فقه	عمدة الحكام ومرجع القضاة في الأحكام	1243	**31**
11	الفارسي	تاج الدين بن أحمد الدهان	فقه	إجادة الجدة بمنع القصر في طريق جدة	1310	**32**
9	الثلث	أحمد بن محمد الحموي	فقه	القول البليغ في حكم التبليغ	1066	**33**
8	الديواني	العلامة تاج الدين الدهان	فقه	رسالة في القنوت في النوازل	1171	**34**
34	الثلث	أحمد بن محمد الناطفي	فقه	أحكام الناطفي	1037	**35**
50	الثلث	زين بن نجيم	فقه	الفوائد الزينية في مذهب الحنفية	1239	**36**
274	الرقعة	أكمل الدين	فقه	العناية على شرح الهداية - نسخة أولى	1334	**37**
214	الرقعة	أكمل الدين	فقه	العناية على شرح الهداية - نسخة ثانية	1334	**38**
40	النسخ	عمر بن عمر الزهري الدفري الحنفي	فقه	الدرة المنيفة على مذهب أبي حنيفة	1197	**39**
82	الحر	ملا خسرو	فقه	درر الحكام شرح غرر الأحكام	0	**40**
71	الفارسي	بدر الدين	نحو	شرح التسهيل	0	**41**
254	الرقعة	حسين بن إبراهيم الشهير بزيني زاده	نحو	الفوائد الشافية في إعراب الكافية	1209	**42**
179	الحر	علي النبيتيتي	نحو	شرح الآجرومية	1233	**43**

Table 3.2 Dataset classification according to the manuscripts (Continued)

129	الثلث	الشنواني	نحو	تعليق الدرة الشنوانية على شرح الآجرومية	1019	**44**
82	الثلث	حسن بن أحمد زيني زاده	نحو	تعليق الفواصل على إعراب العوامل	1165	**45**
140	النسخ	ابن هشام النحوي	نحو	شرح شذور الذهب	1233	**46**
119	الرقعة	حاج بابا ابن عثمان الطرسيوي	نحو	لطائف الإعراب في شرح قواعد الاعراب	1086	**47**
9	الرقعة	أحمد بن زيني دحلان	بلاغة	حاشية على متن السمرقندية	1283	**48**
132	الثلث	قاسم الحريري	أدب	مقامات الحريري	1064	**49**
138	النسخ	محمد النواجي المصري	أدب	حلبة الكميت	0	**50**
271	الفارسي	للشهاب أحمد بن محمد الخفاجي	أدب	ريحانة الألبا وزهرة الحياة الدنيا	1330	**51**
6	الرقعة	يوسف الحنفي الشافعي	علم وضع	شرح الرسالة العضدية	1168	**52**
280	الرقعة	محمد الخطيب الشربيني	تفسير	تفسير الخطيب الشربيني	0	**53**
96	الرقعة	محي الدين التالجي	منطق	حاشية على شرح الكافي	1135	**54**
98	النسخ	محمد بن عبدالرسول الشهير بزودي البرزبخي	دين عام	الاشاعة لاشراط الساعة	1368	**55**
103	الثلث	جلال الدين السيوطي	دين عام	شرح الصدور في شرح حال الموتى في القبور	0	**56**
297	الديواني	عبد الوهاب الشعراني	فقه	الأشباه والنظائر الفقهية	1160	**57**
283	الرقعة	الزيلعي	فقه	تبيين الحقائق شرح كنز الدقائق - الجزء الأول	0	**58**
381	الرقعة	الزيلعي	فقه	تبيين الحقائق شرح كنز الدقائق - الجزء الثاني	0	**59**
141	الرقعة	الزيلعي	فقه	تبيين الحقائق شرح كنز الدقائق - الجزء الثالث, القسم الأول	0	**60**
175	الرقعة	الزيلعي	فقه	تبيين الحقائق شرح كنز الدقائق - الجزء الثالث, القسم الثاني	0	**61**
373	الرقعة	الزيلعي	فقه	تبيين الحقائق شرح كنز الدقائق - الجزء الرابع	1132	**62**
170	الرقعة	عثمان ابن محمد قاري الطايفي	فقه	فتح القدير	1233	**63**
237	النسخ	عبدالله بن النسقي	دين عام	أسباب الإختيار	1053	**64**
8638				**Total Number of Images:**		

Table 3.3 Dataset classification according to the authors

Author ID	Arabic Name	English Name	Manuscript(s) Details		
			ID	Title	No. of Images
1	أبو الضياء عبد الرحمن بن علي بن محمد بن عمر بن الربيع الشيباني الشافعي	Abu Theyaa Abdulrahman Bin Ali Bin Mohammed Bin Omar Bin Rabea Alshebani Alshafeai	1	تيسير الوصول إلى جامع الأصول	191
2	المناوي	Al-Manawe	2	شرح الجامع الصغير	42
3	الشيخ حسام الدين الشهير بالمتقي الهندي	Alshaikah Hosam Aldin	3	منتخب كنز العمال (نسخة أولى)	293
			17	منتخب كنز العمال (نسخة ثانية)	292
4	أبو جعفر الطحاوي المصري	Abu Jafar Altahawe Almasri	4	قطعة من شرح معاني الآثار	80
5	محمد الشيبي	Mohammed Alshaibi	5	الأعمال الموجبة	12
6	شمس الدين محمد بن الجزري	Shams Aldin Mohammed Bin Aljazri	6	الهداية في علم الرواية	16
7	الإمام أحمد بن حنبل	Alimam Ahmed Bin Hanbal	7	مسند الإمام أحمد بن حنبل رواية ابنه عبد الله	309
8	سيدي على القارئ	Sidi Ali Alqari	8	مرقاة المفاتيح على مشكاة المفاتيح	313
9	عبد الرحمن بن أبي بكر السيوطي	Abdulrahman Bin Abibakr Alsayoti	9	الجامع الصغير	277
10	أبو زكريا محي الدين النووي	Abu Zakariya Mohe Aldin Alnawawi	10	شرح صحيح مسلم بن الحجاج	162
			13	رياض الصالحين	114
11	ولي الدين التبريزي	Wali Aldin Altbrizi	11	مشكاة المصابيح (نسخة أولى)	260
			12	مشكاة المصابيح (نسخة ثانية)	264
12	عقيل بن عمر	Aqeel Bin Omar	14	ذكر أسباب إصلاح البيوت	16
13	محمد بن إسماعيل البخاري	Mohammed Bin Ismail Albukhari	15	قطعة من صحيح البخاري	114
14	ابن حجر الهيتمي	Ibn Hajar Alhythami	16	شرح الأربعين، المسمى الفتح المبين	101
			18	الزواجر في الكبائر	16
15	محمد بن محمد الأمير	Mohammed Bin Mohammed Alamer	19	ثبت الأمير	30
16	شهاب الدين أحمد ابن محمد بن محمد بن علي ابن حجر الهيتمي	Shihab Aldin Ahmed Ibn Mohammed Bin Mohammed Bin Ali Ibn Hajar Alhythami	20	مسانيد	139
17	حسن الشرنبلالي	Hasan Alshernulaly	21	العقد الفريد لبيان الراجح في جواز التقليد	27
			25	نظم الفوايد شرح المقاصد	216
			28	تحفة التحرير	7
18	محمد أفندي عابدين	Mohammed Afandi Abbdin	22	الرحيق المختوم شرح قلائد المنظوم	44
19	محمد مكي بن عزوز التونسي	Mohammed Maki Bin Azoz Altonisy	23	الأجوبة المكية على الأسئلة الحفظية	10
20	حكمة الهندي	Hekma Alhindi	24	خزانة الروايات	49
21	إبراهيم بن محمد الحلبي	Ibrahim Bin Mohammed Alhalabi	26	ملتقى الأبحر	158
22	عبد المعطي السملاوي	Abdulmoati Alsimlawy	27	المربع في حكم العقد على المذاهب الأربع	6
23	حضر بن أحمد	Hathar Bin Ahmed	29	مقدمة عن الصلاة و شروطها	25

Table 3.3 Dataset classification according to the authors (Continued)

24	حافظ الدين النسفي	Hafez Aldin Alnsfy	**30**	كنز الدقائق	104
25	محب الدين الحموي	Moheb Aldin Alhamawy	**31**	عمدة الحكام ومرجع القضاة في الأحكام	65
26	تاج الدين بن أحمد الدهان	Taj Aldin Bin Ahmed Aldahan	**32**	إجادة الجدة بمنع القصر في طريق جدة	11
			34	رسالة في القنوت في النوازل	8
27	أحمد بن محمد الحموي	Ahmed Bin Mohammed Alhamawy	**33**	القول البليغ في حكم التبليغ	9
28	أحمد بن محمد الناطفي	Ahmed Bin Mohammed Alnatefy	**35**	أحكام الناطفي	34
29	زين بن نجيم	Zain Bin Nejam	**36**	الفوائد الزينية في مذهب الحنفية	50
30	أكمل الدين	Akmal Aldin	**37**	العناية على شرح الهداية (نسخة أولى)	274
			38	العناية على شرح الهداية (نسخة ثانية)	214
31	عمر بن عمر الزهري الدفري الحنفي	Omar Bin Omar Alzahri Aldafri Alhanafy	**39**	الدرة المنيفة على مذهب أبي حنيفة	40
32	ملا خسرو	Mala Khasro	**40**	درر الحكام شرح غرر الأحكام	82
33	بدر الدين	Badr Aldin	**41**	شرح التسهيل	71
34	حسين بن إبراهيم الشهير بزيني زاده	Hussain Bin Ibrahim	**42**	الفوائد الشافية في إعراب الكافية	254
35	علي النبيتيتي	Ali Alnubity	**43**	شرح الأجرومية	179
36	الشنواني	Alshenwany	**44**	تعليق الدرة الشنوانية على شرح الآجرومية	129
37	حسن بن أحمد زيني زاده	Hassan Bin Ahmed Zaini Zadah	**45**	تعليق الفواصل على إعراب العوامل	82
38	ابن هشام النحوي	Ibn Hesham Alnahwi	**46**	شرح شذور الذهب	140
39	حاج بابا ابن عثمان الطرسيوي	Haj Baba Ibn Othman Althrsiwi	**47**	لطائف الإعراب في شرح قواعد الاعراب	119
40	أحمد بن زيني دحلان	Ahmed Bin Zaini Dahlan	**48**	حاشية على متن السمرقندية	9
41	قاسم الحريري	Qasem Alhariry	**49**	مقامات الحريري	132
42	محمد النواجي المصري	Mohammed Alnawajy Almasri	**50**	حلبة الكميت	138
43	أحمد بن محمد الخفاجي	Ahmed Bin Mohammed Alkhafagy	**51**	ريحانة الألبا وزهرة الحياة الدنيا	271
44	يوسف الحنفي الشافعي	Yousef Alhanafi Alshafei	**52**	شرح الرسالة العضدية	6
45	محمد الخطيب الشربيني	Mohammed Alkhatib Alsherbini	**53**	تفسير الخطيب الشربيني	280
46	محي الدين التالجي	Mohi Aldin Altaljy	**54**	حاشية على شرح الكافي	96
47	محمد بن عبدالرسول	Mohammed Bin Abdulrasol	**55**	الاشاعة لاشراط الساعة	98
48	جلال الدين السيوطي	Jalal Aldin Alsayoti	**56**	شرح الصدور في شرح حال الموتى في القبور	103
49	عبد الوهاب الشعراني	Abdulwahab Alshearani	**57**	الأشباه والنظائر الفقهية	297
50	الزيلعي	Alzailai	**58**	تبيين الحقائق شرح كنز الدقائق (الجزء الأول)	283
			59	تبيين الحقائق شرح كنز الدقائق (الجزء الثاني)	381
			60	تبيين الحقائق شرح كنز الدقائق (الجزء الثالث، القسم الأول)	141
			61	تبيين الحقائق شرح كنز الدقائق (الجزء الثالث، القسم الثاني)	175
			62	تبيين الحقائق شرح كنز الدقائق (الجزء الرابع)	373
51	عثمان ابن محمد قاري الطايفي	Othman Ibn Mohammed Qari Altaifi	**63**	فتح القدير	170
52	عبدالله بن النسقي	Abduallah Bin Alnasqi	**64**	أسباب الإختيار	237
Total:			**64**		**8638**

Table 3.4 Dataset classification according to the calligraphies

Calligraphy ID	Arabic Name	English Name	Manuscripts Details		Images Number	Total No. of images
			Numbers	Titles		
1	النسخ	Al-Nask	16	قطعة من شرح معاني الآثار	80	1952
				تيسير الوصول إلى جامع الأصول	191	
				شرح صحيح مسلم بن الحجاج	162	
				مشكاة المصابيح - جزئين	524	
				مسانيد	139	
				شرح شذور الذهب	140	
				حلبة الكميت	138	
				أسباب الإختيار	237	
				الزواجر في الكبائر	16	
				العقد الفريد لبيان الراجح في جواز التقليد	27	
				خزانة الروايات	49	
				تحفة التحرير	7	
				كنز الدقائق	104	
				الدرة المنيفة على مذهب أبي حنيفة	40	
				الاشاعة لاشراط الساعة	98	
2	الثلث	Al-Thulth	12	الأعمال الموجبة	12	950
				رياض الصالحين	114	
				الرحيق المختوم شرح قلائد المنظوم	44	
				نظم الفوايد شرح المقاصد	216	
				تعليق الدرة الشنوانية على شرح الآجرومية	129	
				مقدمة عن الصلاة و شروطها	25	
				القول البليغ في حكم التبليغ	9	
				أحكام الناطفي	34	
				الفوائد الزينية في مذهب الحنفية	50	
				تعليق الفواصل على إعراب العوامل	82	
				مقامات الحريري	132	
				شرح الصدور في شرح حال الموتى في القبور	103	

Table 3.4 Dataset classification according to the calligraphies (Continued)

3	الرقعة	Al-Reqaa	**16**	ذكر أسباب إصلاح البيوت	16	**2821**
				العناية على شرح الهداية - جزئين	488	
				الفوائد الشافية في إعراب الكافية	254	
				لطائف الإعراب في شرح قواعد الاعراب	119	
				تفسير الخطيب الشربيني	280	
				تبيين الحقائق شرح كنز الدقائق – خمسة أجزاء	1353	
				ثبت الأمير	30	
				حاشية على متن السمرقندية	9	
				شرح الرسالة العضدية	6	
				حاشية على شرح الكافي	96	
				فتح القدير	170	
4	الحر	Al-Hur	**12**	شرح الجامع الصغير	42	**2073**
				شرح الأربعين، المسمى الفتح المبين	101	
				المربع في حكم العقد على المذاهب الأربع	6	
				منتخب كنز العمال - جزئين	585	
				مسند الإمام أحمد بن حنبل رواية ابنه عبد الله	309	
				مرقاة المفاتيح على مشكاة المفاتيح	313	
				الجامع الصغير	277	
				قطعة من صحيح البخاري	114	
				عمدة الحكام ومرجع القضاة في الأحكام	65	
				درر الحكام شرح غرر الأحكام	82	
				شرح الآجرومية	179	
5	الديواني	Al-Diwani	**4**	الأجوبة المكية على الأسئلة الحفظية	10	**473**
				ملتقى الأبحر	158	
				رسالة في القنوت في النوازل	8	
				الأشباه والنظائر الفقهية	297	
6	الفارسي	Al-Farsi	**4**	الهداية في علم الرواية	16	**369**
				إجادة الجدة بمنع القصر في طريق جدة	11	
				شرح التسهيل	71	
				ريحانة الألبا وزهرة الحياة الدنيا	271	
Total			**64**			**8638**

3.3 The Dataset Challenges and their Associated Solutions

The low-quality images treated utilizing the deep learning extraction of the high-level features. Moreover, the "weighted categorical cross-entropy" loss algorithm used to weight the classes, including the minimum number of images more than the classes, including a more significant number of images. This algorithm is used to fix the unbalanced images within the dataset. Because the unbalanced ratio of images may result in an accuracy paradox problem were the generated results could be biased and overfitted to the manuscript, including the largest number of images [112]. The mathematical representation of the weighted categorical cross-entropy loss function is presented in equation (3.1) [113]:

$$L = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M} w_k * y_m^k * log(h_\theta(x_m, k)) \quad (3.1)$$

Where *M* is the number of training images, *K* is the number of classes, w_k is the weight for class k, y_m^k is the target label for the *m* training image belonging to *k* class, x_m refers to the input for *m* training image, and h_θ is the model with neural network weights θ.

Moreover, we employed the online data augmentation method to enhance the classification process since it increases the original dataset size through generating new arbitrary modified versions of the images, which assist the model in getting more generalized with the data. The data augmentation method modifies the original images by changing their brightness, colors, noising, rotation, zooming, twisting, stretching, cropping, and flipping.

The data augmentation method could be implemented offline to generate the new images before training and have the new samples existed on the hard disk, or it could

be implemented online, so the new samples will not exist on the hard drive. Instead, the new augmented samples will be generated and used during the training. The main difference between the two types of augmentation methods is that the online real-time augmentation saves more space on the users' hard disk. The data augmentation method works well with visual-based images (spatial-based) such as images, including faces, animals, clothing, flowers, etc. Thus, we performed wisely online data augmentation on our ancient Arabic manuscripts' images utilizing the "ImageDataGenerator" function under "Keras" deep learning library. Five different modifications to the images' angles implemented as follows:

- Rotate the images up-to 30 degrees from the center
- Zoom up-to 10% more inside images
- Increase both the width and height by 10%
- Twist/shear images by pulling them from the top toward the right or left up-to 20%
- Fill the corner of images through repeating closest values to each pixel

Figure 3.3 illustrates examples of the augmented images.

(a) **(b)**

Figure 3.3: Data augmentation. (a) Original manuscripts' images, (b) Augmented images

Chapter IV

The Proposed Arabic Manuscripts Image Retrieval System

This thesis introduces a novel image retrieval system that utilizes the supervised deep learning technique to automatically extract the features and retrieve similar images to a user query image. Moreover, we developed a deep reinforcement learning model to retrieve the images of the Arabic manuscript.

The dataset used includes historical Arabic manuscript images, which are text-based images. Therefore, we employed the visual features existing in the images to retrieve similar images, as well as, we extracted the text from the images and retrieved the similar images according to the textual features of the text written inside the images. In addition, we fused both visual and textual models to experiment the retrieval system according to the fused models. Thus, the proposed Arabic manuscripts image retrieval system consists of four main steps as following:

1. Visual-based image retrieval
    - Image classification
    - Similarity measurement
2. Textual-based image retrieval
    - Text classification
    - Similarity measurement

3. Image retrieval using the fusion model
    - Image and text classification
    - Similarity measurement
4. Image retrieval using the DRL model

Considering that most image retrieval systems consist of two main steps, which are: the classification and the similarity measurement. Thus, we divided the first three main steps further into classification and similarity measurement. The classification section explains classifying the images or text according to specific labels. While the similarity measurement section explains the method of matching the classified images or text to retrieve similar images to a user query image.

Figure 4.1 highlights all the methods that we went through to reach the most accurate image retrieval system.

From figure 4.1, we notice that we developed four different models for retrieving the Arabic manuscripts' images. They are visual, textual, fusion, and deep reinforcement learning models. The methods highlighted using the bold blue color are to indicate that they were the most accurate methods when we had to experiment more than one method for performing the same task.

4.1 Visual-based Image Retrieval

Since the used handwritten ancient Arabic manuscripts' images are including visual contents, such as drawings, signatures, tables, ...etc. Thus, it is essential to extract the deep learning visual features of the images and to process them according to their extracted visual features. Figure 4.2 illustrates the proposed framework for visual-based image retrieval.

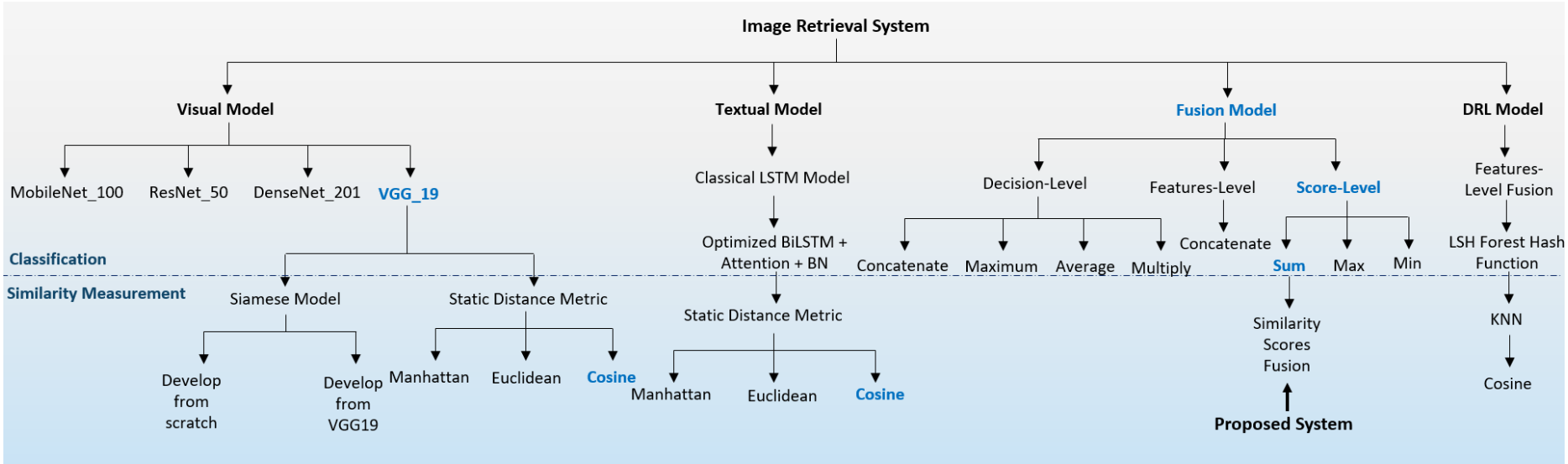


Figure 4.1: The conceptual tree of the proposed IR system

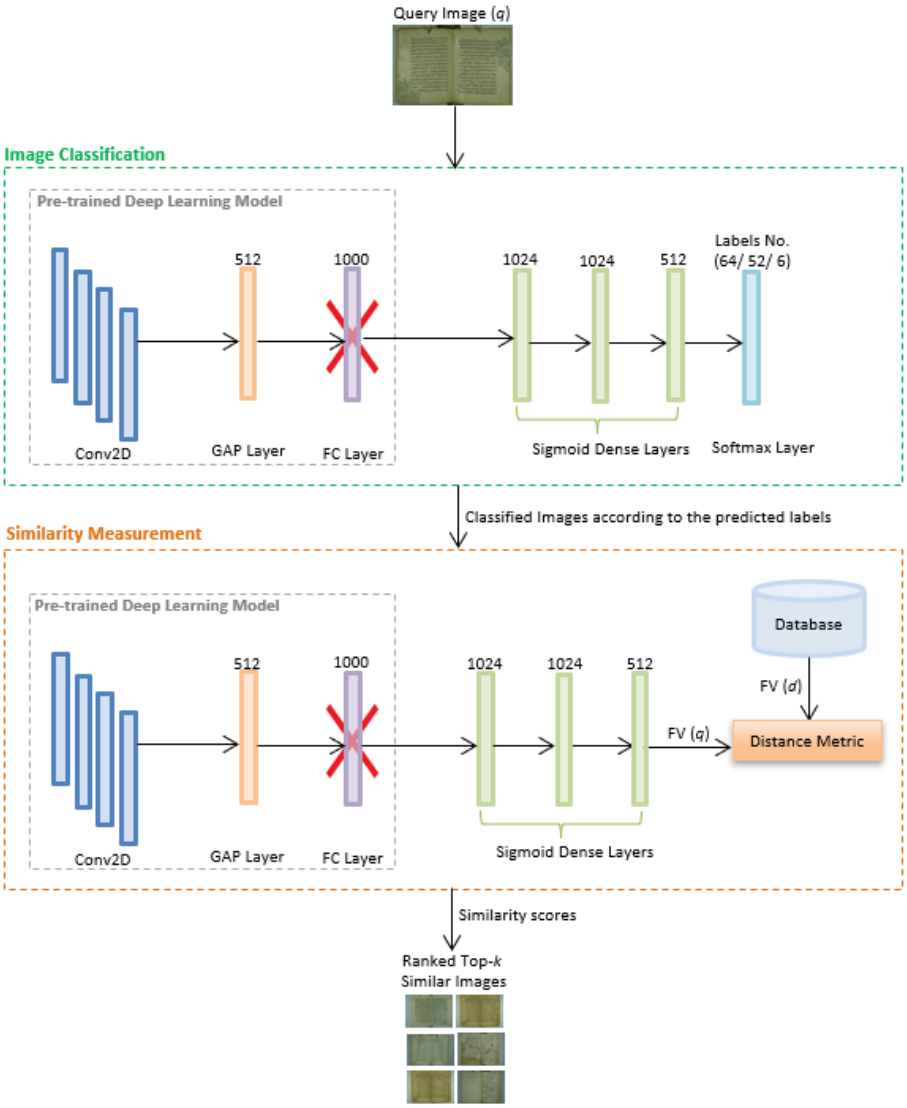


Figure 4.2: Proposed method for visual-based image retrieval

To classify the images, we transfer learning from all the layers and weights of a pre-trained deep learning model. But the last fully connected layer that comes with the original model and usually has a feature vector of size (1000) should be removed and add instead of it a “Softmax” layer, including the number of labels that we aim to classify the images according to. Considering that the used ancient Arabic manuscripts dataset includes (64) manuscripts written by (52) authors using (6) Arabic

calligraphies. Thus, the number of labels in the last added “Softmax” classification dense layer, will include either (64/ 52/ 6) labels.

Before the final “Softmax” classification dense layer, there are three “Sigmoid” dense layers to improve the classification accuracy. After classifying the images, the feature vector of the user query image, denoted as FV(*q*), is generated utilizing the “Sigmoid” dense layer. Moreover, the features vectors of all the images in the dataset denoted as FV(*d*); are generated and saved in a database to enter them along with the feature vector of the user query image into a distance metric to measure the distances among the images and display their similarity scores. Finally, the similarity scores are ranked to output the top-*k* similar images to a user query image. The following subsections explain the image classification and similarity measurement in more detail.

4.1.1 Image Classification

The image classification step has two phases, which are the training phase and the testing phase, as illustrated in figure 4.3.

In the training phase, all the dataset images enter the customized deep pre-trained CNN to classify the images according to the predicted labels. After classifying the images in the training phase, the features vectors of the classified images, denoted as FV(*d*), are generated and saved in a database for faster retrieval during the testing phase.

In the testing phase, the user query image enters the deep CNN to generate its feature vector, denoted as FV(*q*), utilizing the “Sigmoid” dense layer that falls just before the last fully connected layer and employed (512) dimensional feature size.

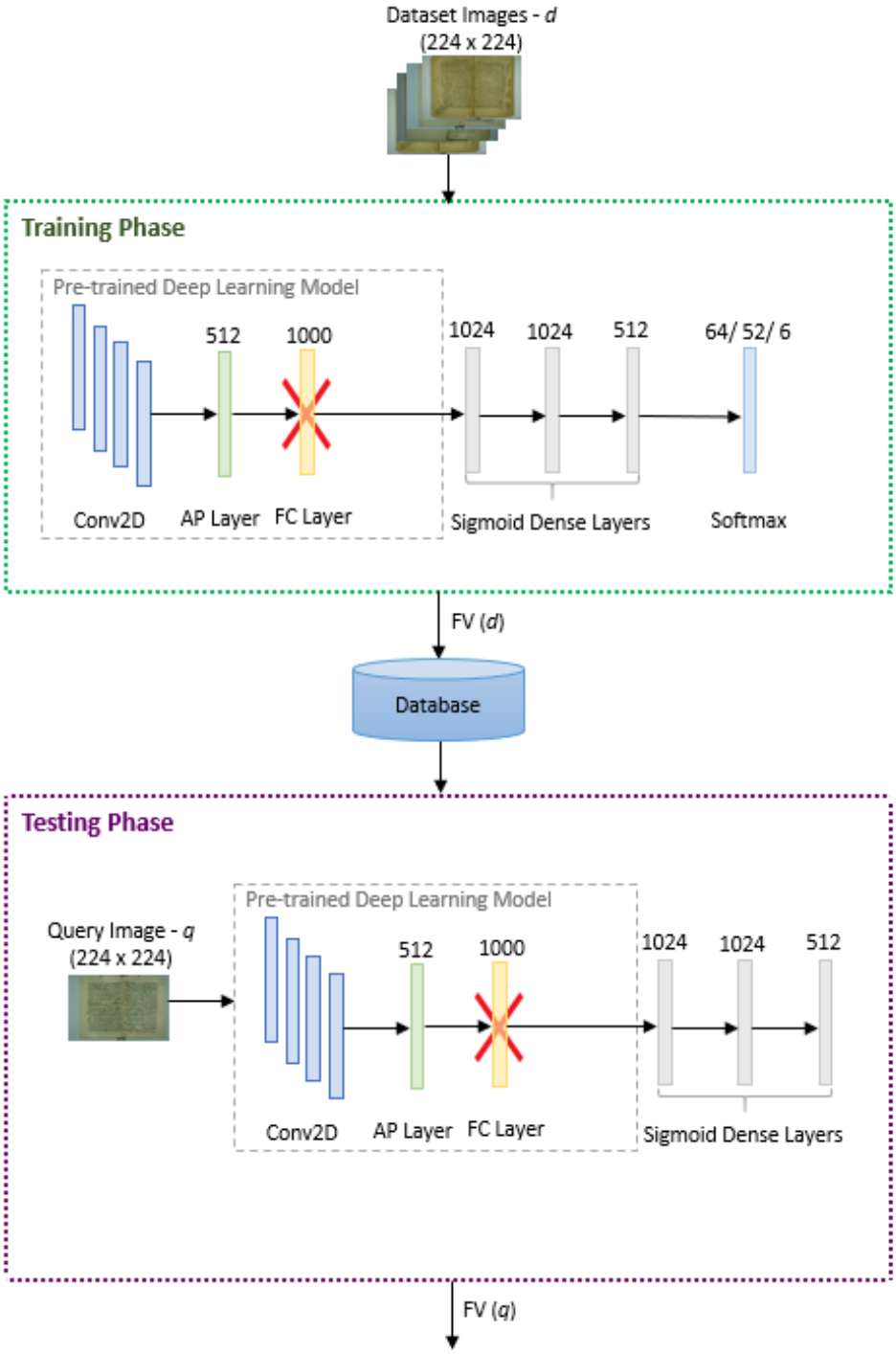


Figure 4.3: Training and testing of the CNN for image classification

There are many open-source deep learning packages that researchers can use to develop their models. Including Theano, Caffe, Torch, PyTorch, MLC++, OpenCV, OpenNN, Scikit, Accord, cuDNN, BigDL, Chainer, Deeplearning4j, Dlib, Keras, Microsoft Cognitive Toolkit (CNTK), Apache MXNet, Apache SINGA, PlaidML, and

Tensorflow. The first library, called MLC++, which released in 1994. While the most recent deep learning library is Tensorflow that released in 2016 [114].

We leveraged four pre-trained deep learning models that all fall under "Tensorflow"[6] deep learning library to classify the images according to their extracted visual features. All utilized models initially trained to classify images from the "ImageNet Large Scale Visual Recognition Challenge" that conducted in the year of 2012 and abbreviated as (ILSVRC-2012-CLS)[7] [115]. The developed deep CNNs for image classification are four as follows:

a) MobileNet_V1_100_244

MobileNetV1 deep learning model is the simplest model we used in our experiments. That is because it consists of a small number of layers contained within plain blocks and stacked on the top of each other without any residual connections between them. Instead, the convolutional layers connected linearly, and the signals move only in forward propagation. Moreover, MobileNetV1 model decreases the spatial dimensions among its tensors, which makes it small compared with other large deep learning models. This characteristic enables it to execute faster and in a short time. Concerning the number of multi-adds, MobileNetV1 includes 569 million of them that authorize the model to realize and comprehend the learned features efficiently [116].

[6] https://tfhub.dev/

[7] http://www.image-net.org/challenges/LSVRC/2012/

b) ResNet_V2_50

ResNet50 is a deep residual convolutional neural network. In other words, it includes residual connections and multiple branches between its 50 convolutional layers, which makes it a non-linear model. Hence, its signals can move in a backpropagation or forward propagation manner, making skip connections as needed. The second version of ResNet50 model uses batch normalization as a pre-activation function before calculation the weights to improve the training on the extracted features [117]. The model includes over 25 million of learning parameters that makes it efficient in the learning process [118].

c) DenseNet_201

DenseNet201 deep learning model is an extensive model since it includes (201) convolutional layers. Each layer in the DenseNet201 model is passing its features to all incoming next layers while collecting previous knowledge from all preceding layers [119]. This increases the number of channels moving forward in the model. However, every two contiguous blocks in the model are separated by one convolutional layer and one average pooling layer to decrease the model's complexity. DenseNet201 model is similar to ResNet50 in that it also uses the batch normalization before the weights' computation function.

d) VGG_19

According to Tsang [120], the performance of VGG19 deep learning model outperformed all other models since it won the ILSVRC-2014 competition for classifying images. In addition, VGG19 generated the highest evaluation parameters on both Caltech and VOC datasets. Thus, it is a rigid deep learning model even though it

includes the least number of convolutional layers comparing it with the other three experimented deep learning models. In contrast, VGG_19 model is having the largest number of learning parameters, among other utilized models. It includes 144 million parameters [121]. This means that increasing the number of layers without efficient use of the other learning parameters will not improve the learning process.

Figure 4.4 Illustrates the layers' structure of the leveraged convolutional neural networks. (a) MobileNet-V1 model [116], (b) ResNet-50 model [29], (c) DenseNet-201 model [119], and (d) VGG-19 model [29].

The figure highlights the output size written in orange color to the right side of each layer. If the same layer repeated then, we indicated this by the dashed blue square with the number of repetitions written in blue above the output size. The straight arrows are for the plain blocks with forwarding propagated signals, while the slanted arrows are referencing the residual blocks with both forward and backward propagated signals.

Even though the residual connections in ResNet50 model exist after each 3-layers block, for simplicity, we drew them between main blocks.

The (1x1/ 3x3/ 7x7) written before each layer indicates the size of the kernel. On the other hand, the number of filters highlighted inside the layers' boxes after the "-" symbol. The number written with the Fully Connected (FC) layer indicating the size of the feature that produced from previous training and features extraction steps and entering the layer.

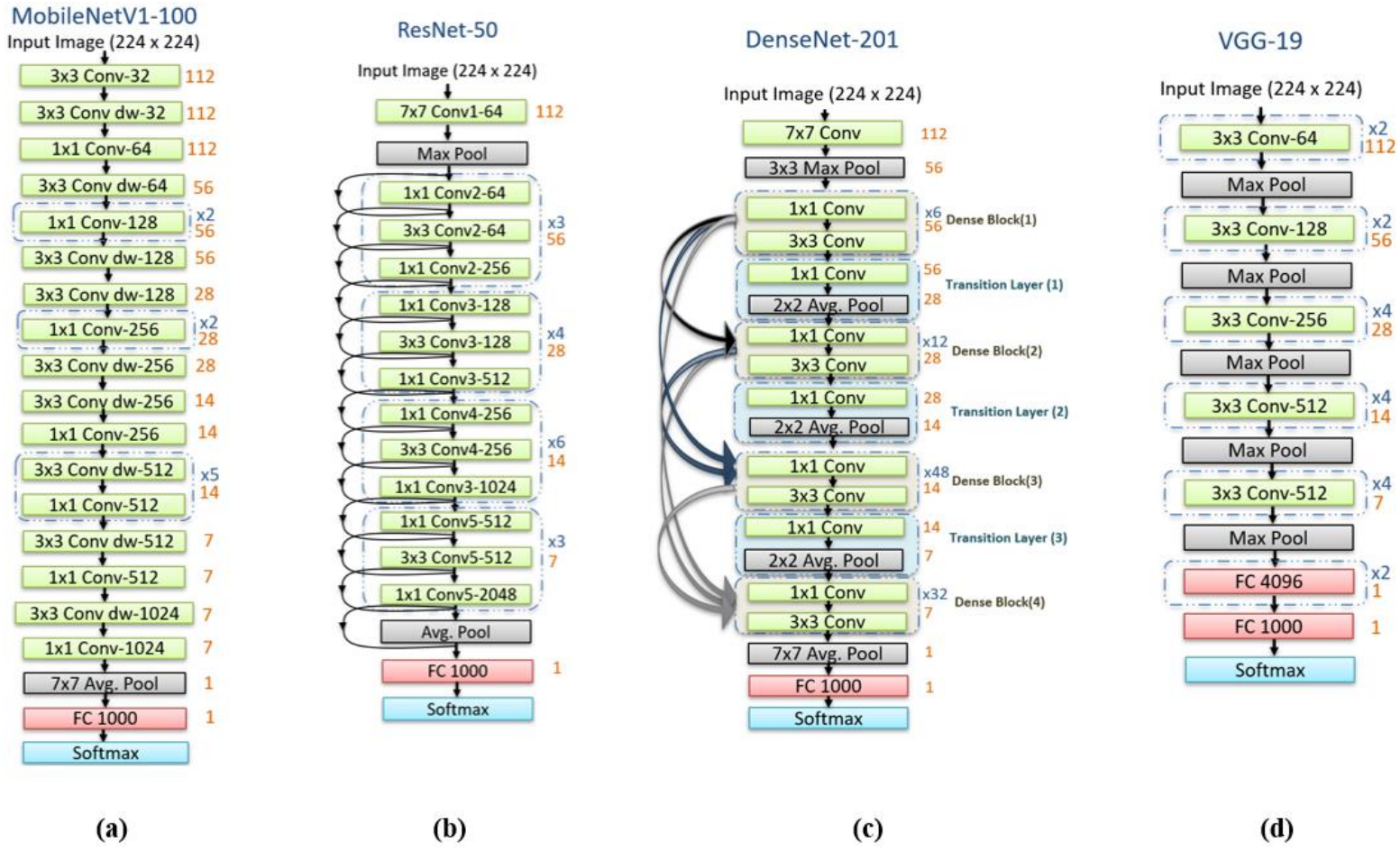


Figure 4.4: The structure of the CNNs. (a) MobileNet100, (b) ResNet50, (c) DenseNet201, and (d) VGG19.

4.1.2 Similarity Measurement

a) Image Matching and Retrieval using the Siamese Deep Learning Model

We started by developing the image matching and retrieval using the Siamese deep learning model because it has been used by many researchers and recorded high satisfying retrieval results [50], [52], [53], [54], and [55].

The second step in the visual-based image retrieval takes the outputs from the images' classification first step, which are the feature vector of the user query image and the features vectors of all classified images stored in the dataset and enters them as inputs to the Siamese model as illustrated in figure 4.5.

From figure 4.5, we notice that during the training phase, the Siamese model enters each one of the input images features vectors into one of the twin Siamese networks. The model uses a max-pooling layer to reduce the spatial dimensionality that exists within the tensors, followed by a flatten layer to convert the two-dimensional matrix into one vector. Then, a dense layer including the "Sigmoid" activation function, to predict the similarities among images. Hence, the last employed layer by the model during the training is the "Cross-entropy loss" shared layer, which includes only one neuron to classify the images as either similar denoted by (1) or not similar, denoted by (0).

In the testing phase, the Siamese model re-produces the feature vector of the images after minimizing the distances among their classified classes and computes the distances between the feature vector of the user query image and the features vectors of all other images in the dataset to generate the ranked similarity scores after computing the sigmoid function of the computed distances. The final outputs from the model are the ranked top-*k* similar images to a user query image.

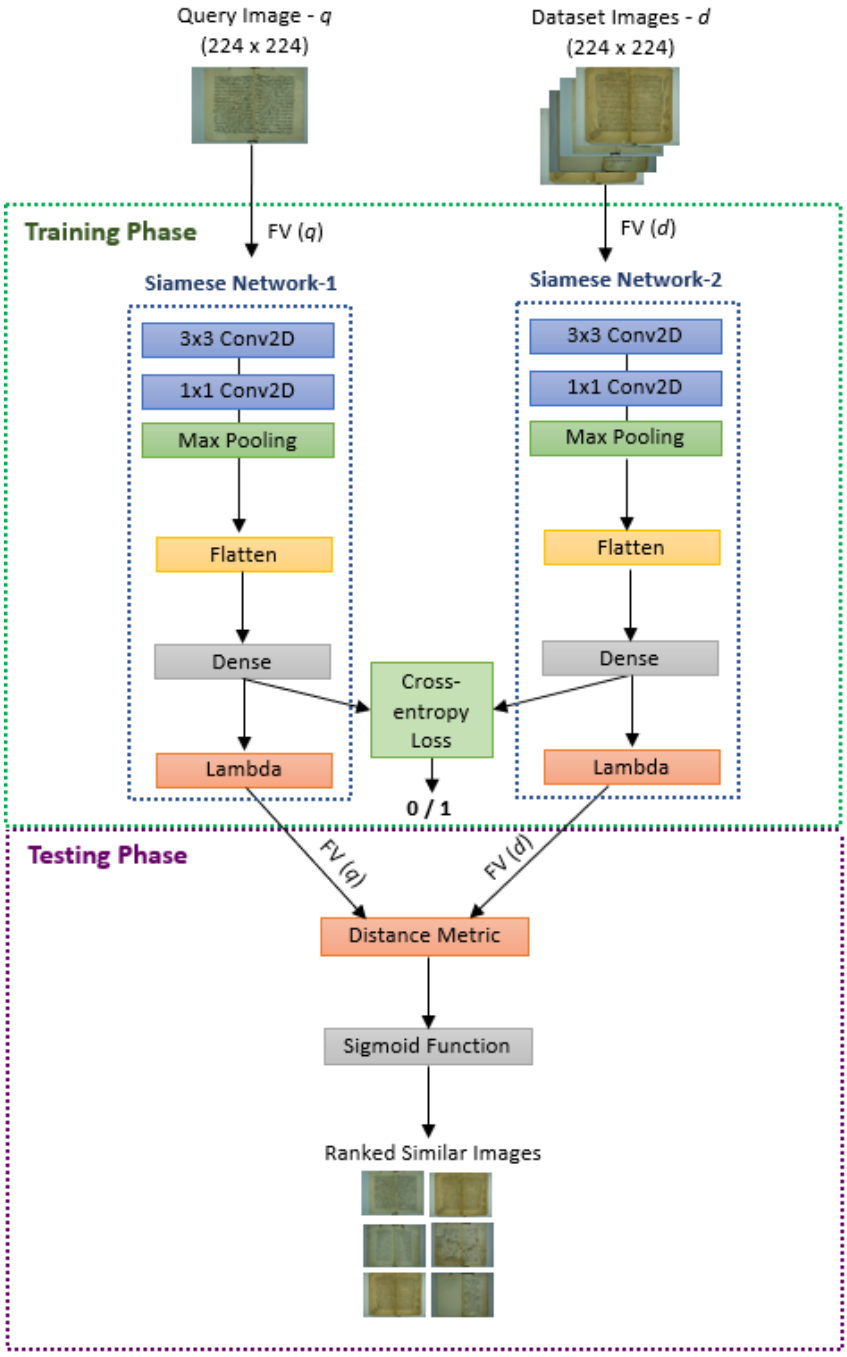


Figure 4.5: Training and testing the Siamese model for retrieval

b) Mathematical Representation of the Siamese Model

The Siamese model doesn't learn any classification; instead, it learns to compute the similarity between classified images [122]. It consists of twin convolutional networks

that share the same weights and structures and are joint through their calculated weights and the computed loss function by the last fully connected dense layer. The main goal of the loss function is to reduce the difference between the probability distribution of the true labels and the predicted labels. The minimization function presented in equation (4.1) [123], ensures that the classified images from the same class preserve analogous feature vector.

$$Ls_i = -\sum_{k=1}^{C} t_k(y_i)\ log\ P(y_i = k|b_i; w_k) \tag{4.1}$$

Where Ls is the loss function of the i^{th} feature in a dataset including a total number of N samples, $i = 1,2,3, \dots, N$. k is the class, and C is the classes' classifier. y_i is the true label in the Y set of true labels, $Y = \{y_i\}_{i=1}^{N},\ y_i \in \{1,2,3, \dots, C\}$ and $t_k(y_i)$ is the distribution of y_i.

$P(y_i = k|b_i; w_k)$ is the probability distribution of the predicted label. b_i is the binary representation of the i^{th} feature, $B = \{b_i\}_{i=1}^{N},\ b_i \in \{-1, 1\}$. w_k is the weight of the classifier.

After computing the loss function, the Siamese model usually contains a pure static distance metric to compute the exact similarity score of the images. Hence, it subtracts the feature vector of the user query image from the features vectors of all other images stored in the dataset. And after calculating the difference between the two extracted features vectors, it converts the computed difference into one single number using the "Sigmoid" activation function. The output number from the final computed "Sigmoid" function presented in (4.2) [51] is called the similarity score (*SC*).

$$SC = \sigma\left(\sum_j \propto_j |h_{1,L-1}^{(j)} - h_{2,L-1}^{(j)}\right) \tag{4.2}$$

Where σ is the "Sigmoid" activation function, *j* refers to the number of neurons on a layer. $\propto_j$ is the final computed difference between the two feature vectors by the twins' Siamese neural networks, *L* represents the number of layers in each Siamese neural network. Hence, *L-1* is the layer before the last fully connected layer.

$h_{1,L-1}^{(j)}$ represents the hidden features falling in the layer before the last fully connected layer within the first Siamese neural network. Similarly, $h_{2,L-1}^{(j)}$ represents the hidden features falling in the layer before the last fully connected layer within the second twin of the Siamese neural network.

c) Siamese Model Development

There are two different developments for the Siamese deep learning model. The first is to develop the model and train from scratch. While the second method is to develop the Siamese model using the structure and the weights of a pre-trained deep learning model. Table 4.1 illustrates the layers, weights, and the overall architecture of the built Siamese model from scratch.

Table 4.1: The architecture of the Siamese DNN

Layer	Type	Output	Parameters	Connected to
Input_1	Input Layer	(224, 224, 3)	0	NA
Input_2	Input Layer	(224, 224, 3)	0	NA
Conv2d_1	Conv2D	(148, 98, 64)	1792	Input_1 [0][0] Input_2 [0][0]
Conv2d_2	Conv2D	(148, 98, 64)	4160	Conv2d_1 [2][0] Conv2d_1 [3][0]
Max_pooling2d_1	Maxpooling2D	(74, 49, 64)	0	Conv2d_2 [2][0] Conv2d_2 [3][0]
Conv2d_3	Conv2D	(72, 47, 128)	73856	Max_pooling2d_1 [2][0] Max_pooling2d_1 [3][0]
Conv2d_4	Conv2D	(72, 47, 64)	8256	Conv2d_3 [2][0] Conv2d_3 [3][0]

Table 4.1: The architecture of the Siamese DNN (Continued)

Max_pooling2d_2	Maxpooling2D	(36, 23, 64)	0	Conv2d_4 [2][0] Conv2d_4 [3][0]
Conv2d_5	Conv2D	(34, 21, 256)	147712	Max_pooling2d_2 [2][0] Max_pooling2d_2 [3][0]
Conv2d_6	Conv2D	(34, 21, 64)	16448	Conv2d_5 [2][0] Conv2d_5 [3][0]
Max_pooling2d_3	Maxpooling2D	(17, 10, 64)	0	Conv2d_6 [2][0] Conv2d_6 [3][0]
Conv2d_7	Conv2D	(15, 8, 256)	147712	Max_pooling2d_3 [2][0] Max_pooling2d_3 [3][0]
Conv2d_8	Conv2D	(15, 8, 64)	16448	Conv2d_7 [2][0] Conv2d_7 [3][0]
Max_pooling2d_4	Maxpooling2D	(7, 4, 64)	0	Conv2d_8 [2][0] Conv2d_8 [3][0]
Flatten_1	Flatten	(1792)	0	Max_pooling2d_4 [2][0] Max_pooling2d_4 [3][0]
Dense_1	Dense	(256)	459008	Flatten_1 [2][0] Flatten_1 [3][0]
Lambda_2	Lambda	(256)	0	Dense_1 [2][0] Dense_1 [3][0]
Dense manuscript	Dense	(1)	257	Lambda_2 [0][0]

Inspired by Singh [124], the second developed Siamese model is according to the structure of the pre-trained VGG19 deep learning model, as illustrated in table 4.2.

Table 4.2: The architecture of the VGG19-Siamese deep learning model

Layer	Type	Output	Parameters	Connected to
Input_1	Input Layer	224, 224, 3	0	NA
Input_2	Input Layer	224, 224, 3	0	NA
Vgg19	Model	Multiple	20024384	Input_1 [0][0] Input_2 [0][0]
Global_max_pooling2d_19	Global	512	0	Vgg19 [1][0]
Global_average_pooling2d_20	Global	512	0	Vgg19 [1][0]
Global_max_pooling2d_20	Global	512	0	Vgg19 [2][0]

Table 4.2: The architecture of the VGG19-Siamese deep learning model (Continued)

Global_average_pooling2d_21	Global	512	0	Vgg19 [2][0]
Concatenate_28	Concatenate	1024	0	Global_max_pooling2d_19 [0][0] Global_average_pooling2d_20 [0][0]
Concatenate_29	Concatenate	1024	0	Global_max_pooling2d_20 [0][0] Global_average_pooling2d_21 [0][0]
Multiply_29	Multiply	1024	0	Concatenate_28 [0][0] Concatenate_28 [0][0]
Multiply_30	Multiply	1024	0	Concatenate_29 [0][0] Concatenate_29 [0][0]
Subtract _19	Subtract	1024	0	Concatenate_28 [0][0] Concatenate_29 [0][0]
Lambda_10	Lambda	1	0	Concatenate_28 [0][0] Concatenate_29 [0][0]
Subtract _20	Subtract	1024	0	Multiply_29 [0][0] Multiply_30 [0][0]
Multiply_28	Multiply	1024	0	Subtract _19 [0][0] Subtract _19 [0][0]
Concatenate_30	Concatenate	2049	0	Lambda_10 [0][0] Subtract_20 [0][0] Multiply_28 [0][0]
Dense_23	Dense	100	205000	Concatenate_30 [0][0]
Dropout_10	Dropout	100	0	Dense_23 [0][0]
Dense_24	Dense	1	101	Dropout_10 [0][0]

From table 4.1 and table 4.2, we notice that the siamese model accepts two input images. In contrast, all other classical deep learning models take only one input image. The total number of trainable parameters, within the built Siamese model from scratch, equals 857,649. While it equals 20,229,485 in the Siamese model built from the structure of the VGG19 pre-trained deep learning model.

d) Siamese Model Learning Strategy

One-shot learning strategy is employed, which allows the model to predict the true labels after seeing only one example from each class. It is accomplished through

defining two customized lists for the training subset, named: images and external. The images customized training list takes one random image from each class and shows it to the model as an example of a similar matching image. In contrast, the external customized training list shows the model one random image that is from any different manuscript than the user query image to show the model an example of non-similar images. This strategy of learning saves training time, and it is useful when some of the dataset classes include only few examples.

e) Image Matching and Retrieval using the Static Distance Metrics

An alternative solution to the Siamese deep learning model is to perform the image matching and retrieval utilizing the static distance metrics. The "Sigmoid" dense layer with (512) feature size within the VGG19 deep CNN is used to generate the images features vectors employing the architecture illustrated in table 4.3.

Table 4.3: The architecture of VGG19 model for visual features extraction

Layer	Type	Output Shape	Parameters Number
Input_1	Input Layer	3	0
Block1_conv1	Conv2D	64	1792
Block1_conv2	Conv2D	64	36928
Block1_pool	MaxPooling2D	64	0
Block2_conv1	Conv2D	128	73856
Block2_conv2	Conv2D	128	147584
Block2_pool	MaxPooling2D	128	0
Block3_conv1	Conv2D	256	295168
Block3_conv2	Conv2D	256	590080
Block3_conv3	Conv2D	256	590080
Block3_conv4	Conv2D	256	590080
Block3_pool	MaxPooling2D	256	0
Block4_conv1	Conv2D	512	1180160
Block4_conv2	Conv2D	512	2359808
Block4_conv3	Conv2D	512	2359808
Block4_conv4	Conv2D	512	2359808
Block4_pool	MaxPooling2D	512	0
Block5_conv1	Conv2D	512	2359808
Block5_conv2	Conv2D	512	2359808
Block5_conv3	Conv2D	512	2359808

Table 4.3: The architecture of VGG19 model for visual features extraction (Continued)

Block5_conv4	Conv2D	512	2359808
Block5_pool	MaxPooling2D	512	0
Global_average_pooling2d_2	Global_average_pooling2D	512	0
Dense_5	Dense	1024	525312
Dense_6	Dense	1024	1049600
Dense_7	Dense	512	524800

From table 4.3, we notice that the model accepts one input image and then, pass it into five main convolutional blocks. Afterward, there is a global average pooling layer followed by three dense layers using the "Sigmoid" activation function. The final "Sigmoid_Dense_7" layer used to extract the features vectors of the images and save them in a database. The total number of trainable parameters used for extracting the images' features vectors is (22,124,096).

Afterward, the feature vector of the user query image along with the saved features vectors of all other images in the dataset are used to measure the similarity among them utilizing static distance metrics. The similarity score is defined as the difference between the distances of the two input images features vectors. If the difference is high, then they are not similar, and their generated similarity score will be closer to (0). On the other hand, if the difference in their distance is low. Then they are more similar, and their produced similarity score will be closer to (1).

There are many distance metrics for calculating the difference between the images' feature vectors. We examined the following three well-known distance metrics looking for the one that generates the most accurate evaluation parameters:

- Manhattan (L1)
- Euclidean (L2)
- Cosine

The static formula of the Manhattan distance method, which is also known as City-block distance and $L1$ method, computes the absolute difference between the coordinates of the two input images, as illustrated in equation (4.3) [31].

$$L1(A, \mathrm{B}) = \sum_{l=1}^{L} |A_i - B_i| \tag{4.3}$$

Where $L1$ is the distance between the two features vectors, *A* and *B* are the extracted feature vector from the two input images, *A* is the feature vector of the query image, $A = \{A_0, A_1, \ldots, A_{n-1}\}$ and *B* is the feature vector of the stored image in the ancient Arabic manuscripts dataset, $B = \{B_0, B_1, \ldots, B_{n-1}\}$, n is the number of dimensions in each image's feature vector. *L* is the number of the features in each vector, and i represents the i^{th} feature in the vector.

Regarding the Euclidean distance method, which is also known as the *L2* method. It is the squared root form of the *L1* method. Its equation is presented in (4.4) [1].

$$L2(A, B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2} \tag{4.4}$$

The Cosine distance is a static similarity method that measures the cosine angle between the two input images' feature vectors in which the similarity score between the two images increases as much as the computed cosine angle between their vector decreased [125]. Its equation presented in (4.5) [126].

$$Cos\theta\ (A, B) = \frac{A_i \cdot B_i}{|A_i||B_i|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i}\sqrt{\sum_{i=1}^{n} B_i}} \tag{4.5}$$

After calculating the similarity scores for all the dataset images, the scores are ranked in a descending order, and the most similar top-*k* images to a user query image are retrieved. The mean accuracy is computed to evaluate the retrieval process. Algorithm-

4.1, explains the pseudo-code for calculating the mean accuracy per the top-*k* retrieved images.

Algorithm 4.1 (Calculate the retrieval mean accuracy)

mean accuracy = 0
count = 0
A_q = feature vector of the query image q
B_d = features vectors of all the images in the dataset, $B_d = \{b_1, b_2, b_3, \ldots, b_j\}$
For (*N*), where *N*= dataset size
{ similarity array = distance metric (A_q , B_d)
retrieve top-*k*, where $k \in \{10, 25, 50, 100\}$
count +=1
}
c = number of correct predictions out of *k*
if (c = = total number of images in the manuscript)
{ $accuracy = 1$ }
else
{ $accuracy = c / k$ }
mean accuracy = $\sum_{i=1}^{N} accuracy \ /N$

4.2 Textual-based Image Retrieval

Since our Arabic manuscripts' dataset includes text images, thus, its more logical to recognize the text written inside the images and extract it to be able to retrieve the images according to their textual features. However, there are many challenges in recognizing and extracting the Arabic text written inside the ancient Arabic manuscripts that is because the Arabic language is morphologically complex [110]. Despite the fact that the Arabic characters are difficult to process, the ancient Arabic manuscripts that are handwritten, very old, and having low-quality resolution are much

more difficult to manipulate. These characteristics make them complex to be visualized and being able to retrieve the relevant images correctly. To overcome all challenges associated with the handwritten Arabic language, we set-up the plan illustrated in figure 4.6 for recognizing and extracting the text.

From figure 4.6, we notice that the work starts by uploading the Arabic manuscripts dataset into the "Google Cloud Platform"[8]. After setting-up our data and having them ready to use, we started the following steps, which are explained in detail in separate sections:

1. Text segmentation and recognition
2. Text extraction
3. Cleaning text
4. Convert cleaned text into a feature vector

After extracting the text and preprocess it, we save the cleaned extracted text along with its associated generated textual features vectors in the freely available "MongoDB Atlas"[9] database management system to facilitate manipulating our big data.

4.2.1 Text Segmentation and Recognition

To implement the text segmentation and recognition steps, we leveraged the "CV2/OpenCV"[10], which is an abbreviation for (Open Source Computer Vision Library). It is a software package specialized in machine learning and including more

[8] https://console.cloud.google.com

[9] https://www.mongodb.com/cloud

[10] https://opencv.org/about.html

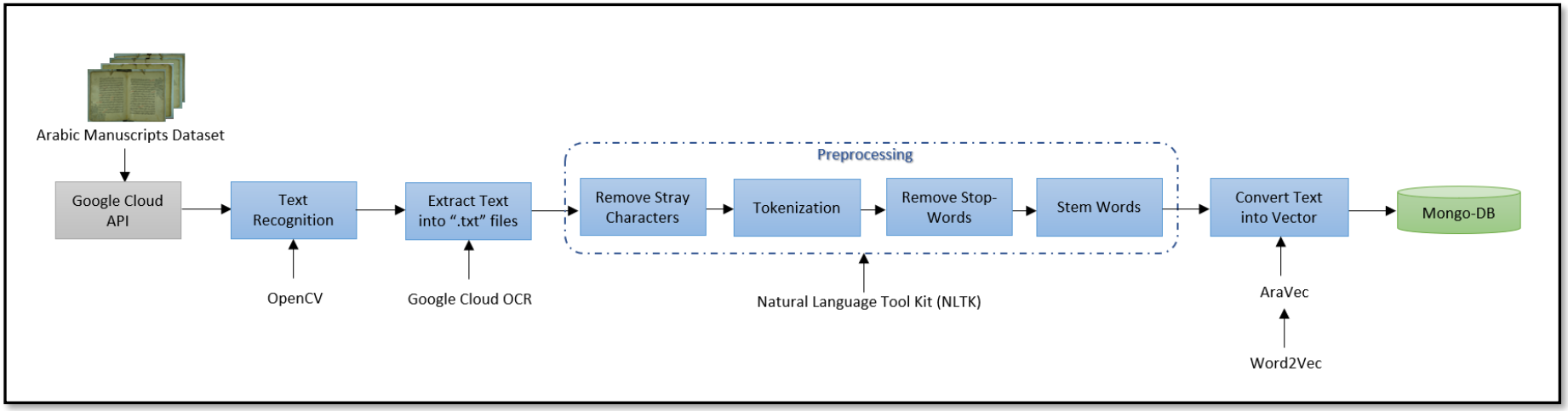


Figure 4.6: Framework for text recognition and extraction

than 2500 enhanced algorithms that facilitate both the physical and organizational structures for various computer vision-based applications [127].

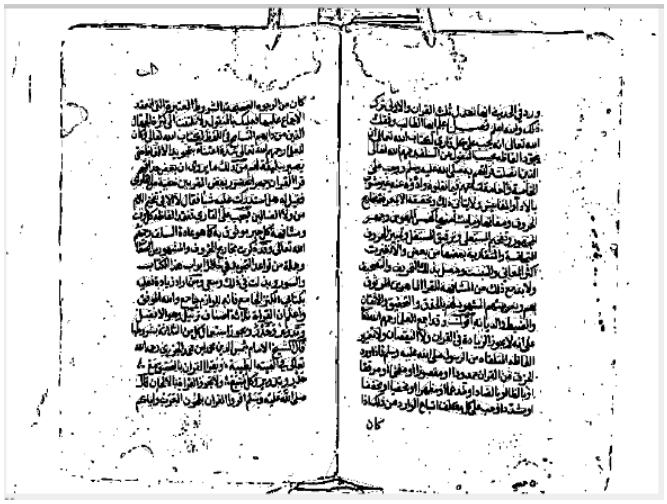
After importing all significant python libraries, we started by segmenting the text-images into lines. Then, segmenting the lines further into words because we need to discover the connected parts within each individual word that represents the Arabic characters/letters. Afterward, we identified the layers of the images to be able to identify the Arabic words written through different layers.

That is because each individual image is extended to more than one layer. Note that partitioning the images into convex polygons only, which performs well with the English language, didn't provide good results with the Arabic handwritten manuscripts. Hence, identifying the layers allowed us to identify written Arabic words through various layers successfully.

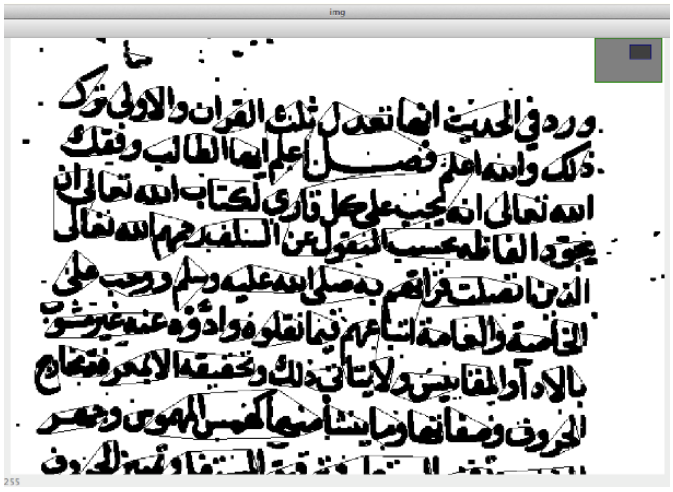
a) Identify the Layers

We employed the Maximally Stable Extremal Regions (MSER) extraction algorithm to get the list of pixels set for each word. We resized the images so that (MSER) can perform better, and we also had to convert the colored images into a grey-scale version because the grey-scale images processed faster than the RGB images. The "DetectRegions" function in the MSER algorithm utilized to detect the regions of each word. Then, we drew a minimum polygon around the detected regions of each identified word using the "ConvexHull" function. Therefore, the ConvexHull highlights the words as polygons, as illustrated in figure 4.7.

(a)



(b)



(c)

Figure 4.7: Polygons forms of words. (a) Original image, (b) Drawn polygons on complete image, (c) Zoomed part of the image

The advantage of being able to recognize the connected components through drawn polygons is to extract words based on their shapes instead of the lines they lay on.

b) Horizontal and Vertical Projection Profile

The structure of words in the ancient Arabic manuscripts' dataset is very difficult to recognize due to the old writers' handwriting styles, as well as, the individual words are spread over multiple lines/rows and not laying on one separate line. Therefore, we had to create a two-dimensional array to detect the maximum peaks in each word and the counterpart minimum peaks. That is done utilizing the "local_maxima" and the "local_minima" functions.

The main purpose of the horizontal projection profile function is to segment images into lines/rows. It is a top-down text-lines recognition method. Keeping in mind that what interests us are the black pixels only because they are the pixels that include words. Furthermore, we concern about the least/minimum peaks in the image because they are the points that correspond to each line boundary. In other words, the bottom/underneath points are the lines between the black pixels. Hence, we can benefit from the local minima points to draw lines between the recognized lines in the original image, as illustrated in figure 4.8.

After lines separation, we can display each individual line, as illustrated in figure 4.9.

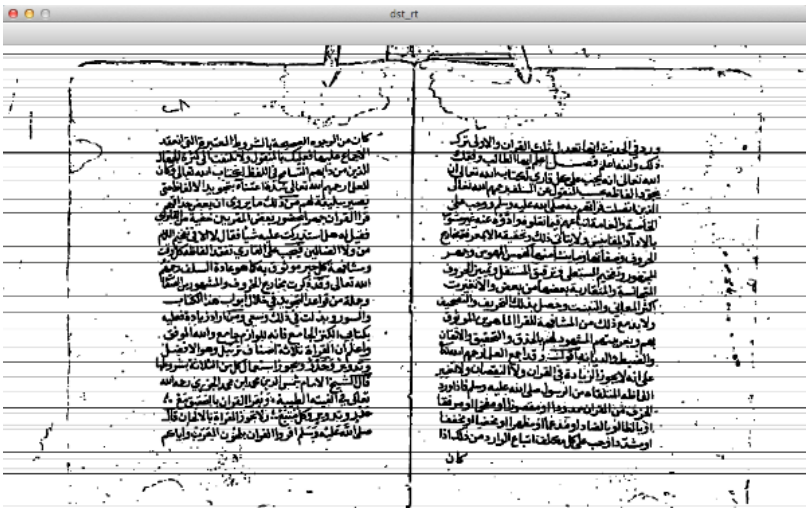
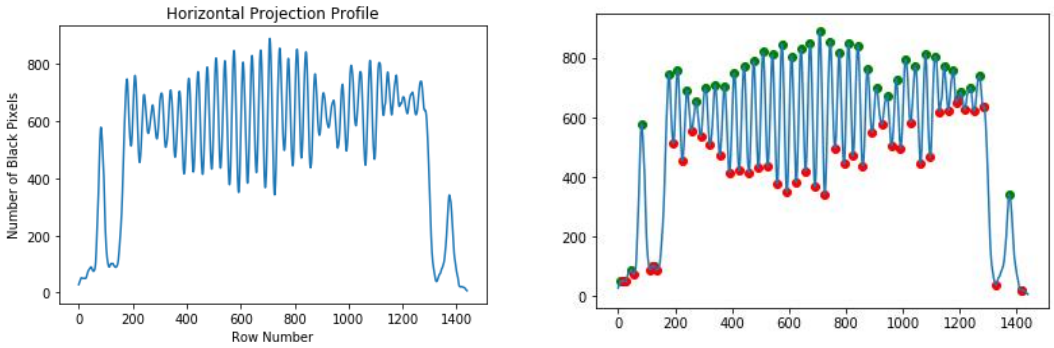


Figure 4.8. Lines display utilizing the local minimum points

Figure 4.9. Separate lines of an image

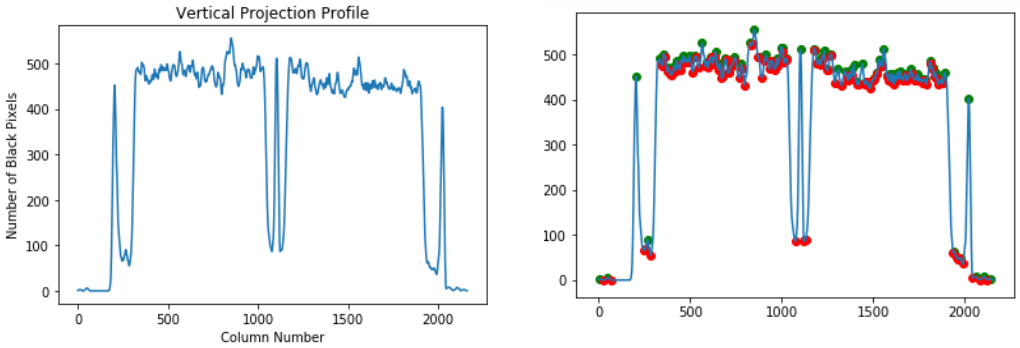
Figure 4.10 illustrates the horizontal projection profile processing on an image. Note that, the left plot clarifies that our segmented image includes 1400 row/line and that the total number of detected black pixels in the image are more than 800 pixels. While the right plot determines lines as peaks whereas the maximum peak is presented by the green color, and the minimum peaks are presented by the red color.



(a) **(b)**

Figure 4.10 Horizontal projection profile. (a) Detected lines, (b) Maximum and minimum lines peaks

Like the horizontal projection profile, we here segment the images into columns instead of rows. Therefore, segmenting the lines further into words since this method determines words as peaks. i.e., it represents the maximum black characters in words as peaks, as illustrated in figure 4.11. Note that there are two big deeps in the middle of the figure corresponding to the separator between the two manuscript pages. Because most of the scanned images in our used dataset include two images, not only one.



(a) **(b)**

Figure 4.11: Vertical projection profile. (a) Detected words, (b) Maximum and minimum words peaks

c) Label Connected Components (Identify Words)

This step of the segmentation phase did label the detected words with numbers to identify them. Thereby, each detected word (black pixel) in the image will have 0 number. In contrast, each detected background (white pixel) in the image will have 255 numbers. We did also smooth the curves in the words to enhance visualizing them. Zooming inside the output result, we can notice the labeled words, as illustrated in figure 4.12.

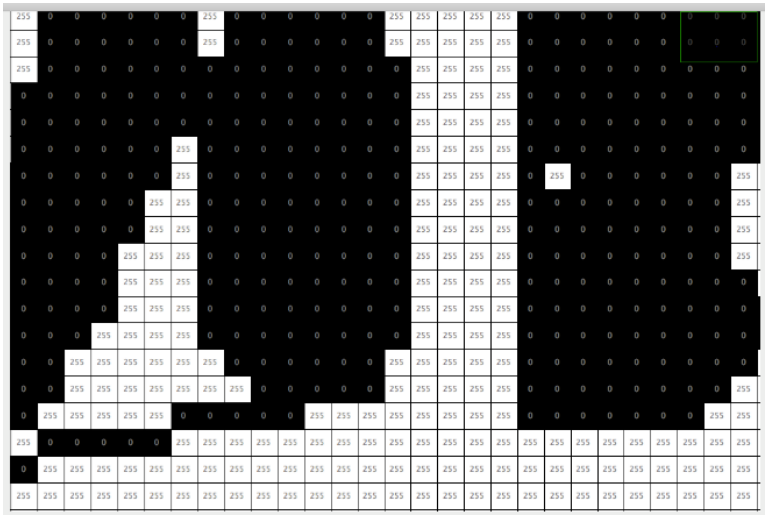


Figure 4.12: Labeled words (numerical representation of the text)

4.2.2 Text Extraction

To extract the text, we need to use an optical character recognition tool that is able to extract the Arabic text correctly. Therefore, we started by experimenting the "tesseract" OCR, which supports the Arabic language, but, it was not accurate in recognizing the handwritten text in the ancient Arabic manuscripts' images. Thus, we employed the "Google OCR Vision" to extract the handwritten Arabic text from each segmented image, and it was accurate. Therefore, we repeated the extraction phase on all the images in our dataset. Finally, the extracted text is saved in a MongoDB file

associated with each individual image. Figure 4.13 illustrates the extracted text from an image. The words highlighted in yellow are correct Arabic words. We notice that a good number of words extracted correctly, which prove the success of the Google OCR tool with the Arabic language.

4.2.3 Cleaning Text

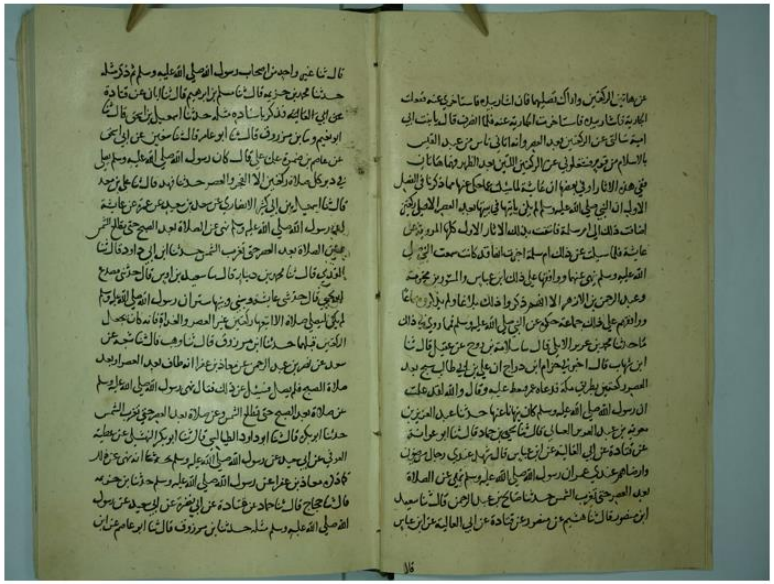
Since our ultimate intention is to retrieve images according to their textual features and not to do sentiment analysis. Then, we can minimize the number of extracted texts further through cleaning the text. This will help in reducing the size of the text and the size of its related feature vector significantly, which will expedite the execution of the code. Utilizing the Natural Language Tool Kit "NLTK 3.4.5"[11], which is a library for natural language processing in conjunction with the "PyArabic 0.6.6"[12] python library specialized in processing Arabic language text, we were able to clean our text through implementing the following steps:

1- Remove the stray characters such as the punctuations, special characters, digits, and non-Arabic letters.

2- Text tokenization, which is a process of splitting the main text into keywords (tokens) according to the existence of the white spaces between the characters to be able to process the tokens further individually.

[11] https://www.nltk.org/

[12] https://pypi.org/project/PyArabic/



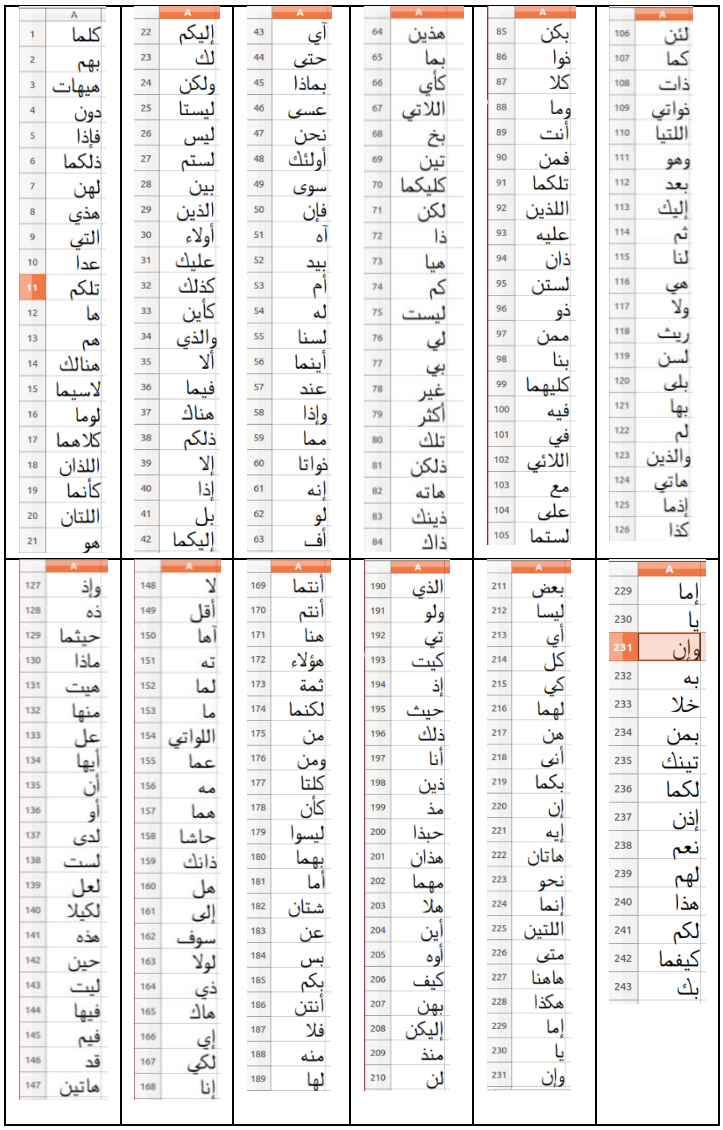
عن هاتين الركعتين وارا لا فان اثاربده فاتاخريعه علت اجاره ناتاري فاتاخرت اجارب عند فلا النقاليات الي امية سالت عن الركيز
عبدالحموانداناتيناسج د القيس بالاسلام مزورخلوني عن الركعتين التنبنجر الظهروماهانان فيهن اللاتا راد في ها ان ا لايدلحنها مما
ذكرنا في العمل الاول ان النيل العلو المانياتها في بيتناسب العمر الاماكن اضات ذلك الارملة فانت تلك الاثار الاول و المروز عاد
فليكن ذلك امس اجتاهاتدات عن ال پي الله عليه وسلم عنها ووافقتها على ذلك ابرعباس والمستورز مجره وعبدالرحمن بن الازهر
الاالضحكوا ذلك ماغاوم مزارون كا ورافقهم على ذلك جاء حكم عن النحل العلوم نيا روي ذلك اماحناح رحرير الاطفال الانروح عنعنا
قال ابن شهاب الاضاحزام ادراج انعلنے طالبجليل العمرركنطري مله ترعاه عرمط علي وقال والله لقتل علت أن رسول الله صلى
المطلومكان ينهانا عنها ح نا عبدالعزيزن محوني مزعل الغرب العالي والا تحاد قال ابوعوانة عن قتادة عن أبي الغالعزازع باسقالارعدي
رحال زن وارضاه عند عمران رسول صل الله علوا من الصلاة لعل العمر حزب المحاماك عبر الامن الناسعد ابز مضور الناهم عن ضرر
عرقتادة عن الي الحالبرابزعاب تالا عني واحد من اصحاب رسول صلى الله عليه وسلمثله حن ا مخربزيزيد فالناما بنابرهم فالغنا ابان عن قتادة
عن ابي العالية تذكريا ساده متاها لاعتقالنا ابراغيم وابنهزرون قال ان ابوعامر قالناسفين عن اراستن عن عام زنم عن عوال كان رسول اهل
اشعلم ورعلى في دبر كل صلاة الفيزا الجوالعصرحا فهد والاعلى نمود والا ايلين الكن الفارينسحلبعبعتعتانة يا رسول التعلي التعلم وم نيز الصلاة
لجدالمحت ظلالئم عن الصلاة لجد الصحرب التحدنا انارداردقانا المودي فالنامون دينار الاسود بزاب الحانصدع البركي الحنى عابدونى داستان
رسول انهالتعليم العين للصلاة الاابا رأتين عن العصر والعراق فاندان نجعل اللتي قلما حدا انرزرق الاوه الناتجهن مدعياض عبدالرحمن معاذزعفا
انعطاف لحد الحصراوجل صلاة الصب قلما لعزة الكفالتى رسول اهلى العلوم عزمه نورالصعصيطل الشروعة لعل المحور الشمس حدنا ابربك قالى
ابوداود الطالبار ابرالها عن عطية العوني عزابيلعن رسول اللااعل وسمحا ان نعلم ادرم معاذبنعناعن رسول العلى العلم واحنا بنحره والاجاج قال
احمدعرقتادة عنابة عن السعيدعن رسول الله صلى الله علم و ملحدنا بن مرزوق قال ابر عام شراب. فال

Figure 4.13: Sample of an image along with its extracted uncleaned text

3- Eliminate the stop words; there is a list of (243) Arabic stop words ready with NLTK corpus, as shown in table 4.4. Thus, we used the list to clean our text. After analyzing the resulted cleaned text, we found more stop words included within our dataset and repeated a lot inside the manuscripts' images while not being included within the NLTK list. This high frequency of stop words emergence within the main text might cause problems in converting the text into vector since the vector size will be huge with no meaningful information. Thereby, we added our stop words list shown in table 4.5 to filter them from the cleaned text too.

4- Stem words using "ISRIStemmer". Words stemming is a technique that retrieves the root of each word. It is helpful for reducing the size of used text while keeping it in its original base form. Hence, it can be used for querying and retrieving images, including similar stemmed text. For example, the stem word of "علوم" is "علم" which is the singular root form of the original word.

Table 4.4: List of NLTK Arabic stop words



كلما	إليكم	أي	هذين	بكن	لئن
بهم	لك	حتى	بما	ذوا	كما
هيهات	ولكن	بماذا	كأي	كلا	ذات
دون	ليستا	عسى	اللاتي	وما	ذواتي
فإذا	ليس	نحن	بخ	أنت	اللتيا
ذلكما	لستم	أولئك	تين	فمن	وهو
لهن	بين	سوى	كليكما	تلكما	بعد
هذي	الذين	فإن	لكن	اللذين	إليك
التي	أولاء	آه	ذا	عليه	ثم
عدا	عليك	بيد	هيا	ذان	لنا
تلكم	كذلك	أم	كم	لستن	هي
ما	كأين	له	ليست	ذو	ولا
هم	والذي	لسنا	لي	ممن	ريث
هنالك	ألا	أينما	بي	بنا	لسن
لاسيما	فيما	عند	غير	كليهما	بلى
لوما	هناك	وإذا	أكثر	فيه	بها
كلاهما	ذلكم	مما	تلك	في	لم
اللذان	إلا	ذواتا	ذلكن	اللائي	والذين
كأنما	إذا	إنه	هاته	مع	هاتي
اللتان	بل	لو	ذينك	على	إذما
هو	إليكما	أف	ذاك	لستما	كذا
وإذ	لا	أنتما	الذي	بعض	إما
ذه	أقل	أنتم	ولو	ليسا	يا
حيثما	آها	هنا	تي	أي	وإن
ماذا	ته	هؤلاء	كيت	كل	به
هيت	لما	ثمة	إذ	كي	خلا
منها	ما	لكنما	حيث	لهما	بمن
عل	اللواتي	من	ذلك	هن	تينك
أيها	عما	ومن	أنا	أنى	لكما
إن	مه	كلتا	ذين	بكما	إذن
أو	هما	كأن	مذ	إن	نعم
لدى	حاشا	ليسوا	حبذا	إيه	لهم
لست	ذانك	بهما	هذان	هاتان	هذا
لعل	هل	أما	مهما	نحو	لكم
لكيلا	إلى	شتان	هلا	إنما	كيفما
هذه	سوف	عن	أين	اللتين	بك
حين	لولا	بس	أوه	متى	
ليت	ذي	بكم	كيف	هاهنا	
فيها	هاك	أنتن	بهن	هكذا	
فيم	إي	فلا	إليكن	إما	
قد	لكي	منه	منذ	يا	
هاتين	إنا	لها	لن	وإن	

Table 4.5: List of our additional filtered Arabic stop words

ما	كان	عند	انه	على	عن	ان	و	في
عليه	له	اذا	لا	مع	كا	ومن	بن	به
من	با	الى	عنه	او	لان	بين	الا	را

4.2.4 Convert Cleaned Text into Feature Vector

To convert the cleaned tokenized text into a feature vector, we used the “AraVec3.0” tool under the “Word2Vec” words embedding tool. “AraVec” is a free pre-trained

word embedding tool specialized in Arabic text processing. The recent version of "AraVec" has been trained on two main sources of Arabic text, which are Twitter and Wikipedia [128]. For our dataset, we employed the unigram model trained on Twitter with 100 vector size.

There are two types of converting text into a vector:

1- One hot representation.

2- Distributed representation.

The one-hot encoding of text converts it into a binary vector that includes only 0's and 1's. On the other hand, the distributed representation of text converts it into different numbers. They can be positive or negative numbers, and this type is the one used in the "AraVec" tool. The distributed representation of words takes into account the frequencies of mutual words that occurred within the text. The mutual information is one of the machine learning algorithms to classify text. It diagnoses the frequent occurrence between two words in a text. The advantage of using the mutual information algorithm over other algorithms is that it increases the classification accuracy [129].

The "AraVec" word embedding tool has its own dictionary of Arabic words. Thus, before converting the raw text into a distributed vector, the tool compares the text with its dictionary, and if the word is not included within the dictionary. This means that it might be extracted in a wrong way, and hence, it will not be converted into vector. This approach ensures that all converted text is accurate. Figure 4.14 illustrates an image along with its cleaned extracted text and the generated distributed feature vector from the preprocessed textual contents.

Algorithm 4.2 describes the way to convert the cleaned text into a textual feature vector.

(500 x 100)

Figure 4.14: Sample of an image along with its extracted cleaned text and feature vector

Algorithm 4.2 (Convert Text into Feature Vector)

Download the "Twitter-CBOW unigram model"[13] version of Aravec tool

Let *N* be the size of our dataset

M is the set of manuscripts in our dataset, $M = \{m_1, m_2, m, \dots\dots, m_i\}$.

W is the number of words in our manuscripts' images,

$W = \{w_1, w_2, w_3, \dots\dots, w_j\}$.

V is the number of words in AraVec vocabulary, $V = \{v_1, v_2, v_3, \dots\dots, v_k\}$.

For (*N*)

{ 1- Load cleaned text files from the dataset.

If ($w \epsilon v$)

{Then the extracted word is correct Arabic word exist within the AraVec vocabularies dictionary}

else

{Then the extracted word is incorrect and will be ignored}

2- Calculate the frequency of correct mutual words using the following Pointwise Mutual Information (PMI) equation:

$$\text{PMI}\ (w, m) = \log \frac{count\ (w,m) * N}{count\ (w) * count\ (m)} \qquad (4.6)$$

3- Generate vector for each correct frequent word using the Aravec model.

4- Write the generated vector for each image into a "pickle", which is a special python-format file used to save and process data easily.

5- Append vectors of all images together into one compressed vector, including each image id along with its feature vector.

6- Save the one complete vector in a "numpy" array on the hard disk.

}

[13] https://github.com/bakrianoo/aravec/blob/master/README.md

4.2.5 Text Classification

After extracting and preprocessing the text, the generated textual features vectors are passed into the Long Short Term Memory (LSTM) deep learning model to classify the extracted textual contents according to specific labels. The LSTM is a special type of the RNN that solves the vanishing gradient problem existed in the RNN through adding memory blocks that are capable of learning long-dependent correlations. The LSTM deep learning model includes hidden state at time step (*t*) denoted as (h_t), as well as, it takes textual content input denoted as (x_t). The input along with the hidden state passes through three LSTM gates named: input (*i*), forget (*f*), and output (*o*) to update the weights (*W*) at a cell activation vector (*c*).

The mathematical representation of the LSTM is presented in the following equations from (4.7) to (4.12) [79]:

$$i_t = \sigma(W_{xi}{}^{x_t} + W_{hi}{}^{h_{t-1}} + b_i) \tag{4.7}$$

$$f_t = \sigma(W_{xf}{}^{x_t} + W_{hf}{}^{h_{t-1}} + b_f) \tag{4.8}$$

$$o_t = \sigma(W_{xo}{}^{x_t} + W_{ho}{}^{h_{t-1}} + b_o) \tag{4.9}$$

$$\tilde{c}_t = tanh(W_{xc}{}^{x_t} + W_{hc}{}^{h_{t-1}} + b_c) \tag{4.10}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{4.11}$$

$$h_t = o_t \otimes tanh(c_t) \tag{4.12}$$

Where *b* refers to the bias vector at the input, forget, and output gates, as well as at the cell activation vector. The $\tilde{c}_t$ refers to the new updated memory cell at time step *t*, and the *t-1* indicates the previous forgotten memory cell. The symbol ⊗ represents the element-wise multiplication operation.

Both σ and *TanH* refers to the Sigmoid and the Hyperbolic Tangent activation functions, respectively, and their computations are presented in equations (4.13) and (4.14) [130]:

$$\sigma\,(x) \;=\; \frac{1}{(\,1+\,e^{-x}\,)} \tag{4.13}$$

$$TanH(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{1-\,e^{-2x}}{1+\,e^{-2x}} \tag{4.14}$$

Table 4.6 illustrates the components of the initial used LSTM deep learning model.

Table 4.6 The components of the LSTM deep learning model

Layer	Output Shape	Parameters No.
Embedding_1	(500, 100)	5000000
Spatial_dropout1d_1	(500, 100)	0
LSTM_1	(500, 64)	42240
Droupout_1	(64)	0
Dense_1	(64)	8256

From table 4.6, we notice that the LSTM deep learning model accepts textual contents of the shape (500, 100). The (500) refers to the maximum average of words in each input sequence or in each input sentence, while the (100) refers to the dimension of the words embedding. The textual contents are passed through the "Spatial dropout" layer to drop the entire one-dimensional feature map. This is helpful for supporting independence among the features vectors before entering them into the LSTM layer. Afterward, the textual features vectors enter the LSTM layer that includes (500) cells or hidden states with a (64) time steps for every hidden state. Then, there is a regular "Dropout" layer to support the independence between the LSTM neural units, which avoids the overfitting problem. Followed by a final "Softmax" classification dense

layer, including the (64) manuscripts' labels that we aim to classify the text according to.

Figure 4.15 illustrates the layered architecture of the initial proposed model for text classification. For simplicity purposes, the dropout layers were eliminated from the model's architecture.

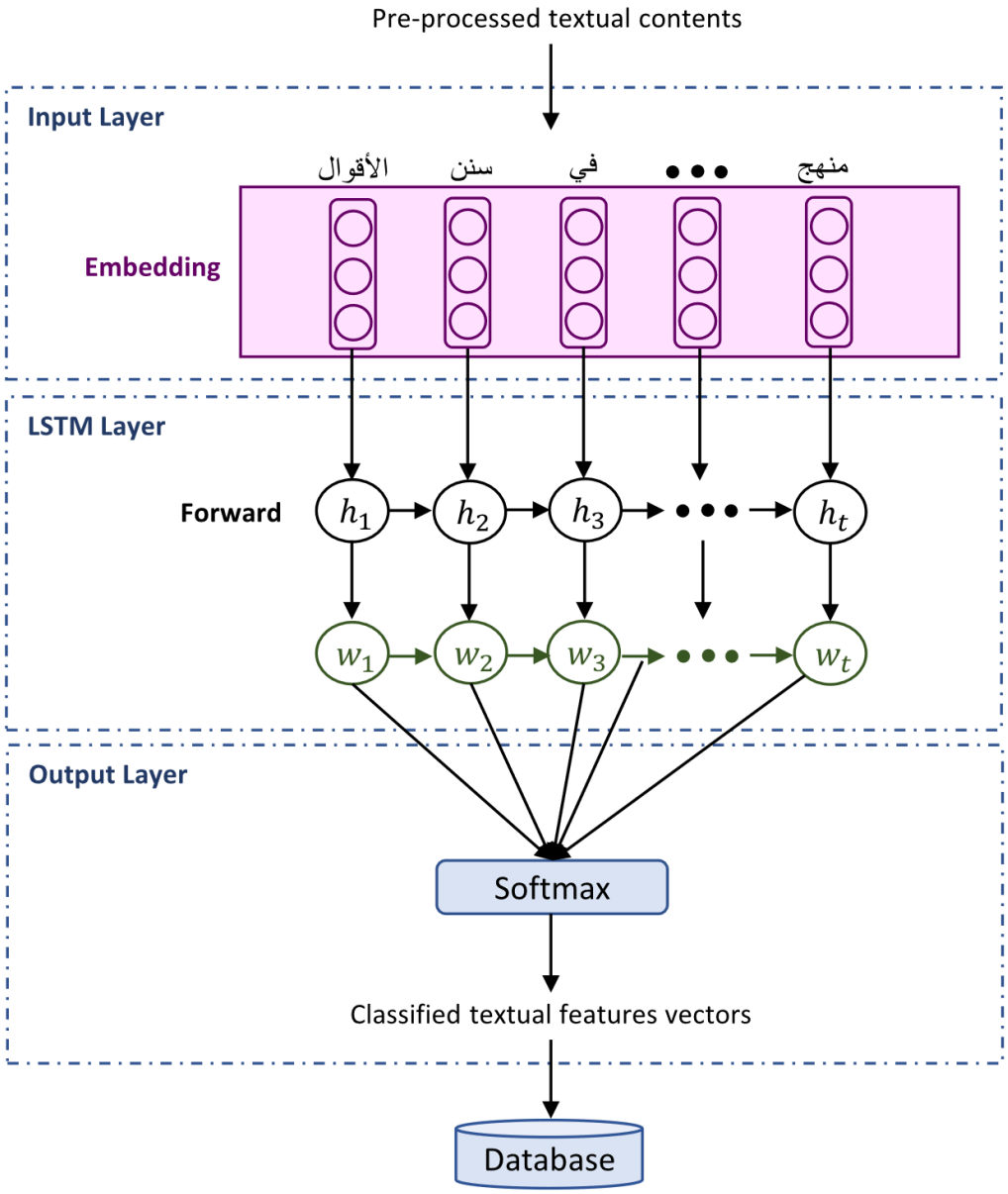


Figure 4.15 The layered architecture of the proposed model for text classification

From figure 4.15, we notice that the cleaned pre-processed textual contents, which are saved in “MongoDB”, enter the embedding layer to generate the features vectors of the text. The features vectors are then passed into the hidden states of the LSTM layer, denoted as $\{h_1, h_2, h_3, \dots, h_t\}$ to generate the weight of each input word, denoted as $\{w_1, w_2, w_3, \dots, w_t\}$. The LSTM layer moves the weights in a forward manner only. The weights are then used by the Softmax dense layer to display the output classified textual features vectors according to specific labels. The features vectors are finally saved in a database for easier future access. The architecture of the initial LSTM deep learning model optimized through implementing the following steps:

1- Make the LSTM layer bidirectional instead of unidirectional (BiLSTM). The bidirectional approach assists in making the model’s weights flow in both forward and backward directions, which allow the model to read the inputs in two ways and increase its memory [78]. The authors in [77], [79], [81], and [108] improved their accuracy using the bi-directional technique.

2- Add the (Attention) layer after the LSTM layer. The attention layer is a custom layer from the “Keras” deep learning library to assist the model, focusing on the important textual contents for a more successful classification process. The “glorot_uniform” distribution is used to initialize the attention layer’s weights. These initial weights get updated with every time step through adjusting the weights exited from the preceding LSTM layer according to the more significant words.

   Many studies handling the sequence classification of words to labels, tend to employ the attention layer to improve the performance of their classification models, such as the studies done in [77 - 83]. With the attention layer, the LSTM

deep learning model focuses on every time step hidden state and updates its weights utilizing the “TanH” activation function followed by the “Softmax” function [76]. This technique stimulates the humans' brains in capturing and focusing the important words only and keep memorizing them for future actions. The mathematical representation of the attention layer is presented in equation (4.15) [131]:

$$Attention\,(Q, K, V) = softmax\,(\frac{QK^T}{\sqrt{d_k}}) * V \qquad (4.15)$$

Where *Q*, *K*, and *V* are three real-number metrics for word embeddings representing the Query (*Q*), the Key (*K*), and the Value (*V*). According to the similarities between the text entries, the QK^T is generated, representing the new weighted matrix. The weighted matrix is then scaled using the $\sqrt{d_k}$ dimensional factor and multiplied by the value matrix to obtain the adjusted attention’s weights.

3- Add Batch Normalization (BN) layer before the final classification dense layer. Adding the batch normalization to the deep learning model has several advantages, one of them is expediting the training process, and more importantly, is increasing the validation accuracy [130]. With every training batch, the batch normalization layer normalizes the weights coming from the previous layers to feed the updated weights to the final classification dense layer. In other words, the batch normalization layer computes both the mean and the standard deviation of the input words to preserve the mean activation value close to (0) and the standard deviation activation value close to (1), which normalizes the weights. The mathematical representation of the batch normalization is presented in equations (4.16) to (4.19) [130]:

$$\mu_{\mathcal{B}} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.16}$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{\mathcal{B}})^2 \tag{4.17}$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{4.18}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{4.19}$$

Where $\mathcal{B}$ refers to a mini-batch including x_i activation values, $\mathcal{B} = \{x_1, x_2, x_3, \dots, x_n\}$, μ_β refers to the mini-batch mean, and $\sigma_{\mathcal{B}}^2$ refers to the mini-batch variance, $\hat{x}_i$ is the normalized activation value, γ is the learned scale parameter, and β is the learned shift parameter for calculating the final Batch Normalization (BN) function. Hence, the $BN_{\gamma,\beta}$ takes the input activation value x_i and normalizes it to get the output activation value y_i.

Afterward, the normalized activation values enter the final "Softmax" classification dense layer to calculate the probabilities of the text classification. The Softmax equation for calculating the probabilities is presented in (4.20) [132]:

$$p(w_t|w_{t-k},\dots w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_{i=1}^{n} e^{y_i}} \tag{4.20}$$

Where p refers to the calculated probability, $\{w_1, w_2, w_3, \dots w_t\}$ refers to the training words matrix, and y_i is the normalized activation value.

Table 4.7 illustrates the optimized components of the LSTM deep learning model. The cells highlighted using the green color are for visualizing the added layers on the initial architecture.

Table 4.7: Optimized components of the LSTM deep learning model

Layer	Output Shape	Parameters No.
Embedding_1	(500, 100)	5000000
Spatial_dropout1d_1	(500, 100)	0
Bidirectional_LSTM_1	(500, 128)	84480
Attention_1	(128)	628
Droupout_1	(128)	0
Batch_normalization_1	(128)	512
Dense_1	(64)	8256

From table 4.7, we notice that after modifying the LSTM layer from unidirectional to bidirectional, the time steps for every hidden state used within the LSTM layer doubled from (64) to become (128). As well as, the number of the LSTM trainable parameters doubled from (42240) to become (84480), which emphasizes the more knowledge the model gained utilizing the bidirectional technique. Figure 4.16 illustrates the optimized layered architecture for text classification.

From figure 4.16, we notice that both the BiLSTM hidden states and weights are doubled compared with the hidden states and weights in figure 3.16. We also notice that the attention layer reads the textual contents again through entering it into the "TanH" activation function and then, multiply it with the output LSTM weights. Thereby, all the output weights from the BiLSTM layer denoted as w_t, get updated to u_t. The updated weights pass into the first "Softmax" activation function to calculate the attention weights denoted as a_t. Afterward, the batch normalization layer takes the attention weights and normalize them to generate the normalized weights denoted as y_t. The final adjusted normalized weights enter the second "Softmax" activation function to output the classified textual features vectors and save them in a database.

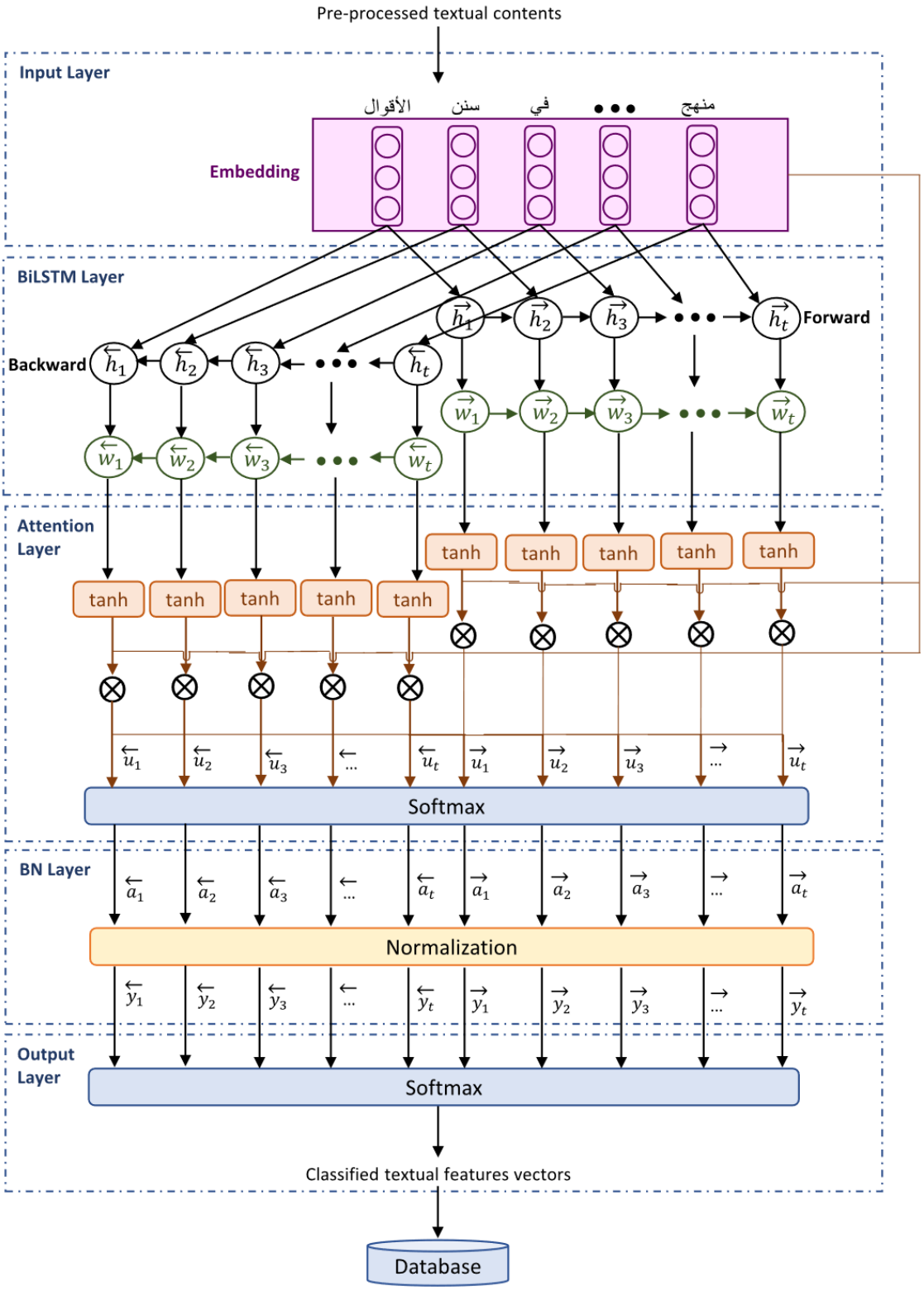


Figure 4.16: The optimized layered architecture for text classification

4.2.6 Similarity Measurement

To measure the similarity among the textual contents, we calculate the distance between the features vectors of all classified text, which is already saved in the database, and the feature vector of the textual contents associated with the user-chosen query image as illustrated in figure 4.17.

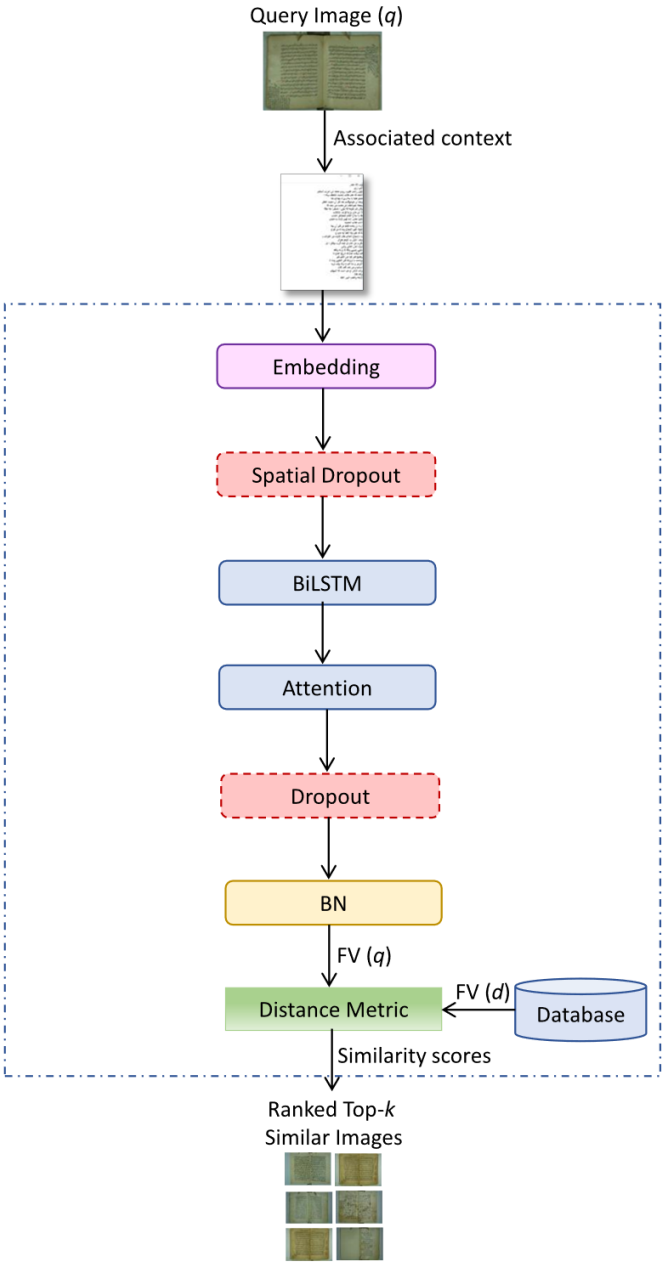


Figure 4.17: The similarity measurement of the classified text

From figure 4.17, we notice that the textual feature vector of the user query image is taken from the Batch Normalization (BN) layer. The distance is then measured between the textual feature vector of the user query image denoted as FV(q) and between all the other textual features vectors saved previously in the database denoted as FV(d). As much as the textual contents are similar, as much as the computed similarity score will be close to (1). In contrast, as much as the textual contents are different, as much as the computed similarity score will be close to (0). The generated similarity scores are ranked in a descending order, and according to the highest measured similarities, the top-k similar images to the user query image will display in the final output result.

4.3 Fusion-based Image Retrieval

Since most of the images are neither purely visual nor completely textual. Thus, fusing both visual and textual models into one fusion model is a crucial step for improving the performance of each model separately [96]. Therefore, we begin in this section by fusing the models into one model for image classification. Afterward, the similarity scores are measured to retrieve similar images using the best-experimented fusion model for image classification.

4.3.1 Image and Text Classification

After extracting the visual features from the images of the ancient Arabic manuscripts' dataset using the pre-trained deep CNN. As well as, after extracting the textual features from the same images using the optimized BiLSTM deep learning model, we experimented the fusion of both models using three different fusion methods named: decision-level fusion, features-level fusion, and score-level fusion; looking for the

most accurate fusion method that increases the classification accuracy of each used model individually.

a) Decision-level Fusion

The decision-level fusion method base on the predicted labels resulted from the final classification of dense layers. Hence, both the inputs and the outputs from the decision-level fusion model are the classified labels. The main purpose of the decision-fusion model is to classify the images according to the new fused labels more accurately. The decision-level fusion model illustrated in figure 4.18 is applied to the predicted (64) manuscripts' labels from both the visual pre-trained CNN and from the textual BiLSTM optimized deep learning model.

From figure 4.18, we notice that the fusion model accepts two inputs. The first input of shape (224 x 224) colored pixels representing the images. While the second input of shape (500 x 100) representing the textual contents. Then, the model takes the predicted outputs from both final "Softmax" classification dense layers in the visual model and in the textual model to merge them in one decision-level fusion model that predicts the labels of both models. The outputs from the decision-level fusion model are the fused (64) manuscripts classified labels.

To develop the decision-level fusion model, a merged layer should be added after getting the visual and textual classified labels. Thus, we developed the model using four different merge layers as following: concatenate, maximum, average, and multiply. Equation (4.21) to equation (4.24) illustrates the mathematical representation of the merge layers.

$$Concatenate = VCL, TCL \tag{4.21}$$

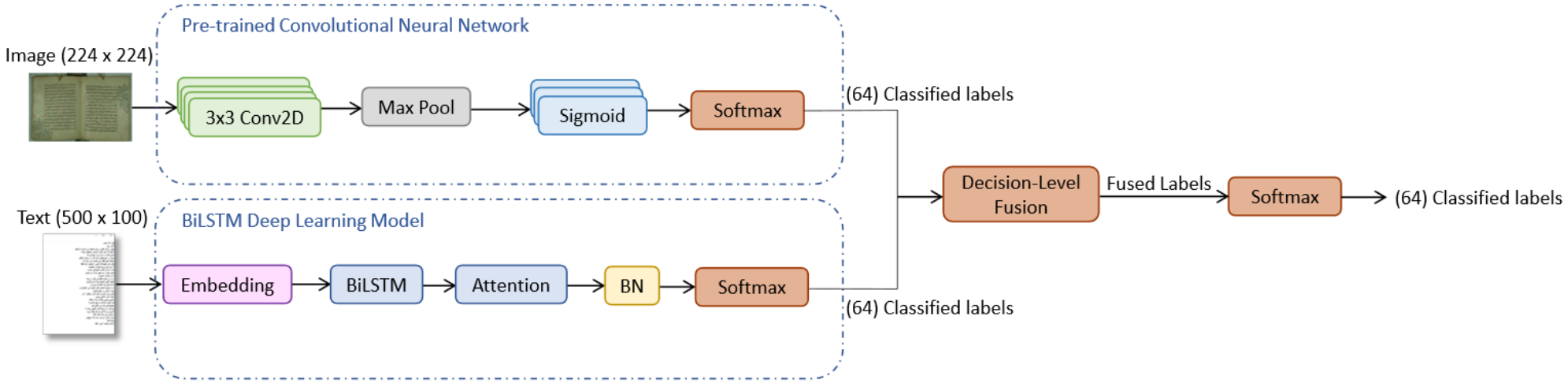


Figure 4.18: Decision-level fusion model

$$Maximum = Max_{i=1}^{N}(VCL, TCL) \quad (4.22)$$

$$Average = \frac{\sum_{i=1}^{N}(VCL+TCL)}{N} \quad (4.23)$$

$$Multiply = VCL * TCL \quad (4.24)$$

Where *VCL* represents the classified visual labels, *TCL* represents the classified textual labels, and *N* is the total number of labels.

b) Features-level Fusion

The features-level fusion method is based on fusing the extracted features from the deep learning models. Hence, both the inputs and the outputs from the features-level fusion model are the extracted features. The main purpose of the features-level fusion model is to obtain new fused features that are better characterizing the images. The features-level fusion model illustrated in figure 4.19 is applied to the extracted visual and textual features from both the pre-trained CNN and the BiLSTM deep learning models.

From figure 4.19, we notice that the fusion model takes the extracted visual features from the pre-trained CNN with (512) dimensions. As well as, it takes the extracted textual features from the optimized BiLSTM deep learning model with (128) dimensions to concatenate them in one feature-level fusion model. Therefore, the output shape from the concatenation layer equals (512 visual feature vector + 128 textual feature vector = 640 fused feature vector). The "Softmax" classification layer is added after the concatenation layer to predict the labels of the fused features from both the visual and the textual models. The outputs from the features-level fusion model are the classified (64) manuscripts labels according to the fused features vectors.

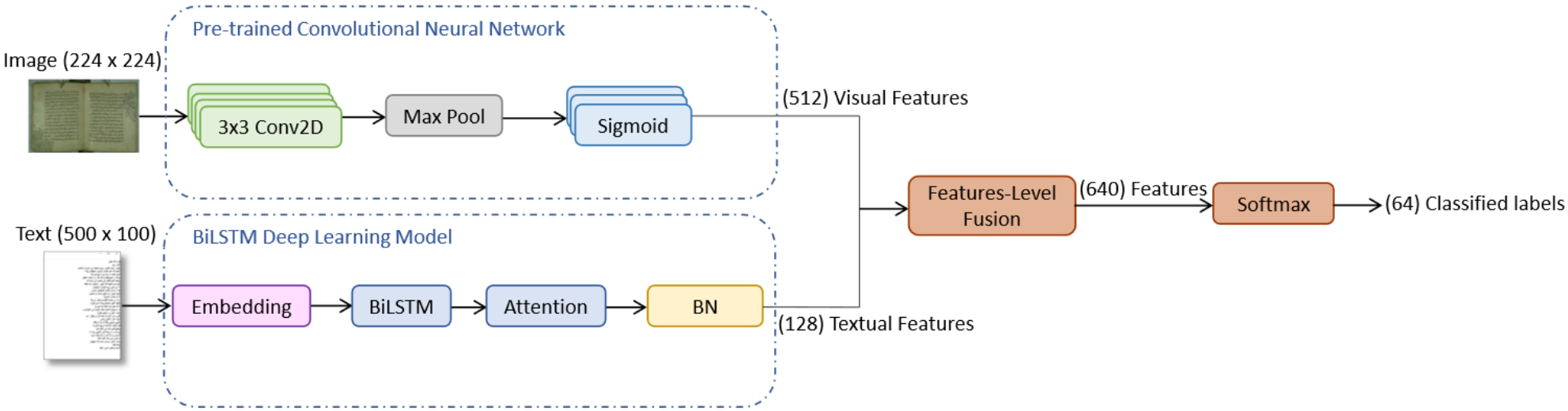


Figure 4.19: Features-level fusion model

c) Score-level Fusion

The score-level fusion method accepts the probability scores from the visual and from the textual models' classifiers. Then it fuses the scores into one fusion model using a rule to better classify the images according to the fused scores. The classifiers' probability scores are heterogeneous because one classifier is visually based, while the other is textually based. Thus, it is necessary to normalize the scores before performing the fusion to keep them within a similar numerical scale [133].

The score-level fusion model illustrated in figure 4.20 is applied to the generated Visual Probability Scores (VPS) from the Softmax classifier of the pre-trained CNN, as well as on the generated Textual Probability Scores (TPS) from the Softmax classifier of the optimized BiLSTM deep learning model. According to Chitroub [133], the scores obtained from the classifiers' probabilities are including the substantial details of the input data.

From figure 4.20, we notice that the score-level fusion model accepts visual and textual inputs to generate the VPS from the CNN classifier and the TPS from the BiLSTM classifier. These probabilities are generated using the "model.predict()" built-in function from "Keras" deep learning library to predict the probabilities of the inputs to the models. Afterward, the generated probability scores are normalized using the "TanH" score normalization function, as illustrated in equation (4.25) [100]:

$$(PS)' = 0.5 * TanH * \frac{0.01 * (PS - PS^{G}_{Mean})}{PS^{B}_{SD}} + 1 \qquad (4.25)$$

Where $(PS)'$ is the normalized probability score, wither it's visual or textual. PS^{G}_{Mean} is the mean estimation of the genuine probability score distribution, and PS^{B}_{SD} is the standard deviation estimation of both the genuine and the impostor probability score distributions.

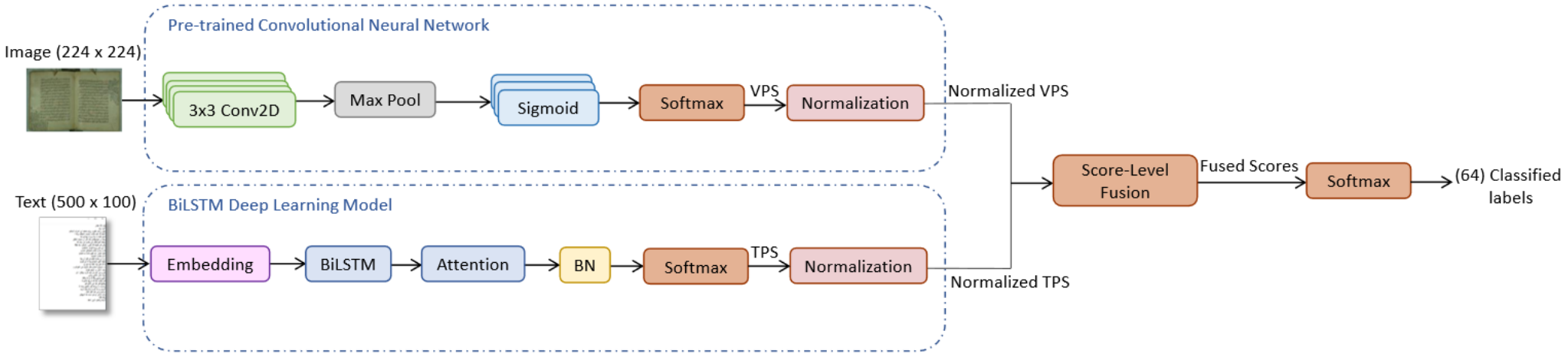


Figure 4.20: Score-level fusion model

After normalizing the VPS and the TPS, they are fused using one score-level fusion model. To do the fusion, we should apply some rules to the scores, such as the max rule, min rule, or sum rule. Equation (4.26) to equation (4.28) [100] illustrates the mathematical equations of the fusion rules.

$$Max\_rule = Max_{i=1}^{N}(PS_i)' \quad (4.26)$$

$$Min\_rule = Min_{i=1}^{N}(PS_i)' \quad (4.27)$$

$$Simple\_sum\_rule = \sum_{i=1}^{N}(PS_i)' \quad (4.28)$$

Where *N* is the total number of the final normalized probability scores.

4.3.2 Similarity Measurement

After classifying the images, the top-*k* similar images to a user query image are retrieved through fusing the similarity scores from both the visual and the textual models, as illustrated in figure 4.21.

From figure 4.21, we notice that the image retrieval step is performed on three phases. Two phases are accomplished offline. The first offline phase generates the Visual Features Vectors from the entire dataset images using the deep pre-trained CNN, denoted as VFV(*d*), and saves them in a database. Similarly, the second offline phase generates the Textual Features Vectors from the entire dataset context using the optimized BiLSTM deep learning model, denoted as TFV(*d*), and saves them in a database.

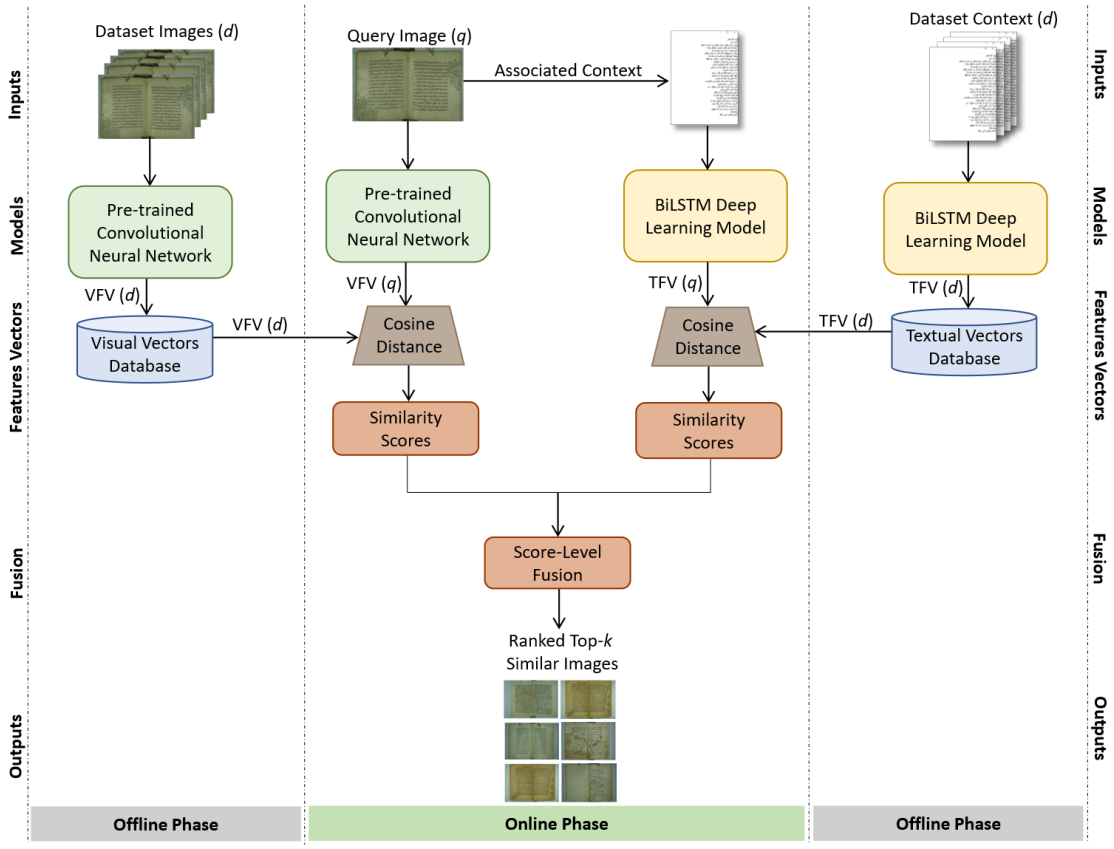


Figure 4.21: Similarity measurement using the score-level fusion model

During the online phase, the user enters a query image. Then, its associated text will be retrieved to get both the visual and the textual features vectors of the query image. Afterward, the model generates two different similarity scores, one from the visual model and the other from the textual model. The scores are fused using the sum-rule score-level fusion model to obtain one fused similarity score from both visual and textual models. The final fused similarity scores are then ranked in a descending order to retrieve the top-k similar images from the ranked list.

4.4 Image Retrieval using the DRL model

The image retrieval task requires a dynamic and interactive search system [62]. Therefore, we formalize the image retrieval problem as a sequential decision-making

problem, where the agent is considered the search engine. The environment sends an image by image to the agent for retrieving the most similar images to each query image. To solve the retrieval problem, the Deep Reinforcement Q-learning Network (DRQN) is used. It involves an end-to-end ordered interaction between the environment and the agent.

Table 4.8 lists the main used components along with their definitions.

Table 4.8: The definitions of the DRQN components

Component	Definition
1) Environment	Includes the dataset, $D = \{(I_1, C_1), (I_2, C_2), ..., (I_n, C_n)\}$.
2) Agent	Select and retrieve the similar images in terms of features utilizing the K-NN algorithm.
3) State ($\boldsymbol{S_t}$)	The fused extracted features from the visual and the textual models.
4) Action ($\boldsymbol{a_t}$)	Array of the indices of the relevant retrieved images after computing the distance using the Cosine metric.
5) Policy ($\boldsymbol{\pi}$)	Boltzmann exploration policy.
6) Reward ($\boldsymbol{r_t}$)	The average computed distance of relevant retrieved images.
7) Observation ($\boldsymbol{o_t}$)	Array of the indices of the irrelevant retrieved images.

From table 4.8, we notice that the deep reinforcement learning approach is based on seven main components, which are explained in detail as following:

1) **Environment**: it includes the dataset, which consists of the images along with their associated textual contexts, $D = \{(I_1, C_1), (I_2, C_2), \ldots., (I_n, C_n)\}$.

   The environment is created using the "gym" reinforcement learning library. Once, a new environment is initiated, then it has access to the dataset images and their associated textual contents. After creating the environment, build an environment's constructor. At each time step (t), the constructor shuffles all the

train images and select one image randomly. The visual features of the selected image are extracted using the pre-trained VGG19 CNN. Then, the textual contents of the selected image are recalled to extract the textual features from them using the BiLSTM deep learning model. Both extracted visual and textual features are fused into one features-level fusion model using the concatenation merge layer from Keras deep learning library. Then, the fused features are sent to the agent. The following algorithm illustrates the approach of sending images from the environment to the agent.

Algorithm 4.3 (Send Images to the Agent)

Let *n* be the total number of train images

Create Environment using “gym”

Build Environment’s Constructor

Create Search Session

For (*n*)

{

1. Shuffle images and select one image at time step *t* randomly.
2. Extract the visual features from the selected images using VGG19.
3. Take the corresponding textual content of the selected image.
4. Extract the textual features from the retrieved contents using BiLSTM deep learning model.
5. Fuse both extracted features using the concatenate layer.

6. Send the fused features to the search agent

}

---

2) **Agent**: works as a search engine that selects and retrieves the most similar images to the sent image's features from the environment. Hence, the agent has access to the fused features being sent. Once the agent receives states from the environment, it takes actions as correct as possible utilizing the KNN algorithm to classify and measure the distances using the Cosine distance metric, trying to reach the optimal relevant list of images. Each search episode ends when the agent's goal is reached, which is finding the top-k similar images correctly.
3) **State (S_t)**: the fused extracted features from the visual and the textual models of each chosen image at a discrete-time step (t).
4) **Action (a_t)**: compute the distance between the entered query image and each retrieved image using the Cosine distance metric associated with the KNN algorithm. If the computed distance is less than the threshold, then save the index of the image in the action array because it is relevant image and belong to the same manuscript as the query image. Otherwise, the image is irrelevant.

   Considering that the goal is to maximize the cumulative rewards to reach the optimal action-value function (Q^*), which is computed as following [18]:

$$Q^*(s_t, a_t) = Q(s_t, a_t) + \alpha\,[r_t + \gamma \max_{\acute{a}} Q(\acute{s_t}, \acute{a_t}) - Q(s_t, a_t)] \quad (4.29)$$

   Where $Q(s, a)$ is the initial action-value function, α refers to the learning step size, r is the environment's reward based on the chosen action, γ refers to the

discount factor $0 \leq \gamma \leq 1$, $\acute{s}$ is the newly updated state sent from the environment, and $\acute{a}$ is the new updated action taken by the agent. Equation (4.30) clarifies the computation of the initial action-value function [58]:

$$Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, \boldsymbol{a}) \tag{4.30}$$

Where $\boldsymbol{a}$ is the set of all actions that the agent can choose from.

5) **Policy ($\boldsymbol{\pi}$)**: the Boltzmann exploration policy[14] is used to direct the agent with a set of rules to follow. The policy works with a discrete action space. Hence, it receives the predicted actions from the agent as probabilities and converts them into distributions representing the true action that will be applied to the environment.

   The implementation of the Boltzmann exploration policy on an action at a time is illustrated in equation (4.31) [18]:

$$\pi_t(a) = \frac{e^{P_t(a)}}{\sum_{i=1}^{n} e^{P_t(i)}} \tag{4.31}$$

   Where P_t is the computed probabilities, and *n* is the total number of images in the dataset.

6) **Reward ($\boldsymbol{r_t}$)**: according to the action taken by the agent, which is the list of all relevant images to the user query image, the average of the relevant images is returned as a reward to the agent's action. Hence, the environment rewards the agent with the average of the retrieved relevant images distances. The reward is a session-based computed value that gets updated with every new episode that includes a new query image (state).

[14] https://nervanasystems.github.io/coach/_modules/rl_coach/exploration_policies/boltzmann.html#Boltzmann

The following algorithm 4.4, explains the computation of the reward.

Algorithm 4.4 (Reward Computation)

Let *k* be the number of the total retrieved similar images.

r_t is the reward from the environment to the agent at the time step *t*.

Ƭh = 0.2, is the distance threshold.

For (*k*)

{

If (Cosine distance = (1 – Cosine similarity) < = Ƭh)

{ set the image as relevant image (*RI*)}

else

{add the image's index to the observation array}

}

$$r_t = \frac{\sum_{i=0}^{k} distances\ (RI)}{k} \tag{4.32}$$

7) **Observation (o_t)**: The indices of all images that are irrelevant (not from the same manuscript as the query image) are saved in an observation array (o_t) in the experience replay memory to assist the agent in the learning process.

Figure 4.22 illustrates the architecture of the proposed method.

From figure 4.22, we notice that the developed model to retrieve the images consists of six main steps as following:

1. Start from the environment, which has the dataset images and their related contexts. At each discrete time step, the environment enters a random shuffled image (I_t) into the pre-trained deep VGG19 CNN to extract the visual features from it. Moreover, the corresponding textual features of the image (C_t) are extracted using an attentional BiLSTM deep learning model.

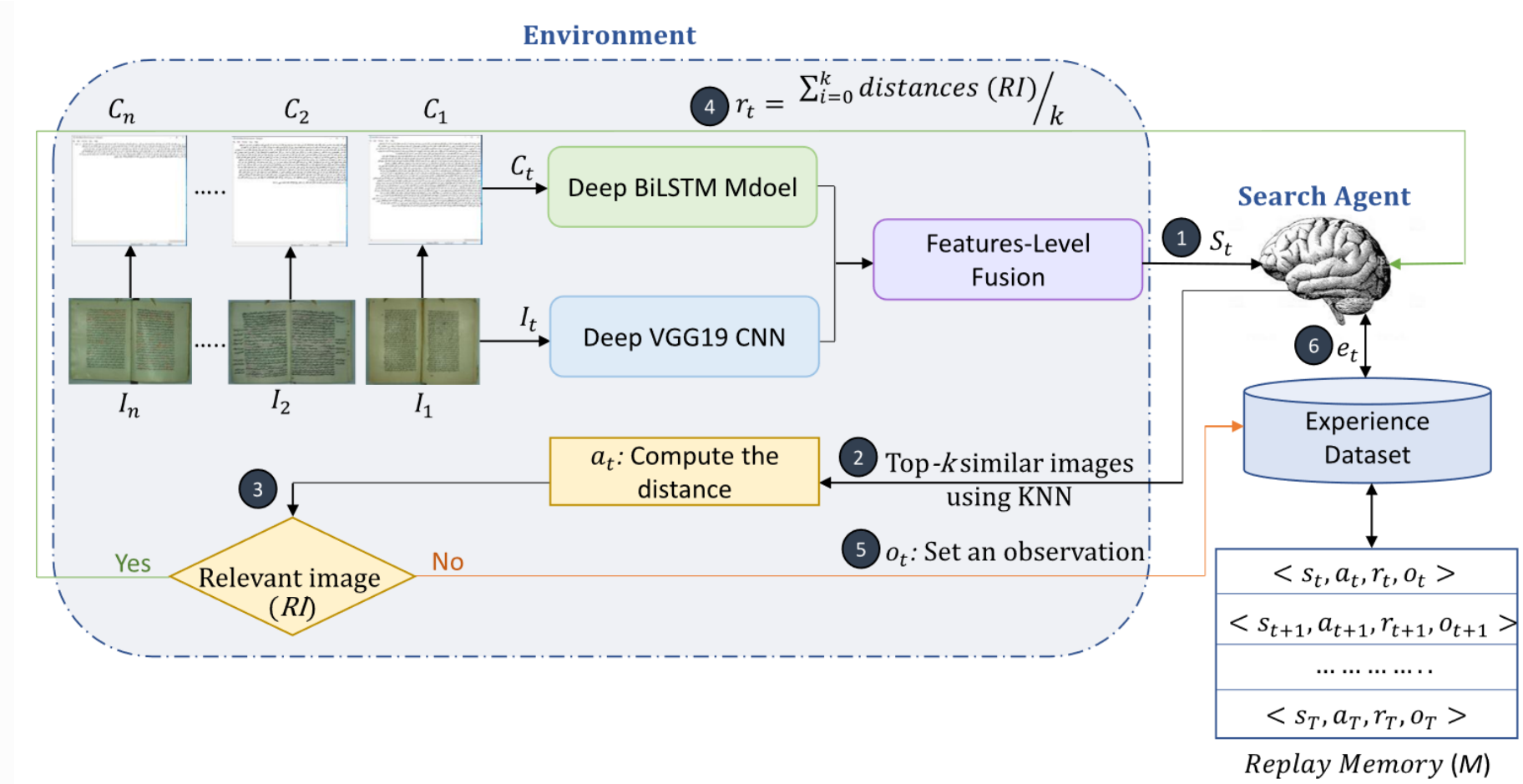


Figure 4.22: Framework of the proposed DRL model for IR

The features from both the visual and textual models are fused using the concatenation merge layer to generate the state (S_t).

2. Once the agent receives a state from the environment, it uses the KNN algorithm and follows the policy rules while checking the previous saved experience as an attempt to take the correct action. To create the agent's experience dataset, we store all the previously taken actions as an experience in a replay memory. Note that, the first action is taken with no previous saved experience. i.e., The $e_t(a) = 0, \forall\ a$. However, as much as the agent is trained, as much as it builds a better experience depending on the previously taken actions as illustrated in the following equation [18]:

$$e_{t+1}(a) = e_t(a) + \alpha\ (r_t)\ \pi_t(a) \tag{4.33}$$

3. After the agent retrieves the top-k similar images utilizing the KNN algorithm, the distance is computed between the query image and the rest of the retrieved images to assign the proper reward to the agent's taken action.
4. If the retrieved images by the agent are from the same manuscript as the user's query image, then the agent gets a reward equals to the average of distances of the relevant retrieved images.
5. In contrast, if the retrieved image is from a different manuscript than the user's query image, then the environment saves the retrieved image's index in an observation array.
6. The detailed information of the reached decisions are saved in the replay memory (*M*) to assist the agent in learning more efficiently through time.

4.4.1 DRL Model Enhancement

The evaluation results from the initial developed DRL model were not very high. Thus, we enhanced the model through hashing the generated fused features to reduce the dimensionalities among them, which assist in improving the clustering of the images belonging to the same manuscript. The hash function is one of the most popular solutions for approximating the nearest neighbor search [134]. That is because it makes the environment more stable by removing the randomness in the search process. The hash function was suggested by many researchers to improve image retrieval accuracy [135-143]. In addition, the authors of paper [58] and paper [75] recommended combining the DRL technique with the hashing technique to increase the image retrieval accuracy. Therefore, the initially proposed method for the images' retrieval increased one step, as illustrated in figure 4.23.

From figure 4.23, we notice that the retrieval steps became seven instead of six. The added step is to take the fused visual and textual features of the images and hash them. Then, send the generated reduced hashed features to the search agent. This step removes the randomness existed in the environment and assisted the agent in clustering and identifying similar images more accurately. The proposed hashing function is the Locality Sensitive Hashing (LSH) Forest method.

The author in [144] recommended using LSH Forest to hash textual data as it improves the retrieval accuracy. The LSH Forest is a data-independent, unsupervised learning method that clusters the images in groups according to their similarities. It receives the fused features as input and converts them into binary hash codes after reducing the dimensionalities among them. Thus, the images from the same manuscript will have similar hash codes.

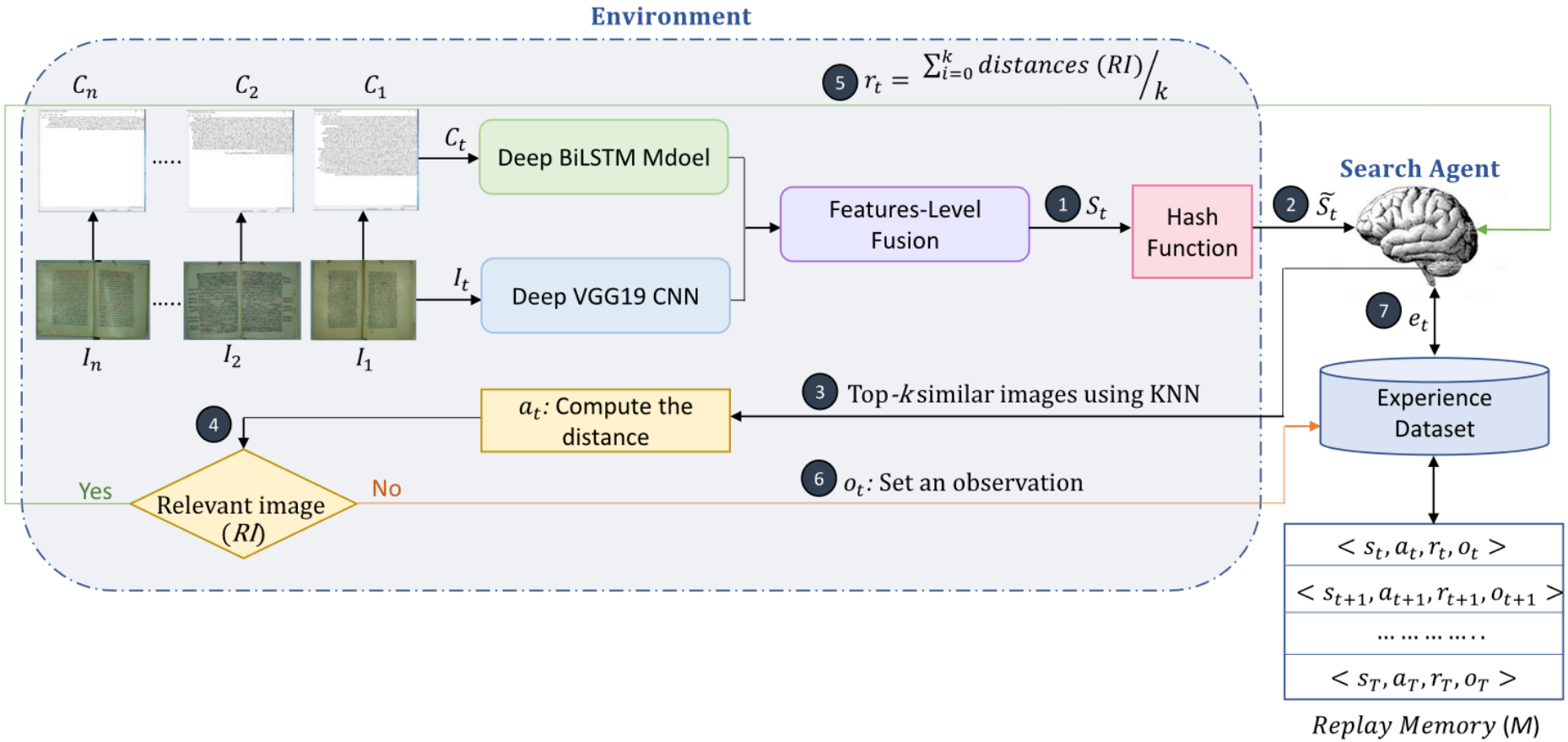


Figure 4.23: Enhanced DRL model including the hash function for IR

The mathematical representation of the LSH Forest method utilizing the min hash codes is presented in equation (4.34)[133]:

$$\widetilde{S_t} = argmin_{g=1}^{2^G} h_g^T . S_t \quad (4.34)$$

Where $\widetilde{S_t}$ is the new binary hashed features generated from the original fused features S_t. 2^G is the size of the randomly generated projection hyperplane $h = \{h_1, h_2, h_3, \ldots, h_{2^G}\}$. Figure 4.24 illustrates an example of the testing phase by the developed model. From figure 4.24, we notice that the testing phase includes entering the user's query image to the model to retrieve the most similar images to it. Therefore, the text associated with the entered query image is recalled to extract both the visual and the textual features of the image. Then, fuse them using one features-level fusion model. The fused features are then hashed using the LSH Forest method to reduce the dimensionalities among the features and remove the environment's randomness. The hashed fused features are passed into the search agent to select and retrieve the most similar images to the user query image using the K-NN algorithm. During the agent's search process, it checks its experience replay memory, which was built during the training phase as an attempt to take the correct action as possible. The Cosine distance is measured between the query image and each retrieved image to decide their relevancy. Considering that the distance threshold is set to (0.2), then all the retrieved images within the distance threshold are relevant and belonging to the same manuscript as the query image. In contrast, all the retrieved images with a distance that is greater than the threshold are irrelevant since they from different manuscripts than the true manuscript of the query image. Note that, the distance of the first retrieved image in figure 4.24 is (0.000), meaning that the model retrieved the exact same user's query image.

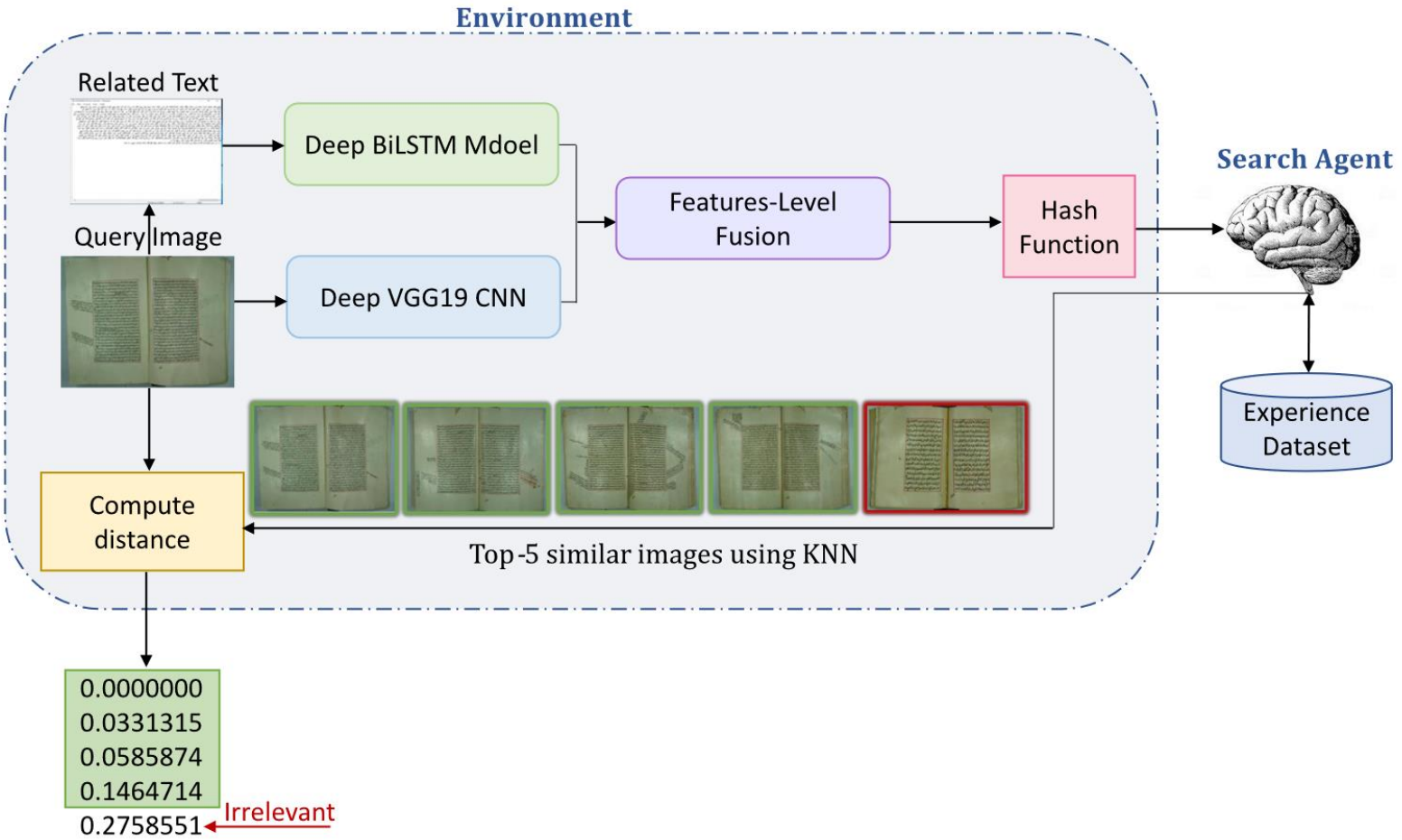


Figure 4.24: The testing phase of the enhanced DRL model

Chapter V

The Experimental Results and Discussion

This chapter begins by explaining the used hardware and software for conducting the experiments. Followed by illustrating the mathematical equations used for the model evaluation. Afterward, we start experimenting the image classification and retrieval according to the images' deep visual features. Within the visual features experiments, we first experiment the best categorization of the dataset. Then, we tune the learning hyperparameters through experimenting three hypotheses and test a range of five various values from each hypothesis on four different deep learning models named: MobileNetV1, DenseNet201, ResNet50, and VGG19 to increase the classification accuracy. In addition, we experiment the image classification and retrieval according to the deep textual features. Moreover, we test various fusion methods for fusing both the visual and the textual models into one model to reach the optimized image retrieval system. Finally, we experiment the developed DRL model to assess its performance in retrieving the Arabic manuscripts' images.

5.1 Used Hardware and Software

To conduct our experiments, we implemented the models using the Python programming language version 3.7 and the Jupyter notebook web application

interface on Ubuntu 16.04 Operating System. Note that “Tensorflow” and “Keras” were two of the main deep learning libraries that we used at the backend.

Concerning the used hardware device, it is the “ABS Battelbox” PC, including Intel Core i7-9700K 3.60 GHz with eight-core processors and Nvidia GeForce RTX 2080.

5.2 Performance Evaluation Metrics

To evaluate the classification models, we recorded the generated accuracy by each model. The equation for calculating the accuracy evaluation metric presented in (5.1) [71]:

$$Accuracy = \frac{S_{cw}}{T_w} \tag{5.1}$$

Where S_{cw} represents the number of successfully predicted images and T_w represents the total number of images.

Moreover, we evaluated the effectiveness of the developed deep learning models by computing the Precision (*P*), Recall (*R*), and the F-score (*F-score*). Following equations (5.2) - (5.4) illustrate their computations [29]:

$$P = \frac{number\ of\ correctly\ retrieved\ images}{total\ number\ of\ retrieved\ images} \tag{5.2}$$

$$R = \frac{number\ of\ correctly\ retrieved\ images}{total\ number\ of\ relevant\ images\ in\ the\ dataset} \tag{5.3}$$

$$F\text{-}score = 2 * (P * R) | (P + R) \tag{5.4}$$

We depended on both the validation accuracy and the average F-score metrics in evaluating the performance of our conducted experiments. That is because there is a trade-off between the recall and the precision. However, the F-score metric combines the measurements of both the recall and the precision [145]. Thus, we can rely on it

as a trustable general evaluation parameter for evaluating the developed deep learning models.

For the image retrieval task, we computed the mean Average Precision (mAP). Hence, first the Average Precision (AP) for each query until position *N* is computed as in equation (5.5):

$$AP = \frac{1}{GT} \sum_{i=1}^{N} \frac{TP}{i} \tag{5.5}$$

Where *GT* are the ground truth labels, and *TP* are the true positive labels. Afterward, the mean of the computed average precision is calculated as illustrated in equation (5.6):

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{5.6}$$

5.3 Image Classification and Retrieval According to the Visual Features

5.3.1 Dataset Categorization

The dataset divided into three main categories as the following: training, testing, and validation. The training portion of the dataset should always contain the largest amount of data to train the model successfully. Therefore, we experimented three different ratios of the datasets as follows:

- Allocate 60% from the original size of the main dataset for the training purpose and divide the rest 40% equally between the testing and the validation subsets.
- Assign 70% from the data for the training and split the rest 30% equally between the testing and the validation subsets.

- Allocate 80% from the data for the training and split the remaining 20% evenly between the testing and validation.

Afterward, we transferred learning from the pre-trained MobileNetV1 deep CNN to extract the visual features in the images and to classify them according to the Arabic calligraphies. We assigned a small learning rate that equals (1e-3) to allow the model to learn the extracted features more efficiently. Then, we trained the model by running the learning algorithm ten times (10 epochs).

We ran the same customized pre-trained model and re-used the same hyperparameters on all the three datasets ratios to ensure the fairness of implementation. Eventually, we recorded the Precision (P), Recall (R), and the F-score (F) per each calligraphy and for the three experimented datasets ratios, as presented in table 5.1. Note that the highest generated results were highlighted by bold, and the lowest generated results were highlighted by the red colour for easier visualization.

Table 5.1: Evaluation parameters per each calligraphy

Calligraphy ID	60% Training, 40% Testing and Validation			70% Training, 30% Testing and Validation			80% Training, 20% Testing and Validation		
	P	R	F	P	R	F	P	R	F
1	0.0000	0.0000	0.0000	0.9756	0.9756	0.9756	**1.0000**	0.9726	**0.9861**
2	0.6042	0.9355	0.7342	0.9318	0.9762	0.9535	0.9375	0.8955	0.9160
3	0.6610	0.9512	0.7800	0.9474	0.9231	0.9351	0.9275	0.9697	0.9481
4	0.7295	**0.9674**	0.8318	0.9750	0.9512	0.9630	0.9571	0.9853	0.9710
5	**1.0000**	0.8750	0.9333	0.9762	**1.0000**	0.9880	0.9853	0.9571	0.9710
6	0.9714	0.9315	**0.9510**	**1.0000**	0.9783	**0.9890**	0.9643	**1.0000**	0.9818

From table 5.1, we notice that the 60% training and 40% testing and validation was the worst-performing ratios of datasets. That is because the first calligraphy was having (0.0000) for all the evaluation parameters, which means that the model was not able to predict the first calligraphy and classify the images according to it. That is most likely due to the small size of the training dataset, which didn't allow the

model to see any images written using the first calligraphy, and hence, it was not capable of recognizing it.

However, there was a fluctuation between the highest recorded evaluation metrics. Since the highest recorded precision was by the fifth calligraphy, which was 100% recognized and successfully classified by the model. While the highest recorded recall was by the fourth calligraphy as (0.9674), and the highest recorded F-score was by the last calligraphy as (0.9510). We conclude that the best classified Arabic calligraphy using the 60% training and 40% testing and validation were the last three calligraphies because they generated the highest metrics. While the worst classified Arabic calligraphy was the first one named “Al-Nask”.

The second experimented ratios of the datasets, which are 70% training and 30% testing and validation performed very well since it recorded (1.0000), which is the best result we can achieve, under the precision evaluation parameter of the last calligraphy. As well as, it recorded (1.0000) under the recall for the fifth calligraphy. The model also generated the best F-score for the last calligraphy as (0.9890).

Regarding the lowest recorded results, they were all above 90%, which affirms the effectiveness of the deep learning model in accurately learning and classifying the calligraphies using the 70% training and 30% testing and validation datasets ratios.

We conclude that the best classified Arabic calligraphy using the 70% training and 30% testing and validation was the last one since it recorded the highest precision and F-score. In contrast, the worst classified Arabic calligraphy was the third one because it includes the smallest recall and F-score.

Increasing the size of the training portion from the Arabic manuscripts dataset to become 80% instead of 70%, it also generated good satisfying results. Sine, the first calligraphy recorded the highest precision as (1.0000) and the highest F-score as

(0.9861). Moreover, the last calligraphy recorded the highest recall as (1.0000). In contrast, the lowest recorded precision was by the third calligraphy as (0.9275). While the lowest recorded recall and F-score was by the second calligraphy as (0.8955) and (0.9160), respectively.

Hence, we conclude that the best classified Arabic calligraphies using the 80% training and 20% testing and validation were the first one named “Al-Nask” and the last one named “Al-Farsi”. On the other hand, the worst classified Arabic calligraphy was the second one named “Al-Thulth”.

Eventually and after analyzing the results generated using the three different ratios of datasets, we admit that the 60% training and 40% testing and validation was the worst-performing categorization. On the other hand, both the other categorizations of datasets were generating better results.

To confirm our reached conclusion and to more accurately assess the performance of our developed model, we computed the final Validation Accuracy (VA), Average Precision (AP), Average Recall (AR), and Average F-Score (AF) for each one of the three experimented datasets ratios as illustrated in table 5.2.

Table 5.2: Computed metrics for different datasets ratios

Evaluation	60% Training, 40% Testing, and Validation	70% Training, 30% Testing, and Validation	80% Training, 20% Testing, and Validation
VA	0.8164	**0.9688**	0.9375
AP	0.6610	**0.9677**	0.9619
AR	0.7768	**0.9674**	0.9634
AF	0.7051	**0.9673**	0.9623

Based on the results in table 5.2, we notice that the 60% training and 40% testing and validation generated the lowest results for all the evaluation parameters. In contrast, the 70% training and 30% testing and validation generated the highest results for all

the metrics. Therefore, the generated results proved that the ultimate dataset categorization is 70% for the training portion and 30% for both the testing and the validation portions. Thus, we used this categorization for the rest of the experiments.

5.3.2 Modulating the Learning Hyperparameters While Classifying the Image According to the Author

The dataset images resized into (224 x 224) pixels to prepare them for entering the deep CNNs and then, classified using four pre-trained deep CNNs as following:

- MobileNet_V1_100_244
- ResNet_V2_50
- DenseNet_201
- VGG_19

The experiments started by executing the four deep CNNs ten times (10 epochs), utilizing (1e-3) learning rate. In addition, we used one final classification dense layer that includes "Softmax" activation function with “adam” optimizer and the “weighted categorical cross-entropy” loss. We used a global shuffling buffer while building the “TF” record, which is a zipped simplified version of our collected dataset. As well as, we performed another local shuffle of (64) buffers while doing the training on our data. We trained the models on the same dataset and used the same batch size as (32). Table 5.3 summarizes the results from the initial execution of the four pre-trained deep CNNs for classifying the images according to the authors.

Table 5.3: Initial image classification according to the author

Evaluation	MobileNetV1	ResNet50	DenseNet201	VGG19
VA	0.3542	0.4006	0.2957	**0.8737**
AP	0.2167	0.2402	0.1589	**0.8371**
AR	0.3716	0.3932	0.2972	**0.8533**
AF	0.2516	0.2775	0.1896	**0.8362**

From table 5.3, we notice that all the models were not able to perform well in classifying the images according to the authors, except VGG19 deep CNN. That is because all the models classified only around 30% of the images successfully, except VGG19, which classified approximately 80% of the images according to the authors correctly. We also notice that the DenseNet201 deep CNN was the weakest in classifying the images since it generated the lowest results among the other tested deep CNNs. Therefore, we set a goal to modulate and tune the primary hyperparameters essential in the learning process of the deep CNNs to reach the best strategy for recognizing the visual features of the ancient Arabic manuscripts' images and classifying them successfully. Hence, we experimented three hypotheses to reach the best evaluation metrics.

Hypothesis (1): Minimizing the learning rate allows the model to learn slowly, and hence it will improve the learning process.

The learning rate is the step size in seeking images within the dataset to get trained on them. Therefore, it shouldn't be too small, either too large to enable the deep learning model to learn effectively with a suitable speed in memory [146].

To test the correctness of the hypothesis, we conducted new experiments that employ different learning rates ranging from 1e-2 (0. 01) to 1e-6 (0.000001). The generated results are summarized in table 5.4.

Table 5.4: Learning process through different learning rates

Evaluation	1e-2	1e-3	1e-4	1e-5	1e-6
	MobileNetV1				
VA	0.3542	0.3678	0.3710	0.3750	**0.4135**
AP	0.2167	0.2199	0.2212	0.2299	**0.2653**
AR	0.3716	0.3790	0.3861	0.3779	**0.4047**
AF	0.2516	0.2490	0.2685	0.2688	**0.2965**

Table 5.4: Learning process through different learning rates (Continued)

	ResNet50				
VA	0.3349	0.4006	0.4087	**0.4415**	0.4295
AP	0.1859	0.2402	0.2526	**0.2884**	0.2583
AR	0.3545	0.3932	0.4162	**0.4598**	0.4431
AF	0.2247	0.2775	0.2973	**0.3398**	0.3110
	DenseNet201				
VA	0.3165	0.2957	0.3357	0.3253	**0.3438**
AP	0.1609	0.1589	0.1890	0.1914	**0.2072**
AR	0.3219	0.2972	0.3346	0.3366	**0.3506**
AF	0.1888	0.1896	0.2230	0.2289	**0.2408**
	VGG19				
VA	0.8478	0.8737	0.8622	0.8686	**0.9431**
AP	0.8331	0.8371	0.8936	0.8577	**0.9184**
AR	0.8631	0.8533	0.8559	0.8801	**0.9304**
AF	0.8418	0.8362	0.8588	0.8638	**0.9217**

Analysis and findings from table 5.4 results:

1. Even though there is a little bit fluctuation in the results, three deep CNNS (MobileNetV1, DenseNet201, and VGG19) recorded the highest evaluation parameters at (1e-6) learning rate. The average F-score recorded by MobileNetV1 deep CNN was 0.2965, and the average F-score recorded by DenseNet201 deep CNN was 0.2408, as well as, the average F-score recorded by VGG19 deep CNN was 0.9217. Hence, we can claim that the hypothesis holds true, and we will use (1e-6) for the rest of the experiments.
2. All the four deep CNNs recorded the lowest evaluation parameters at (1e-2) and (1e-3) learning rates. That is because the lowest average F-scores were 0.2490 and 0.8362 recorded at (1e-3) by MobileNetV1 and VGG19 deep CNNs, respectively. While the lowest average F-scores were 0.2247 and 0.1888 recorded at (1e-2) by ResNet50 and DenseNet201 deep CNNs, respectively. Thus, we shouldn't use fast learning rates for training our deep CNNs.

To easily visualize the improvements in the learning process, we drew the F-score values of the four models at the different learning rates in Figure 5.1.

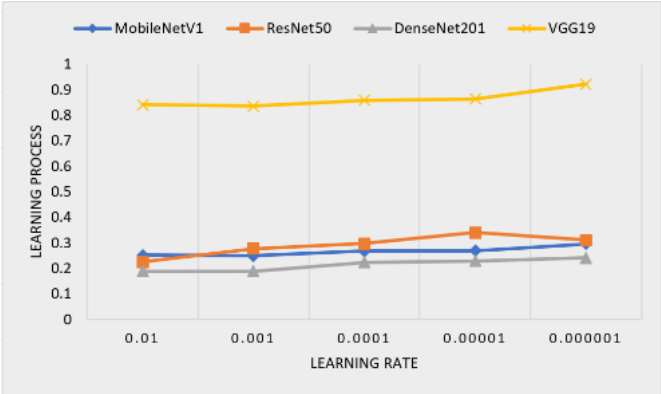


Figure 5.1: Visualizing the learning process through different learning rates

In general, there were no considerable improvements in the classification results after minimizing the learning rate. Thus, we had to tune another learning hyperparameter that is crucial to the models' operation. Hence, we made all the models deeper through increasing their layers in the next hypothesis.

Hypothesis (2): Increasing the number of final classification dense layers improve the classification accuracy.

To test the correctness of this hypothesis, we added more classification dense layers before the formerly existing "Softmax" layer, denoted as (F). The cases we tested are as following:

- Add "ReLU" dense layer, denoted as (R)
- Add "Sigmoid" dense layer, denoted as (G)
- Add both "ReLU" and "Sigmoid" dense layers
- Add two "ReLUs" and one "Sigmoid" dense layer

We set the number of neurons in all added new classification dense layers to (256). Table 5.5 summarizes the generated results.

Table 5.5: Learning process through different classification layers

Evaluation	F	F + R	F + G	F + R + G	F + 2R + G
	MobileNetV1				
VA	0.4135	0.2804	0.9511	**0.9631**	0.9495
AP	0.2653	0.1521	**0.9613**	0.9611	0.9526
AR	0.4047	0.2859	0.9576	**0.9581**	0.9486
AF	0.2965	0.1804	0.9555	**0.9578**	0.9454
	ResNet50				
VA	0.4295	0.3197	0.9583	**0.9631**	0.9567
AP	0.2583	0.2019	**0.9670**	0.9578	0.9607
AR	0.4431	0.3249	**0.9665**	0.9576	0.9600
AF	0.3110	0.2266	**0.9655**	0.9569	0.9599
	DenseNet201				
VA	0.3438	0.0176	0.9399	**0.9599**	0.9327
AP	0.2072	0.0004	0.9567	**0.9634**	0.9340
AR	0.3506	0.0192	0.9555	**0.9629**	0.9308
AF	0.2408	0.0009	0.9539	**0.9685**	0.9300
	VGG19				
VA	0.9431	0.9471	**0.9583**	0.9551	0.9503
AP	0.9184	0.9459	**0.9659**	0.9543	0.9497
AR	0.9304	0.9441	**0.9657**	0.9548	0.9504
AF	0.9217	0.9439	**0.9647**	0.9539	0.9492

Analysis and findings from table 5.5 results:

1. Making the convolutional neural networks deeper through adding two “ReLUs” and one “Sigmoid” classification dense layers didn’t record the highest results in any one of the four tested deep CNNs. That is because the recorded average F-scores were 0.9454, 0.9599, 0.9300, and 0.9492 by MobileNetV1, ResNet50, DenseNet201, and VGG19 dep CNNs respectively, which were not the highest recorded values. Thus, we can’t claim that this hypothesis holds true.
2. Both MobileNetV1 and DenseNet201 recorded their highest results when we added one “ReLU” and one “Sigmoid” classification dense layers before the

existing “Softmax” classification layer. MobileNetV1 CNN recorded 0.9578, and the DenseNet201 CNN recorded 0.9685 average F-scores, which were the highest recorded F-scores by both models during the entire experiments. On the other hand, both ResNet50 and VGG19 deep CNNs recorded their most top results when we added the “Sigmoid” classification dense layer before the existing “Softmax” layer. ResNet50 CNN recorded 0.9655, and the VGG19 CNN recorded 0.9647 average F-scores, which were the highest recorded F-scores by both models during the entire experiments. Thus, we recommend adding the “Sigmoid” activation function either alone or with the “ReLU” activation function before the original “Softmax” dense layer since it had the highest effects on the results.

3. Three deep CNNs recorded the lowest results when we used the “ReLU” classification dense layer before the existing “Softmax”, which were MobileNetV1, ResNet50, and DenseNet201. In fact, the DenseNet201 deep CNN decreased its evaluation parameters dramatically since it recorded 0.0009 average F-score when using the “ReLU” classification dense layer before the existing “Softmax”. In addition, MobileNetV1 recorded 0.1804, and ResNet50 recorded 0.2266 average F-scores, which were the lowest recorded F-scores by both models. Thereby, it is not recommended to combine between “ReLU” and “Softmax” alone.

To highlight the improvements in the learning process, we drew the F-score values of the four models utilizing the different number of final classification dense layers in figure 5.2.

Since all the four utilized deep CNNs reached higher than 90% successful classification of the images according to the authors after adding the “Sigmoid”

classification dense layer, we accomplished satisfying results. The final recorded validation accuracy of the MobileNetV1 deep CNN raised from 41.35% to 95.11%.

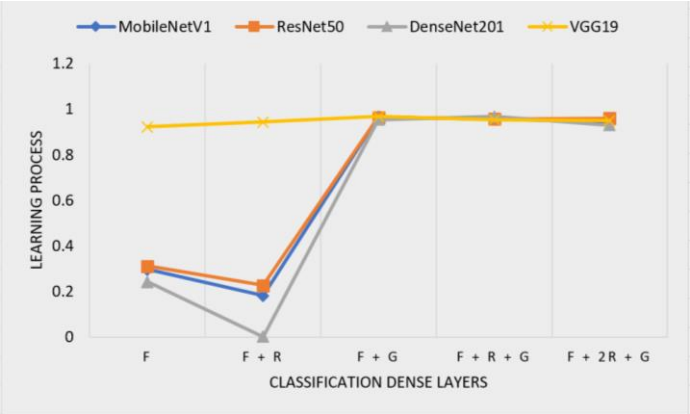


Figure 5.2: Visualizing the learning process through different dense layers

Similarly, the validation accuracy of both ResNet50 and VGG19 deep CNNs increased from 42.95% and from 94.31%, respectively, to become 95.83%. Moreover, the validation accuracy of the DenseNet201 deep CNN raised from 34.38% to 93.99%. Thus, we will use “Sigmoid” in addition to the existing “Softmax” classification dense layer for the rest of the experiments. But we want to conduct one more examination that tests the effects of increasing the neurons number on the added classification dense layer.

Hypothesis (3): Increasing the number of neurons on the last classification layer enhances the learning performance.

To test the correctness of this hypothesis, we increased the number of neurons in the new added "Sigmoid" classification dense layer from (64) to (1024) neurons. The generated results are summarized in table 5.6.

Table 5.6: Learning process through different number of neurons

Evaluation	64	128	256	512	1024
	MobileNetV1				
VA	0.9239	0.9447	0.9511	**0.9639**	0.9559
AP	0.9401	0.9557	0.9613	0.9559	**0.9599**
AR	0.9366	0.9553	0.9576	0.9552	**0.9584**
AF	0.9356	0.9549	0.9555	0.9545	**0.9566**
	ResNet50				
VA	0.9463	0.9471	0.9583	0.9511	**0.9623**
AP	0.9477	0.9518	0.9670	0.9543	**0.9698**
AR	0.9459	0.9516	**0.9665**	0.9538	0.9663
AF	0.9457	0.9508	**0.9655**	0.9494	0.9646
	DenseNet201				
VA	0.9022	0.9431	0.9399	0.9511	**0.9583**
AP	0.9116	0.9398	0.9567	0.9490	**0.9637**
AR	0.9115	0.9345	0.9555	0.9487	**0.9637**
AF	0.9083	0.9347	0.9539	0.9477	**0.9606**
	VGG19				
VA	0.9367	0.9583	0.9583	**0.9663**	0.9591
AP	0.9532	0.9652	0.9659	0.9609	**0.9710**
AR	0.9528	0.9649	**0.9657**	0.9613	0.9652
AF	0.9524	0.9647	0.9647	0.9605	**0.9649**

Analysis and findings from table 5.6 results:

1. All the four tested deep CNNs slightly raised their evaluation parameters by increasing the number of neurons from (64) neurons to become (1024) neurons. The MobileNetV1 deep CNN raised its recorded F-score from 0.9356 to 0.9566. Similarly, The ResNet50 deep CNN increased its recorded F-score from 0.9457 to 0.9646. DenseNet201 deep CNN raised its recorded F-score from 0.9083 to 0.9606, as well as, VGG19 deep CNN raised its recorded F-score from 0.9524 to 0.9649. These slight improvements in the results allow us to admit that the hypothesis holds true.
2. All the four tested deep CNNs recorded the lowest evaluation parameters using the (64) neurons number. That is because the recorded final validation accuracies were 0.9239, 0.9463, 0.9022, and 0.9367 by MobileNetV1, ResNet50,

DenseNet201, and VGG19 deep CNNs respectively. Hence, we recommend not to use this low number of neurons on the last classification dense layer.

To simplify the visualization of the reached results, we summarized the generated F-score of the four models during the used different number of neurons in figure 5.3.

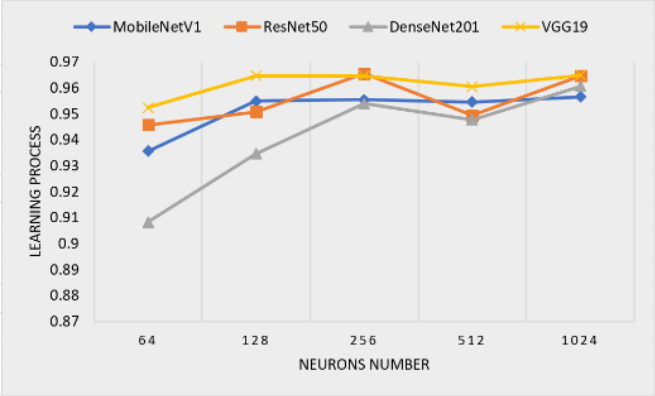


Figure 5.3: Visualizing the learning process through various neurons number

From figure 5.3, we notice that there is a little bit fluctuation in the final recorded F-score. However, all the models improved their performance after increasing the neurons number to (1024) neurons on the final classification "Sigmoid" dense layer. Therefore, we reached a successful classification of the images according to the authors.

After experimenting the effects of various learning hyperparameters on the performance of the four deep neural networks, we conclude that the best strategy to follow on developing our deep CNN utilizing our ancient Arabic manuscripts is as following:

- Employ (1e-6) for the learning rate since it allowed the models to learn the extracted features more slowly, and that made them more knowledgeable.
- Add "Sigmoid" before the original existing "Softmax" classification dense layer as they produced high results.

- Use (1024) neurons in the added “Sigmoid” final classification dense layer.
- Furthermore, we found out that running the learning cycles (10) epochs saved our time and accomplished satisfying results.

From the conducted experiments, we noticed that initially, the accuracy was low in all the models except the VGG19 deep CNN. There was a massive difference in the results between the VGG19 deep CNN and the other three deep CNNs. However, after the wise and careful tuning of the main learning hyperparameters, we were able to increase the evaluation parameters for all the CNNs, which optimized their performance and generated accuracies that were all above 95% successful classification.

After reaching the best strategy for developing our deep CNNs, we compare the four models through computing their precision, recall, and F-score for each author, as illustrated in table 5.7. This comparison is conducted to ensure that we reached our goal, which is confirming that all the tested four deep CNNs are performing well in classifying the images according to the authors using the tuned learning hyperparameter.

From table 5.7, we notice that the four deep CNNs were able to 100% successfully classify a close number of authors out of the existing (52) Arabic authors in our dataset. For instance, the MobileNet model classified (21) authors. ResNet50 model classified (23) authors, DenseNet201 model classified (26) authors, and VGG19 model classified (24) authors. In addition, we notice that none of the authors were completely miss-classified, which validates the effectiveness of the utilized strategy in developing the pre-trained deep CNNs. After reaching and validating the effectiveness of the best strategy to develop our models, we generated the confusion metrics for each model, as illustrated in figure 5.4.

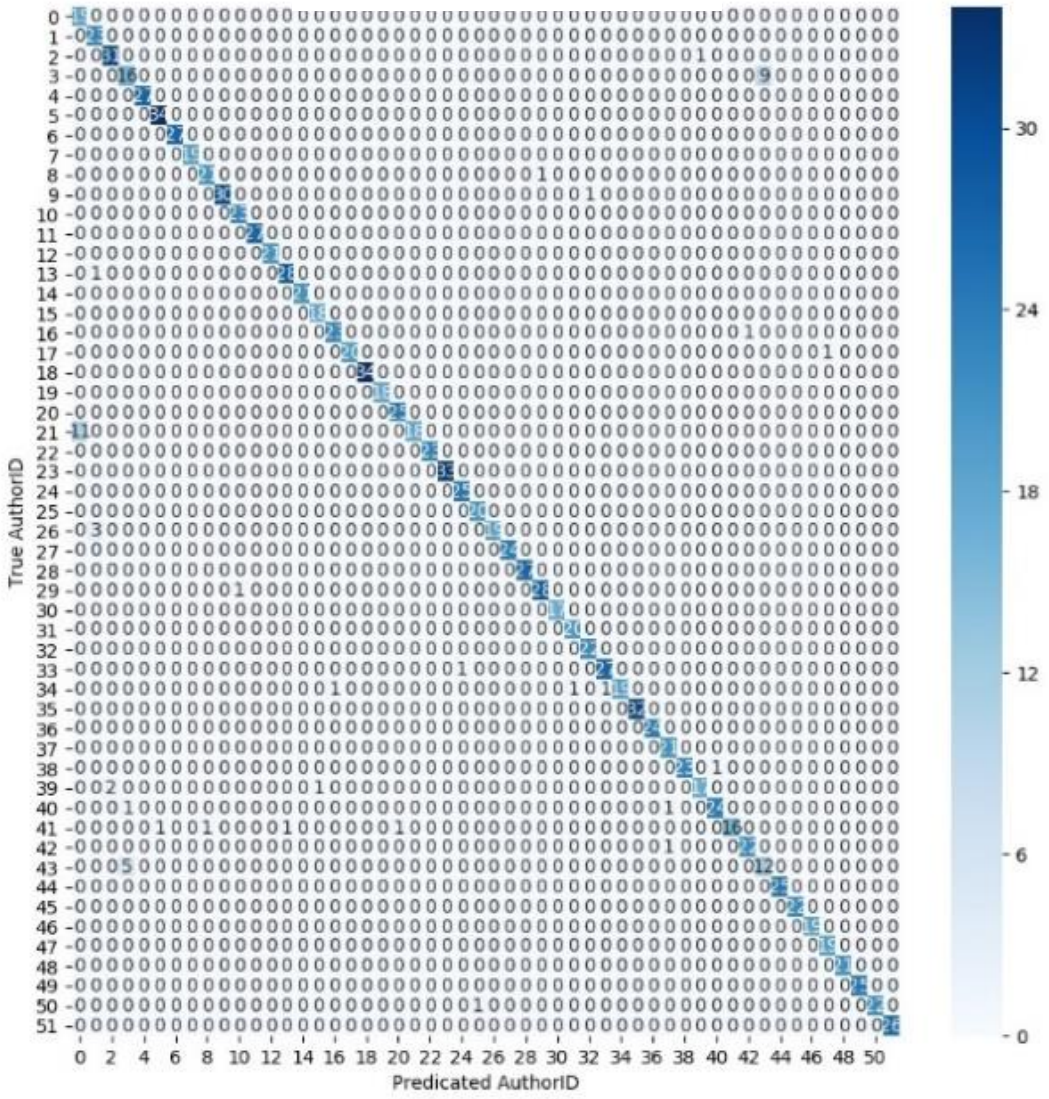
Table 5.7: Comparative analysis of the four CNNs for author classification

Author ID	MobileNet_100			ResNet_50			DenseNet_201			VGG_19		
	Precision	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
1	0.6333	1.0000	0.7755	0.6207	0.9000	0.7346	0.5714	1.0000	0.7273	0.7308	1.0000	0.8444
2	0.8519	1.0000	0.9200	0.8800	1.0000	0.9362	0.8889	1.0000	0.9412	0.8519	1.0000	0.9200
3	0.9394	0.9688	0.9538	0.9375	1.0000	0.9677	0.9143	0.9412	0.9275	0.8709	0.9310	0.9000
4	0.7273	0.6400	0.6809	0.7097	1.0000	0.8302	0.7241	0.9130	0.8077	0.7353	1.0000	0.8475
5	**1.0000**	**1.0000**	**1.0000**	0.9629	1.0000	0.9811	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
6	0.9714	1.0000	0.9855	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
7	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
8	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
9	0.9545	0.9545	0.6545	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
10	1.0000	0.9677	0.9836	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9677	0.9836	0.9032	0.9655	0.9333
11	0.9583	1.0000	0.9787	**1.0000**	**1.0000**	**1.0000**	0.9583	1.0000	0.9787	0.9615	1.0000	0.9804
12	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
13	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9500	0.9744
14	0.9655	0.9655	0.9655	1.0000	0.9643	0.9818	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
15	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9524	1.0000	0.9756	0.9545	1.0000	0.9767
16	0.9474	1.0000	0.9729	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
17	0.9583	0.9583	0.9583	**1.0000**	**1.0000**	**1.0000**	0.9565	1.0000	0.9778	**1.0000**	**1.0000**	**1.0000**
18	1.0000	0.9524	0.9756	**1.0000**	**1.0000**	**1.0000**	0.9545	1.0000	0.9767	0.9545	0.9545	0.9545
19	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
20	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
21	0.9615	1.0000	0.9804	0.9600	1.0000	0.9796	0.9231	1.0000	0.9600	**1.0000**	**1.0000**	**1.0000**
22	1.0000	0.6207	0.7659	0.8636	0.6333	0.7308	1.0000	0.6000	0.7500	0.9130	0.7500	0.8236
23	**1.0000**	**1.0000**	**1.0000**	0.9130	1.0000	0.9545	0.9583	1.0000	0.9787	**1.0000**	**1.0000**	**1.0000**
24	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9688	0.9841	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9667	0.9831
25	0.9615	1.0000	0.9804	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9615	1.0000	0.9804
26	0.9524	1.0000	0.9756	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9545	0.9767

Table 5.7: Comparative analysis of the four CNNs for author classification (Continued)

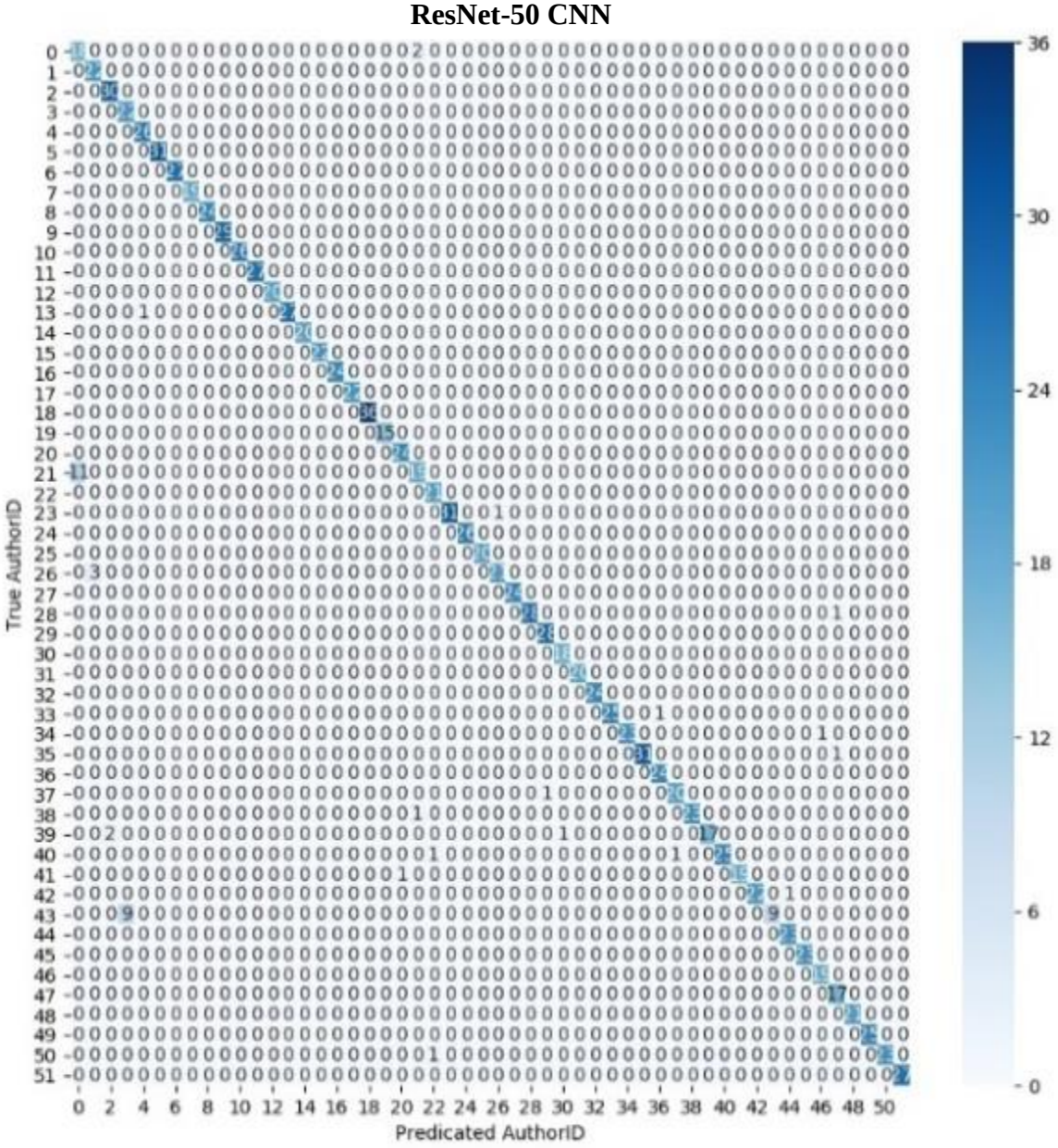
27	1.0000	0.8636	0.9268	0.9545	0.8750	0.9130	0.9524	0.8696	0.9091	0.9524	0.8696	0.9091
28	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
29	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9655	0.9825	1.0000	0.9667	0.9831	1.0000	0.9667	0.9831
30	0.9655	0.9655	0.9655	0.9655	1.0000	0.9825	1.0000	0.9667	0.9831	**1.0000**	**1.0000**	**1.0000**
31	**1.0000**	**1.0000**	**1.0000**	0.9474	1.0000	0.9729	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9444	0.9714
32	0.9524	1.0000	0.9756	**1.0000**	**1.0000**	**1.0000**	0.8696	1.0000	0.9302	**1.0000**	**1.0000**	**1.0000**
33	0.9565	1.0000	0.9778	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
34	0.9643	0.9643	0.9643	1.0000	0.9615	0.9804	0.9615	0.9615	0.9615	1.0000	0.9643	0.9818
35	1.0000	0.8636	0.9268	1.0000	0.9583	0.9787	1.0000	0.9130	0.9545	1.0000	0.9565	0.9778
36	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9687	0.9841	0.9655	0.9333	0.9492	**1.0000**	**1.0000**	**1.0000**
37	**1.0000**	**1.0000**	**1.0000**	0.9600	1.0000	0.9796	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
38	0.9130	1.0000	0.9545	0.9524	0.9524	0.9524	**1.0000**	**1.0000**	**1.0000**	0.9524	0.9524	0.9524
39	1.0000	0.9583	0.9787	1.0000	0.9583	0.9787	1.0000	0.9167	0.9565	1.0000	0.9615	0.9804
40	0.9444	0.8500	0.8947	1.0000	0.8500	0.9189	0.8889	0.8421	0.8649	0.8947	0.8500	0.8718
41	0.9600	0.9231	0.9412	1.0000	0.9259	0.9615	1.0000	0.9231	0.9600	1.0000	0.9643	0.9818
42	1.0000	0.8000	0.8889	1.0000	0.9500	0.9744	1.0000	0.8636	0.9268	1.0000	0.9524	0.9756
43	0.9565	0.9565	0.9565	1.0000	0.9565	0.9778	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
44	0.5714	0.7059	0.6316	1.0000	0.5000	0.6667	0.7857	0.5789	0.6667	1.0000	0.4706	0.6400
45	**1.0000**	**1.0000**	**1.0000**	0.9583	1.0000	0.9787	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9091	0.9524
46	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
47	**1.0000**	**1.0000**	**1.0000**	0.9500	1.0000	0.9744	**1.0000**	**1.0000**	**1.0000**	0.9091	1.0000	0.9524
48	0.9500	1.0000	0.9744	0.8947	1.0000	0.9444	0.8889	1.0000	0.9412	0.9474	1.0000	0.9729
49	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
50	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
51	1.0000	0.9565	0.9778	1.0000	0.9583	0.9787	1.0000	0.9565	0.9778	1.0000	0.9565	0.9778
52	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

MobileNet-V1 CNN



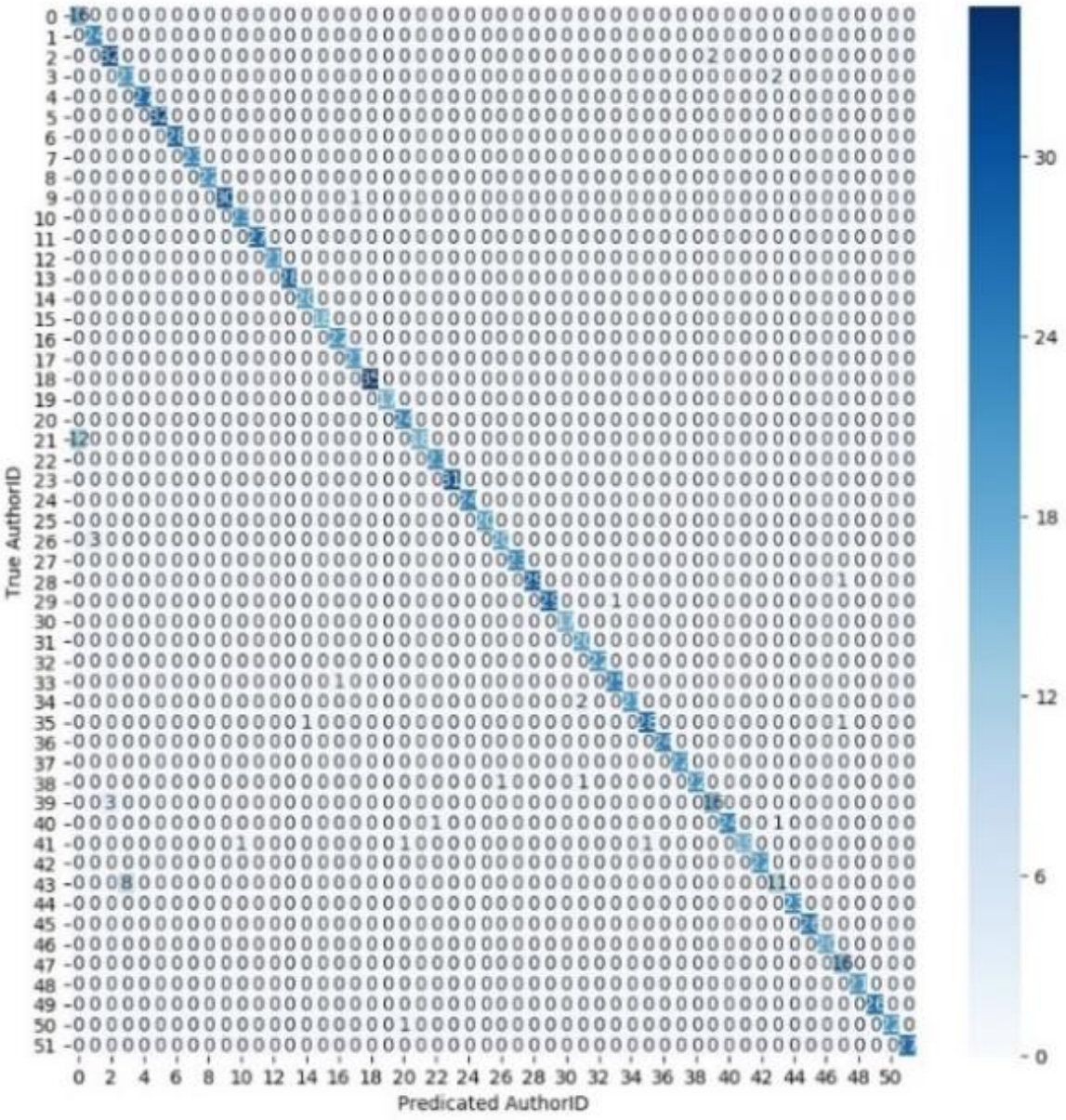
(a)

ResNet-50 CNN



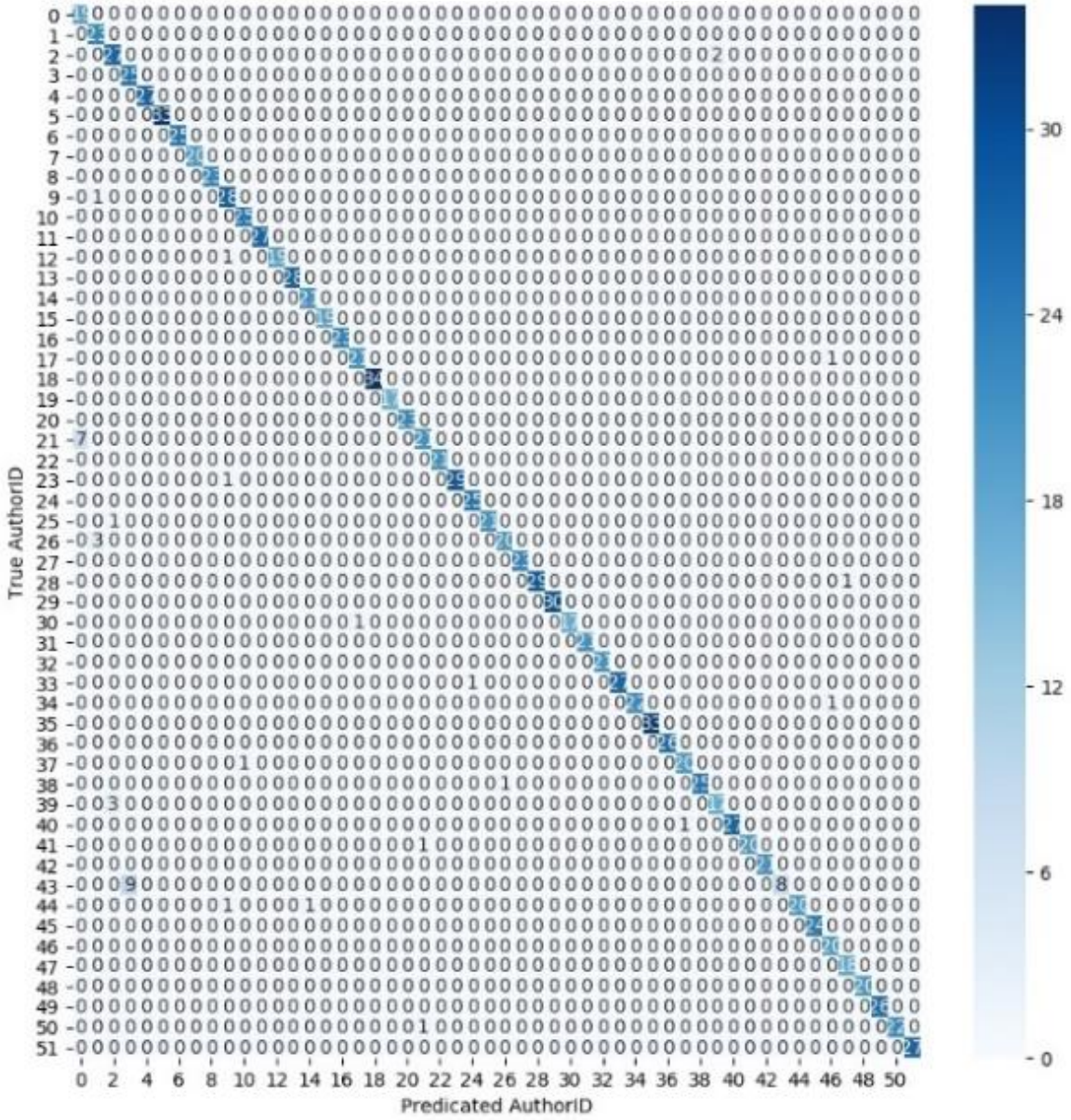
(b)

DenseNet-201 CNN



(c)

VGG-19 CNN



(d)

Figure 5.4: Authors generated confusion matrices by (a) MobileNetV1, (b) ResNet50, (c) DenseNet201, and (d) VGG19

Theses metrics assist in comparing the classified authors with the ground truth ones since the rows include the true authors, while the columns contain the classified authors. From the generated confusion matrices in figure 5.4 by the four deep CNNs,

we admit that the image classification process, according to the authors, is performing well. That is because we can notice a clear diagonal created inside the confusion matrices, including the most significant numbers, which indicates that the models were able to classify the images of our ancient Arabic manuscripts successfully. In addition, we notice that the author with (21) number in the confusion matrix and (22) id in our dataset, as well as, the author with (43) number in the confusion matrix and (44) id in our dataset were the worst classified Arabic authors. That is because they had the largest numbers of miss-classified images. But we find this result satisfactory because even though these two authors only wrote (6) pages, the models were able to classify them, and this was due to the use of the "weighted categorical cross-entropy" algorithm and the performed online data augmentation on the training subset. About the author with (22) id, 76.59%, 73.08%, 75%, and 82.36% of its images were classified successfully using MobileNet, ResNet50. DenseNet201 and VGG19 deep CNNs, respectively. Similarly, the author with (44) id; 63.16%, 66.67%, 66.67%, and 64% of its images was classified successfully using MobileNet, ResNet50. DenseNet201 and VGG19 deep CNNs, respectively.

5.3.3 Image Classification According to the Manuscript

Table 5.8 summarizes the evaluation of the four pre-trained deep CNNs for image classification according to the manuscript criteria.

Table 5.8: Image classification according to the manuscript

Evaluation	MobileNet_100	ResNet_50	DenseNet_201	VGG_19
VA	0.9744	0.9752	0.4679	**0.9768**
AP	0.9782	0.9733	0.4954	**0.9836**
AR	0.9756	0.9724	0.4802	**0.9811**
AF	0.9762	0.9719	0.4265	**0.9817**

From table 5.8, we notice that both the MobileNet100 and the ResNet50 deep CNNs accomplished around 97% successful classification of the images according to the manuscripts. However, the VGG19 pre-trained deep CNN outperformed the other three deep CNNs in classifying the dataset images because it recorded 98% successful classification. On the other, the DenseNet201 deep CNN was the worst model for classifying the images.

Table 5.9 illustrates the calculated precision, recall, and F-score per each classified manuscript.

From table 5.9, we notice that the MobileNet100 deep learning model classified (33) manuscripts without any images being miss-classified since they included (1.0000) under all the evaluation parameters. While the ResNet50 deep learning model classified (29) manuscripts correctly. Regarding the DenseNet201 deep learning model, it was not performing well for image classification. That is because it only classified (2) manuscripts correctly, and it was not able to classify (11) manuscripts. In contrast, the VGG19 deep learning model was the best performing model. Since it classified (38) images 100% successfully.

Figure 5.5 illustrates the generated confusion matrices from the classification process using the four experimented deep CNNs. The sum of the total numbers inside the confusion matrices equals the test subset of the entire dataset. Hence, the total number of images inside each confusion matrix equals (1248) classified images, which represent the 15% test subset from the dataset size.

From figure 5.5, we notice that the MobileNet deep CNN was able to classify (1217) images successfully and was not able to classify only (31) images.

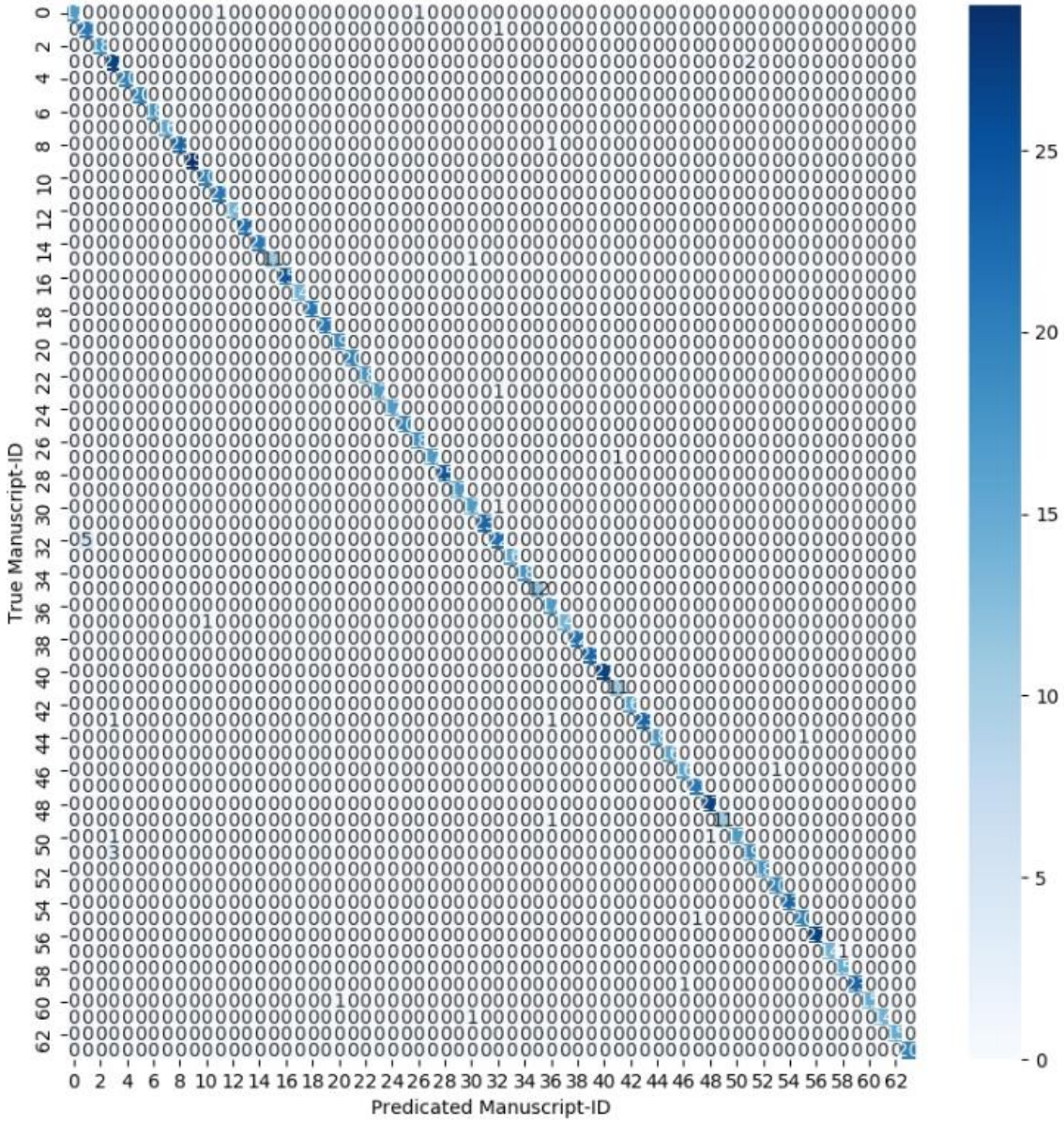
Table 5.9: Comparative analysis of the four CNNs for manuscript classification

Manuscript ID	MobileNet_100			ResNet_50			DenseNet_201			VGG_19		
	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
1	1.0000	0.8947	0.9444	1.0000	0.8889	0.9412	0.6364	0.3889	0.4828	1.0000	0.8824	0.9375
2	0.8077	0.9545	0.8750	0.8214	1.0000	0.9020	0.3261	0.6522	0.4348	0.7692	0.9524	0.8511
3	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.3864	0.9444	0.5484	**1.0000**	**1.0000**	**1.0000**
4	0.8438	0.9310	0.8852	0.8846	0.8519	0.8679	0.7222	0.4815	0.5778	0.9600	0.8889	0.9231
5	**1.0000**	**1.0000**	**1.0000**	0.9500	1.0000	0.9744	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
6	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.5714	0.4211	0.4848	**1.0000**	**1.0000**	**1.0000**
7	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.7083	1.0000	0.8293	**1.0000**	**1.0000**	**1.0000**
8	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.5161	1.0000	0.6809	**1.0000**	**1.0000**	**1.0000**
9	1.0000	0.9583	0.9787	1.0000	0.9545	0.9767	0.8261	0.8261	0.8261	1.0000	0.9583	0.9787
10	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
11	0.9524	1.0000	0.9756	**1.0000**	**1.0000**	**1.0000**	1.0000	0.0556	0.1053	**1.0000**	**1.0000**	**1.0000**
12	0.9545	1.0000	0.9767	0.9000	1.0000	0.9474	0.5000	0.9524	0.6557	0.9545	1.0000	0.9767
13	**1.0000**	**1.0000**	**1.0000**	0.9375	1.0000	0.9677	0.3462	0.6923	0.4615	**1.0000**	**1.0000**	**1.0000**
14	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9615	0.9804	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
15	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9545	0.9767	0.6667	0.1739	0.2759	**1.0000**	**1.0000**	**1.0000**
16	1.0000	0.9167	0.9565	1.0000	0.9231	0.9600	0.3600	0.6000	0.4500	1.0000	0.9231	0.9600
17	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9286	1.0000	0.9630	**1.0000**	**1.0000**	**1.0000**
18	**1.0000**	**1.0000**	**1.0000**	0.8889	1.0000	0.9412	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
19	**1.0000**	**1.0000**	**1.0000**	0.9524	1.0000	0.9756	1.0000	0.3158	0.4800	**1.0000**	**1.0000**	**1.0000**
20	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9545	1.0000	0.9767	**1.0000**	**1.0000**	**1.0000**
21	0.9500	1.0000	0.9744	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
22	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.2394	0.7727	0.3656	0.9583	1.0000	0.9787
23	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.1714	1.0000	0.2927	**1.0000**	**1.0000**	**1.0000**
24	1.0000	0.9444	0.9714	0.9474	0.9474	0.9474	0.4146	0.9444	0.5763	1.0000	0.9375	0.9677
25	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.6842	0.7647	0.7222	**1.0000**	**1.0000**	**1.0000**
26	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.8125	0.8125	0.8125	**1.0000**	**1.0000**	**1.0000**
27	0.9474	1.0000	0.9730	**1.0000**	**1.0000**	**1.0000**	0.2000	0.0952	0.1290	**1.0000**	**1.0000**	**1.0000**
28	1.0000	0.9444	0.9714	**1.0000**	**1.0000**	**1.0000**	0.0667	0.2222	0.1026	**1.0000**	**1.0000**	**1.0000**
29	**1.0000**	**1.0000**	**1.0000**	0.9615	0.9615	0.9615	0.0000	0.0000	0.0000	0.9643	1.0000	0.9818
30	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.7647	0.7647	0.7647	0.9444	1.0000	0.9714
31	0.8947	0.9444	0.9189	1.0000	0.9444	0.9714	0.0000	0.0000	0.0000	1.0000	0.8947	0.9444
32	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000	0.0000	0.9565	1.0000	0.9778

Table 5.9: Comparative analysis of the four CNNs for manuscript classification (Continued)

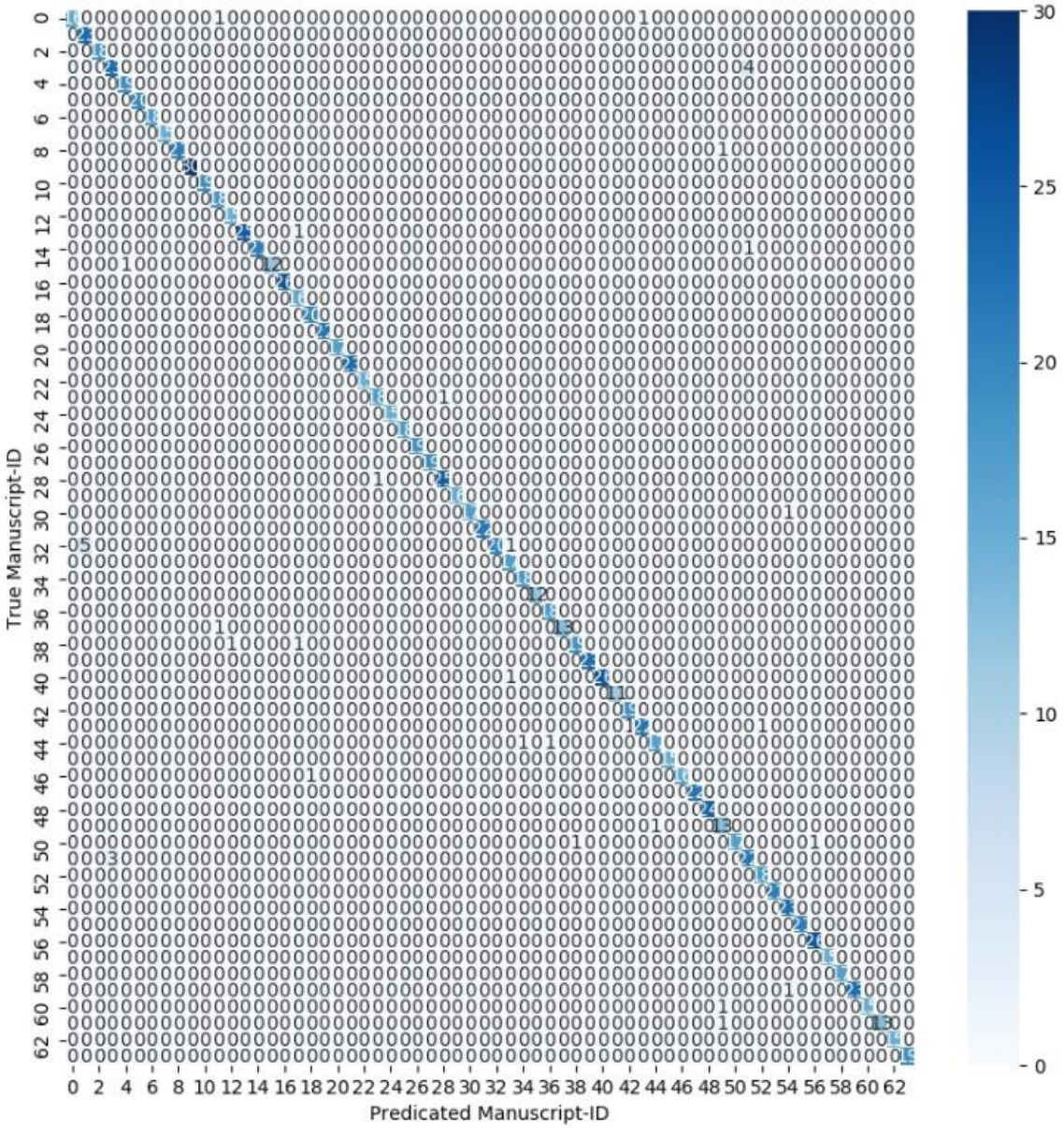
33	0.8800	0.8148	0.8462	1.0000	0.7692	0.8696	0.5000	0.0370	0.0690	0.9167	0.8148	0.8627
34	**1.0000**	**1.0000**	**1.0000**	0.8947	1.0000	0.9444	0.0901	0.7143	0.1600	**1.0000**	**1.0000**	**1.0000**
35	**1.0000**	**1.0000**	**1.0000**	0.9474	1.0000	0.9730	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
36	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.3214	0.8182	0.4615	**1.0000**	**1.0000**	**1.0000**
37	0.8500	1.0000	0.9189	0.9474	1.0000	0.9730	0.8824	0.8824	0.8824	**1.0000**	**1.0000**	**1.0000**
38	1.0000	0.9333	0.9655	1.0000	0.9286	0.9630	0.5455	0.8571	0.6667	1.0000	0.9286	0.9630
39	**1.0000**	**1.0000**	**1.0000**	0.9500	0.9048	0.9268	0.1250	0.0500	0.0714	0.9600	1.0000	0.9796
40	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0000	0.1000	0.1818	**1.0000**	**1.0000**	**1.0000**
41	**1.0000**	**1.0000**	**1.0000**	1.0000	0.9630	0.9811	0.6364	0.5385	0.5833	1.0000	0.9630	0.9811
42	0.9167	1.0000	0.9565	**1.0000**	**1.0000**	**1.0000**	1.0000	0.8000	0.8889	**1.0000**	**1.0000**	**1.0000**
43	**1.0000**	**1.0000**	**1.0000**	0.9500	1.0000	0.9744	0.8571	0.7059	0.7742	**1.0000**	**1.0000**	**1.0000**
44	1.0000	0.9200	0.9583	0.9565	0.9167	0.9362	0.0000	0.0000	0.0000	1.0000	0.9167	0.9565
45	1.0000	0.9474	0.9730	0.9444	0.8947	0.9189	0.6875	0.5789	0.6286	0.9091	1.0000	0.9524
46	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.5000	0.0667	0.1176	**1.0000**	**1.0000**	**1.0000**
47	0.9412	0.9412	0.9412	1.0000	0.9412	0.9697	1.0000	0.5263	0.6897	**1.0000**	**1.0000**	**1.0000**
48	0.9545	1.0000	0.9767	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
49	0.9643	1.0000	0.9818	**1.0000**	**1.0000**	**1.0000**	0.8462	0.8800	0.8627	0.9615	1.0000	0.9804
50	1.0000	0.9167	0.9565	0.8125	0.9286	0.8667	1.0000	0.4000	0.5714	**1.0000**	**1.0000**	**1.0000**
51	1.0000	0.8947	0.9444	1.0000	0.8947	0.9444	1.0000	0.2632	0.4167	1.0000	0.9474	0.9730
52	0.9048	0.8636	0.8837	0.8148	0.8800	0.8462	0.0000	0.0000	0.0000	0.8261	0.9500	0.8837
53	**1.0000**	**1.0000**	**1.0000**	0.9474	1.0000	0.9730	0.4750	1.0000	0.6441	**1.0000**	**1.0000**	**1.0000**
54	0.9524	1.0000	0.9756	**1.0000**	**1.0000**	**1.0000**	0.8000	0.6316	0.7059	**1.0000**	**1.0000**	**1.0000**
55	**1.0000**	**1.0000**	**1.0000**	0.9200	1.0000	0.9583	0.0000	0.0000	0.0000	0.9167	1.0000	0.9565
56	0.9524	0.9524	0.9524	**1.0000**	**1.0000**	**1.0000**	0.2778	0.2381	0.2564	**1.0000**	**1.0000**	**1.0000**
57	**1.0000**	**1.0000**	**1.0000**	0.9630	1.0000	0.9811	0.7727	0.6800	0.7234	**1.0000**	**1.0000**	**1.0000**
58	1.0000	0.9333	0.9655	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
59	0.9375	1.0000	0.9677	**1.0000**	**1.0000**	**1.0000**	0.0000	0.0000	0.0000	**1.0000**	**1.0000**	**1.0000**
60	1.0000	0.9583	0.9787	1.0000	0.9583	0.9787	0.3333	0.9600	0.4948	1.0000	0.9565	0.9778
61	1.0000	0.9375	0.9677	1.0000	0.9375	0.9677	1.0000	0.7222	0.8387	1.0000	0.9375	0.9677
62	1.0000	0.9333	0.9655	1.0000	0.9286	0.9630	0.2500	0.0667	0.1053	1.0000	0.9375	0.9677
63	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	1.0000	0.3333	0.5000	**1.0000**	**1.0000**	**1.0000**
64	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9500	1.0000	0.9744

MobileNet-V1 CNN



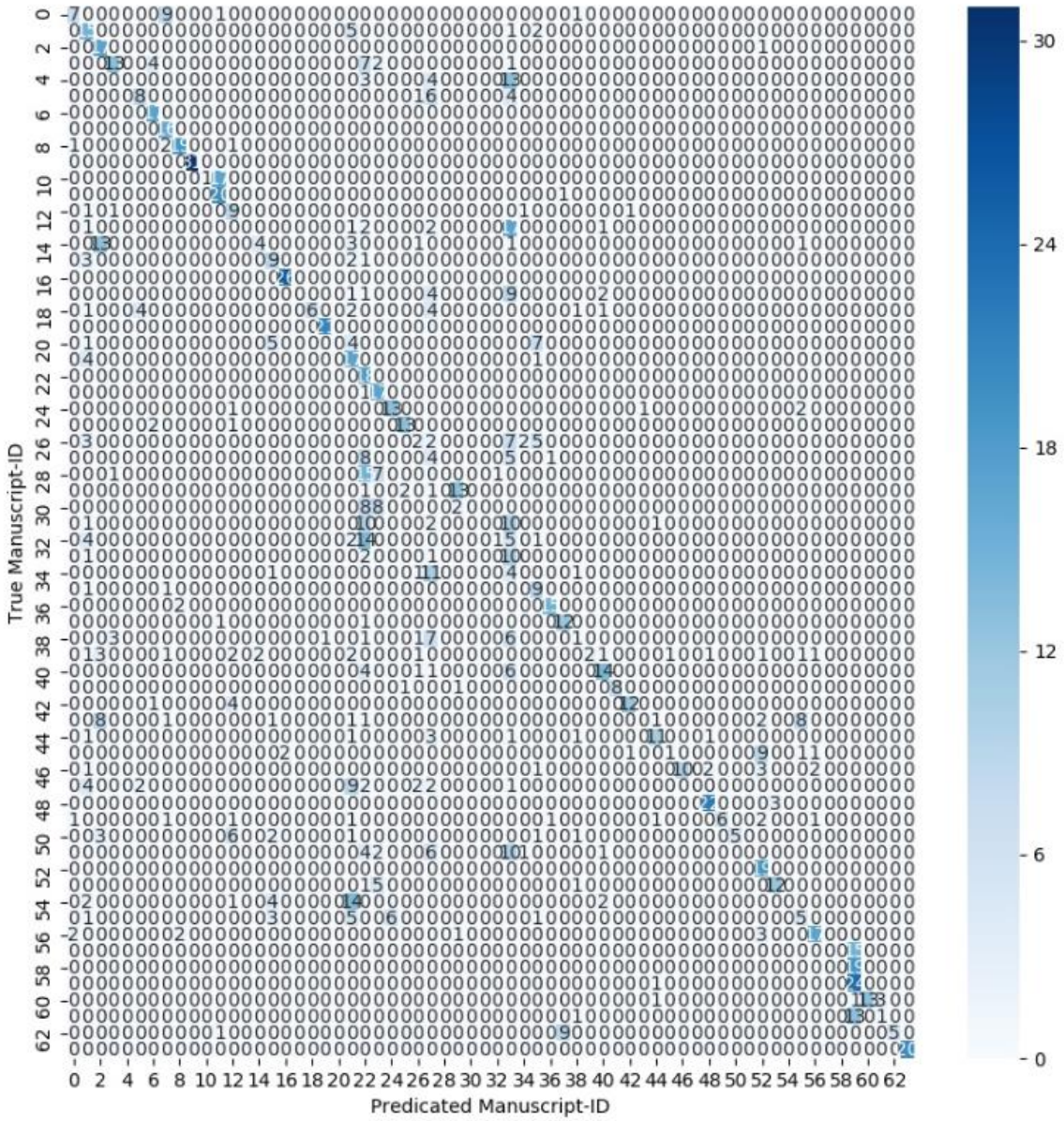
(a)

ResNet-50 CNN



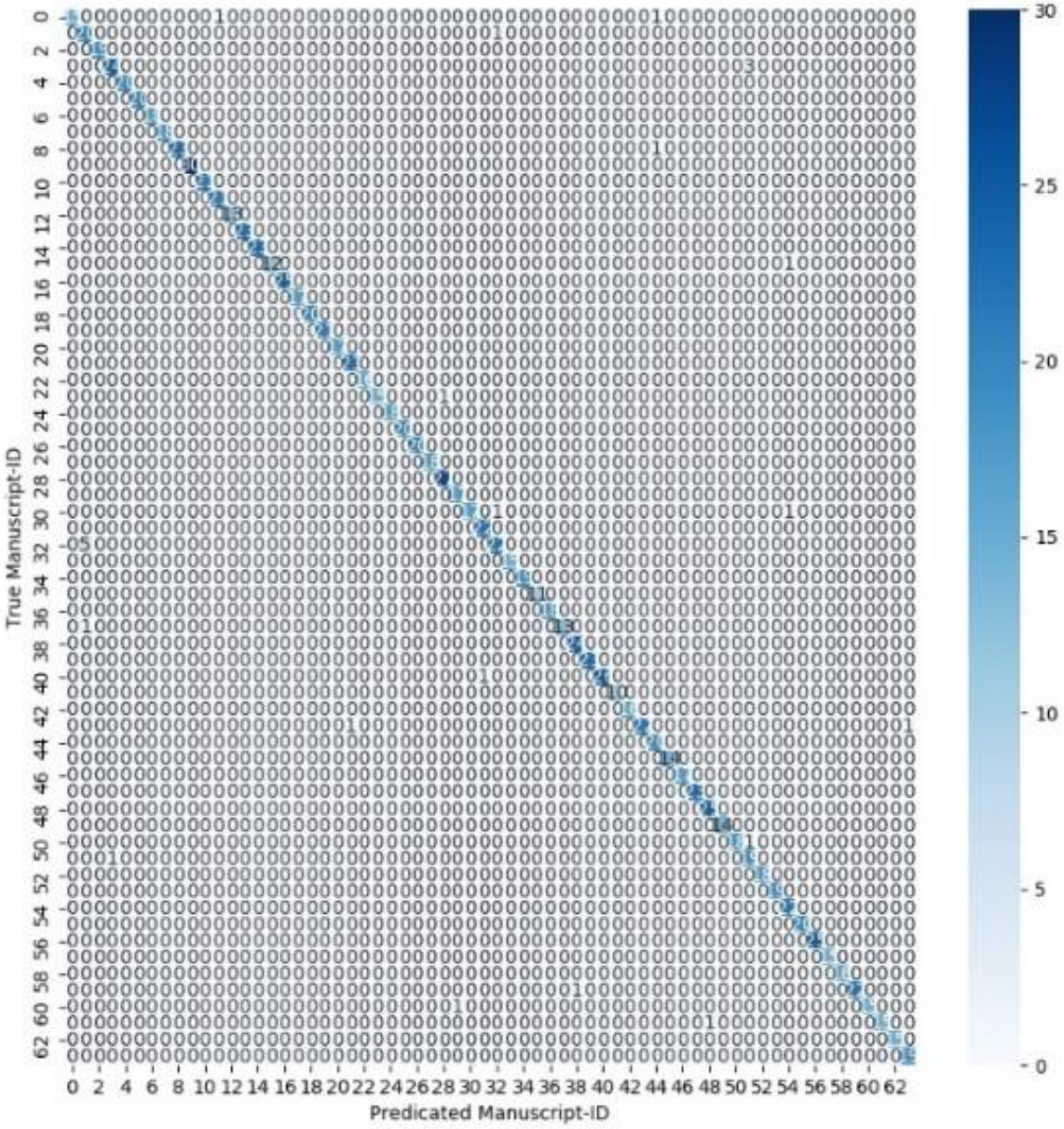
(b)

DenseNet-201 CNN



(c)

VGG-19 CNN



(d)

Figure 5.5: Manuscripts generated confusion matrices by (a) MobileNetV1, (b) ResNet50, (c) DenseNet201, and (d) VGG19

Similarly, the ResNet50 deep CNN classified a total of (1210) images successfully, and the number of miss-classified images is only (38) images. However, we conclude that the DenseNet201 deep CNN was the worst model for classifying the images according to the manuscripts among the four tested models. That is because it

classified only (656) images successfully and miss-classified (592) images. On the other hand, we conclude that most of the manuscripts' images classified correctly using the VGG19 deep learning model. That is because (1223) images were 100% successfully classified. While only (25) images were miss-classified as belonging to another different manuscript than the ground truth one.

5.3.4 Image Classification According to the Calligraphy

Table 5.10 summarizes the results of the four deep CNNs on classifying the images according to the calligraphy.

Table 5.10: Image classification according to the calligraphy

Evaluation	MobileNet_100	ResNet_50	DenseNet_201	VGG_19
VA	0.9688	0.6741	**0.9732**	0.9688
AP	0.9677	0.8089	0.9727	**0.9892**
AR	0.9674	0.6942	0.9713	**0.9879**
AF	0.9673	0.6907	0.9714	**0.9884**

From table 5.10, we notice that the model is performing the best classification of images according to the calligraphy features using the VGG19 deep CNN. That is because the model generated the highest evaluation parameters when classified using the VGG19. In contrast, the classification of images using the ResNet50 deep CNN included the lowest evaluation parameters. Table 5.11 compares between the four deep CNNs for calligraphy classification.

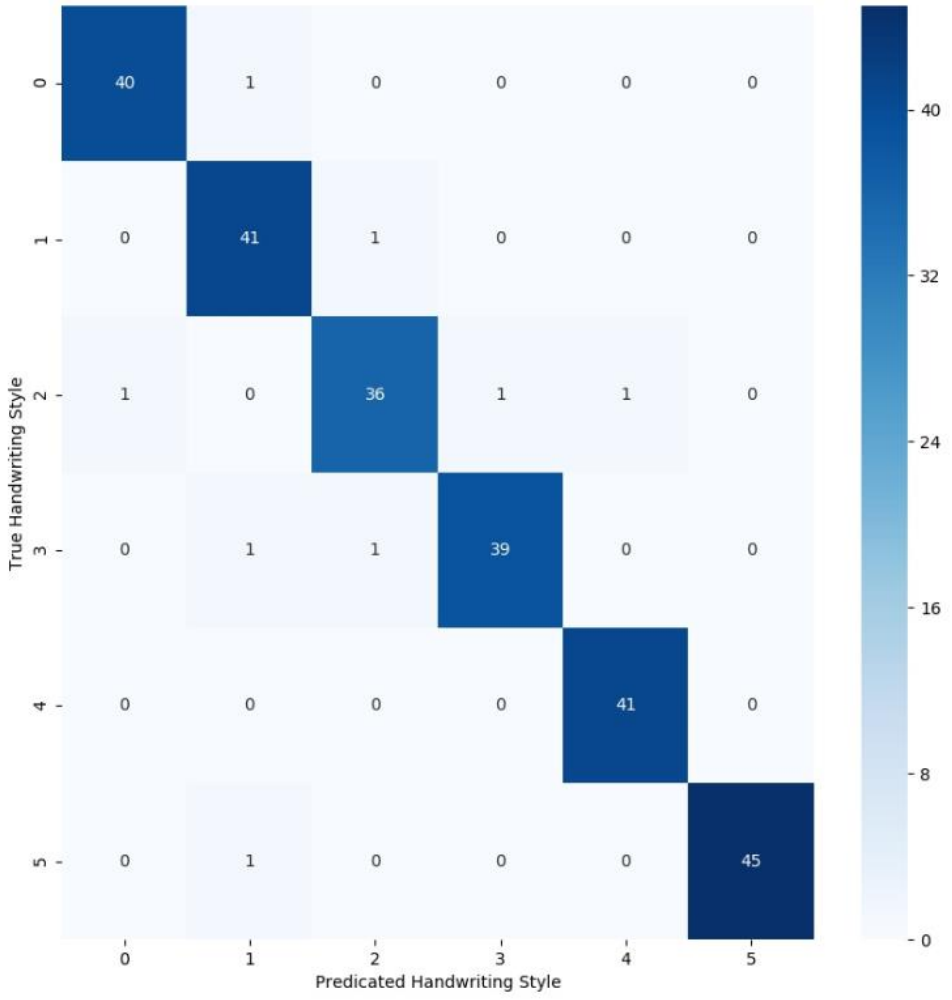
Table 5.11: Comparative analysis of the four CNNs for calligraphy classification

ID	MobileNet_100			ResNet_50			DenseNet_201			VGG_19		
	P	R	F	P	R	F	P	R	F	P	R	F
1	0.9756	0.9756	0.9756	0.8889	0.7111	0.7901	0.9778	0.9778	0.9778	**1.0000**	**1.0000**	**1.0000**
2	0.9318	0.9762	0.9535	0.6667	0.9744	0.7917	0.9762	1.0000	0.9880	1.0000	0.9762	0.9880
3	0.9474	0.9231	0.9351	0.4565	0.9767	0.6222	0.9556	1.0000	0.9773	0.9583	1.0000	0.9787
4	0.9750	0.9512	0.9630	0.9565	0.5641	0.7097	0.9268	0.9500	0.9383	1.0000	0.9744	0.9870
5	0.9762	1.0000	0.9880	1.0000	0.3636	0.5333	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
6	1.0000	0.9783	0.9890	0.8846	0.5750	0.6970	1.0000	0.9000	0.9474	0.9769	0.9767	0.9767

From table 5.11, we notice that both the DenseNet201 and the VGG19 deep CNNs were able to classify the images under the fifth calligraphy successfully. Moreover, the VGG19 deep CNN also classified all the images written using the first calligraphy successfully. Hence, we admit the effectiveness of the VGG19 deep CNN in classifying the images more accurately than the other tested deep CNNs.

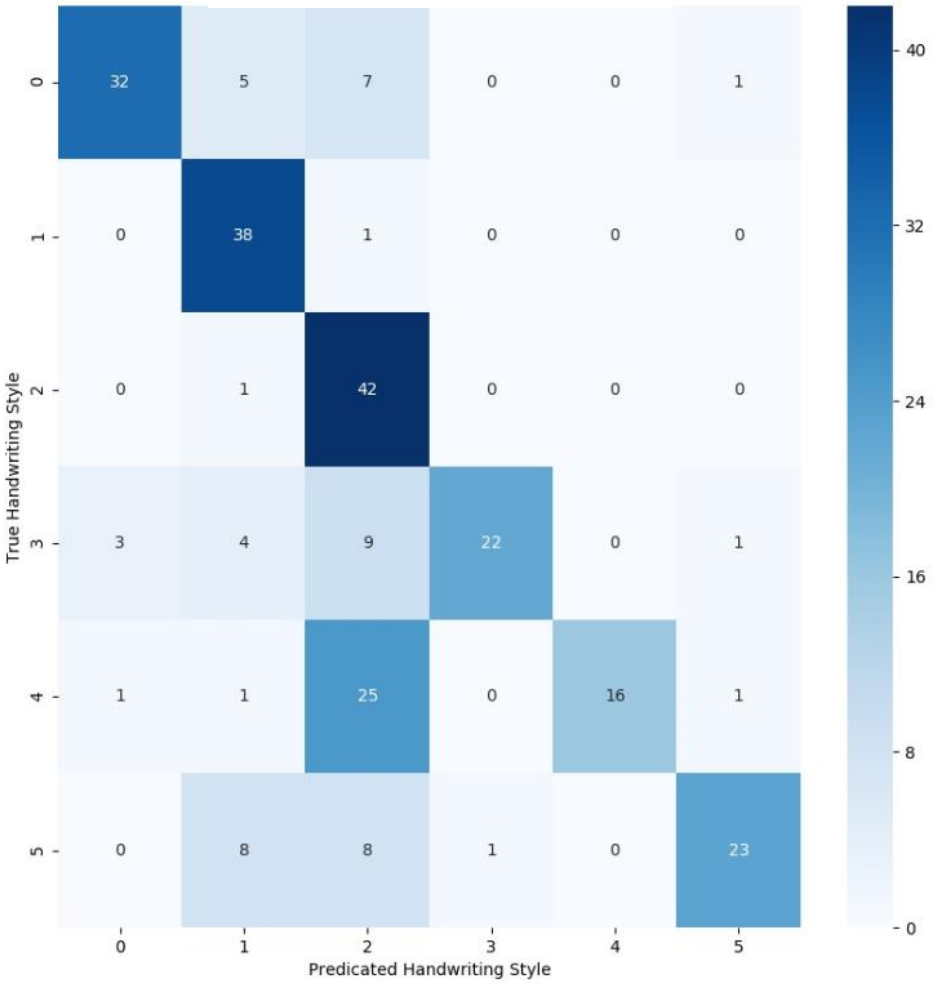
Figure 5.6 illustrates the generated confusion matrices using the four deep CNNs for classifying the images according to the calligraphy criteria.

MobileNet-V1 CNN



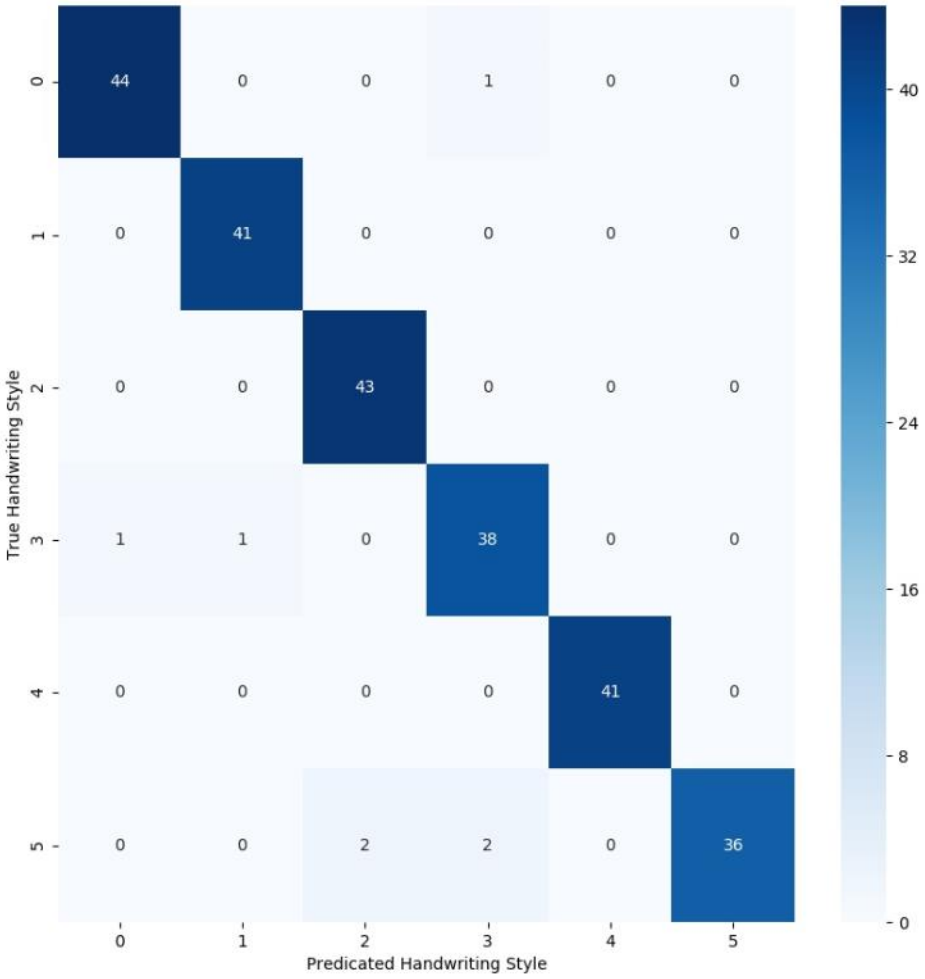
(a)

ResNet-50 CNN



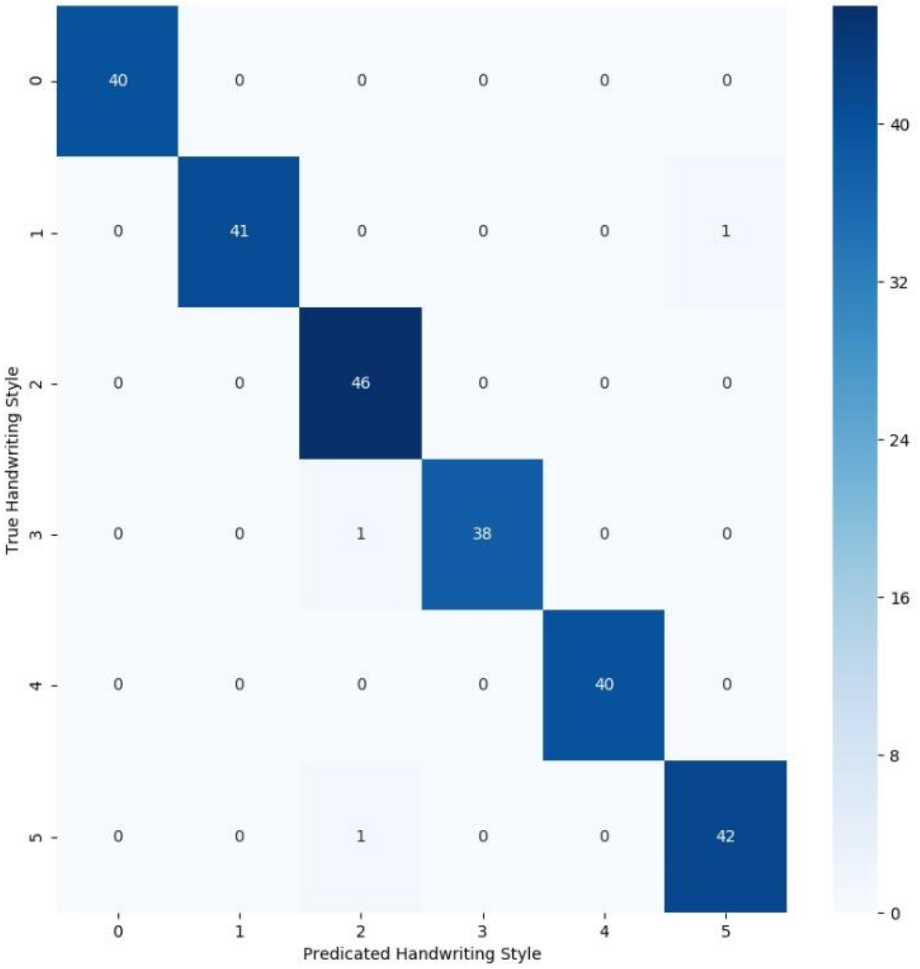
(b)

DenseNet-201 CNN



(c)

VGG-19 CNN



(d)

Figure 5.6: Calligraphies generated confusion matrices by (a) MobileNetV1, (b) ResNet50, (c) DenseNet201, and (d) VGG19

From figure 5.6, we notice that the MobileNet deep CNN was able to classify the images under the calligraphy number (4) in the confusion matrix only successfully, and all the remaining five calligraphies were including miss-classified images. Moreover, we notice that the ResNet deep CNN was not able to perform well in classifying the images according to the calligraphy. That is because the diagonal is

not showing clearly within the confusion matrix due to the huge number of miss-classified images.

Both the DenseNet and the VGG19 deep CNNs were including three calligraphies that 100% successfully classified. For example, the DenseNet deep CNN classified the calligraphies numbered (1), (2), and (4) in the confusion matrix successfully with only (7) images being miss-classified as written using different calligraphy than the true ground one. While, the VGG19 deep CNN classified all the images falling under the calligraphies numbered (0), (2), and (4) in the confusion matrix successfully with only (3) images that were miss-classified.

From all the image classification generated results using the four deep CNNs on the three experimented search criteria, we conclude that both DenseNet201 and ResNet50 deep CNNs were the weakest among the four tested models. In contrast, we conclude that the image classification process was performing the best, utilizing the VGG19 deep CNN. That is because it recorded the highest evaluation parameters, which indicates it rigidness in classifying the images successfully.

Figure 5.7 summarizes the image classification according to the three search criteria and using the four pre-trained deep CNNs. Note that the highest generated metrics highlighted in bold, while the lowest generated metrics highlighted using the red color for easier visualization of the results.

From figure 5.7, we notice that both DenseNet201 and ResNet50 deep CNNs were the weakest among the four tested models. In contrast, we conclude that the image classification process was performing the best, utilizing the VGG19 deep CNN. That is because it recorded the highest evaluation parameters.

Therefore, we decided to use the VGG19 deep CNN for the rest of the experiments.

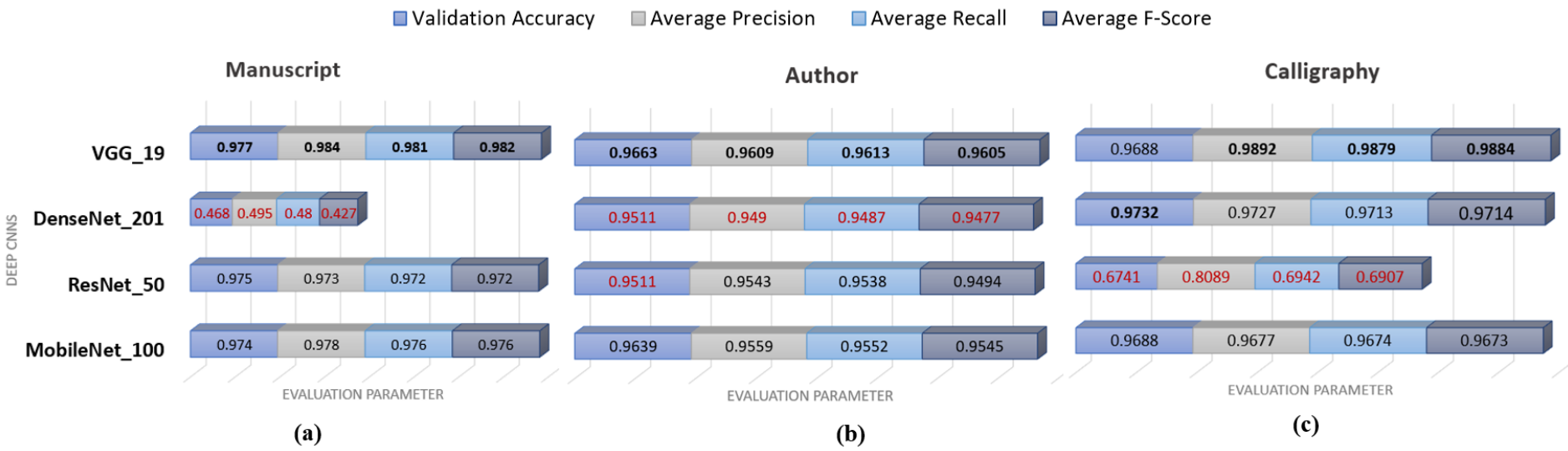


Figure 5.7: Image classification using four pre-trained CNNs and according to, (a) Manuscript, (b) Author, and (c) Calligraphy

Table 5.12 compares between the three search criteria when classified using the VGG19 deep CNN.

Table 5.12: Evaluation of the image classification using the VGG19 CNN

Classification Criteria	Validation Accuracy	Average Precision	Average Recall	Average F-Score
Manuscript	**0.9768**	0.9836	0.9811	0.9817
Author	0.9663	0.9609	0.9613	0.9605
Calligraphy	0.9688	**0.9892**	**0.9879**	**0.9884**

From table 5.12, we notice that the VGG19 deep CNN is performing the best classification of images according to the calligraphy features. That is because the model generated the highest evaluation parameters when classified according to the calligraphy. In contrast, the classification of images, according to the author included the lowest evaluation parameters. But generally, the classification model is performing very well because all the generated results were above 96% successful classification.

5.3.5 Image Matching and Retrieval

To measure the similarities among the classified images, we started by testing the image matching and retrieval using the Siamese deep learning model. We experimented two developments for the Siamese model. One is the development of the Siamese model from scratch, and the other is, its development from the VGG19 pre-trained deep CNN. Considering that both developed models were using the Euclidean (*L2*) distance metric. The evaluation parameters summarized in table 5.13 to assess the performance of the two developed models. Moreover, we visualized the same results in figure 5.8.

Table 5.13: Evaluating the developed Siamese deep learning models

Evaluation Parameter	Built from scratch	Built from VGG19
Validation Accuracy	0.4765625	0.6171875
Average Precision	0.4651163	0.6190476
Average Recall	0.3125000	0.6093750
Average F-Score	0.3738318	0.6141732

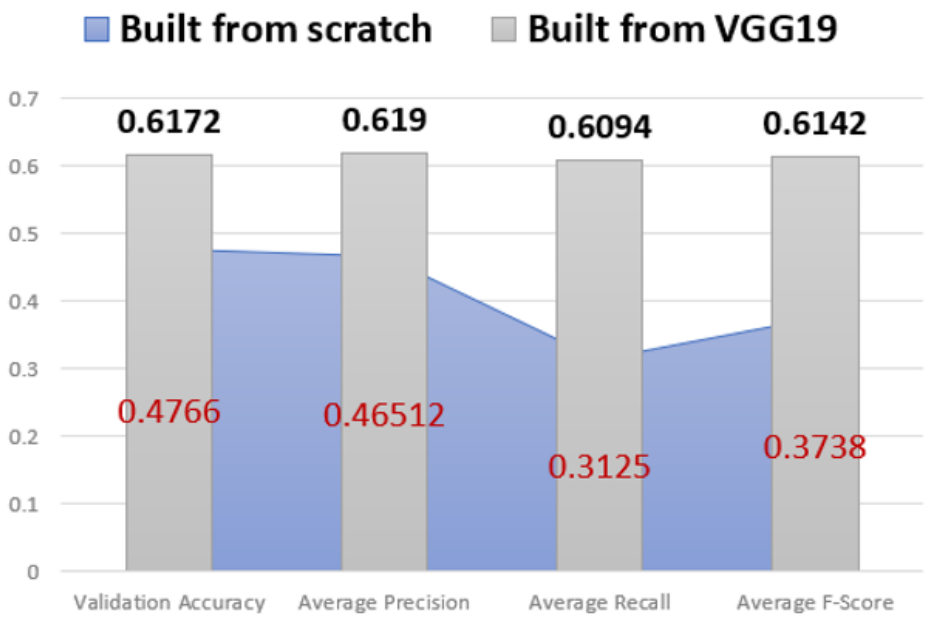


Figure 5.8: Developed Siamese deep learning models

From both table 5.12 and figure 5.8, we notice that the resulted evaluation parameters were higher when we used the structure and the weights of the VGG19 pre-trained deep CNN to build the Siamese deep neural network. However, we aim to improve its performance further. Therefore, we experimented using the Siamese model in conjunction with different distance metrics than the Euclidean metric. We tried both the Manhattan (*L1*) and the Cosine distance metrics since they are commonly used and summarized the results in table 5.14, as well as, we visualized the results in figure 5.9.

Table 5.14: Siamese deep learning model combined with three distance metrics

Evaluation Parameter	Manhattan	Euclidean	Cosine
Validation Accuracy	0.5937500	0.6171875	0.9508929
Average Precision	0.8750000	0.6190476	0.9697674
Average Recall	0.2187500	0.6093750	0.9308036
Average F-Score	0.3500000	0.6141732	0.9498861

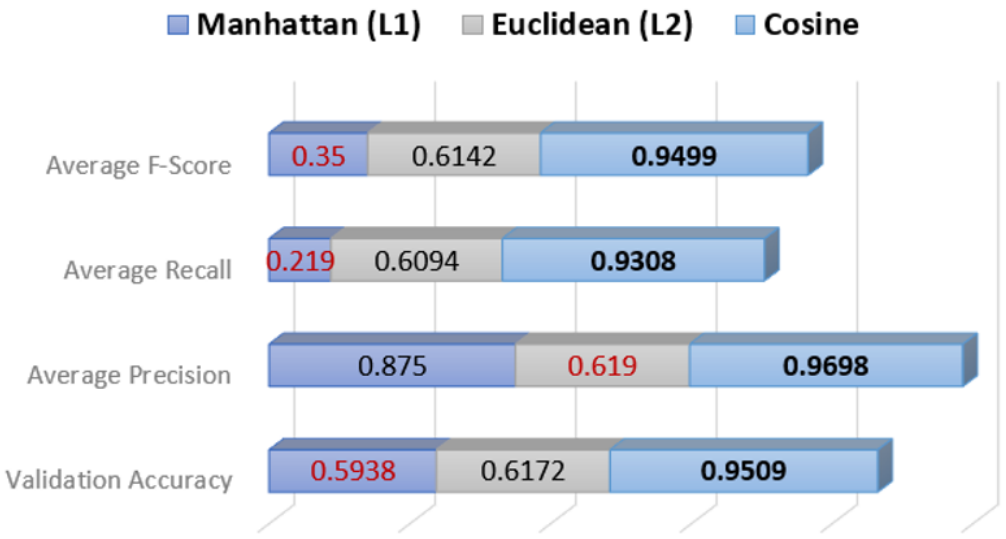


Figure 5.9: Siamese model with three distance metrics

From table 5.14 and figure 5.9, we notice that both the Manhattan and the Euclidean distance metrics generated around 60% accuracy. On the other hand, the highest evaluation parameters generated using the Cosine distance metric. Thus, we generated its Precision-Recall Curve (PRC), as illustrated in figure 5.10.

The PRC summarizes the various generated probabilities among different discrimination threshold values. From the figure, we notice a steady improvement in the precision values as much as the recall values increase, which reflects the good learning ability of the model using the Cosine distance metric in conjunction with the Siamese deep learning model reaching both precision and recall values that are above 93% successful classification of images.

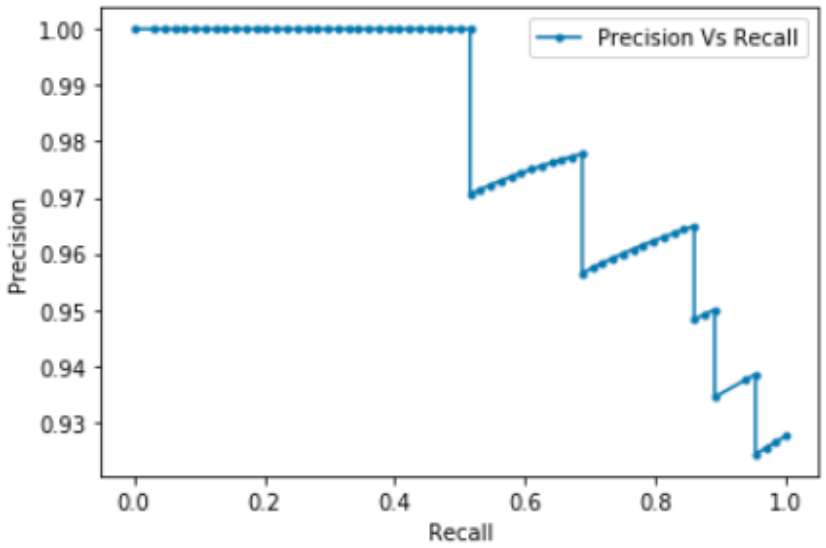


Figure 5.10: Precision-Recall Curve of the Cosine distance metric

Considering that the PRC doesn't reflect the true negatives existed in our data, then it didn't show the non-similar images that were classified correctly. Thereby, we also generated the Receiver Operating Characteristic (ROC) curve of the Cosine distance metric, as illustrated in figure 5.11.

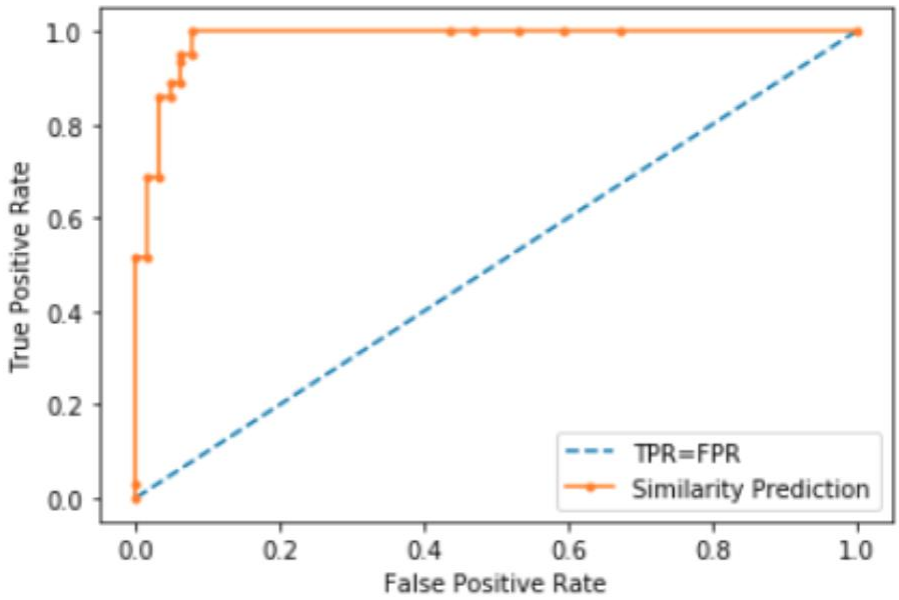


Figure 5.11: ROC Curve of the Cosine distance metric

The orange dots within the ROC curve are representing the generated probability values by the Siamese deep learning model using the Cosine distance metric. While the blue dashed line appears in the middle of the curve is to easily visualize the values when the true positive rate equals the false positive rate. From the figure, we notice that the true positive rates are increasing, reaching a high rate that is close to (1). This result reflects the 93% of the predictions classified as true-positive rates. In other words, most of the dataset images correctly classified as either similar or not similar. Whereas, there are few images classified as false-positive rates, which is good because there are only a few false alarms generated by the dataset.

However, the model spent many hours training the Siamese model. And even after the training, it consumed a long time to generate the output results, and that is due to the Siamese nature in taking two input images, which increased the dataset size exponentially.

Therefore, we installed and activated both the CUDA toolkit and the cuDNN deep neural network library on our machine to run the code on the GPU instead of running it on the CPU. Considering that the installed NVIDIA GPU on our machine is "GeForce RTX 2080". Thus, its compatible with (10.0.130) CUDA driver and cuDNN version (7). After compiling and executing the same code using the GPU, we reached good enhancements on the results, as illustrated in Table 5.15.

Table 5.15: Comparison between the Siamese execution on the CPU and GPU

Comparison Criteria	CPU	GPU
Total Training Time	11:53 hours	6 hours
Image Retrieval Time	57 seconds	21.406 seconds

We notice from table 5.15 that the overall training time for the ten learning cycles was around 12 hours using the CPU. While it took half of the time using the GPU.

Thus, we can save lots of time utilizing a rigid GPU. Regarding the retrieval time of the top-10 similar images to one query input image, there is a dramatic decrease in computing it using the GPU. However, it still taking a long time that is not acceptable for real-time interactive applications.

Since our goal is to reach an accurate and instantaneous image retrieval system, we decided to measure the similarities among images utilizing the static distance metrics and employing the GPU. Table 5.16 illustrates the mAP for each retrieved top-*k* similar images calculated from the entire dataset and utilizing the VGG19 deep CNN. The same results are visualized in figure 5.12.

Table 5.16: Retrieval mean accuracy per *k* similar images

Distance Metric	Top-10	Top-25	Top-50	Top-100
	Manuscript			
Manhattan	97.2707%	96.4376%	95.9357%	95.5366%
Euclidean	97.2699%	96.4504%	**95.9522%**	95.5637%
Cosine	**97.2751%**	**96.4512%**	95.9481%	**95.5640%**
	Author			
Manhattan	95.3311%	94.4575%	93.7768%	93.0069%
Euclidean	95.3716%	94.4895%	93.8312%	93.0612%
Cosine	**95.4214%**	**94.5024%**	**93.8541%**	**93.0927%**
	Calligraphy			
Manhattan	98.7439%	98.2294%	97.8423%	97.5650%
Euclidean	98.7567%	**98.2498%**	97.8520%	97.5855%
Cosine	**98.7614%**	98.2420%	**97.8593%**	**97.5921%**

From table 5.16 and figure 5.12, we conclude the following:

- As much as the number of *k* is smaller, as much as the model records higher retrieval mean accuracy.
- The highest mean accuracies are generated by the image retrieval according to the visual calligraphy features.

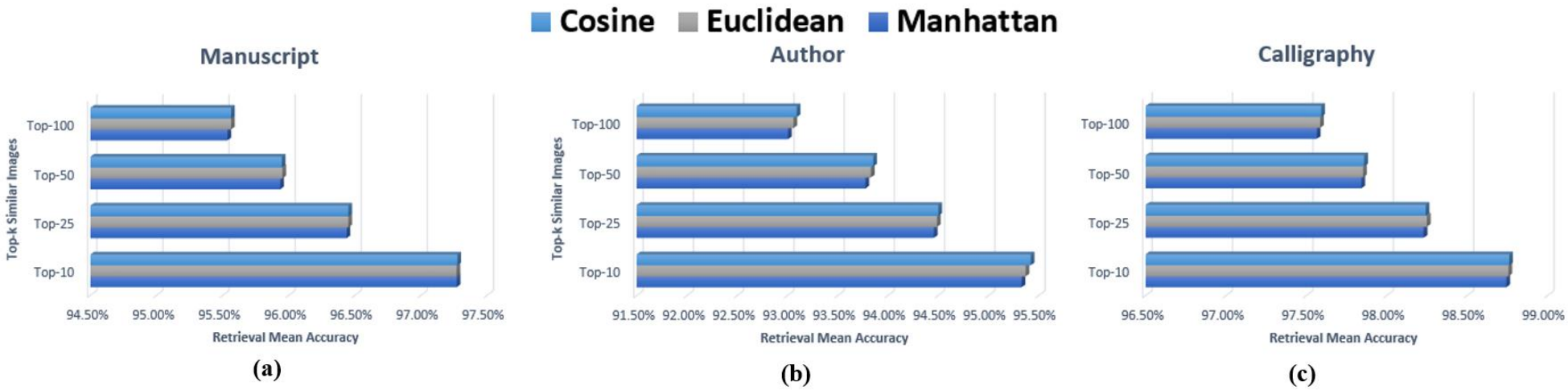


Figure 5.12: Retrieval mean accuracy per *k* similar images and according to (a) Manuscript, (b) Author, and (c) Calligraphy

- The lowest mean accuracies are computed by retrieving images according to the author's criteria.
- The used handwriting style in writing the images is the best visual feature for distinguishing the images and retrieve them when compared with both the manuscript and the authors' visual features.
- The three tested distance metrics are all performing well in measuring the similarities, and their generated values are close to each other.
- The generated mean accuracies by the Manhattan distance metric are the lowest compared with the other two distance metrics.
- The mean accuracies generated using the Cosine distance metric are the highest.

According to the findings, we decided to use the VGG19 classification model combined with the Cosine distance metric in the image retrieval system according to the visual features.

5.4 Image Classification and Retrieval According to the Textual Features

The dataset categorized into 70% training, and the remaining 30% of the total dataset size is divided equally between the validation and the testing subsets. Regarding the learning parameters, the dropout rate is set to (0.5) and the learning rate is set to (1e-6). The model compiled using "Adam" optimizer with the "weighted categorical cross-entropy" loss function. The batch size for training the model set to (128). Then, the model trained using (10) learning cycles on both non-cleaned and cleaned textual data and the generated evaluation parameters using the initial LSTM deep learning model summarized in table 5.17.

Table 5.17: Classical LSTM classification according to the manuscript

Evaluation Parameter	Non-Cleaned Text	Cleaned Text
Validation Accuracy	0.3573	0.5720
Average Precision	0.2662	0.4815
Average Recall	0.2662	0.4815
Average F-Score	0.2662	0.4815

From table 5.17, we notice that the generated evaluation parameters by the initial LSTM deep learning model on both non-cleaned and cleaned text were not satisfying. Thus, we optimized the classical LSTM deep learning model through making the LSTM layer bidirectional and through adding both the attention and batch normalization layers.

Table 5.18 summarizes the evaluation of the optimized BiLSTM deep learning model on both non-cleaned and cleaned textual data for classifying them according to the manuscript, after adding the attention layer to the Bidirectional LSTM (BiLSTM), and after adding both the attention and the Batch Normalization (BN) layers.

Table 5.18: BiLSTM classification according to the manuscript

Evaluation	BiLSTM + Attention	BiLSTM + Attention + BN	BiLSTM + Attention	BiLSTM + Attention + BN
	Non-Cleaned Text		Cleaned Text	
VA	0.5107	0.6333	0.7249	0.8381
AP	0.5304	0.6452	0.7454	0.8546
AR	0.5304	0.6498	0.7454	0.8484
AF	0.5304	0.6475	0.7454	0.8515

Figure 5.13 summarizes the performance of both the classical and optimized LSTM deep learning models for text classification according to the manuscripts and using both the cleaned and the non-cleaned textual contents.

We notice that the classification accuracy of non-cleaned text improved from 36% (as illustrated in figure 5.13(a)) to 63% (as illustrated in figure 5.13(b)) after optimizing the LSTM deep learning model.

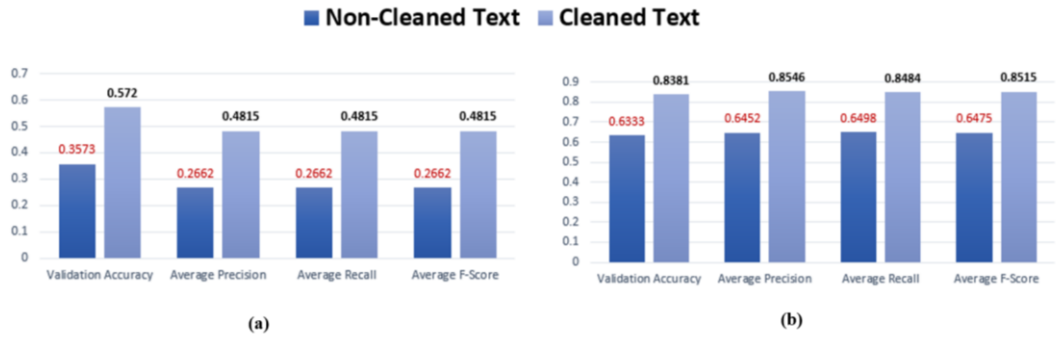


Figure 5.13: Evaluation of text classification according to the manuscript using (a) Classical LSTM model, and (b) Optimized BiLSTM model

Similarly, the validation accuracy of the cleaned text improved from 57% to become 84%, which is a successful and satisfying result. Moreover, we notice that the cleaned textual data are generating higher evaluation parameters than the non-cleaned textual data. Therefore, we will use the optimized BiLSTM architecture, including both the attention and the batch normalization layers, to classify the cleaned textual data for the rest of the experiments.

5.4.1 Text Classification According to the Manuscript

After deciding to classify the manuscripts' text using the optimized attentional BiLSTM deep learning model, we list in detail the computed precision, recall, and F-Score for each manuscript in table 5.19.

Table 5.19: Text classification evaluation parameters per manuscript

Manuscript ID	Precision	Recall	F-Score	Manuscript ID	Precision	Recall	F-Score
1	0.93	0.93	0.93	**33**	0.38	1.00	0.55
2	0.90	1.00	0.95	**34**	0.75	1.00	0.86
3	1.00	0.67	0.80	**35**	0.88	1.00	0.93
4	0.68	1.00	0.81	**36**	0.86	0.86	0.86
5	0.89	0.80	0.84	**37**	**1.00**	**1.00**	**1.00**
6	**1.00**	**1.00**	**1.00**	**38**	0.94	0.76	0.84
7	0.93	1.00	0.97	**39**	0.71	1.00	0.83
8	0.80	0.73	0.76	**40**	1.00	0.88	0.93

Table 5.19: Text classification evaluation parameters per manuscript (Continued)

9	0.79	0.90	0.84	**41**	**1.00**	**1.00**	**1.00**
10	0.00	0.00	0.00	**42**	1.00	0.62	0.77
11	0.67	1.00	0.80	**43**	0.00	0.00	0.00
12	**1.00**	**1.00**	**1.00**	**44**	0.86	0.80	0.84
13	0.77	0.98	0.92	**45**	0.84	0.84	0.84
14	0.73	0.92	0.81	**46**	0.92	0.86	0.89
15	1.00	0.96	0.98	**47**	0.94	0.91	0.92
16	1.00	0.50	0.67	**48**	0.00	0.00	0.00
17	1.00	0.14	0.25	**49**	0.81	0.92	0.86
18	0.89	0.77	0.83	**50**	0.57	0.50	0.53
19	0.71	0.95	0.82	**51**	0.69	0.82	0.75
20	0.00	0.00	0.00	**52**	1.00	0.60	0.75
21	0.00	0.00	0.00	**53**	0.86	0.82	0.84
22	0.50	0.33	0.40	**54**	0.83	0.94	0.88
23	0.96	0.89	0.92	**55**	0.93	0.85	0.89
24	0.71	0.67	0.69	**56**	0.00	0.00	0.00
25	0.50	1.00	0.67	**57**	1.00	0.12	0.22
26	0.00	0.00	0.00	**58**	0.67	0.71	0.69
27	0.00	0.00	0.00	**59**	0.73	0.86	0.79
28	0.00	0.00	0.00	**60**	0.89	1.00	0.94
29	0.00	0.00	0.00	**61**	1.00	0.86	0.92
30	0.67	1.00	0.80	**62**	**1.00**	**1.00**	**1.00**
31	0.84	0.93	0.89	**63**	**1.00**	**1.00**	**1.00**
32	1.00	0.89	0.94	**64**	0.97	1.00	0.98

From table 5.19, we notice that a total of (6) manuscripts were 100% correctly classified, while a total of (10) manuscripts while completely miss-classified.

5.4.2 Text Classification According to the Author

The evaluation of the optimized BiLSTM for text classification according to the author is summarized in table 5.20.

Table 5.20: BiLSTM classification according to the author

Evaluation Parameter	BiLSTM
Validation Accuracy	0.9030
Average Precision	0.9061
Average Recall	0.9097
Average F-Score	0.9079

From table 5.20, we notice that all the evaluation parameters of the optimized BiLSTM text classification according to the author are above 90%, which confirms the success of the model. Table 5.21 lists the recorded parameters per author.

Table 5.21: Text classification evaluation parameters per author

Author ID	Precision	Recall	F-Score	Author ID	Precision	Recall	F-Score
1	1.00	0.94	0.97	**27**	**1.00**	**1.00**	**1.00**
2	1.00	0.81	0.89	**28**	0.97	1.00	0.98
3	0.93	0.98	0.95	**29**	0.86	0.71	0.77
4	1.00	0.25	0.40	**30**	0.71	1.00	0.83
5	1.00	0.93	0.96	**31**	1.00	0.88	0.93
6	0.80	0.80	0.80	**32**	0.90	0.64	0.75
7	0.00	0.00	0.00	**33**	0.88	0.88	0.88
8	0.88	0.93	0.90	**34**	1.00	0.86	0.92
9	0.89	0.64	0.74	**35**	0.00	0.00	0.00
10	1.00	0.50	0.67	**36**	1.00	0.93	0.97
11	0.00	0.00	0.00	**37**	0.81	0.93	0.87
12	0.67	1.00	0.80	**38**	0.80	0.91	0.85
13	**1.00**	**1.00**	**1.00**	**39**	0.00	0.00	0.00
14	0.79	0.79	0.79	**40**	0.94	1.00	0.97
15	0.00	0.00	0.00	**41**	0.91	1.00	0.95
16	0.67	0.67	0.67	**42**	0.00	0.00	0.00
17	0.29	0.57	0.38	**43**	0.80	0.89	0.84
18	0.50	0.50	0.50	**44**	0.90	0.97	0.93
19	**1.00**	**1.00**	**1.00**	**45**	0.95	0.92	0.94
20	0.00	0.00	0.00	**46**	0.97	0.99	0.98
21	**1.00**	**1.00**	**1.00**	**47**	0.79	1.00	0.88
22	**1.00**	**1.00**	**1.00**	**48**	0.93	1.00	0.96
23	0.98	1.00	0.99	**49**	0.00	0.00	0.00
24	0.91	0.94	0.93	**50**	0.97	0.97	0.97
25	0.60	0.75	0.67	**51**	0.96	1.00	0.98
26	**1.00**	**1.00**	**1.00**	**52**	**1.00**	**1.00**	**1.00**

From table 5.21, we notice that a total of (7) authors were 100% successfully classified, and (8) authors miss-classified.

5.4.3 Text Classification According to the Calligraphy

The evaluation of the optimized BiLSTM for text classification according to the calligraphy is summarized in table 5.22.

From table 5.22, we notice that all the evaluation parameters of the optimized BiLSTM text classification according to the calligraphy are above 84%. However, we also computed the evaluation metrics per calligraphy in table 5.23.

Table 5.22: BiLSTM classification according to the calligraphy

Evaluation Parameter	BiLSTM
Validation Accuracy	0.8360
Average Precision	0.8402
Average Recall	0.8380
Average F-Score	0.8391

Table 5.23: Text classification evaluation parameters per calligraphy

Calligraphy ID	Precision	Recall	F-Score
1	0.85	0.80	0.83
2	0.72	0.62	0.67
3	0.82	0.94	0.88
4	0.92	0.81	0.86
5	0.86	0.88	0.87
6	0.77	0.86	0.81

From table 5.23, we notice that none of the calligraphies were 100% successfully classified neither none of them were completely miss-classified. The lowest classification F-Score was recorded by the second calligraphy as (0.67). While the best-classified calligraphy was the third one, recording (0.88) F-Score.

Table 5.24 compares between the three search criteria when classified using the optimized BiLSTM deep learning model.

Table 5.24: Text classification using the optimized BiLSTM model

Classification Criteria	Validation Accuracy	Average Precision	Average Recall	Average F-Score
Manuscript	0.8381	0.8546	0.8484	0.8515
Author	**0.9030**	**0.9061**	**0.9097**	**0.9079**
Calligraphy	0.8360	0.8402	0.8380	0.8391

From table 5.24, we notice that the model is performing the best classification of text according to the authors' features. In contrast, the classification of text according to the calligraphy included the lowest evaluation parameters. But generally, the classification model is performing well because all the generated results were above

84% successful classification. In addition, we conclude that the classification according to the textual features generated the opposite results of the classification according to the visual features. Because the classification according to the visual features, was performing the best using the calligraphy and the worst using the author. That might be due to the use of the same context by the author, which distinguished the textual features but, looking at the visual features, the calligraphy, which is like a drawing, distinguished the images more.

5.4.4 Text Matching and Image Retrieval

Table 5.25 illustrates the mAP for each top-*k* calculated from the entire dataset according to the three search criteria and utilizing the optimized BiLSTM deep learning model. The same results are visualized in figure 5.14.

Table 5.25: Retrieval mean accuracy per *k* similar texts

Distance Metric	Top-10	Top-25	Top-50	Top-100
	Manuscript			
Manhattan	63.7192%	57.9827%	53.9149%	49.7222%
Euclidean	64.0321%	58.6654%	54.9090%	**51.0960%**
Cosine	**64.7836%**	**59.2452%**	**55.1781%**	51.0182%
	Author			
Manhattan	71.7534%	66.3000%	61.9309%	57.2254%
Euclidean	72.4189%	67.0870%	62.8150%	58.1295%
Cosine	**72.6399%**	**67.5653%**	**63.5470%**	**59.2347%**
	Calligraphy			
Manhattan	49.4963%	43.7296%	39.1502%	35.6149%
Euclidean	48.6023%	44.0473%	40.4890%	35.9469%
Cosine	**49.6216%**	**44.8548%**	**40.8139%**	**36.4654%**

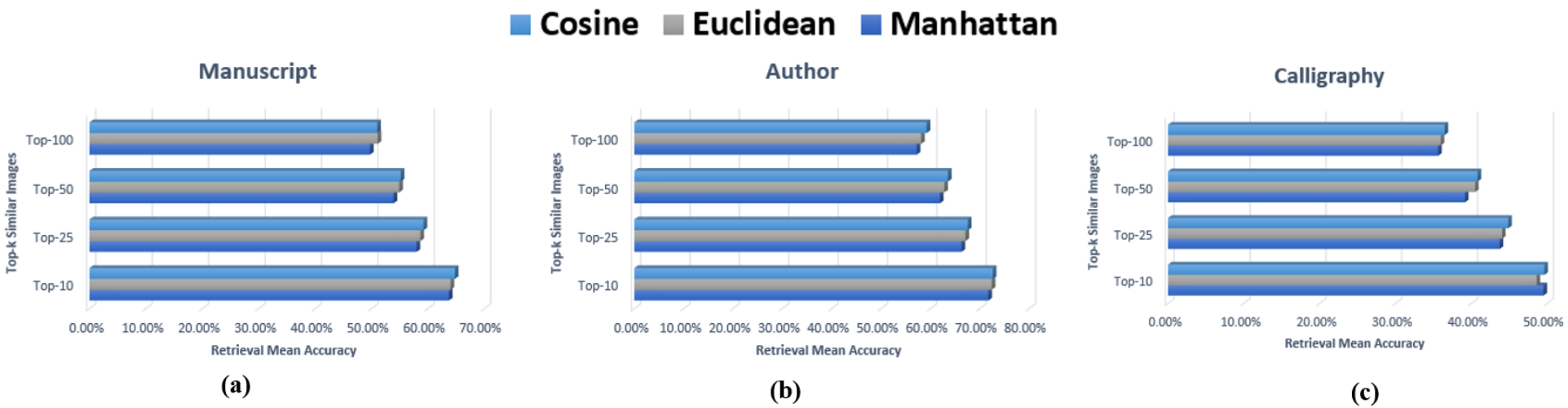


Figure 5.14: Retrieval mean accuracy per *k* similar texts and according to (a) Manuscript, (b) Author, and (c) Calligraphy

From table 5.25 and figure 5.14, we conclude the following:

- As much as the number of *k* is smaller, as much as the model records higher retrieval mean accuracy.
- The highest mean accuracies are generated by the image retrieval according to the authors' textual features.
- The lowest mean accuracies are computed by retrieving images according to the calligraphy criteria.
- The generated mean accuracies by the Manhattan distance metric are the lowest compared with the other two distance metrics.
- The mean accuracies generated using the Cosine distance metric are the highest.

According to the findings, we decided to use the optimized BiLSTM classification model combined with the Cosine distance metric in the image retrieval system according to the textual features.

5.5 Image Classification and Retrieval According to the Fusion Models

"Keras"[15] deep learning library provides several built-in merge layers that can be used with the fusion model. We tested four different merge layers with the decision-level fusion model for classifying images and texts according to the manuscripts looking for the merge layer that will increase the classification accuracy. The evaluation results of the developed decision-level fusion model are summarized in table 5.26.

[15] https://keras.io/api/layers/merging_layers/

Table 5.26: Evaluation of the decision-level fusion model

Evaluation	Concatenate	Maximum	Average	Multiply
VA	0.9470	0.9603	**0.9727**	0.9355
AP	**0.9741**	0.9468	0.9408	0.9501
AR	0.9598	**0.9680**	**0.9680**	0.9633
AF	**0.9670**	0.9573	0.9542	0.9567

From table 5.26, we notice that the decision-level fusion model generated close results using the four tested merge Keras built-in layers and that the results were all-around 96%. The reached results are higher than the manuscript classification using the textual model, which reached 83.81% validation accuracy. But they are not higher than the manuscript classification using the visual model, which reached 97.68% validation accuracy. Therefore, we experimented another fusion type called features-level fusion.

The features-level fusion model developed using the "Concatenate" merge layer from the "Keras" deep learning library only. Because all the other types of merge layers require the inputs to the merge layer to be of the same size. While, unlike the previous tested decision-level fusion model, the features vectors are of different sizes. Thus, we used the "Concatenate" merge layer in conjunction with the developed features-level fusion model for classifying both images and text according to the manuscripts. The model evaluation results are summarized in table 5.27.

Table 5.27: Evaluation of the features-level fusion model

Evaluation Parameter	Concatenate
Validation Accuracy	0.9836
Average Precision	0.9745
Average Recall	0.9836
Average F-Score	0.9790

From table 5.27, we notice that the features-level fusion model generated slightly better results than the decision-level fusion model. That is because the evaluation

parameters of the features-level fusion model were above 97%. Whereas, the evaluation parameters using the decision-level fusion model were around 96%.

The score-level fusion method with three different rules named: Min rule, Max rule, and Sum rule also tested, and its generated results summarized in table 5.28.

Table 5.28: Evaluation of the score-level fusion model

Evaluation Parameter	Min Rule	Max Rule	Sum Rule
Validation Accuracy	0.9837	0.9804	**0.9896**
Average Precision	0.9885	**0.9956**	0.9906
Average Recall	0.9906	0.9904	**0.9959**
Average F-Score	0.9895	0.9930	**0.9916**

From table 5.28, we notice that the score-level fusion model using the simple-sum rule generated higher results than using the min and max rules. Furthermore, we notice that the evaluation parameters are higher than the parameters generated using the decision-level and the features-level fusion models. Because the evaluation parameters of the score-level fusion model were all above 98%, which is higher than the classification using the images visual and textual models separately.

Figure 5.15 summarizes the image and text classification using the three different fusion models.

After extensive experimentation of multiple fusion methods, we reached that the best performing fusion method is the simple sum score-level fusion. That is because it recorded the highest classification accuracy according to the manuscripts. Therefore, we decided to use it for fusing the classification of images.

Figure 5.16 highlights the evaluation parameters of the image classification using the deep VGG_19 CNN, the optimized BiLSTM deep learning model, and the score-level fusion model. The same results are summarized in table 5.29.

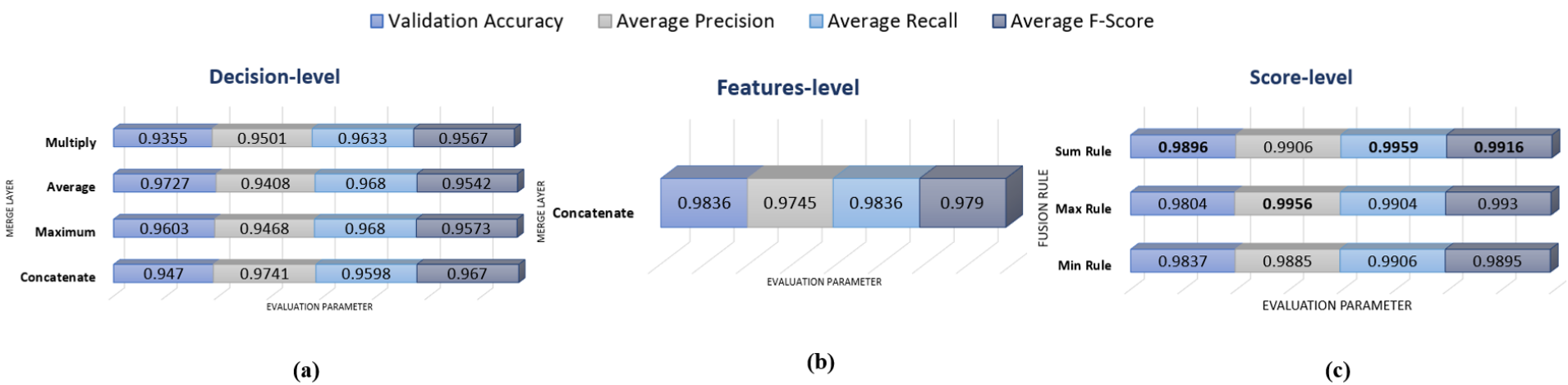


Figure 5.15: Classification using (a) Decision-level fusion model, (b) Features-level fusion model, and (c) Score-level fusion model

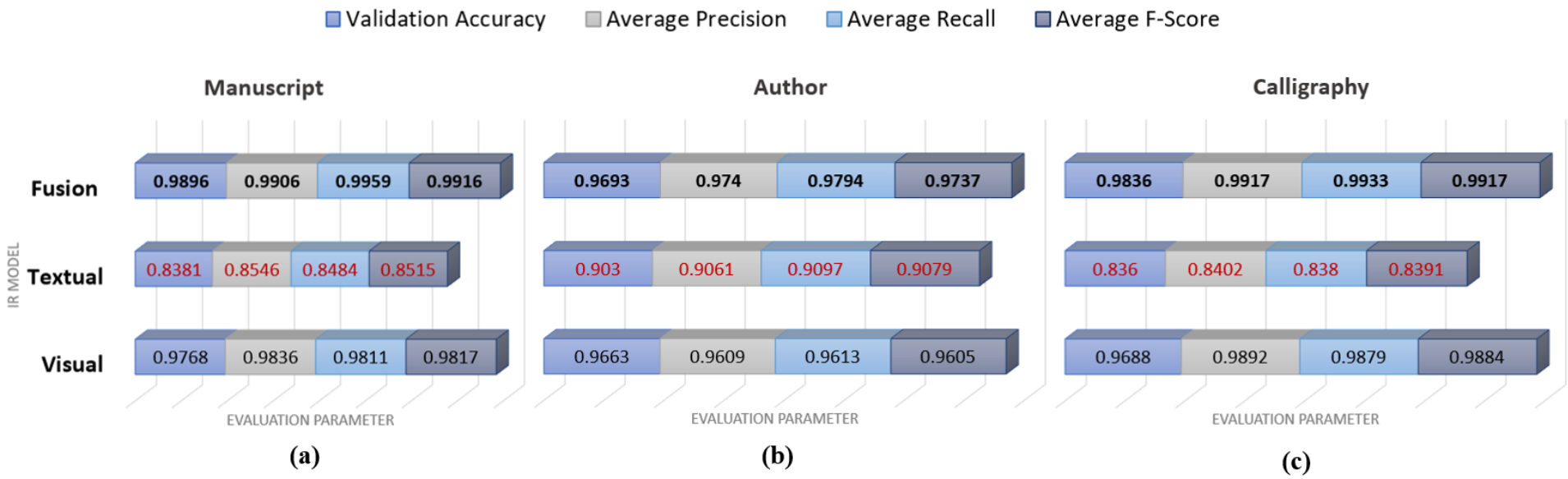


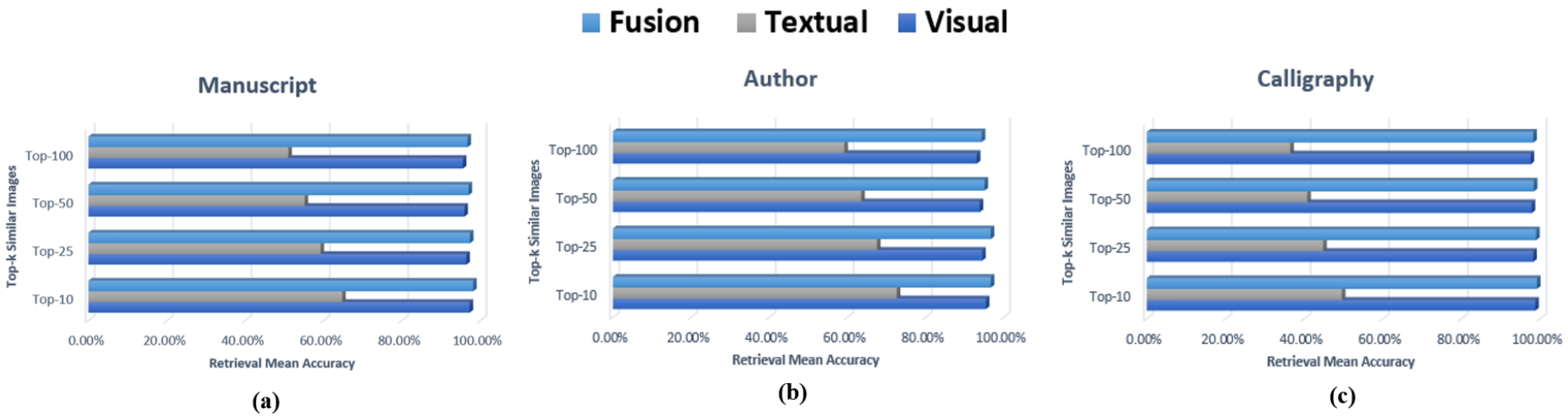
Figure 5.16: Classification using visual, textual, and fusion models, according to (a) Manuscript, (b) Author, and (c) Calligraphy

Table 5.29: Image classification using visual, textual, and fusion models

Evaluation Parameter	Visual	Textual	Fusion
	Manuscript		
Validation Accuracy	0.9768	0.8381	**0.9896**
Average Precision	0.9836	0.8546	**0.9906**
Average Recall	0.9811	0.8484	**0.9959**
Average F-Score	0.9817	0.8515	**0.9916**
	Author		
Validation Accuracy	0.9663	0.9030	**0.9693**
Average Precision	0.9609	0.9061	**0.9740**
Average Recall	0.9613	0.9097	**0.9794**
Average F-Score	0.9605	0.9079	**0.9737**
	Calligraphy		
Validation Accuracy	0.9688	0.8360	**0.9836**
Average Precision	0.9892	0.8402	**0.9917**
Average Recall	0.9879	0.8380	**0.9933**
Average F-Score	0.9884	0.8391	**0.9917**

From figure 5.16 and table 5.29, we conclude that the score-level fusion of both the visual and the textual models improved the classification accuracy more than using each model separately. That is because the fusion model has more information about the images than each single model, which increased its ability to identify the images and classifying them.

Figure 5.17 highlights the computed mAP using the Cosine distance metric per the retrieved top-*k* similar images using the VGG19 deep CNN, optimized BiLSTM deep learning model, and using the score-level fusion model. The same results are summarized in table 5.30.



(a) (b) (c)

Figure 5.17: Mean accuracy per k similar images using visual, textual, and fusion models and according to (a) Manuscript, (b) Author, and (c) Calligraphy

Table 5.30: Mean accuracy per k similar images using visual, textual, and fusion models

Top-*k*	Visual	Textual	Fusion
	Manuscript		
Top-10	97.2751%	64.7836%	**98.1819%**
Top-25	96.4512%	59.2452%	**97.3695%**
Top-50	95.9481%	55.1781%	**97.0304%**
Top-100	95.5640%	51.0182%	**96.7362%**
	Author		
Top-10	95.4214%	72.6399%	**96.7003%**
Top-25	94.5024%	67.5653%	**96.6676%**
Top-50	93.8541%	63.5470%	**95.0795%**
Top-100	93.0927%	59.2347%	**94.3602%**
	Calligraphy		
Top-10	98.7614%	49.6216%	**99.1792%**
Top-25	98.2420%	44.8548%	**99.0042%**
Top-50	97.8593%	40.8139%	**98.3961%**
Top-100	97.5921%	36.4654%	**98.2467%**

From figure 5.17 and table 5.30, we notice that the textual model was including the lowest accuracies for image retrieval. In contrast, the score-level fusion model was including the highest accuracies for retrieving images.

After deciding to use the Cosine distance metric to measure the similarity among the query image and the rest of the images stored in the dataset, we summarize in table 5.31 the average retrieval time in seconds using the GPU for each retrieved top-*k* similar images calculated from the entire dataset and utilizing the VGG19 deep CNN for the visual model, attentional BiLSTM for the textual model, and the simple-sum score-level for the fusion model.

Table 5.31: Average retrieval time in seconds per *k* similar images

Retrieval Criteria	Top-10	Top-25	Top-50	Top-100
	Visual model			
Manuscript	0.21907	0.71320	1.93504	4.56307
Author	0.23559	0.59772	1.81280	3.92415
Calligraphy	0.29484	0.77313	2.28103	4.76673

Table 5.31: Average retrieval time in seconds per *k* similar images (Continued)

	Textual model			
Manuscript	3.83926	6.28944	7.48153	13.49186
Author	3.80893	5.29627	7.97847	12.48483
Calligraphy	3.92394	5.99753	6.95799	10.96536
	Fusion model			
Manuscript	4.08392	5.94171	7.31670	13.38190
Author	4.60893	6.84184	8.16917	13.48415
Calligraphy	3.92394	6.33289	7.91967	11.57053

From table 5.31, we notice that as much as the number of retrieved images increase, as much as the average retrieval time per seconds also increase. Moreover, we notice that the visual model is faster in its retrieval from the textual and the fusion models because the textual and fusion models extract the text written inside the images and preprocess it using the "Google OCR" tool online. However, the image retrieval system of all the three models is fast in performing the retrieval. Thus, the developed image retrieval system is considered an instantaneous and real-time interactive application.

5.6 Image Retrieval Using the DRL Model

The "Keras-rl" library is used to develop the policy value to action, and the "gym" library is used to create the reinforcement learning model's environment. From "rl.agents.dqn", we imported the "DQNAgent". Moreover, from the "rl.policy", we imported the "BoltzmannQPloicy", and from the "rl.memory", we imported the "SequantialMemory". The "K-NN", as well as, the "LSH Forest" similarity search algorithms combined with the "Cosine" distance metric with a (reward value function=10) to compute the distances among the top-10 similar images to the user's query image and retrieve the most relevant images.

Regarding the learning hyperparameters, we set the min_hash_match to (4). Then, trained the deep reinforcement learning model for (190 episodes), including (100 interaction steps) between the agent and the environment per episode. The environment seed equals (123) to initialize the complete environment per episode, while the agent's warmup steps equal (30). The model's render mode is set to "human" to keep the learning more stable. Finally, we compiled the model using (Adam) optimizer with (0.001) learning rate.

To evaluate the performance of the developed deep reinforcement learning models, we computed the classification accuracy, and the mAP of the top-10 retrieved similar images utilizing both the initial DRQN and the enhanced DRQN, including the hashing function. The results are summarized in table 5.32. The same results are visualized in figure 5.18.

Table 5.32: Evaluation of the DRL models according to the manuscript

Evaluation Parameter	DRQN	DRQN + Hashing
Classification Accuracy	0.9218495	**0.9686520**
Top-10 mAP	0.8963009	0.9431034

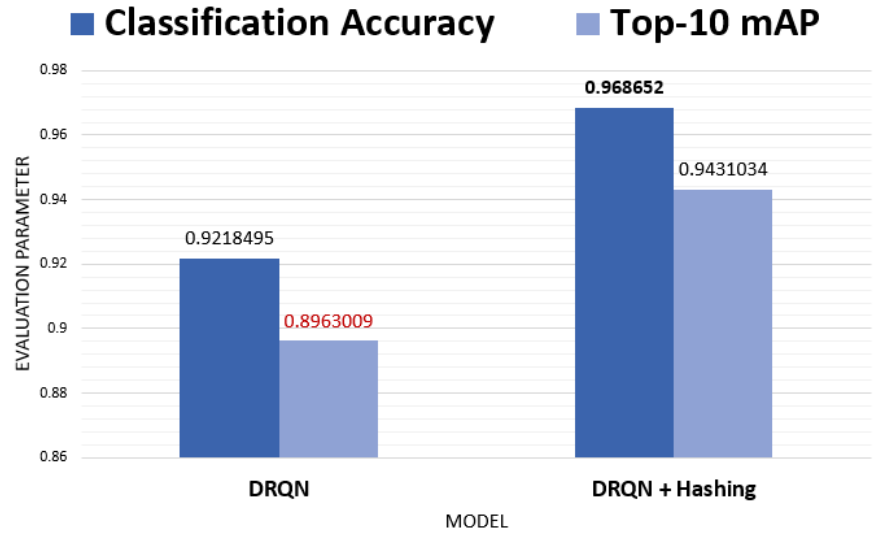


Figure 5.18: Evaluation of the DRL model according to the manuscript

From table 5.32 and figure 5.18, we notice that the recorded evaluation parameters are higher when we employed the hashing function to reduce the dimensionality among the fused features vectors. The use of K-NN algorithm alone to retrieve the similar images were including some randomness in searching the most similar images to the user query image while combining the K-NN algorithm with the LSH Forest method eliminated the existed randomness in the environment and assisted the agent in its learning process, which resulted in a more accurate images retrieval.

Note that, the average retrieval time in seconds using the GPU for each retrieved top-10 similar images calculated from the entire dataset and utilizing the Cosine distance metric within the KNN algorithm equals 5.60275 seconds using the DRQN algorithm alone and 5.92057 seconds using the DRQN algorithm with the hashing function.

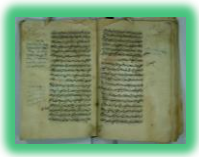
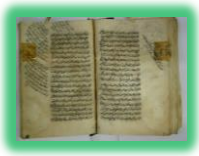
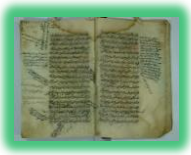
After deciding to use the score-level fusion model of the deep pre-trained VGG19 CNN and the optimized BiLSTM deep learning model, we illustrate in table 5.32 to table 5.34 examples of the output results from the same input query image using the visual VGG19 CNN, the textual BiLSTM deep learning model, and the proposed score-level fusion model according to the three search criteria. Note that, for the image retrieval according to the manuscript search criteria, we experimented the DRL model. Thus, its outputs are shown in table 5.33.

The output images highlighted using the green color are retrieved correctly from the same label as the query image. In contrast, the images highlighted using the red color are retrieved incorrectly from different labels.

Table 5.33: Query input image and its ranked top-5 similar output images according to the manuscript

Input Image

Manuscript ID: **26,** Title: ملتقى الأبحر
Author ID: **21,** Arabic Name: إبراهيم بن محمد الحلبي
Calligraphy ID: **5,** Arabic Name: الديواني

Retrieval from the visual-based model				
Retrieval time: 0.184 seconds Mean accuracy: 80%				
				
1.00000 (26)	0.98794 (26)	0.97930 (26)	0.97442 (26)	0.97273 (16)
Retrieval from the textual-based model				
Retrieval time: 4.413 seconds Mean accuracy: 60%				
				
0.95978 (26)	0.95940 (41)	0.95735 (26)	0.95677 (35)	0.95647 (26)
Retrieval from the fusion model				
Retrieval time: 4.591 seconds Mean accuracy: 100%				
				
0.97598 (26)	0.97596 (26)	0.97596 (26)	0.97591 (26)	0.97587 (26)
Retrieval from the DRL model				
Retrieval time: 5.202 seconds Mean accuracy: 100%				
				
0.99198 (26)	0.98361 (26)	0.98148 (26)	0.97664 (26)	0.97268 (26)

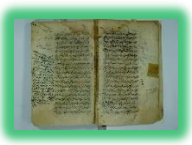
From table 5.33, we notice that the input query image is belonging to manuscript id (26). The similarity score and the manuscript id of each retrieved image are displayed under it. Both the score-level fusion model and the enhanced DRL model, which is based on the features-level fusion of both the visual and the textual features and including hash function, were able to 100% successfully retrieve the top-5 similar images from the same manuscript as the query image. In addition, we notice that there is a variation in the generated similarity scores and that they are in a descending order, which indicates the ability of the models to visualize and distinguish between the Arabic manuscripts' images successfully.

Table 5.34 displays the top-5 similar images according to the author search criteria.

Table 5.34: Query input image and its ranked top-5 similar output images according to the author

Input Image				
				
Manuscript ID: **26,** Title: ملتقى الأبحر Author ID: **21,** Arabic Name: إبراهيم بن محمد الحلبي Calligraphy ID: **5,** Arabic Name: الديواني				
Retrieval from the visual-based model				
Retrieval time: 0.2 seconds Mean accuracy: **100%**				
				
0.96295 (21)	0.96177 (21)	0.94581 (21)	0.94241 (21)	0.9375 (21)
Retrieval from the textual-based model				
Retrieval time: 4.214 seconds Mean accuracy: **100%**				
				
1.00000	1.00000	1.00000	1.00000	1.00000

Table 5.34: Query input image and its ranked top-5 similar output images according to the author (Continued)

(21)	(21)	(21)	(21)	(21)
Retrieval from the fusion model				
Retrieval time: 4.907 seconds Mean accuracy: **100%**				
				
1.00000 (21)	1.00000 (21)	1.00000 (21)	1.00000 (21)	1.00000 (21)

From table 5.34, we notice that the input query image is written by author id (21), and all the three models were able to retrieve the top-5 similar images from the same author as the query image. Table 5.35 displays the top-5 similar images according to the calligraphy search criteria.

Table 5.35: Query input image and its ranked top-5 similar output images according to the calligraphy

Input Image				
				
Manuscript ID: **26,** Title: ملتقى الأبحر Author ID: **21,** Arabic Name: إبراهيم بن محمد الحلبي Calligraphy ID: **5,** Arabic Name: الديواني				
Retrieval from the visual-based model				
Retrieval time: 0.207 seconds Mean accuracy: 80%				
				
0.99545 (5)	0.99522 (5)	0.99516 (5)	0.99513 (5)	0.99410 (1)
Retrieval from the textual-based model				
Retrieval time: 4.01 seconds Mean accuracy: 40%				

Table 5.35: Query input image and its ranked top-5 similar output images according to the calligraphy (Continued)

				
0.98839 (5)	0.98802 (4)	0.96870 (5)	0.96130 (3)	0.95642 (3)
Retrieval from the fusion model				
Retrieval time: 4.849 seconds Mean accuracy: **100%**				
				
0.99421 (5)	0.99150 (5)	0.98037 (5)	0.97738 (5)	0.97279 (5)

From table 5.35, we notice that the query input image is written using “Al-Diwani” Arabic calligraphy with id (5). The visual model retrieved four images written using the same calligraphy as the query image. While the textual model was the weakest in retrieving images written using “Al-Diwani” calligraphy, that is because it retrieved only two images successfully. In contrast, the score-level fusion model was able to 100% successfully retrieve the top-5 similar images to the query input image.

5.7 Theoretical Comparison with State-of-the-art Methods

In this section, we compare the results reached using our proposed method with the other methods worked on the Arabic manuscripts, authors, and calligraphies. We divide the papers according to the features as either handcrafted or deep learning features. The comparison is relative because the datasets used to conduct the studies and the employed classification techniques are different. Moreover, some papers worked on the classification only. While other papers worked on both the

classification and retrieval. Table 5.36 compares the studies that worked on the Arabic manuscripts with our proposed method.

From table 5.36, we notice that four papers [13], [33], [68], and [70] addressed the retrieval of the Arabic manuscript' images. However, all of them used handcrafted features.

For the classification, the researchers used traditional approaches such as the index pages in XML files, as well as; they used classical neural networks. Regarding the used datasets of Arabic manuscripts, there are Sahih Al-Bukhari, Mawaqeet Al-Haj wa Al-Umra, KHATT dataset, Ibn Sina database, ADAB, SANAD, NADiA, and manually collected Arabic manuscripts.

Some of the researchers in table 5.36 indicated the size of the used data as the number of manuscripts images. While other researchers indicated the number of the Arabic characters within the images of the manuscript. The highest recorded recognition rate was using the deep learning features of the SANAD Arabic dataset in [78] as 96.94%. However, we recorded 98.96% recognition and classification accuracy using the supervised deep learning fusion model. Furthermore, we notice that one study [88] used a fusion model, but it utilized handcrafted features. From table 5.36 and after reviewing all the papers worked on the Arabic manuscripts image retrieval. We found that none of the papers used the fusion model of both the visual and textual deep learning features to retrieve the images of the Arabic manuscripts. Moreover, none of the papers used the DRL technique to retrieve the images of the Arabic manuscripts. Thus, it is considered a contribution for us to introduce and develop these new techniques.

Table 5.36: Comparison according to the Arabic manuscripts

	Reference# (Year)	Problem Domain	Input Data	Feature Extraction	Classification	Dataset			Results
						Name	Size	Lang.	
Handcrafted Features	** [13] (2011)	Retrieve Arabic manuscripts' images	Subword images	Word spotting through LSI	Singular Value Decomposition (SVD) algorithm	Sahih Al-Bukhari and Mawaqeet Al-Haj wa Al-Umra	Two pre-scanned ancient Arabic manuscripts	Arabic	Recall: 78.8% using the circular polar grid features set
	[32] (2009)	Recognize handwritten Arabic letters	Image containing one Arabic character	LeNet CNN	Error signal function	Manually collected Arabic texts with different handwriting styles.	758 segmented Arabic characters.	Arabic	There is a need to handle the non-textual parts within manuscripts as images not as text
	** [33] (2019)	Retrieve Arabic manuscripts' images	Image contains one Arabic word	SURF	Compare points of interest and compute the distance using the Euclidean or the Mahalanobis metrics.	HADARA80P	20 images	Arabic	95.27% recognition accuracy.
	[66] (2016)	Arabic texts recognition	Handwritten text image	SVMs or ASMs	Levenstein distance	IESK-arDB SynWords manuall collected dataset	50,000 famous Arabic words vocabularies.	Arabic	Recall: 65.1279% Precision: 64.716%
	[67] (2014)	Arabic texts extraction	Handwritten Arabic text Image	Word spotting through FCM	Euclidean distance	Arabic Handwritten Data-Base	25 images.	Arabic	Extraction rate: 84.8%
	** [68] (2015)	Retrieve Arabic manuscript images	Arabic word image	Word spotting through BoWFs	Histogram intersection and Earth movers distance	Manually scan and collect 120 ancient Arabic images	120 manually scanned ancient Arabic images.	Arabic	Recall: 50% Precision: 89.60%
	[69] (2016)	Arabic texts extraction	Subset of historical text images	OIC transparent neural network	Compute the upper and lower limits of diacritic point	KHATT Arabic dataset	100 handwritten Arabic text.	Arabic	Extraction rate: 74%

Table 5.36: Comparison according to the Arabic manuscripts (Continued)

	** [70] (2017)	Retrieve Arabic manuscript images using text search	Arabic text word	Optical Shape Recognition (OSR)	Index pages in XML file	Ibn Sina" database	51 historical manuscripts' images.	Arabic	NA
	[71] (2009)	Classify Arabic manuscript images	Word binary image	Feedforward technique of multi-language processing neural network	Error signal function	One historical Arabic manuscript named " كشف اللثام عن وجه الإسلام".	27 images.	Arabic	89.3% average accuracy.
	[88] (2014)	Arabic characters recognition	Binary Arabic word	GA-HS fused model	Harmony search character algorithm	Manually collected dataset consisting of 4500 Arabic words and ADAB dataset	4500 Arabic words (24,960 individual characters) to conduct the study.	Arabic	Recognition rate: 93.6% using manual collected dataset Recognition rate: 94.68%-96.33% using ADAB
Deep Learning Features	[78] (2020)	Classify Arabic text	Arabic articles	Attention-GRU	Sigmoid and binary cross entropy	SANAD Arabic dataset	8836 articles.	Arabic	96.94% classification accuracy.
						NADiA Arabic datasets	485k articles.		88.68% classification accuracy.
	Proposed method	Classify and retrieve Arabic manuscript images	Text-based image	Pretrained VGG19 CNN	Softmax dense layer	Collected ancient Arabic manuscripts	8638 images	Arabic	97.68% classification accuracy and 97.28% mean accuracy on the top-10 image retrieval.
				Attentional BiLSTM					83.81% classification accuracy and 64.78% mean accuracy on the top-10 image retrieval.
				Score-level fusion of VGG19 and BiLSTM					**98.96%** classification accuracy and **98.19%** mean accuracy on the top-10 image retrieval.
				Features-level fusion of VGG19 and BiLSTM	DRQN with hash function				96.87% classification accuracy and 94.31% mean accuracy on the top-10 image retrieval.

The papers highlighted using the blue color addressed the Arabic text recognition and classification.

The ** written before the reference number is to highlight that the study addressed the retrieval of the Arabic manuscripts' images.

The green color is to highlight our proposed method.

Table 5.37 summarizes the papers worked on classifying and/or retrieving the authors.

From table 5.37, we found four papers addressing the Arabic authors' classification or retrieval problem and three papers addressing English Authors.

Looking at the paper worked on identifying Arabic authors and utilizing the deep learning features [109], the researchers used a fusion model between the deep CNN and the TF-IDF method, which stands for (Term-Frequency-Inverse Document Frequency). The TF-IDF method is a weighting method often used for information retrieval. However, they recorded only 64% authors' identification accuracy, and that might be due to the use of the TF-IDF traditional approach. On the other hand, using our proposed fusion model of both VGG19 and the attentional bidirectional LSTM, we were able to reach 96.93% authors identification and classification accuracy.

After reviewing the papers worked on the authors' classification and retrieval in table 5.37, we found that we are the only study utilized a fusion of two deep learning models to classify and retrieve the images of the Arabic manuscripts according to the author.

Table 5.38 summarizes the papers that handled the calligraphies.

From table 5.38, we found only four studies addressed the calligraphies. Three of them handled the Arabic calligraphies, while one study handled the Chinese calligraphies. We notice from the table that all the four studies extracted the features using the handcrafted techniques.

Moreover, none of the papers used the fusion model or addressed the image retrieval problem. While “to the best of our knowledge” we are the only study that utilized the fusion deep learning technique for extracting the high-level features and retrieved similar images according to the recognized Arabic calligraphies.

Table 5.37: Comparison according to the authors

	Reference#(Year)	Problem Domain	Features Extraction	Classification	Dataset			Results
					Name	Size	Lang.	
Handcrafted Features	** [45] (2019)	Arabic manuscripts' images retrieval	SURF and BRISK CBIR	Hamming distance and Sum of square distance	Manually collected Arabic manuscripts	1670 images.	Arabic	61% overall accuracy using SURF technique and 37% using BRISK
	** [64] (2018)	Arabic manuscripts' images retrieval	Sparse representation-based technique and handwriting style-based features	K-nearest neighbor	KERTAS ancient Arabic manuscripts	2505 images.	Arabic	94.77% accuracy with predefined folds and 42.31% accuracy with random train/test split using (50×50) size
	** [65] (2017)	Arabic manuscripts' images retrieval	Modified contour-based feature and local key point descriptors	Cosine or Chi-square distance metric	KHATT and IHP ancient Arabic manuscripts	IHP contains 2313 images and KHATT contains 4000 images.	Arabic	88.9% and 73% classification accuracy using KHATT and IHP datasets respectively. 34.6% retrieval accuracy of the top10 images using M-CBF.
Deep Learning Features	[76] (2017)	Authorship Identification	The authors tested four deep learning models named: sentence-level GRU, article-level GRU, article-level LSTM, and article-level Siamese network	Softmax dense layer	"Reuters_50_50" and "Gutenberg" datasets	Reuters contains 5000 images and Gutenberg contains 1286 images.	English	Article-level GRU was the best performing model recording 69.1% and 89.2% accuracy on Reuters and Gutenberg datasets respectively
	[85] (2015)	Authorship Identification	Multi-headed Recurrent Neural Network (RNN)	Rectified Shifted Square Root (ReSQRT)	PAN 2014	__	English	Higher than 80% AUC

Table 5.37: Comparison according to the authors (Continued)

	[86] (2018)	Writer Identification based on single handwritten word images	The authors tested three methods: 1) Baseline, 2) linear adaptive, and 3) deep adaptive learning	Sigmoid dense layer	CVL and IAM datasets	CVL contains 99513 images and IAM contains 49625 images.	English	The deep adaptive learning was the best method recording 78.6% and 69.5% top-1, as well as, 93.7% and 86.1% top-5 recognition rates using the CVL and IAM datasets respectively
	[109] (2017)	Authorship Identification	Confusion between deep CNN and (TFIDF) model	Softmax dense layer	Four tweet collections from Twitter	—	Arabic	64% Arabic authors identification accuracy
	Proposed method	Classify and retrieve Arabic manuscript images	Pretrained VGG19 CNN	Softmax dense layer	Collected ancient Arabic manuscripts	8638 images.	Arabic	96.63% classification accuracy and 95.42% mean accuracy on the top-10 image retrieval.
			Attentional BiLSTM					90.30% classification accuracy and 72.64% mean accuracy on the top-10 image retrieval.
			Score-level fusion of VGG19 and BiLSTM					**96.93%** classification accuracy and **96.70%** mean accuracy on the top-10 image retrieval.

The papers highlighted using the blue color addressed the Arabic text recognition and classification.

The ** written before the reference number is to highlight that the study addressed the retrieval of the Arabic manuscripts' images.

The green color is to highlight our proposed method.

Table 5.38: Comparison according to the calligraphies

	Reference# (Year)	Problem Domain	Feature Extraction	Classification	Dataset			Calligraphy		Results
					Name	Size	Lang.	No.	Type	
Handcrafted Features	[34] (2019)	Classify Arabic manuscript images	SIFT algorithm	Gaussian Naive Bayes (GNB)	Two historical Islamic Arabic books	200 images	Arabic	2	• Naskh • Reqaa	92% recognition accuracy
	[72] (2016)	Classify Arabic manuscript images	Genetic algorithm	Neural network module	Local dataset written by calligraphers	89 images	Arabic	3	• Thuluth • Reqaa • Kufi	8.02% recognition error rate
					Public dataset generated by the Computer	113284 images	Arabic	10	• AdvertisingBold • Andalus • ArabicTransparent • DecoTypeNaskh • DecoTypeThuluth • DiwaniLetter • MUnicodeSara • SimplifiedArabic • Tahoma • TraditionalArabic	7.55% recognition error rate
	[73] (2011)	Classify Arabic manuscript images	Edge direction Matrices (EDMS)	Backpropagation neural network	Selected Arabic images	14 images	Arabic	7	• Thuluth • Andalusi • Diwani • Persian • Kufi • Naskh • Roqaa	43.7% recognition accuracy

Table 5.38: Comparison according to the calligraphies (Continued)

	[74] (no date)	Classify Arabic manuscript images	HOG descriptor	Softmax Regression	Single character images	2000 images	Chinese	5	• Seal • Clerical • Regular • Running • Cursive	95.55% recognition accuracy
Deep Learning Features	Proposed method	Classify and retrieve Arabic manuscript images	Pretrained VGG19 CNN	Softmax dense layer	Collected ancient Arabic manuscripts	8638 images.	Arabic	6	• Al-Nask • Al-Thulth • Al-Reqaa • Al-Hur • Al-Diwani • Al-Farsi	96.88% classification accuracy and 98.76% mean accuracy on the top-10 image retrieval.
			Attentional BiLSTM							83.60% classification accuracy and 49.62% mean accuracy on the top-10 image retrieval.
			Score-level fusion of VGG19 and BiLSTM							**98.36%** classification accuracy and **99.18%** mean accuracy on the top-10 image retrieval.

The papers highlighted using the blue color addressed the Arabic text recognition and classification.

The ** written before the reference number is to highlight that the study addressed the retrieval of the Arabic manuscripts' images.

The green color is to highlight our proposed method.

Looking at the used datasets in table 5.38, we found that the first study [34] classified only two calligraphies from 200 images. While the second study [72] classified 13 Arabic calligraphies. Three calligraphies belonging to 89 handwritten images, while ten Arabic calligraphies belong to 113284 public computer printed images. The third study [73] classified seven Arabic calligraphies but, from 14 images only. Regarding the paper worked on the Chinese calligraphies [74], five Chinese calligraphies recognized from 2000 images, and this study recorded the highest recognition accuracy as 95%. However, we classified six Arabic calligraphies from 8638 images and recorded 98% recognition and classification accuracy.

Chapter V

Conclusion and Future Works

6.1 Conclusion

The study introduces a novel solution to classify and retrieve the top-k similar images to a user query image according to three search criteria, which are the manuscript, Author, and calligraphy. Thus, four different models developed based on the visual features, textual features, a fusion of both visual and textual models, and using the reinforcement learning approach.

After classifying the images according to their extracted visual features, the similarity is investigated using three different static distance metrics, in addition to investigating the performance of the Siamese deep neural network with the visual model. The Siamese deep neural network performs well in matching and retrieving similar images, but it increases the dataset size exponentially, which made it slow.

Compared with the Manhattan and Euclidean static distance metrics, the Cosine distance metric is the most accurate method for computing the similarity scores.

After experimenting four pre-trained deep CNNs for extracting the images visual features, we reached that the VGG19 combined with the Cosine distance metric is the best performing model recording 97.28%, 95.42%, 98.76% mAP on the top-10 image retrieval according to the manuscript, author, and calligraphy, respectively.

Besides classifying the images, the text also classified using the classical LSTM deep learning model. The initial recorded results from the classical LSTM deep learning model for text classification were not satisfying. Thus, the textual model optimized through using the bidirectional technique, as well as through adding the attention and the batch normalization layers.

The optimized BiLSTM model combined with the Cosine static distance metric recorded the highest results as 64.78%, 72.64%, and 49.62% mAP on the top-10 image retrieval according to the manuscript, author, and calligraphy, respectively.

After classifying the images and textual contents, three different fusion models are investigated. Reaching that the simple-sum score-level fusion of both the VGG19 CNN and the attentional BiLSTM deep learning model is outperforming the other experimented fusion methods, and it's generating higher evaluation parameters than each used model separately, recording **98.19%, 96.70%, and 99.18%** mAP on the top-10 image retrieval according to the manuscript, author, and calligraphy, respectively.

Finally, an enhanced DRL model is developed, including the LSH Forest hashing function and recorded 96.87% classification accuracy and 94.31% mAP on the top-10 image retrieval according to the manuscript criteria.

6.2 Limitations

The solution focused on the retrieval according to the complete image only and not according to the text or part of an image. Moreover, only three search criteria were used, which are manuscript, author, and calligraphy. While we could employ the retrieval according to the genre also. However, the reason that made us not use the genre criteria is that because most of the manuscripts in our ancient Arabic dataset

are religious, and only a few of them were linguistics and literature. Thus, we found that classifying the images according to the genre criteria will not be distinguishing.

6.3 Future Work

- Integrate the image retrieval system with a big data solution method to facilitate the management and control of the data along with their extracted features vectors.
- Make the fusion model parallel and experiment assembling all the three fusion models into one unified ensembled model.
- Make this image retrieval system the largest database for the ancient Arabic manuscripts and the main search engine using the manuscripts' pictures. Therefore, we will increase the number of the Arabic manuscripts in the dataset, as much as possible, trying to collect all the manuscripts in one dataset.
- Improve the web-based prototype by making quality analysis of the scalability and response time.
- Enhance the solution further by making it GPU-architecture independent. So, the image retrieval system should run successfully using any GPU architecture.
- Experiment the developed DRL model using the score-level and the decision-level fusion models, as well as customize it to retrieve images according to the author and calligraphy criteria.

APPENDIX

We developed and deployed a web-based prototype for the image retrieval system. The dataset uploaded into the "Google Cloud Platform". "Flask"[16] is used for the backend codes, while the "Angular Material"[17] component is used for the front-end design. The domain name is bought from "Godaddy"[18].

Link to the web-based prototype: https://arabic-image-retrieval.com/

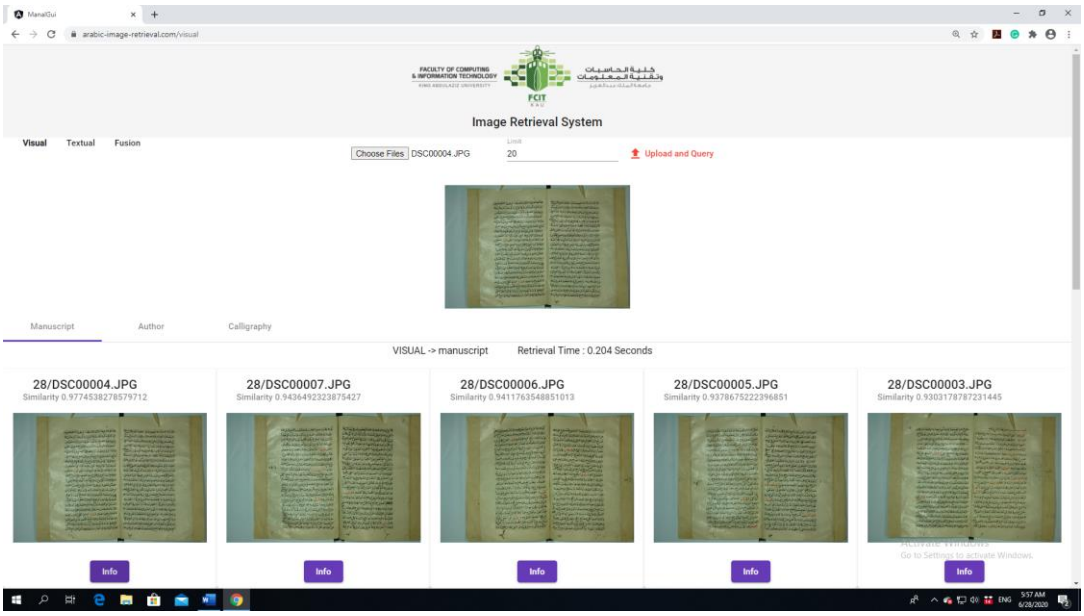
Link to the supervised deep learning models code in Github:

https://github.com/ManalKhayyat/Arabic_Manuscripts_Image_Retrieval.git

[16] https://flask.palletsprojects.com/

[17] https://material.angular.io/

[18] https://ae.godaddy.com/



We notice from the interface that there are three different image retrieval models located at the top-left of the home page, which are: Visual, Textual, and Fusion.

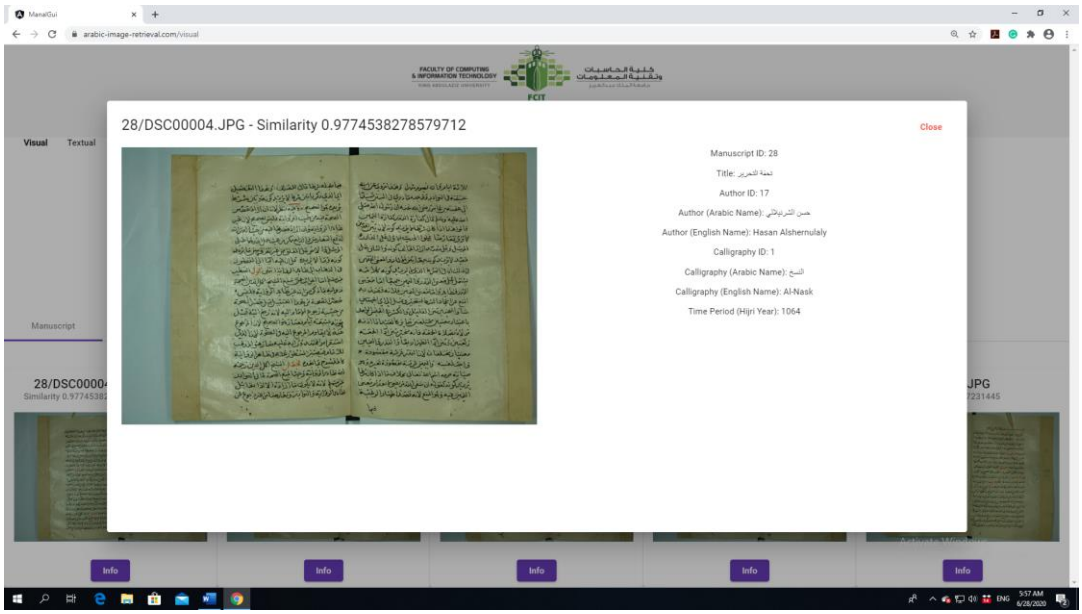
The user can choose any query image and upload it to the system. The limit for the top-similar retrieved images is set to (20) images by default, but the user can select any other different limit.

After uploading the input query image, the user should click on the “Upload and Query” button to be able to visualize the similar images to the user query image.

The name and the similarity score of each retrieved image is displayed on the top of it. The similarity scores are ranked in a descending order. Thus, the first image has the highest similarity score. The user can check the retrieval time in seconds, as well as, can check the performance of each model according to three search criteria, which are the manuscript, author, and calligraphy.

The detailed information of the images including the manuscript they belong to, the author wrote the manuscript, the used calligraphy, and the time period of the

manuscript in the “Hijri” date, can be displayed by clicking on the small purple button named “info” under each retrieved image.



REFERENCES

[1] S. P. Rana, M. Dey, and P. Siarry, "Boosting content based image retrieval performance through integration of parametric & nonparametric approaches," *J. Vis. Commun. Image Represent.*, vol. 58, pp. 205–219, 2019.

[2] W. Zhou and J. Jia, "A learning framework for shape retrieval based on multilayer perceptrons," *Pattern Recognit. Lett.*, vol. 117, pp. 119–130, 2018.

[3] Z. Xia, J. Lin, and X. Feng, "Trademark image retrieval via transformation-invariant deep hashing," *J. Vis. Commun. Image Represent.*, vol. 59, pp. 108–116, 2019.

[4] V. Tyagi, "Research Issues for Next Generation Content-Based Image Retrieval," *Springer Nat. Singapore Pte Ltd*, pp. 295–302, 2017.

[5] C. Beecks, S. Kirchhoff, and T. Seidl, "On stability of signature-based similarity measures for content-based image retrieval," *Multimed. Tools Appl.*, vol. 71, no. 1, pp. 349–362, 2014.

[6] R. Shah, J. Vaghela, K. Surve, and R. Shah, "Comparative Performance Study of Various Content Based Image Retrieval Methods," vol. 50, pp. 397–407, 2016.

[7] P. Altoe, Class Lecture, Topic: "Fundamentals of Deep Learning for Computer Vision", *Supercomputing Laboratory, King Abdullah University of Science and Technology. KAUST, Jeddah,* Mar. 2019.

[8] M. Al-Ayyoub, A. Nuseir, K. Alsmearat, Y. Jararweh, and B. Gupta, "Deep learning for Arabic NLP: A survey," *J. Comput. Sci.*, vol. 26, pp. 522–531, 2018.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[10] S. Ren, K. He, R. Girshick, and S. Jian, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," vol. 3, no. 1, pp. 1–14, 2016.

[11] M. Al-Yahya, "Stylometric analysis of classical Arabic texts for genre detection," vol. 36, no. 5, pp. 842–855, 2018.

[12] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "PCANet: A Simple Deep Learning Baseline for Image Classification?," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5017–5032, 2015.

[13] M. H. N. Yahia, "Content-Based Retrieval of Arabic Historical Manuscripts

Using Latent Semantic Indexing," King Fahd university of Petroleum and Minerals, 2011.

[14] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, 2015.

[15] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative Deep Learning for Recommender Systems," pp. 1235–1244, 2014.

[16] B. Marr, "Artificial Intelligence: What's The Difference Between Deep Learning And Reinforcement Learning?," 22-Oct-2018 [Online] *Forbes*. Available at: https://www.forbes.com/sites/bernardmarr/2018/10/22/artificial-intelligence-whats-the-difference-between-deep-learning-and-reinforcement-learning/#6e010ff7271e (accessed 05 Jun 2020).

[17] M. Wiering, and M. V. Otterlo, "Reinforcement Learning: State-of-the-Art," *New York, NY, USA: Springer*, 2012.

[18] R. S. Sutton, and A. G. Barto, "Reinforcement Learning : An Introduction," *Cambridge, MA, USA: MIT Press*, 2015.

[19] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: a review," *Eurasip J. Wirel. Commun. Netw.*, vol. 211, no. 1, pp. 1–12, 2017.

[20] A. V. N. Reddy and C. P. Krishna, "A Survey on Applications and Performance of Deep Convolution Neural Network Architecture for Image Segmentation," *Int. J. Pure Appl. Math.*, vol. 118, no. Special Issue 19A, pp. 43–60, 2018.

[21] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," *Rom. J. Phys.*, vol. 61, no. 5–6, pp. 1120–1132, 2017.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *J. Geotech. Geoenvironmental Eng.*, vol. 12, p. 04015009, 2015.

[23] V. Nigam, "Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning," Lect, 11-Sep-2018. [Online] *Towards Data Science*. Available at: https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90 (accessed 17 Apr 2019).

[24] Colah, "Understanding LSTM Networks," Posted blog on 27-Aug-2015 [Online]. Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed 04 Dec 2019).

[25] S. Elmowafy, "Neural Network Recurrent Neural Network in Arabic," Youtube

Lect9 (1-2) posted on 17-May-2017 [Online]. Available at: https://www.youtube.com/watch?v=baD2JuEju28 (accessed 26 Mar 2019).

[26] S. Hochreiter and J. Schmidhuber, "LONG SHORT-TERM MEMORY," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available at: cs231n.github.io/convolutional-networks/#case (accessed 15 Jul 2019).

[28] F. Altenberger, and C. Lenz, "A Non-Technical Survey on Deep Convolutional Neural Network Architectures," 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Enzyme Microb. Technol.*, vol. 19, no. 2, pp. 107–117, 2015.

[30] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," pp. 1–60, 2018.

[31] F. Alaei, A. Alaei, U. Pal, and M. Blumenstein, "A Comparative Study of Different Texture Features for Document Image Retrieval," *Expert Syst. Appl.*, vol. 121, pp. 97–114, 2018.

[32] R. Al-Jawfi, "Handwriting Arabic Character Recognition LeNet Using Neural Network," *Int. Arab J. Inf. Technol.*, vol. 6, no. 3, pp. 304–309, 2009.

[33] N. El-Makhfi, "A Word Spotting Method for Arabic Manuscripts Based on Speeded Up Robust Features Technique," vol. 4, no. 6, pp. 99–107, 2019.

[34] M. Ezz, M. A. Sharaf, and A. A. Hassan, "Classification of Arabic Writing Styles in Ancient Arabic Manuscripts," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no.10, pp. 409–414, 2019.

[35] Y. A. Aslandogan and C. T. Yu, "Techniques and Systems for Image and Video Retrieval," *World Wide Web Internet Web Inf. Syst.*, vol. 11, no. 1, pp. 1–19, 1999.

[36] A.M. Marshall, and S. Gunasekaran, "A Survey on Image Retrieval Methods," *CIET-ECE DEPT Conf.*, pp. 1–16, 2014.

[37] M. Saleem, R. Senthilkumar, and T. S. Prakash, "Image Retrieval System by Automatic Annotation," vol. 1, no. 8, pp. 286–290, 2014.

[38] A. Kumar et al., "Adapting content-based image retrieval techniques for the semantic annotation of medical images," *Comput. Med. Imaging Graph.*, vol. 49, pp. 37–45, 2016.

[39] J. S. Hare, P. H. Lewis, P. G. B. Enser, and C. J. Sandom, "Mind the Gap: Another look at the problem of the semantic gap in image retrieval," *Multimed. Content Anal. Manag. Retr.*, p. 607-309, 2006.

[40] K. S. Reddy and K. Sreedhar, “Image Retrieval Techniques: A Survey,” *IOSR Journal of Computer Engineering (IOSR-JCE)*, pp. 75–79, 2019.

[41] Y. Rui, T. S. Huang, and S. Mehrotra, “Content-Based Image Retrieval With Relevance Feedback in MARs,” pp. 815–818, 2002.

[42] S. A. Wadhai and S. S. Kawathekar, “Techniques of Content Based Image Retrieval : A Review,” *IOSR Journal of Computer Engineering (IOSR-JCE)*, pp. 75–79, 2019.

[43] G. F. Ahmed and R. Barskar, “A Study on Different Image Retrieval Techniques in Image Processing,” *Journal of Computing and Engineering (IJSCE)*, vol. 1, no. 4, pp. 247–251, 2011.

[44] A. Dureja and P. Pahwa, “Image retrieval techniques: a survey,” *Int. J. Eng. Technol.*, vol. 7, nos. 1-2, p. 215-219, 2018.

[45] B. Bagasi, and L. Elrefaei, “Arabic Manuscript Content Based Image Retrieval: A Comparison between SURF and BRISK Local Features,” *Int. J. Comput. Digit. Syst.*, vol. 7, no. 6, pp. 355–364, 2019.

[46] F. Chen, B. Li, and L. Li, “3D object retrieval with graph-based collaborative feature learning,” *J. Vis. Commun. Image Represent.*, vol. 58, pp. 261–268, 2019.

[47] K. T. Ahmed, S. Ummesafi, and A. Iqbal, “Content based image retrieval using image features information fusion,” *Inf. Fusion*, vol. 51, no. 3, pp. 76–99, 2019.

[48] S. Raju, K. Sreelatha, and S. Kumari, “An Efficient Approach to Improve Retrieval Rate in Content Based Image Retrieval Using MPEG-7 Features,” *Adv. Intell. Syst. Comput.*, vol. 248, pp. 337–347, 2014.

[49] F. Radenovic, G. Tolias, and O. Chum, “Fine-tuning CNN Image Retrieval with No Human Annotation,” *Tpami.*, vol. 2, pp. 1–14, 2018.

[50] O. Seddati, S. Dupont, S. Mahmoudi, and M. Parian, “Towards Good Practices for Image Retrieval Based on CNN Features,” *Proc. - 2017 IEEE Int. Conf. Comput. Vis. Work. ICCVW*, pp. 1246–1255, 2018.

[51] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese Neural Networks for One-shot Image Recognition,” *Proc. 32 Nd Int. Conf. Mach. Learn. Lille, Fr. JMLR W&CP*. vol. 37, pp. 1–8, 2015.

[52] Y. Ge, S. Jiang, Q. Xu, C. Jiang, and F. Ye, “Exploiting representations from pre-trained convolutional neural networks for high-resolution remote sensing image retrieval,” vol. 77, pp. 17489–17515, 2018.

[53] E. Ong, S. Husain, and M. Bober, “Siamese Network of Deep Fisher-Vector Descriptors for Image Retrieval,” pp. 1–12, 2017.

[54] K. Qiu, Y. Ai, B. Tian, B. Wang, and D. Cao, “Siamese-ResNet : Implementing

Loop Closure Detection based on Siamese Network," *IEEE Intell. Veh. Symp.*, pp. 716–721, 2018.

[55] U. Chaudhuri, B. Banerjee, and A. Bhattacharya, "Siamese graph convolutional network for content based remote sensing image retrieval," *Comput. Vis. Image Underst.*, vol. 184, pp. 22–30, 2019.

[56] K.L. Wiggers, A.S. Britto, L. Heutte, A.L. Koerich, and L.S. Oliveira, "Image Retrieval and Pattern Spotting using Siamese Neural Network," pp. 1–8, 2018.

[57] S. Ioffe, and C. Szegedy, "Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift," pp. 1–11, 2015.

[58] D. Zhao, Y. Chen, and L. Lv, "Deep Reinforcement Learning With Visual Attention for Vehicle Classification," IEEE Trans. Cogn. Dev. Syst., vol. 9, no. 4, pp. 356–367, 2017.

[59] E. Lin, Q. Chen, and X. Qi, "Deep Reinforcement Learning for Imbalanced Classification," *Sch. Comput. Sci. Eng. South China Univ. Technol. Guangzhou, China*, pp. 1–9, 2019.

[60] W. Nie, W. Wang, A. Liu, and C. Chen, "Characteristic Views Extraction Modal Based-On Deep Reinforcement Learning For 3d Model Retrieval," *2019 IEEE Int. Conf. Image Process.*, pp. 2389–2393, 2019.

[61] H. Wang, K. Wang, Y. Wu, Z. Wang, and L. Zou, "User preference-aware video highlight detection via deep reinforcement learning," *Multimed. Tools Appl.*, vol. 79, pp. 15015–15024, 2020.

[62] J. Zhou and E. Agichtein, "RLIRank : Learning to Rank with Reinforcement Learning for Dynamic Search," *In Proceedings of TheWeb Conference 2020 (WWW'20), ACM, New York, NY, USA*, pp. 2842–2848, 2020.

[63] J. Yao, Z. Dou, J. Xu, and J. Wen, "RLPer : A Reinforcement Learning Model for Personalized Search," *In Proceedings of TheWeb Conference 2020 (WWW'20), ACM, New York, NY, USA*, pp. 2298–2308, 2020.

[64] K. Adam, A. Baig, S. Al-Maadeed, A. Bouridane, and S. El-Menshawy, "KERTAS: dataset for automatic dating of ancient Arabic manuscripts," *Int. J. Doc. Anal. Recognit.*, vol. 21, no. 4, pp. 283–290, 2018.

[65] A. Asi, A. Abdalhaleem, D. Fecker, V. Märgner, and J. El-Sana, "On writer identification for Arabic historical manuscripts," *Int. J. Doc. Anal. Recognit.*, vol. 20, no. 3, pp. 173–187, 2017.

[66] L. Dinges, A. Al-Hamadi, M. Elzobi, and S. El-Etriby, "Synthesis of common arabic handwritings to aid optical character recognition research," *Sensors (Switzerland)*, vol. 16, no. 3, pp. 1–25, 2016.

[67] A. Al-Dmour and F. Fraij, "Segmenting Arabic Handwritten Documents into Text lines and Words," vol. 6, no. 3, pp. 109–119, 2014.

[68] R. A. A. Othman, “Arabic Manuscripts Analysis and Retrieval,” *PhD Thesis, King Fahad Univerity of Petrolum and Minerals*, pp. 1–199, 2015.

[69] S. Snoussi, F. Ghazouani, and Y. Wahabi, “Text lines Segmentation of Handwritten Arabic Script using Outer Isothetic Cover,” *Graphics, Vision and Image Processing Journal, USA*, vol. 16, no. 1, pp. 47–55, 2016.

[70] S. Al-Maadeed, F. Issawi, and A. Bouridan, “Word Retrieval System for Ancient Arabic Manuscripts,” *2017 9th IEEE-GCC Conf. Exhib.*, pp. 1–5, 2017.

[71] Z. Al Aghbari and S. Brook, “Word Stretching for Effective Segmentation and Classification of Historical Arabic Handwritten Documents,” *Proc. 2009 3rd Int. Conf. Res. Challenges Inf. Sci. RCIS 2009*, pp. 217–224, 2009.

[72] S. R. Allaf, and R. Al-Hmouz, “Automatic Recognition of Artistic Arabic Calligraphy Types,” *JKAU: Eng. Sci.*, vol. 27, no. 1, pp. 3–17, 2016.

[73] B. Bataineh, S. N. H. Abdullah, and K. Omar, “Arabic Calligraphy Recognition Based on Binarization Methods and Degraded Images,” *International Conference on Pattern Analysis and Intelligent Robotics*, vol. 11, pp. 65–70, 2011.

[74] C. Yu-Sheng, S. Guangjun, and L. Haihong, “Machine Learning for Calligraphy Styles Recognition,” (no date).

[75] Y. Peng, J. Zhang, and Z. Ye, “Deep Reinforcement Learning for Image Hashing,” *arXiv:1802.02904*, vol. 2, pp. 1–12. 2018.

[76] C. Qian, T. He, and R. Zhang, “Deep Learning based Authorship Identification,” *Report, Stanford University*, pp. 1–9, 2017.

[77] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu, “AttentionXML : Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification,” pp. 1–17. 2019.

[78] A. Elnagar, R. Al-Debsi, and O. Einea, “Arabic text classification using deep learning models,” *Inf. Process. Manag.*, vol. 57, pp. 102-121, 2020.

[79] G. Liu, and J. Guo, “Bidirectional LSTM with attention mechanism and convolutional layer for text classificatio,” *Neurocomputing.*, 2019.

[80] J. Du, L. Gui, R. Xu, and Y. He, “A Convolutional Attention Model for Text Classification,” vol. 3, pp. 183–195, 2018.

[81] T. Liu, S. Yu, B. Xu, and H. Yin, “Recurrent networks with attention and convolutional networks for sentence representation and classification,” *Appl. Intell.*, 2018.

[82] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical Attention Networks for Document Classification,” *Proc. NAACL-HLT.*, pp.

1480–1489, 2016.

[83] S. Gao, A. Ramanathan, and G. Tourassi, “Hierarchical Convolutional Attention Networks for Text Classification,” *Proc. Ofthe 3rd Work. Represent. Learn. NLP.*, pp. 11–23, 2018.

[84] K. Saeed and M. Albakoor, “Region growing based segmentation algorithm for typewritten and handwritten text recognition,” *Appl. Soft Comput. J.*, vol. 9, no. 2, pp. 608–617, 2009.

[85] D. Bagnall, “Author identification using multi-headed recurrent neural networks, ” *CLEF (Working Notes) arXiv:1506.04891v2*, pp. 1–11, 2015.

[86] S. He, and L. Schomaker, “Deep Adaptive Learning for Writer Identification based on Single Handwritten Word Images,” *Pattern Recognit. arXiv:1809.10954v1* , pp. 06–27, 2018.

[87] T. Chen, Z. Wang, G. Li, and L. Lin, “Recurrent Attentional Reinforcement Learning for Multi-label Image Recognition,” *SenseTime Gr. Ltd.*, pp. 1–8, 2017.

[88] M. Y. Potrus, U. K. Ngah, and B. S. Ahmed, “An evolutionary harmony search algorithm with dominant point detection for recognition-based segmentation of online Arabic text recognition,” *Ain Shams Eng. J.*, vol. 5, no. 4, pp. 1129–1139, 2014.

[89] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, “CNN-RNN: A Unified Framework for Multi-label Image Classification,” *J. Japan Soc. Cancer Ther. 13th Cong.*, pp. 245–246, 2016.

[90] U. Sharif, Z. Mehmood, T. Mahmood, M.A. Javid, A. Rehman, and T. Saba, “Scene analysis and search using local features and support vector machine for effective content-based image retrieval,” *Artif. Intell. Rev.*, vol. 52, pp. 901–925, 2018.

[91] L. Guo, D. Zhang, L. Wang, H. Wang, and B. Cui, “CRAN : A Hybrid CNN-RNN Attention-Based Model for Text Classification,” *Springer International Publishing*, 2018.

[92] A. Koesdwiady, R. Soua, and F. Karray, “Improving Traffic Flow Prediction With Weather Information in Connected Cars: A Deep Learning Approach,” pp. 0018–9545, 2016.

[93] J. Li, T. Qiu, C. Wen, K. Xie, and F. Wen, “Robust Face Recognition Using the Deep C2D-CNN Model Based on Decision-Level Fusion,” vol. 18, pp. 1–27 (2018).

[94] Y. Su, K. Zhang, J. Wang, and K. Madani, “Environment Sound Classification Using a Two-Stream CNN Based on Decision-Level Fusion,” vol. 19, pp. 1–15, 2019.

[95] Y. Xie, J. Zhang, Y. Xia, M. Fulham, and Y. Zhang, "Fusing Texture, Shape and Deep Model-Learned Information at Decision Level for Automated Classification of Lung Nodules on Chest CT," *Inf. Fusion.*, pp. 1566–2535. 2017.

[96] P. Liu, J. Guo, C. Wu, and D. Cai, "Fusion of Deep Learning and Compressed Domain features for Content Based Image Retrieval," pp. 1057–7149, 2017.

[97] D. Sudha, and M. Ramakrishna, "Comparative Study of Features Fusion Techniques," *2017 Int. Conf. Recent Adv. Electron. Commun. Technol.*, vol. 39, pp. 235–239, 2017.

[98] H. Suk, S. Lee, and D. Shen, "Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis," *Neuroimage.*, pp. 1053–8119, 2014.

[99] S. Zhang, S. Zhang, T. Huang, W. Gao, and Q. Tian, "Learning Affective Features with a Hybrid Deep Model for Audio-Visual Emotion Recognition," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 1051–8215, 2017.

[100] K. Vishi, and V. Mavroeidis, "An Evaluation of Score Level Fusion Approaches for Fingerprint and Finger-vein Biometrics," *Proc. Ofthe 10th Nor. Inf. Secur. Conf. (NISK 2017), Oslo, Norw.*, pp. 1–11, 2017.

[101] C.C. Lip, and D.A. Ramli, "Comparative Study on Feature, Score and Decision Level Fusion Schemes for Robust Multibiometric Systems," *Front. Comput. Educ.*, vol. 133, pp. 941–948, 2012.

[102] H. Kaya, F. Gürpınar, and A.A. Salah, "Video-based emotion recognition in the wild using deep transfer learning and score fusion," *Image Vis. Comput.*, vol. 65, pp. 66-75, 2017.

[103] B. N. Kang, Y. Kim, and D. Kim, "Deep Convolutional Neural Network using Triplets of Faces , Deep Ensemble, and Score-level Fusion for Face Recognition," *IEEE Conf. Comput. Vis. Pattern Recognit. Work., Honolulu, HI, USA*, pp. 109–116, 2017.

[104] S.N.B. Bhushan, and A. Danti, "Classification of text documents based on score level fusion approach," *Pattern Recognit. Lett.*, pp. 0167–8655, 2017.

[105] A. George, and A. Routray, "A Score-level Fusion Method for Eye Movement Biometrics," pp. 1-11, 2016.

[106] Y. Jhansi, and E.S. Reddy, "A Methodology for Sketch based Image Retrieval based on Score level Fusion," Int. J. Comput. Appl., vol. 109, no. 3, pp. 0975–8887, 2015.

[107] M. Jovic, Y. Hatakeyama, F. Dong, and K. Hirota, "Image Retrieval Based on Similarity Score Fusion from Feature Similarity Ranking Lists," Springer-Verlag Berlin Heidelb., pp. 461–470, 2006.

[108] W. Xue, Q. Li, and Q. Xue, “Text Detection and Recognition for Images of Medical Laboratory Reports With a Deep Learning Approach,” IEEE Access, vol. 8, pp. 407–416, 2020.

[109] N. Schaetti, “UniNE at CLEF 2017 : TF-IDF and Deep-Learning for Author Profiling Notebook for PAN at CLEF 2017,” pp. 1–11, 2017.

[110] H. A. Al-Muzaini, T. N. Al-Yahya, and H. Benhidour, “Automatic Arabic Image Captioning using RNN-LSTM-Based Language Model and CNN,” 2018.

[111] Y. Roh, G. Heo, and S. E. Whang, “A Survey on Data Collection for Machine Learning,” *Sch. Electr. Eng. KAIST, Daejeon, Korea*, pp. 1–20, 2019.

[112] S. Guo, Y. Liu, R. Chen, X. Sun, and X. Wang, “Improved SMOTE Algorithm to Deal with Imbalanced Activity Classes in Smart Homes,” *Neural Process. Lett.*, 2018.

[113] Y. Ho and S. Wookey, “The Real-World-Weight Cross-Entropy Loss Function : Modeling the Costs of Mislabeling,” *IEEE Access*, pp. 1–9, 2017.

[114] P. Goldsborough, “A Tour of TensorFlow,” *arXiv preprint* arXiv:1610.01178, pp. 1-16, 2016.

[115] S. Bahrampour, N. Ramakrishnan, L. Schott, M. Shah, “Comparative Study of Deep Learning Software Frameworks,” *Res. Technol. Center, Robert Bosch LLC.* arXiv:1511.06435, pp. 1-9, 2015.

[116] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” *arXiv preprint* arXiv:1704.04861, pp. 1-9, 2017.

[117] K. He, X. Zhang, S. Ren, J. Sun, “Identity Mappings in Deep Residual Networks,” *European Conference on Computer Vision, Springer,* arXiv:1603.05027, pp. 630-645, 2016.

[118] S. Zagoruyko, N. Komodakis, “Wide Residual Networks,” *Univ. Paris-Est, École Des Ponts, Paris Tech, Fr. arXiv preprint* arXiv:1605.07146, pp. 1-15, 2017.

[119] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, “Densely Connected Convolutional Networks,” *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700-4708, 2018.

[120] S.H. Tsang, “Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014,” 22-Aug-2018 [Online] *Medium.* Available at: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11 (accessed 03 March 2019).

[121] K. Simonyan, A. Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," *Am. J. Heal. Pharm. ICLR 2015 conference paper*, arXiv:1409.1556, vol. 75, p. 398–406, 2015. doi: 10.2146/ajhp170251.

[122] H. Lamba, "One Shot Learning with Siamese Networks using Keras," 21-Jan-2019 [Online] *Medium*. Available at: https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d (accessed 04 Jan 2020).

[123] L. Liu, and H. Qi, "Learning Effective Binary Descriptors via Cross Entropy," pp. 1251–1258, 2017.

[124] P. Singh, "Siamese Network Keras for Image and Text similarity," 24-Aug-2019 [Online] *Medium*. Available at: https://medium.com/@prabhnoor0212/siamese-network-keras-31a3a8f37d04 (accessed 02 Feb 2020).

[125] S. Prabhakaran, "Cosine Similarity – Understanding the math and how it works (with python codes)," 22-Oct-2018 [Online] Machine Learning Plus. Available at: https://www.machinelearningplus.com/nlp/cosine-similarity/ (accessed 29 Feb 2020).

[126] P. Sitikhu, K. Pahi, P. Thapa, and S. Shakya, "A Comparison of Semantic Similarity Methods for Maximum Human Interpretability," *arXiv:1910.09129v2*, pp. 1–4, 2019.

[127] R. Ponakala, H. K. Adda, C. A. Kumar, K. Avula, and K. A. Sheela, "Character Recognition And Extraction of An Indian State License Plates Using K-NN," *Jawaharlal Nehru Technological University Hyderabad*. 2016.

[128] A. Soliman, K. Eissa, and S. El-Beltagy, "AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. Procedia Computer Science," vol. 117, pp. 256-265, 2017.

[129] F. H. Khan, U. Qamar, and S. Bashir, "SentiMI : Introducing point-wise mutual information with SentiWordNet to improve sentiment polarity detection," *Appl. Soft Comput. J.* vol. 39, pp. 140–153, 2016.

[130] F. Schilling, "The Effect of Batch Normalization on Deep Convolutional Neural Networks," *KTH Royal Institute of Technology*, pp. 1-113, 2016.

[131] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017.

[132] Q. Le, and T. Mikolov, "Distributed Representations of Sentences and Documents," vol. 32, pp. 1–5, 2014.

[133] S. Chitroub, "Classifier combination and score level fusion: concepts and practical aspects," *Int. J. Image Data Fusion.*, vol. 1, no. 2, pp. 113–135, 2010.

[134] J. Wang, H. T. Shen, J. Song, and J. Ji, “Hashing for Similarity Search : A Survey,” pp. 1–29, 2014.

[135] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, “Supervised Hashing for Image Retrieval via Image Representation Learning,” *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 2156–2162, 2014.

[136] H. B. Kekre and D. Mishra, “Content based image retrieval using weighted hamming distance image hash value,” pp. 284–285, 2011.

[137] K. Bozas and E. Izquierdo, “Large Scale Sketch Based Image Retrieval Using Patch Hashing,” *Springer-Verlag Berlin Heidelb.*, pp. 210–211, 2012.

[138] R. R. Saritha, V. Paul, and P. G. Kumar, “Content based image retrieval using deep learning process,” *Springer Clust. Comput.*, pp. 1–14, 2018.

[139] X. Shi, M. Sapkota, F. Xing, F. Liu, L. Cui, and L. Yang, “Pairwise based Deep Ranking Hashing For Histopathology Image Classification and Retrieval,” *Pattern Recognit.*, pp. 3–32, 2018.

[140] D. Varga and T. Sziranyi, “Fast content-based image retrieval using Convolutional Neural Network and hash function,” *IEEE International Conference on Systems, Man, and Cybernetics - SMC*, pp. 2636–2640, 2016.

[141] J. Wang, S. Kumar, and S.F. Chang, “Semi-Supervised Hashing for Scalable Image Retrieval,” *IEEE*, pp. 3424–3431, 2010.

[142] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, “Bit-Scalable Deep Hashing With Regularized Similarity Learning for Image Retrieval and Person Re-Identification,” *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, 2015.

[143] F. Zhao, Y. Huang, L. Wang, and T. Tan, “Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval,” *IEEE Xplore*, pp. 1556–1564, 2015.

[144] M. Bawa, T. Condie, and P. Ganesan, “LSH Forest : Self-Tuning Indexes for Similarity Search,” *International World Wide Web Conference Committee (IW3C2)*, pp. 1–10, 2005.

[145] S. Minaee, “20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics,” 28-Oct-2019 [Online] *Medium*. Available at: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce (accessed 25 December 2019).

[146] J. Brownlee, “How to Configure the Learning Rate When Training Deep Learning Neural Networks,” 23-Jan-2019 [Online] *Machine Learning Mastery*. Available at: https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks (accessed 25 September 2019).

استرجاع صور المخطوطات العربية باستخدام التعليم العميق

منال محمود علي خياط

المستخلص

توفر المخطوطات العربية القديمة قيمة من المعلومات التاريخية التي تعكس التعليم وثقافة المجتمع وتقاليده خلال فترات زمنية محددة. لذلك ، نظرًا لأهميتها ودورها الأساسي في إثراء المعلومات التاريخية القيمة ، تهدف هذه الدراسة إلى جمع المخطوطات العربية القديمة في مجموعة بيانات. ثم ، تصنيف صورها حتى نتمكن من استرداد أكثر الصور مشابهه لصورة الاستعلام بدقة وبشكل لحظي.

يمكن بناء نظام الاسترجاع وفقًا لمعايير بحثيه مختلفه. استوفينا في هذه الدراسة الاسترجاع حسب ثلاث معايير وهي استرجاع الصور حسب المخطوطه وحسب المؤلف وحسب نوع الخط.

يعد الاستخراج والتصنيف التلقائي وفقًا لأهم السمات المميزة للصور، خطوة حاسمة لاكتشاف أوجه التشابه بين الصور. مع العلم أن المخطوطات العربية التاريخية هي صور نصية. وبالتالي ، من المهم استخراج النص من الصور واسترجاع الصور وفقًا لخصائصها النصيه.

يتم تنفيذ هذه الخطوه من خلال تطوير نموذج التعلم العميق LSTM ثنائي الاتجاه والمُحسن. بعد ذلك ، يتم قياس أوجه التشابه بين النصوص باستخدام ثلاثة مقاييس مختلفة للمسافة.

صور المخطوطات ليست نصية بحتة ، عوضاً عن ذلك فإنها تشمل توقيعات ورسومات وأشكال وجداول وملاحظات جانبية ، إلخ. وبالتالي ، من الضروري الإهتمام بالأجزاء غير النصيه في الصور واستعادة الصور

وفقًا لخصائصها المرئيه. لتحقيق ذلك ، نقلنا التعلم من أربع شبكات عصبية تلافيفية مدربة مسبقًا تسمى : MobileNetV1 و DenseNet201 و ResNet50 و VGG19 كما تم اختبار نموذج التعلم العميق السيامي إلى جانب مقاييس المسافة الثلاثة لقياس أوجه التشابه بين الصور واسترجاعها.

وأخيراً ، تم دمج نماذج التعلم العميق المرئيه والنصيه الأكثر دقه بإستخدام ثلاثة نماذج مختلفة للانصهار تسمى: مستوى القرار ومستوى الميزات ومستوى النتيجة. حسن نموذج الانصهار على مستوى النتيجة من كل نموذج مستخدم بشكل فردي.

كما وتم تطوير واختبار أداء نموذج تعليم معزز لاسترجاع صور المخطوطات العربية. ومن ثم ، تطرح هذه الدراسه نهجًا جديدًا يوصي بتطبيق نموذج انصهار لكلا النماذج المرئيه والنصيه لتصنيف واسترجاع صور المخطوطات العربية بدقه ونجاح.

استرجاع صور المخطوطات العربية باستخدام التعليم العميق

منال محمود علي خياط

الملخص

نظرًا للكمية المتزايدة بشكل كبير من البيانات التي يتم استخدامها واسترجاعها عبر الإنترنت، بالتالي هناك حاجة ماسة لاكتشاف نماذج قويه قادرة على تصنيف الصور واسترجاعها.

تناولت معظم الدراسات السابقة في مجال استرجاع الصور، استرجاع الصور الحديثه والمطبوعة إلكترونياً مع تجاهل الصور القديمة الباهته والمحتويه على نصوص مكتوبه بخط اليد. لذا ركزت هذه الرساله على تحسين جودة صور المخطوطات العربية القديمه بحيث يمكن استخراج النص منها واسترجاعها بالاعتماد على خصائصها النصيه. كما و تم استخراج الخصائص المرئيه من الصور ودمج كلا الخصائص في نموذج واحد قادر على استرجاع الصور بنجاح.

الأهداف الرئيسية لهذا البحث هي:

1. جمع المخطوطات العربية القديمة وتخزينها في مجموعة بيانات. علماً بعدم توفر مكتبات رقميه متضمنه لجميع المخطوطات العربيه القديمه.
2. تحليل الخوارزميات المستخدمة في كل من رؤية الكمبيوتر وفي مجالات البرمجة اللغوية العصبية للوصول إلى الخوارزميات الأكثر ملائمة لحل مشكلة استرجاع الصور.

3. توظيف تكنولوجيا التعلم العميق لأتمتة مرحلة استخراج الميزات.

4. استخراج الميزات البصرية والنصية للصور. ثم دمجها في نموذج انصهار واحد للحصول على أدق استرجاع.

تنقسم فصول الدراسة الى ستة فصول مرتبة كالاتي:

•الفصل الاول يحتوي على مقدمة عن أنظمة استرجاع الصور والطرق المختلفه لبناء هذه الأنظمه.

•الفصل الثاني يعرض مراجعة أدبية لموضوع البحث. يبدء الفصل بمراجعة التصنيفات المختلفه لتقنيات استرجاع الصور. يليه تصنيف التقنيات إلى ثلاثه أقسام رئيسيه تبعاً لدراستنا وهي تقنيه استرجاع الصور بالاعتماد على خصائصها المرئيه و تقنيه استرجاع الصور بالاعتماد على خصائصها النصيه و تقنيه استرجاع الصور بالاعتماد على دمج كلا الخصائص المرئيه والنصيه في نموذج واحد.

•يقدم الفصل الثالث وصفا تفصيلياً لإستراتيجية جمع قاعدة البيانات وتصنيفها وفقاً للمخطوطة والمؤلف ونوع الخط.

• الفصل الرابع يشرح بشكل تفصيلي النموذج المقترح لإسترجاع صور المخطوطات.

•يقدم الفصل الخامس وصفا لمنهجية تقييم النماذج والإعدادات التي تم اختيارها أثناء تدريب النماذج المقترحه.

كما يقدم النتائج للنماذج بناء على عدة تجارب، ثم يعرض المقارنة مع الدراسات الاخرى ذات الصلة.

•الفصل السادس يوضح الخاتمة والعمل المستقبلي بناء على نموذجنا أو العمل المستقبلي في مجال أنظمة استرجاع الصور.

استرجاع صور المخطوطات العربية باستخدام التعليم العميق

منال محمود علي خياط

بحث مقدم لنيل درجة الدكتوراه في العلوم

(علوم الحاسبات)

بإشرا ف

د. لمياء عبدالله الرفاعي

كلية الحاسبات وتقنية المعلومات
جامعة الملك عبد العزيز
جدة – المملكة العربية السعودية

ذو القعده 1441 هـ – جولاي 2020 م