PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE
COMPUTER SCIENCE
DEPARTMENT
N° :.....

Domain: Mathematics and Computer Science
Branch: Computer Science
Specialty: IA

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER IN COMPUTER SCIENCE

By: Lamri Hamza

Laidoune Zakaria

TOPIC

Arabic Handwritten Letters Recognition

The jury composed of:

Dr. Bentrcia Rahima	University of M'sila	Supervisor
Dr. Barkat Abdelbasset	University of M'sila	Reporter
Mr. Boughrara Seddik	University of M'sila	Examiner

Academic Year: 2021 /2022

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE
COMPUTER SCIENCE
DEPARTMENT
N° :.....

1985
جامعة محمد بوضياف - المسيلة
Université Mohamed Boudiaf - M'sila

Domain: Mathematics and Computer Science
Branch: Computer Science
Specialty: IA

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER IN COMPUTER SCIENCE

By: Lamri Hamza

Laidoune Zakaria

TOPIC

Arabic Handwritten Letters Recognition

The jury composed of:

Dr. Bentrcia Rahima	University of M'sila	Supervisor
Dr. Barkat Abdelbasset	University of M'sila	Reporter
Mr. Boughrara Seddik	University of M'sila	Examiner

Academic Year: 2021 /2022

Zakaria Dedication

Thank Allah

My college journey has come to an end

I am grateful to all those who have had merit in my career and helped me, parents, family and friends

I present to you my dissertation

Hamza Dedication

Praise be to Allah, whose good grace is done
My college journey has come to an end after tiring
and hardship…

And here I am concluding my graduation thesis with all vigor and activity.

I am grateful to everyone who has had a role in my career, and helped me, even if it was a little, parents, family, friends and respected professors..

I present to you my dissertation

ACKNOWLEDGEMENTS

Praise be to Allah who enlightened us the path of science and knowledge, helped us to perform this duty, and agreed to accomplish this work.

Our thanks go to my supervisor because she helped us a lot to accomplish this work and wish her success in her future studies because she has great abilities.

Table of contents

CHAPTER 4: EXPERIMENTAL RESULTS

GENERAL CONCLUSION

List of figures

List of tables

List of abbreviations

AHCD: Arabic Handwritten Characters Dataset

CNN: Convolutional Neural Network

PNG: Portable Graphics Format

SVM: Support Vector Machine

ADBase: Arabic Data Base

HACDB: Handwriting Arabic Characters Data Base

RF: Random Forest

Gist: Gastrointestinal stromal tumor

USPS: United States Postal Service

MNIST: United States Postal Service

RFT: Random Forest Tree

KNN: k-nearest neighbors

ML: Machine learning

CPU: central processing unit

GENERAL INTRODUCTION

General introduction

With the technological development that we observe especially in the field of computer science, and the emergence of what is now called the term artificial intelligence and machine learning, there is an immense need for specific recognition systems in order to facilitate our life. A recognition system determined whether an image contains a specific object, feature, or activity. Based on that, various types of recognition and classification problems were described in the literature and applied in many fields such as optical character recognition, 2D code reading, facial recognition, and Shape Recognition.

Problem statement

Recently, many researchers have focused on the recognition of letters, including Arabic handwritten letters. Unfortunately, this last is very challenging compared to Latin and Chinese characters, and most of the solutions suffer from insufficient recognition rates for two reasons: the first reason is that there are no large and different data sets to train the developed models, and the second reason is the ambiguity in Arabic handwriting due to the existence of dots, vowels, and overlapping characters.

Novel contributions

We have developed two recognition models of Arabic handwritten letters. In the first model, we have used a machine-learning algorithm called Random Forest whereas in the second model, we have extracted 16 features to be fed to Random Forest algorithm. The output of both models is the recognized Arabic handwritten letter.

Thesis organization

In this research, we divided the contents into five chapters. In Chapter 1, we introduced the problem statement of this research, which makes the recognition of Arabic handwriting a challenging task, and the proposed solution.

The second chapter described the Arabic Handwritten Characters Dataset (AHCD) where we divided it into 28 groups of images and each group represents one character written in different forms and by different writers.

In the third chapter, we presented many studies related to Arabic handwriting recognition using different approaches such as Convolutional Neural Network (CNN).

Chapter four explains the proposed recognition system, which consists of three important phases: preprocessing, feature extraction, and model training and testing.

In Chapter five, we discussed the obtained experimental results of the two developed models accompanied by accuracy measurement criteria such as precision and recall.

Finally, the last chapter concluded this research and presented the future directions.

CHAPTER 1: ARABIC HANDWRITTEN CHARACTERS DATASET AHCD

1. Arabic Handwritten Characters Dataset AHCD

1.1. Overview

The Arabic language is considered one of the most complex languages in the world, where we find more than 467 million people using this language. Therefore, many writing styles and types appeared, and this is due to the different nature of people, as each person tends to write letters in his own style based on a special type of fonts which makes even the human mind sometimes unable to distinguish between them, especially in ancient writings and manuscripts. Despite all this, the Arabic alphabet is represented by 28 letters, where some of them have the same body shape and differ in the number or existence or location of the dots. For example, (the letter Baa (ب) the letter Taa (ت) and the letter Thaa (ث)) are completely similar in terms of structure, but they differ in the number and location of dots.

1.2. Arabic Handwritten Characters Dataset

In this study, we have used the **Arabic Handwritten Characters Dataset (AHCD),** taken from Kaggle website and collected by El-Sawy et al. [1]. It contains 16800 images of handwritten Arabic letters (13440 train images and 3360 images for test), 600 images for every letter (480 image for the train and 120 for test). These images are in PNG format with a size of 32X32 pixels, and they have a black background and a white character body. We divided the described dataset into 28 classes where each class represents the Arabic letter, and it consists of 480 images of the handwritten letter written by different persons in different styles.

Table 1.1 introduced sample images of Arabic letter. The first column presents the order of the letter in the alphabet, the second column defines the name of the letter in Arabic, and the third column clarifies the name of the letter in English / French. Finally, the last column shows a sample of images where each letter appears clearly.

N°	Name of class	Name of character	Sample images of the character
1	أ	Alif	
2	ب	Baa	
3	ت	Taa	
4	ث	Thaa	
5	ج	Jeem	

6	ح	Haa	
7	خ	Khaa	
8	د	Dall	
9	ذ	Dhaal	
10	ر	Raa	
11	ز	Zaay	
12	س	Seen	
13	ش	Sheen	
14	ص	Saad	
15	ض	Daad	
16	ط	Ttaa	
17	ظ	Dhaa	
18	ع	Ayn	
19	غ	Ghyan	
20	ف	Faa	
21	ق	Qaaf	
22	ك	Kaaf	
23	ل	Laam	
24	م	Meem	
25	ن	Noon	
26	ه	Haa	
27	و	Waw	

28	ي	Yaa	

Table 1.1:A Sample of Arabic handwritten letter images

CHAPTER 2: LITERATURE REVIEW

2. Literature Review

The recognition of Arabic handwriting occupies a large area in machine learning problems. Different studies have been conducted with various accuracy rates. In [2], El-Sawy et al. collected their own Handwritten Arabic Character dataset from 60 participants to form 16,800 characters. They used a Convolutional Neural Network (CNN) model consisting of two layers to classify the letters with an accuracy of 88%. To improve the CNN performance, different optimization techniques have been used to get 93.93%.

Altwaijry and Turaiki [3] created a new Arabic handwritten letters dataset named "Hijja" which consists of about 47,000 characters from 591 participants. They proposed a CNN model to classify letters. The recognition rates were 88% and 97%, using the Hijja and the Handwritten Arabic Character dataset, respectively.

Moreover, Younis in [4] used two datasets to train the proposed CNN model to recognize Arabic handwritten characters. The model contains three convolutional layers and one final fully connected layer. The accuracy rate reaches 93.7% for the first dataset and 93.8% for the second one.

Latif et al. [5] designed a CNN to recognize a mix of handwriting of multiple languages including Arabic. They used as input images with size of (28 × 28) pixels, passed to two convolutional layers which used the max-pooling function. The model achieved an accuracy of around 99% for each language.

Also, Alrobah and Albahl [6] used the Hijja dataset and two models to recognize Arabic handwritten letters. The first model is the CNN model in order to extract the important features from the letter, and the second one is the SVM model for letter classification. They get an accuracy of 96.3%.

Another CNN model called Visual Geometry Group net architecture was developed by Mudhsh et al., [7] for recognizing Arabic handwritten characters and digits. It contains 13 convolutional layers, two max pooling layers, and 3 final fully connected layers. The model was trained and tested using two different datasets: ADBase and HACDB, with an accuracy of 99.66% and 97.32% for ADBase and HACDB, respectively.

Random Forest algorithm (RF) was applied for handwriting recognition. Shamim et al. [8] presented a comparative study for offline digit recognition where SVM with RF classifier achieved highest results. Do and Pham computed GIST features with RF on USPS, MNIST data-sets. They achieved for the RF model an accuracy of 84.16% and 91.15% for SVM model.

Marwa Rashad and Noura A. Semary [9] used in thier investigation two models, the first one is RFT model and the second one is KNN model using feature extraction to create a vector from their images. They archived an accuracy of 87% and 98% for KNN and RFT respectively.

In the next chapter, we illustrate the main phases of developing a recognition system based on the Random Forest algorithm to recognize Arabic handwritten letters taken from the AHCD.

CHAPTER 3: THE PROPOSED RECOGNITION SYSTEM

3. The Proposed Recognition System

3.1. Introduction

In this chapter, we will discuss the main steps that we followed in order to create and train the proposed Random Forest model.

We used python as a programming language, which is one of the most popular scientific application languages. Its high level of interactivity and rich academic library collection make it a good choice for algorithm development and exploratory data analysis. It includes a well-known module called Scikit-Learn-Tool, which integrates many ML algorithms for supervised and unsupervised problems, such as the Random Forest algorithm [10].

3.1.1. Definition of Random Forest Algorithm

A Random Forest or Random Decision Forest is a learning method for classification, regression, and other tasks that work by building various decision trees at training time. For classification tasks, the output of the RF is the class chosen by the majority of the trees. For regression tasks, it returns the mean or prediction mean for each tree. Random decision forests correct a decision tree's habit of over fitting its training set. Random forests generally outperform decision trees, but are less accurate than gradient boosted trees. However, data attributes can affect its performance.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.[11]

3.1.2. Definition of Decision Trees

In order to understand how random forests works it is necessary to become familiar with decision trees. Decision trees are predictive models that use a set of binary rules to calculate a target value. Two types of decision trees are classification trees and regression trees. Classification trees are used to create categorical data sets such as land cover classification and regression trees are used to create continuous data sets such as biomass and percent tree cover [12].

3.1.3. How Random Forests work

Random Forests, like decision trees, can be used to solve classification and regression problems but it is able to overcome the drawbacks associated with single decision trees while maintaining the benefits. The random forests model calculates a response variable (e.g., land cover, percent tree cover) by creating many (usually several hundred) different

decision trees (the forest of trees) and then putting each object to be modeled (in our case the object is a multi-layered pixel) down each of the decision trees. The response is then determined by evaluating the responses from all of the trees. In the case of classification, the class that is predicted most is the class that is assigned for that object [13]. In other words, if 500 trees are grown and 400 of them predict that a particular pixel is forest and 100 predict it is grassing the predicted output for that pixel will be forest. In the case of regression, the resulting value for an object is the mean of all of the predictions. Since predictions from random forests are derived using a forest of trees, it is not possible to easily illustrate how the predictions are made. To illustrate the process, it would be necessary to draw all of the trees for each prediction which would result in hundreds of decision tree diagrams for each model [12].

The algorithm for random forests is:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

The follow figure 3.1 is an example of how random forest classification work

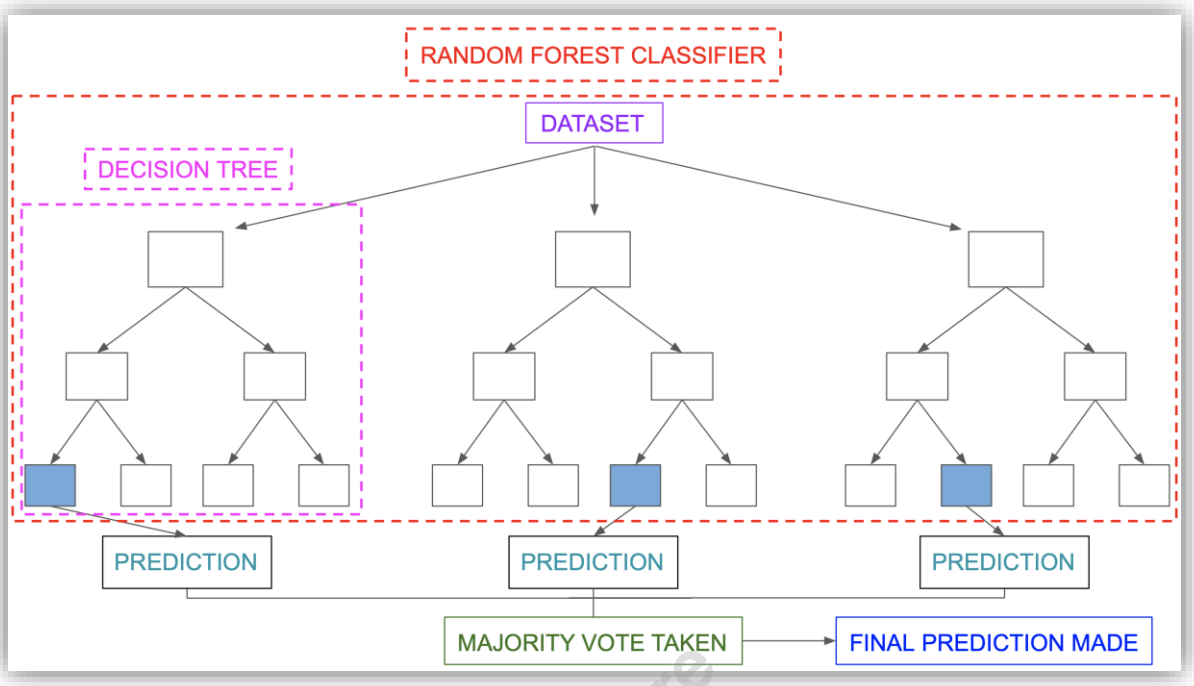


Figure 3.1:how Random Forest work

3.1.4. Advantages and limitations

There are a number of advantages of using random forests. It has been found to be comparable to other machine learning algorithms such as boosting, and support vector machines but with the advantage that random forests are not very sensitive to the parameters used to run it and it is easy to determine which parameters to use [14]. Over fitting is less of an issue than it is with individual decision trees and there is no need for the cumbersome task of pruning the trees. Lastly, the ability of automatically producing accuracy and variable importance and information about outliers makes random forests easier to use effectively. However, there are some limitations when using random forests, especially when using it for regression. Due to the way regression trees are constructed, it is not possible to predict beyond the range of the response values in the training data. For example, if the training data for a biomass model contains low and moderate biomass values but no high values, it will not be possible to accurately predict high biomass values when the model is applied to the full data set. It is extremely important that the training data include samples that cover the entire range of response data values [12].

There are many algorithms in the field of recognizing handwritten Arabic letters. As we mentioned in the previous chapters, we chose the Random Forest algorithm due to its lack of use in this research and its many advantages, which were the most important for us is the ease of creating and training the model, and the speed of implementation.

3.2. **Our Proposed Recognition System**

After data collection, we started our research by preprocessing the images passing through Binarization, thinning, and normalization as clearly presented in Figure 3.2. Two approaches are used in the recognition of the output image. In the first approach, we extracted a set of 16 features from the character image, which is fed to the random forest model. In the second approach, we used the whole image (pixels) as our features, which are passed to the random forest model.

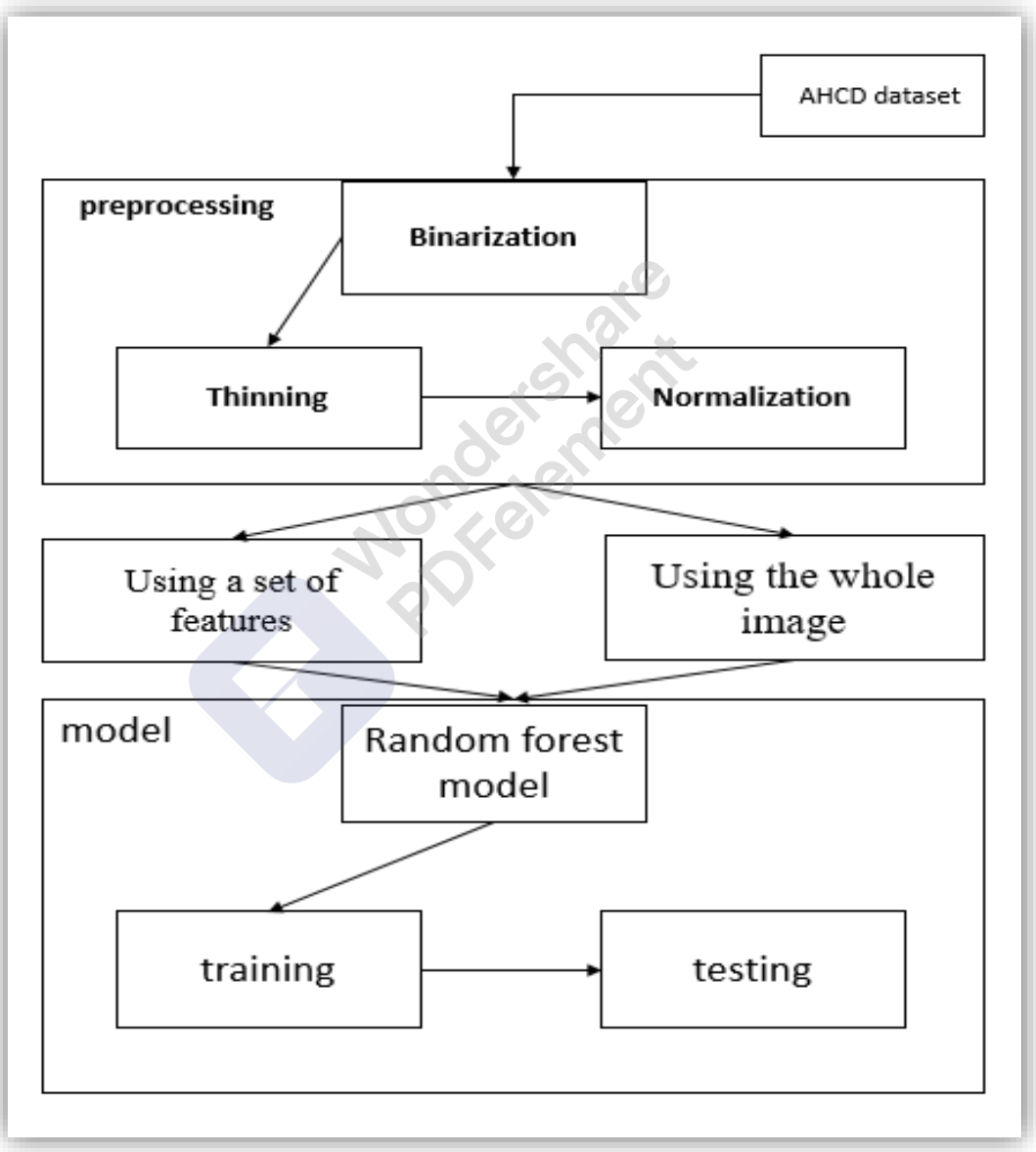


Figure 3.2:The proposed Random Forest recognition model

3.2.1. Preprocessing

First, we have collected the images in one file. Next, we created a function that divides them into 28 folders with different names so that each folder contains a group of images belongs to one letter in different handwriting styles. The files are renamed in ascending numbers from 1 to 480 by using the glop function to move into a folder and find files, and the rename function to change their names, as shown in Figure 3.3.

```
Procedure rename () :
Int i=1 ;
For each filename in glop.glop (‘. /*.png’) do
# glop is a function used to look in a folder for all files with a specific name
        Os.rename (filename,’. /’i’.png’) ;
-#Rename is a function on OS Python library used to rename a file with a new given name
        I=i+1 ;
End for
End
```

Figure 3.3:Pseudocode of glop glop and Rename functions

3.2.1.1. Binarization

Binarization is an important step in any image processing, which is the method of converting any grayscale image into black and white image by finding the threshold value of gray scale and checking whether a pixel is having a particular gray value or not [15].

There is a lot of work in the field of binarization to obtain the ability to extract clear black and white images only from normal color images and preserve the content [16, 17].

In our work, we rely on a function found on GitHub [18] that is known for its open-source algorithms. After modifying it to our own needs, we have a function that allows us to do the work we need from site [19], Figure 3.4 illustrates the Binarization step applied to the letter Alif (ا).

Figure 3.4: The Arabic handwritten letter Alif before Binarization (left) and after (right)

3.2.1.2. Thinning

Thinning is an operation used to remove selected foreground pixels from a binary image, thinning is usually only applied to one binary image, producing another binary image as output. Refinement operations are related to hit and miss transformation [20, 21].

The second model, which used feature extraction in it, we use thinning of image at the data set but in the other hand the first model we did not use it because of the size of data set, which is so small to make thinning on it. This affected the final result by reducing the recognition rate of the letter Alif. The pseudocode is shown in Figure 3.5.

```
Procedure thin() :
Int i=1 ;
For each filename in glop.glop ('./*.png') do
# glop is a function used to look in a folder for all files with a specific name
        Img=io.imread(filename) ;
#Imread is a function from io library in Pithon used to read an image from a path
        Image= thin(img) ;
#Thin is a function from Thin library in Python used to decrease body thickness to one pixel
        Img1=img_as_ubyte(image) ;
#Img.as.ubyte is a function converts images to unsigned byte format,with values between
[0-255]
        Os.imsave('./'i'.png' ,img1) ;
# Imsave is a function from OS library in Python used to save images in a path
        I=i+1 ;
End for
End
```

Figure 3.5: Pseudocode of thin function

Figure 3.6 depicts the effects of the thinning function on the Arabic letter Alif.

Figure 3.6: The Arabic handwritten letter Alif before thinning (left) and after (right)

3.2.1.3. Normalization

Image normalization is a typical process in image processing to change the range of pixel intensity values. Its job is to convert the input image into a series of sensory familiar or normal pixel values.

We used Normalization in specific role on our work, we applique the normalization of image using astype function which is a function from numpy library to make the list of input model in the same type which in this case float32 as we see in Figure 3.7

```
#Normalizing image

X_train =x_train .astype('float32') ;

X_test =x_test .astype('float32') ;

X_train= X_train/255 ;

X_ test = X_ test /255 ;
```

Figure 3.7: Pseudocode of normalization

3.2.2. Features Extraction

This phase is considered very important in character recognition systems. It helps to recognize a character based on specific features. In our first proposed approach, we extracted a set of 16 features from each character image, and we implemented different functions for this task.

The first function is Find Letter Contour Area, which detects the area of the letter in the image to make it easy for other functions to detect the letter. It returns three outputs as described in Figure 3.8.

```
Float* Function findLetterCountourArea (img) :
[Contours, _]=cv2.findContours (img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE) ;
# cv2.findContours is a function retrieves contours from the binary image there are three
arguments in cv2.findContours () function, first one is source image, second is contour
retrieval mode, third is contour approximation method.
# cv2.CHAIN_APPROX_NONE = show all the points on contour.
If length (contours) ==0 then
        Return [-1,-1,-1] ;
End if
Framearea= cv2.countourArea (contours [0]) ;
# cv2. CountourArea in a function for calculi countours area of figure in image
MaxArea=0 ;
index=0 ;
i=1 ;
for i until length (contours) do
        if (cv2.countourArea (countour[i])>maxArea) then
                MaxArea = cv2.countourArea (countour[i]) ;
                index=i ;
        end if
end for
return [MaxArea, countour [index], framearea] ;
end
```

Figure 3.8: Pseudocode of Find Letter Contour Area function

The second function is Count_connected_parts which detected the number of the connected parts in the letter, that is it counts the number of adjacent pixels of the same color (white). It is also used by other functions.

Moreover, we defined another function to extract holes from the letter. It takes as inputs the image and the output of the function Count_connected_parts, and it uses Find Letter Contour Area to detect the number of holes in the letter, as we see in Figure 3.9.

```
Float function count_holes (img) :

[Contours, _]=cv2.findContour (img, RETR_LIST, cv2.CHAIN_APROX_SIMPLE) ;

# cv2.findContours is a function retrieves contours from the binary image there are three
arguments in cv2.findContours () function, first one is source image, second is contour
retrieval mode, third is contour approximation method.

# cv2.CHAIN_APPROX_NONE = show all the points on contour.

return abs (length(Contours)-num_connexcted_parts) ;

end
```

Figure 3.9: Pseudocode of count-holes function

The next function is the height to width ratio, which we can use to compare between letters with different width and height. As an example, the letter Alif has a height greater than the width and whereas the letter Baa has a width greater than the height. This function also used the previous two functions, namely Find Letter Contour Area function and Count Connected Parts function, as shown in Figure 3.10.

```
Float function height _width_ration (img) :

[area, letter_contour, framearea] =findLetterContourArea(img) ;

Whiteatea =framearea -area ;

If area == -1 then

        Return 0 ;

End if

[X, y, w, h} = cv2.boundingrect (letter_contour) ;

# cv2 boundingrect() is a function used to create an approximate rectangle along with the
image

Return h/w ;

End
```

Figure 3.10: Pseudocode of height -width-ratio function

Another function is also developed to determine the percentage of white to black pixels in the image. We can see both of them in figure 3.11.

```
Int function white_black (img) :
If (lenght (img [img ==0]) or lenght (img [img ==1] == 0)) then
        Return 1 ;
End if
Return lenght (img [img ==1] / lenght (img [img ==0]) ;
end
**********
Int fuction white_number (img) :
Return lenght (img [img == 1]) ;
end
```

Figure 3.11: Pseudocode of white-black pixels ratio

Finally, we created a function that collects all the previous features in one 16-elements vector, as illustrated in Figure 3.12.

```
return [num_parts, holes, h_W, vertical_trans, horizontal_trans, ratio0, ratio1, ratio2, ratio3,
ratio4, ratio5, ratio6, ratio7, ratio8, ratio9, ratio10]
```

Figure 3.12: A Sample of extracted features vector

As an example, the set of features extracted from the letter Alif image appears in Figure 3.13 is as follows:

[6, 5, 2.8333333333333335, 6, 2, 1, 1, 1, 1, 0.0, 5.0, 1.0, 10.0, 2.0, 10.0, 2.0]

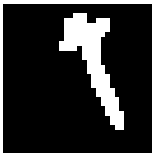


Figure 3.13: The Arabic handwritten letter Alif

We applied the whole steps of the preprocessing phase to each image found in our data set. So, the final result is a matrix of 480*16 elements for each letter.

However, in the second proposed approach, we considered every pixel in the image as a distinctive feature that can help in recognizing the letter.

The collected features are then passed to the recognition system namely Random Forest as explained in the next sections.

3.2.3. Recognition

3.2.3.1. Design

In the introduction to this chapter, we talked about how random forests are a collection of decision trees. In this item, we explain the structure of decision trees more clearly, because we will mention the components that make up them more fully.

A decision tree consists of three components: decision nodes, leaf nodes, and a root node. A decision tree algorithm divides a training dataset into branches, which further segregate into other branches. This sequence continues until a leaf node is attained. The leaf node cannot be segregated further.

The nodes in the decision tree represent attributes that are used for predicting the outcome. Decision nodes provide a link to the leaves. The following diagram shows the three types of nodes in a decision tree [22].

As we see in the following figure 3.14, an example of how decision trees work.

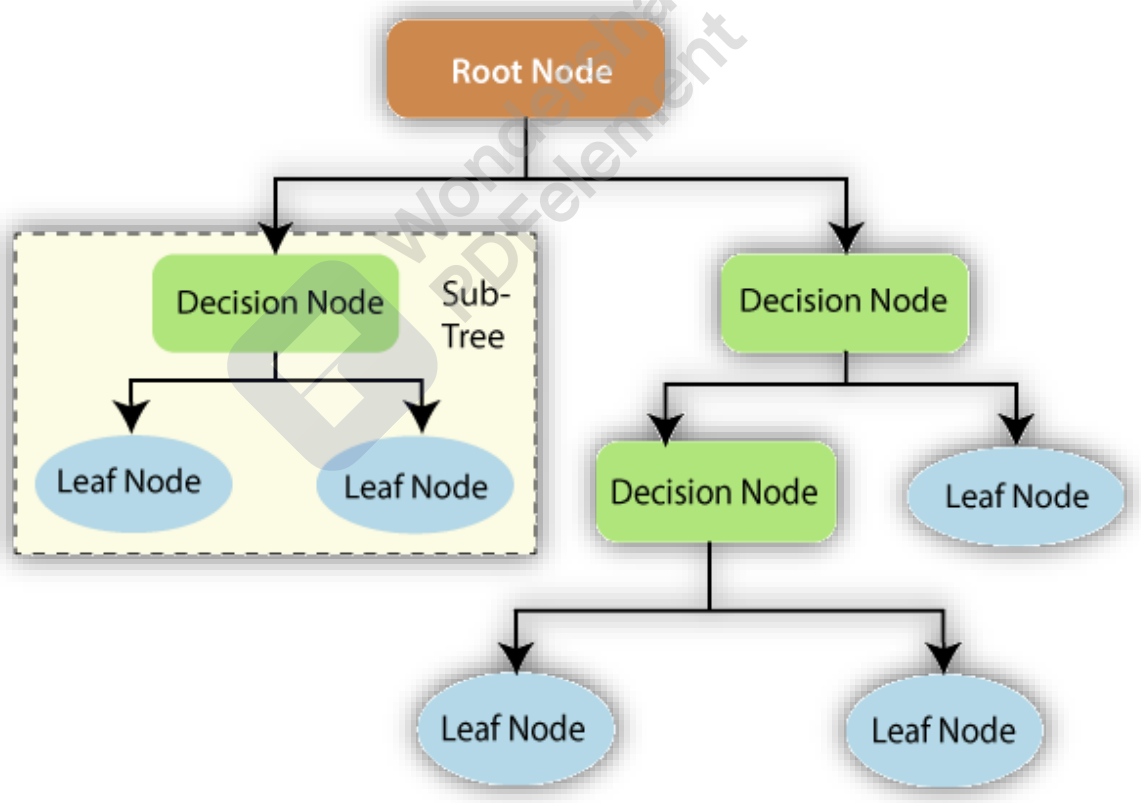


Figure 3.14: Example of how Decision trees work

In the recognition phase, we have used the Random Forest model to be trained with the set of features extracted in the previous phase. We selected the random state equals 2 and 90% of the images to train the model. In addition, we gave to random forest model as parameter 200 estimators (epoch) and max feature 112, true verbose, and we have used two CPUs for training and testing.

The general Random Forest model appears clearly in Figure 3.15.

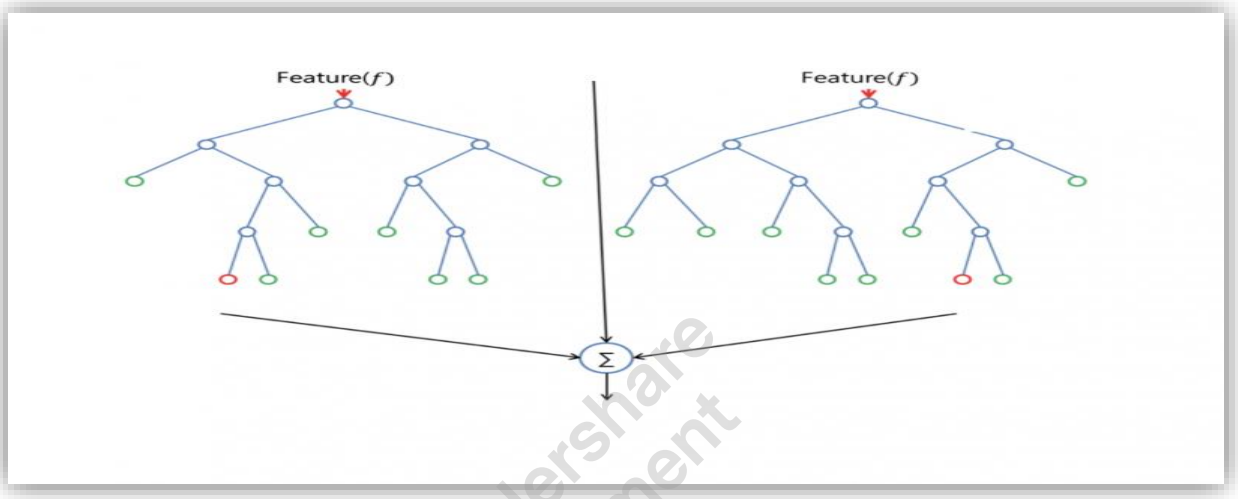


Figure 3.15: The Random Forest model

In Figure 3.16, we introduced the two different approaches which we implemented using the RF Recognition model. The input of the model is the image of the Arabic handwritten letter, whereas the output is the name and image of the recognized letter.

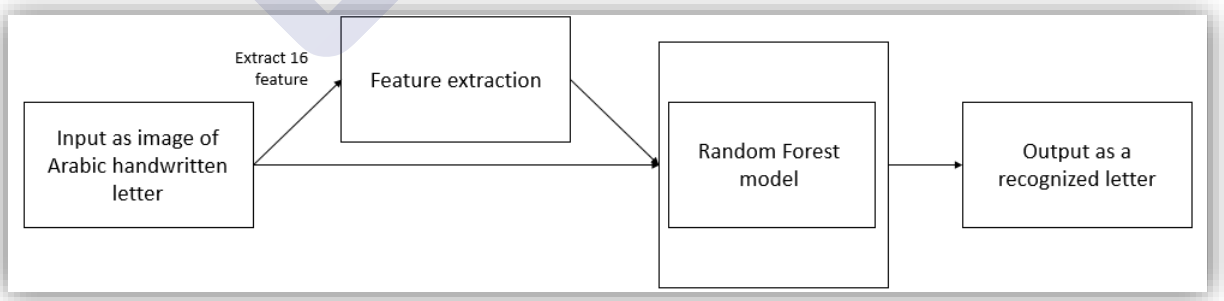


Figure 3.16: The design of the two proposed approaches

3.2.3.2. Training

The training step of Random Forest in both approaches is shown in Figure 3.17. The purpose of model training is to create a Random Forest model that can later on distinguish between Arabic handwritten letters. As we mentioned before, in the first approach, the Random Forest was fed with a vector of 16 features extracted from all images. In the second approach, the model was fed with a vector of 1024 features which are the whole pixels of the image (32*32 pixel).

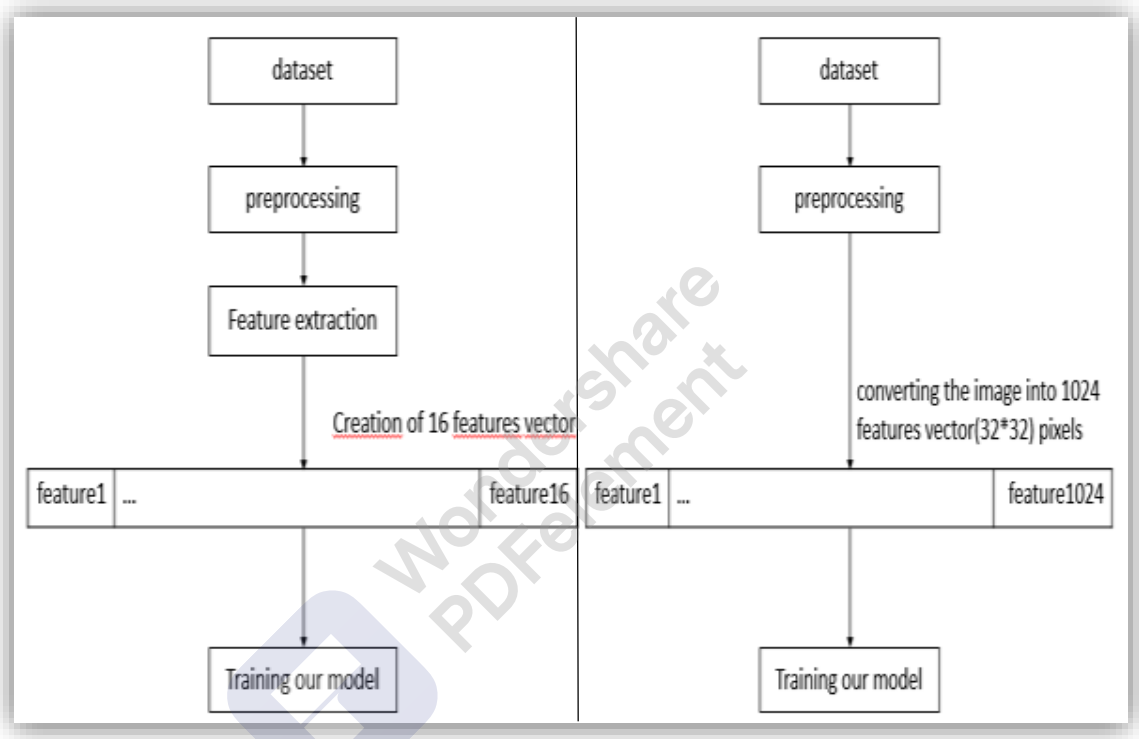


Figure 3.17 :Training RF model with 16 features and Training RF model with all image pixels

3.2.3.3. Testing

After building and training the RF model, we proceed to the last step in our proposed system, which is the testing phase. In the first approach, the obtained accuracy is 40%, and it is lower than that of the second approach, 73%. This is basically related to the extracted features, which are not very useful and efficient in letters recognition. Therefore, we decided to apply the second approach, so we tested the RF model by using 3360 images as presented in Figure 3.18.

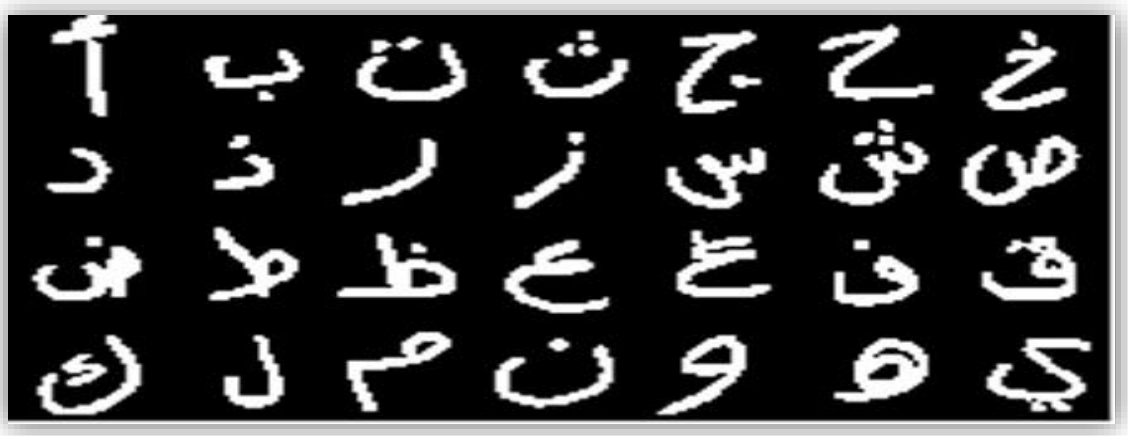


Figure 3.18: Example of testing images

In addition, we create an interface by using the library tkinter (Figure 3.19) and we add a button to browse the testing images one by one as we see in Figure 3.20. The output is the recognized letter which appears in the middle of the interface and the testing image in the lower right, as depicted in Figure 3.21.

```
Procédure Tkinter_UI() :
arabic_letter_window_test = Tk() ;
# Tk().function It helps to display the root window and manages all the other components
of the tkinter application
arabic_letter_window_test.title("arabic letter window test Recognition") ;
arabic_letter_window_test.geometry('550x550') ;
# geometry is a function to define the hiegh and width of tkinter window
procedure browsefunc():
        browsefunc.file_name = filedialog.askopenfilename() ;
# filedialog.askopenfilename is a function to give access to button to open file
          image = Image.open(Path(browsefunc.file_name)) ;
          Image=image_detection1.sliding_window (image) ;
          prediction_num=loaded_rf.predict(image) ;
# loaded_rf.predict is a function for make predict using model
          prediction_letter=num_to_char(prediction_num) ;
          write(prediction_letter) ;
          letter = prediction_letter ;
          path_label.config(text=letter, font=("Courier", 100)) ;
          img = Image.open(Path(browsefunc.file_name)) ;
          img = img.resize((150, 150), Image.ANTIALIAS) ;
          img = ImageTk.PhotoImage(img) ;
          img_label.config(image=img) ;
          img_label.image = img ;
fin procedure
browse_button = Button(arabic_letter_window_test, text="test
letter",command=browsefunc) ;
browse_button.grid(column=2, row=3) ;
browse_button.config(font=("Courier", 25)) ;
path_label = Label(arabic_letter_window_test) ;
path_label.grid(column=5, row=5) ;
img_label = Label(arabic_letter_window_test) ;
img_label.grid(column=20, row=20) ;
arabic_letter_window_test.rowconfigure(5, weight=1) ;
arabic_letter_window_test.columnconfigure(5, weight=1) ;
arabic_letter_window_test.mainloop() ;
end
```

Figure 3.19: tkinter interface function

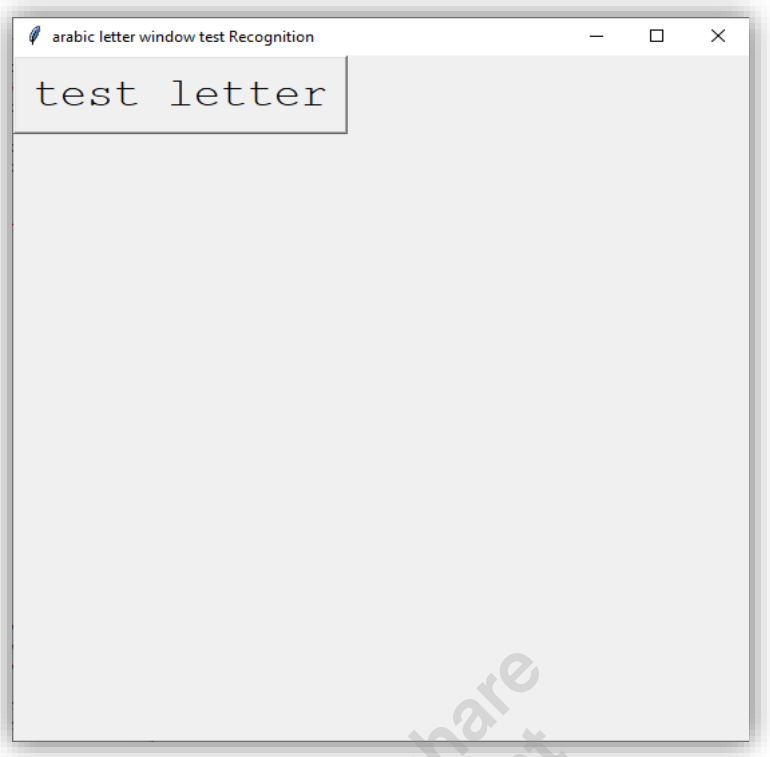


Figure 3.20: tkinter interface

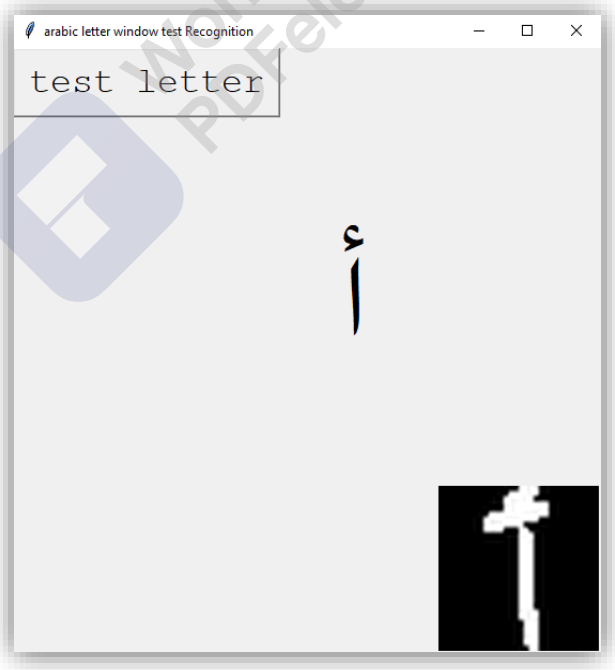


Figure 3.21: The recognized letter displayed on tkinter interface

In the Figures 3.22 and 3.23 examples of well-recognized letters whereas the Figure 3.24 clarifies an example of unrecognized letter.

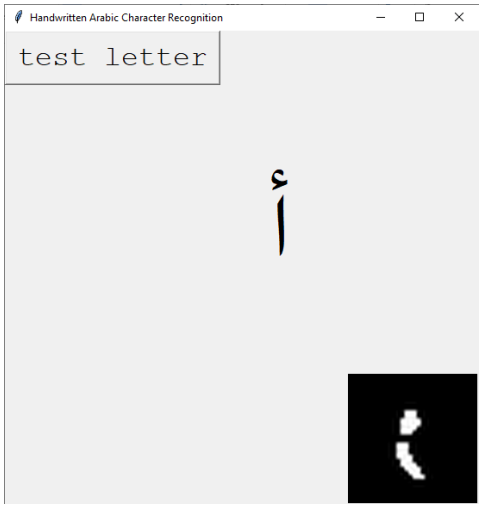


Figure 3.22: Example of well recognized letter 'أ'

Figure 3.23: Example of unrecognized letter' ث'

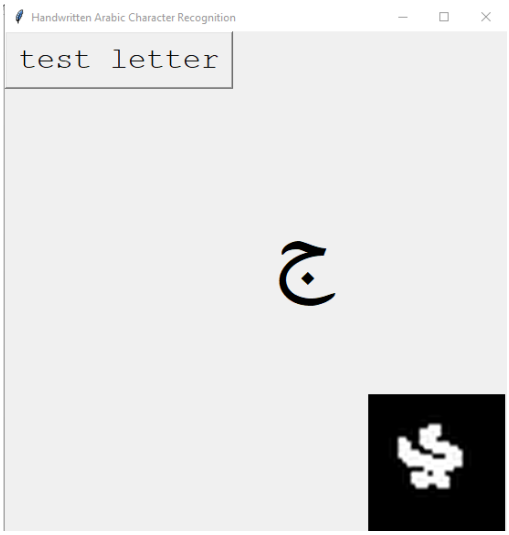


Figure 3.24: Example of unrecognized letter 'ج'

CHAPTER 4: EXPERIMENTAL RESULTS

4. Experimental Results

4.1. Introduction

Model testing is considered the most important phase in any system development process.

After training the proposed RF model, we passed to the testing phase, where we take under consideration both of the training and testing accuracies. Training accuracy means that identical images are used both for training and testing, while test accuracy represents that the trained model identifies independent images that were not used in training.

4.2. Training and Testing Results

First, we start with the accuracy results obtained from the RF model training where we used a collection of 13440 images, 480 images for each letter. Table 4.1 presents the name of the input letter (column 3), the number and the name of the output class which is the recognized letter (column 1 and 3) respectively, and the training percentage.

N°	Name of class	Name of character	Training by letter
1	أ	Alif	98.96%
2	ب	Baa	98.96%
3	ت	Taa	97.29%
4	ث	Thaa	93.79%
5	ج	Jeem	96.67%
6	ح	Haa	97.5%
7	خ	Khaa	96.25%
8	د	Dall	99.17%
9	ذ	Dhaal	95.83%
10	ر	Raa	98.96%
11	ز	Zaay	96.67%
12	س	Seen	97.08%
13	ش	Sheen	97.71%
14	ص	Saad	97.92%
15	ض	Daad	93.38%
16	ط	Ttaa	96.87%
17	ظ	Dhaa	95.83%
18	ع	Ayn	96.04%
19	غ	Ghyan	96.25%

20	ف	Faa	95.83%
21	ق	Qaaf	96.25%
22	ك	Kaaf	98.16%
23	ل	Laam	98.96%
24	م	Meem	98.96%
25	ن	Noon	97.29%
26	ه	Haa	98.13%
27	و	Waw	98.13%
28	ي	Yaa	93.38%

Table 4.1: The training accuracy of each letter

Finally, we tested our model using a different dataset consists of 3360 images, 120 images for every letter. The obtained results are introduced in Table 4.2.

N°	Name of class	Name of character	Testing by letter
1	أ	Alif	96.67%
2	ب	Baa	90.08%
3	ت	Taa	65.83%
4	ث	Thaa	53.16%
5	ج	Jeem	80.00%
6	ح	Haa	80.83%
7	خ	Khaa	60.83%
8	د	Dall	81.66%
9	ذ	Dhaal	66.66%
10	ر	Raa	86.67%
11	ز	Zaay	65.00%
12	س	Seen	83.17%
13	ش	Sheen	71.67%
14	ص	Saad	76.67%
15	ض	Daad	53.33%
16	ط	Ttaa	79.16%
17	ظ	Dhaa	66.66%
18	ع	Ayn	65.83%
19	غ	Ghyan	68.33%

20	ف	Faa	65.83%
21	ق	Qaaf	61.67%
22	ك	Kaaf	75.00%
23	ل	Laam	92.50%
24	م	Meem	83.17%
25	ن	Noon	70.00%
26	ه	Haa	80.00%
27	و	Waw	80.83%
28	ي	Yaa	51.67%

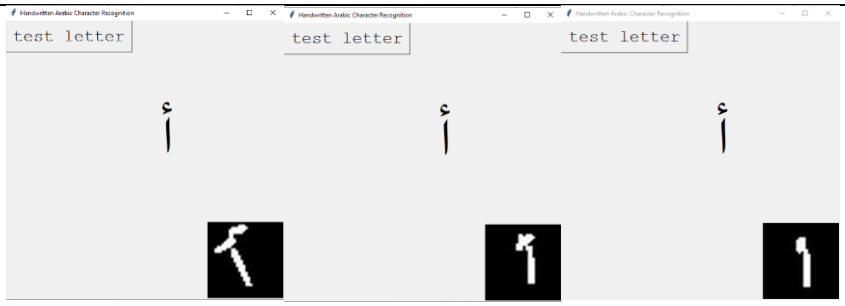
Table 4.2: The testing accuracy of each letter

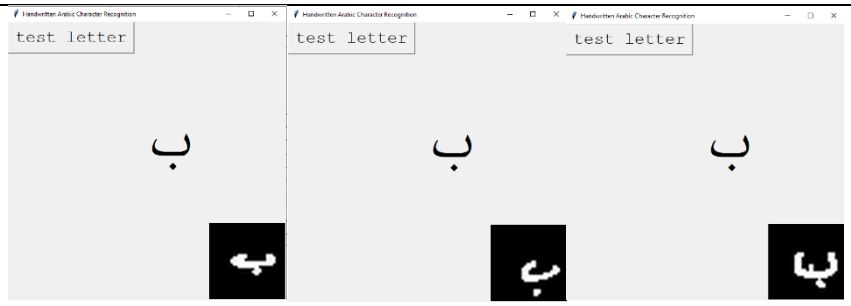
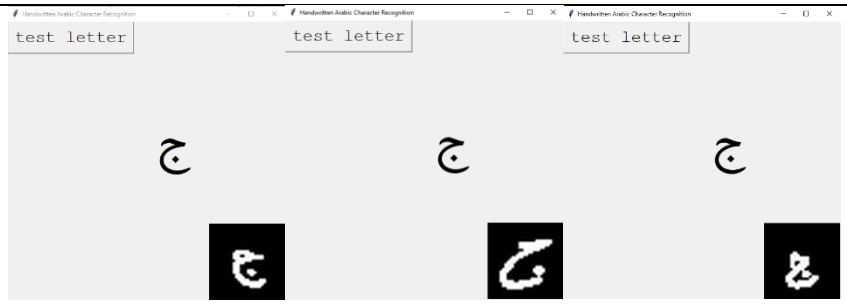
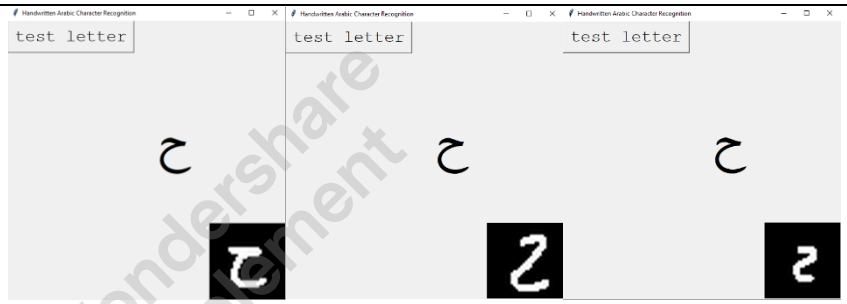
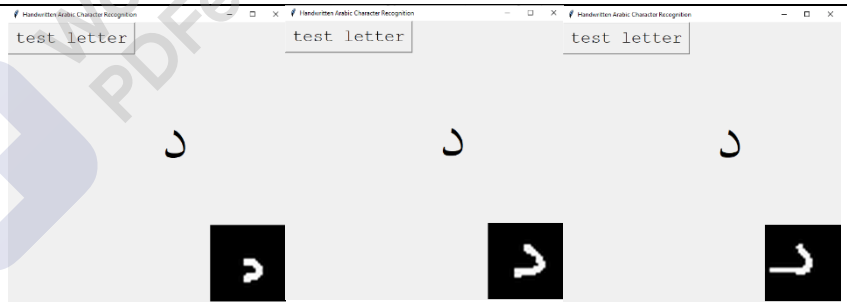
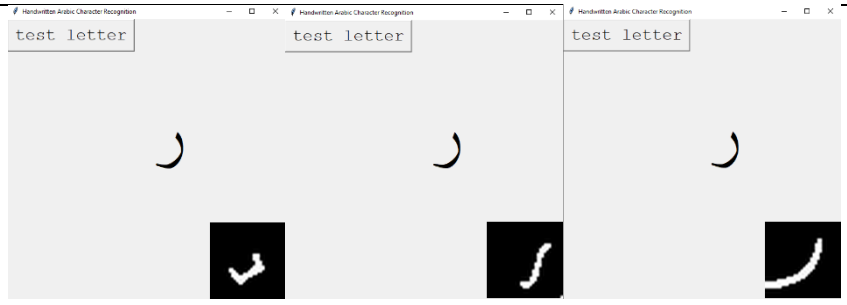
We conclude that the average training accuracy of the proposed Random Forest model is 97.14% where it reaches 73.48% in the testing phase using AHCD dataset.

In Table 4.2, there is a clear difference in the percentages of recognized characters, we find that 11 characters have more than 80% as an accuracy rate, compared to 17 characters that have been recognized by less than 80%. Other letters reach an accuracy less than 52%, such as the letter (ي). This is because of the structure similarity between the letters, which are represented in (ب,ت,ث) ,(ج,ح,خ) ,(د,ذ) ,(ر,ز) ,(س,ش) ,(ص,ض) ,(ط,ظ) ,(ع,غ).

The similarities between these characters are reflected in the character structure, the differences lie in the number of points per character, the position of the special points and how the multiple points are separated or they are connected, which reduces the character recognition rate.

For the characters with recognition rates over 80%, we created Table 4.3 to model characters that got a correct recognition.

N°	Name of class	Examples of recognized letters
1	أ	

2	ب	
3	ج	
4	ح	
5	د	
6	ر	

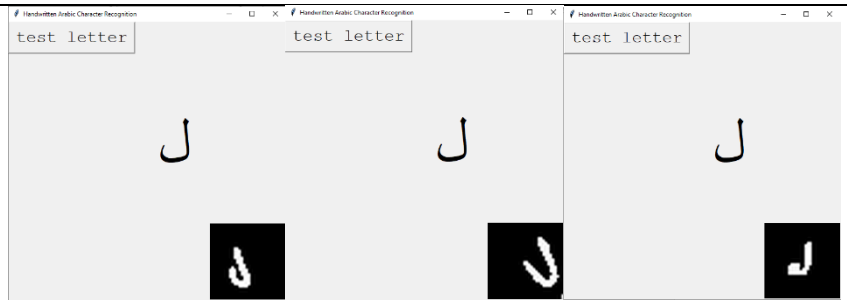
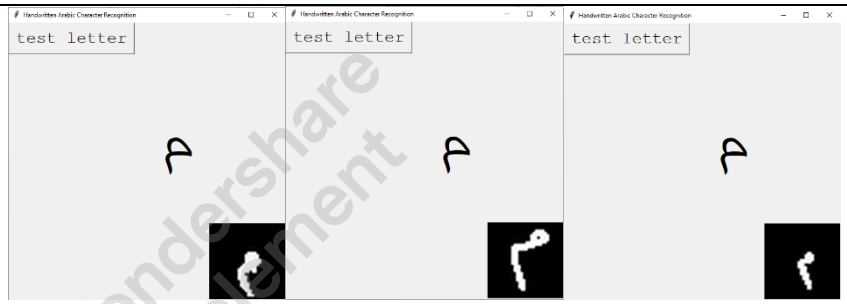
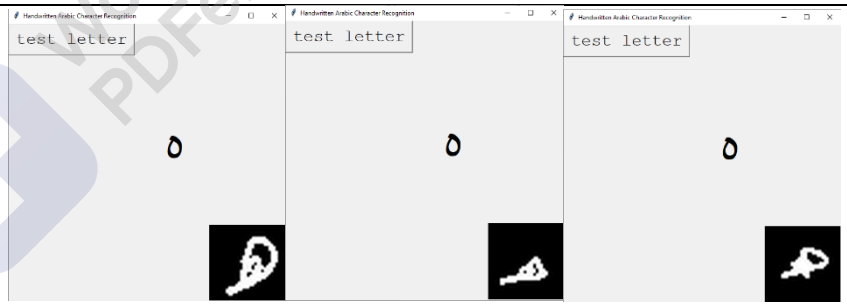
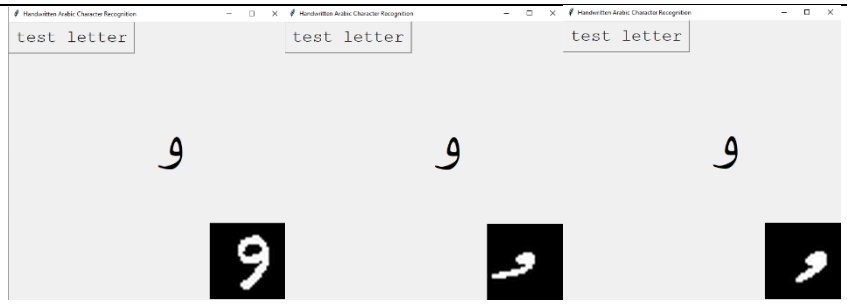
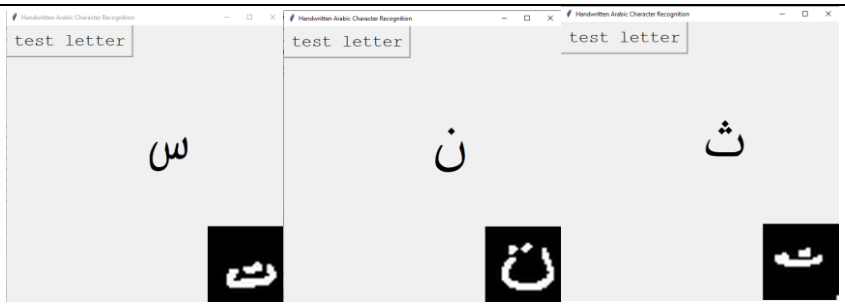
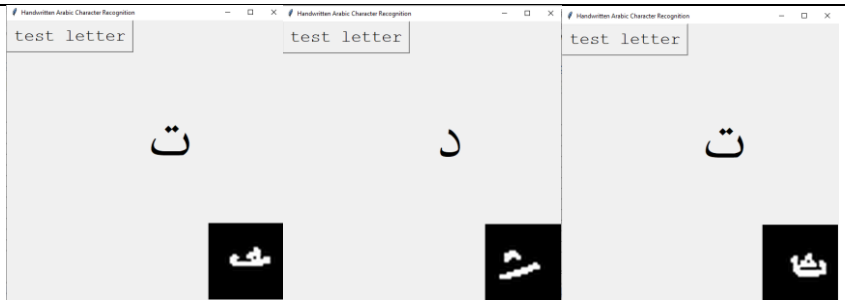
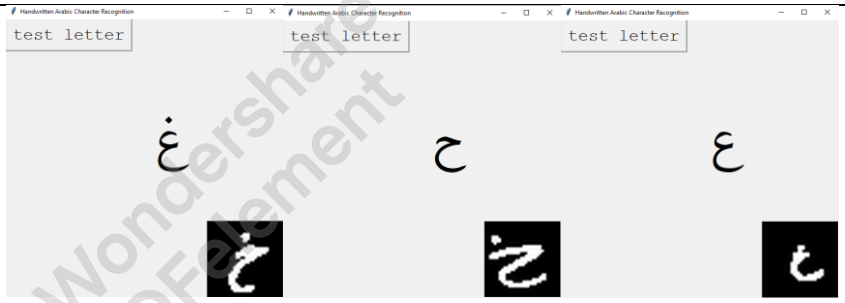
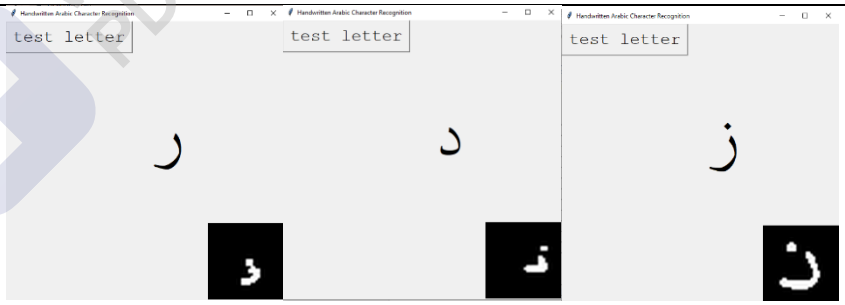
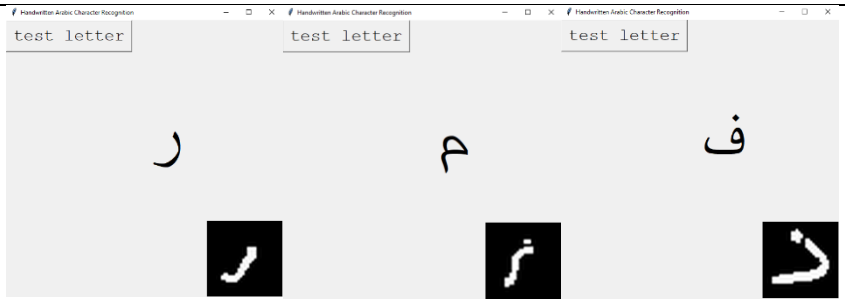
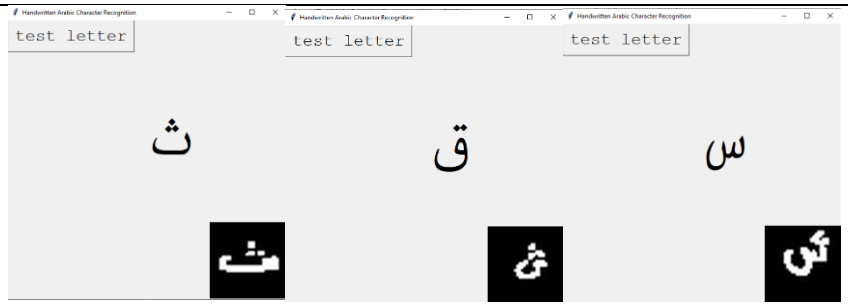
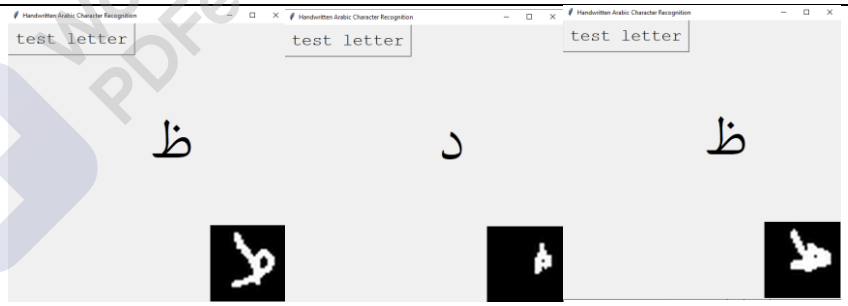
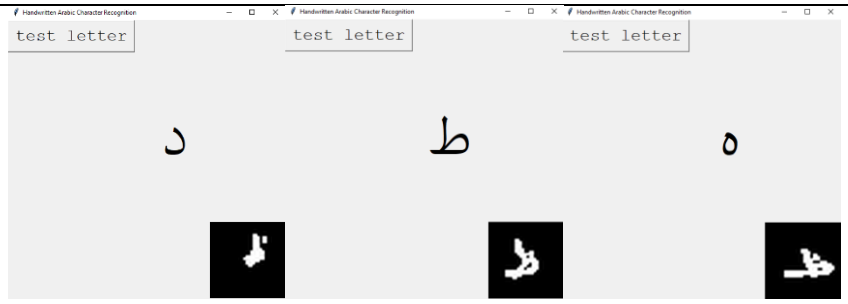
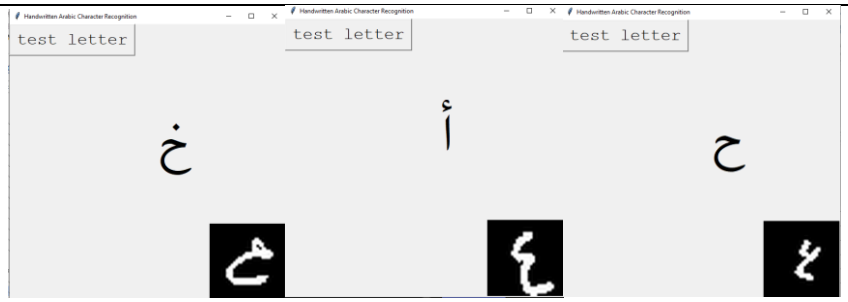
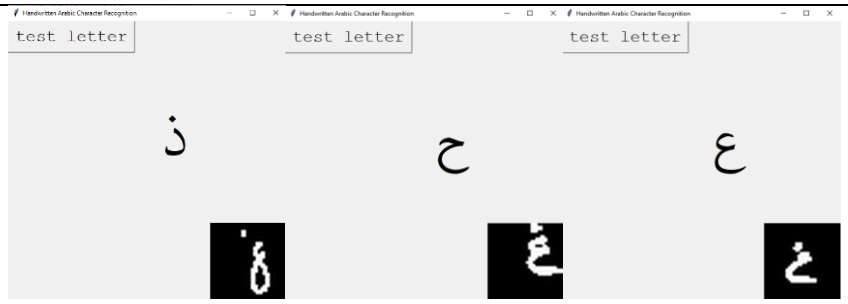
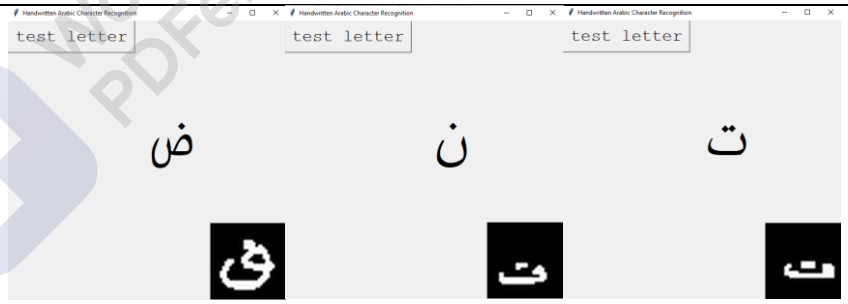
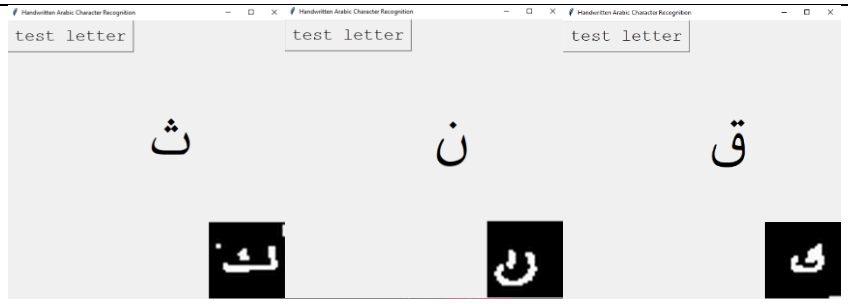
7	س	
8	ل	
9	م	
10	ه	
11	و	

Table 4.3: Examples of letters with recognition rate more than 80%

In Table 4.4, we show some samples unrecognized of letters that have recognition rates of less than 80%.

N°	Name of class	Examples of unrecognized letters
1	ت	
2	ث	
3	خ	
4	ذ	
5	ز	

6	ش	
7	ص	
8	ض	
9	ط	
10	ظ	

11	ع	
12	غ	
13	ف	
14	ق	
15	ك	

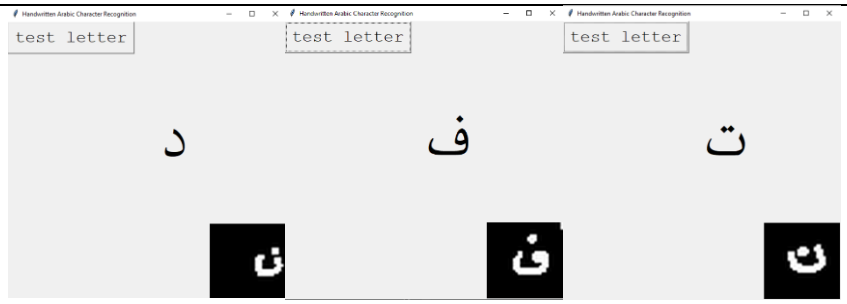
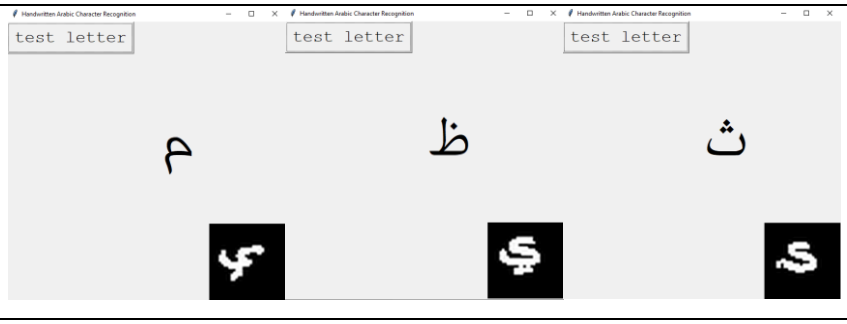
16	ن	
17	ي	

Table 4.4: Examples of letters with recognition rate less than 80%

GENERAL CONCLUSION

5. General Conclusion

The main objective of our work is to exploit the effectiveness of artificial intelligence methods to recognize handwritten Arabic letters. In this work, we use the AHCD dataset to train and test random forest models. We rely on two different methods to identify letters. In the first method, the model uses the whole image as input, while in the second method; the model uses a set of 16 feature from image as input. We use Python as the programming language. We achieved a training accuracy of 97.14% and a testing accuracy of 73.48%.

In chapter 1, we give a general introduction of the topic and an overview of own work.

In the chapter 2, we have discussed the dataset AHCD, from where we got the dataset and its properties, and how we applied the preprocessing phases. We introduced how we split AHCD into 28 files with the same number of images.

In the third chapter, we have done some research on Arabic handwriting recognition systems, and we find a lot of them used CNN system recognition, as it is the most popular recognition system, where some used RF model.

In Chapter 3, we explain our proposed recognition system, which we divide into 3 important steps: preprocessing then feature extraction, and finally modeling, training and testing. We mentioned in the first step the functions we used to implement dataset preprocessing (binarization, thinning, and normalization). On the other hand, in the second step, we assemble the function to extract features from the image and how it works. The final step determines the design of the model and how we train both models, test them and show figures of the testing letter.

In the last chapter, Chapter 4, we discussed the experimental results for each letter as well as our testing and training rates and the overall results of our model. We achieved 97.14% as a training accuracy and 73.48% as a testing accuracy.

5.1. Discussion and future direction

In this part, we talk about our future concerns such as:

- Increase our testing rate to more than 90%.
- Add word segmentation phase to the proposed recognition system.
- Add handwritten digits and different types of writing as a dataset.
- Create Android application and integrate this model in it.
- Use more than one dataset and create our own dataset.

6. REFERENCES

[1] kaggle: A. El-Sawy, M. Loey, and H. EL-Bakry, "Arabic handwritten character's recognition using convolutional neural network," WSEAS Transactions on Computer Research, Volume. 5, pp. 11–19, 2017. https://www.kaggle.com/datasets/mloey1/ahcd1

[2] A. El-Sawy, M. Loey, and H. EL-Bakry, "Arabic handwritten character's recognition using convolutional neural network," WSEAS Transactions on Computer Research, Volume. 5, pp. 11–19, 2017

[3] N. Altwaijry and I. Al-Turaiki, "Arabic handwriting recognition system using convolutional neural network," Neural Computing Applications, Volume. 33, 2020.

[4] K. Younis, "Arabic handwritten characters recognition based on deep convolutional neural networks," Jordan Journal Computers and Information Technology (JJCIT), Volume. 3, 2018

[5] G. Latif, J. Alghazo, L. Alzubaidi, M. M. Naseer, and Y. Alghazo, "Deep convolutional neural network for recog-nition of unified multi-language handwritten numerals," in Proceedings of the 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR), pp. 90–95, London, UK, March 2018.

[6] N. Alrobah and S. Albahli, "A hybrid deep model for rec-ognizing Arabic handwritten characters," IEEE Acces, Volume. 9, pp. 87058–87069, 2021.

[7] M. A. Mudhsh and R. Almodfer, "Arabic handwritten al-phanumeric character recognition using very deep neural network," MDPI, Information, Volume. 8, 2017.

[8] S. M. Shamim, M. B. A. Miah, A. S., M. Rana, A. Al Jobair "Handwritten Digit Recognition Using Machine Learning Algorithms", Global Journal of Computer Science and Technology: D Neural & Artificial Intelligence, Volume 18, pp. 17-23,2018.

[9] M. Rashad and N. A. Semary, "Isolated Printed Arabic Character Recognition Using KNN and Random Forest Tree Classifiers", Advanced Machine Learning Technologies and Applications, Volume 488, pp. 11-17 ,2013.

[10] Steven J. Rigatti, MD, DBIM, DABFM, J Insur Med "Random Forest", journal of insurance medicine, Volume 47, pp. 31–39, 2017.

[11] Random Forest Algorithm, https://www.javatpoint.com/machine-learning-random-forest-algorithm

[12] N. Horning, "Random Forests: An algorithm for image classification and generation of continuous fields data sets", American Museum of Natural History

Center for Biodiversity and Conservation, Central Park West at 79 Street New York, NY 10024 USA, 2010.

[13] Breiman, L. and Cutler, A., "Random Forests". Available at:http://www.stat.berkeley.edu/~breiman/RandomForests/ [Accessed October 12, 2010]

[14] L.Breiman , "Random forests". Machine learning, Volume 45, pp.5–32, 2001.

[15] S. Saha, S.Basu and M. Nasipuri, "Automatic Localization and Recognition of License Plate Characters for Indian Vehicles," in International Journal of Computer Science and Emerging Technologies, Volume 2, pp. 520-533, Aug. 2011.

[16] M. A.Dhali, J. Willem de Wit, L. Schomaker , "BiNet: Degraded-Manuscript Binarization in Diverse Document Textures and Layouts using Deep Encoder-Decoder Networks", BiNet, Department of Artificial Intelligence, Bernoulli Institute University of Groningen The Netherlands, November 20, 2019.

[17] Kumar, Deepak, A. Prasad, M. N., Ramakrishnan, A. G., "Evaluation of document binarization using eigen value decomposition", Document Recognition and Retrieval XX. Proceedings of the SPIE, Volume 8658, pp. 12 ,2013.

[18] github, https://github.com/,

[19] github, https://github.com/pedrofrodenas/image-thresholding-OCR

[20] T. Matsumoto, L.O. Chua, T. Yokohama, "Image thinning with a cellular neural network", IEEE Transactions on Circuits and Systems, Volume 37, pp. 638 – 640, may 1990.

[21] L. Lam, S.W. Lee, C.Y. Suen, "Thinning Methodologies-A Comprehensive Survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 14, pp. 869-885, September 1992.

[22] Introduction to Random Forest in Machine Learning, https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

ملخص

الهدف الرئيسي لعملنا هو استغلال فعالية أساليب الذكاء الاصطناعي للتعرف على الحروف العربية المكتوبة بخط اليد. في هذا العمل، بالنسبة لمجموعة البيانات، نستخدم مجموعة بيانات AHCD لتدريب نموذجنا واختباره، ونستخدم Random Forest كنموذج للتدريب والاختبار. نعتمد على نموذجين Random Forest الأول يأخذ 16 ميزة مستخرجة من الصورة كمدخلات؛ من ناحية أخرى، يأخذ الثاني الصورة كمدخل. نحن نستخدم بايثون كلغة برمجة. في الأخير، حققنا دقة تدريب بلغت 97.14٪ ودقة اختبار 73.48٪.

الكلمات المفتاحية: الذكاء الاصطناعي،Random Forest ، الخصائص المميزة، التعرف على الاحرف العربية المكتوبة بخط اليد، بايثون.

Abstract

The main goal of our work is to exploit the effectiveness of artificial intelligence methods to recognize handwritten Arabic letters. In this work, for the dataset, we use the AHCD dataset to train and test the Random Forest model. We rely on two different approaches to recognize the letter. In the first one, the model takes as inputs 16 features extracted from the image, whereas in the second approach, the model takes the whole image as an input. We use Python as the programming language. We achieved a training accuracy of 97.14% and a testing accuracy of 73.48%.

Keyword: Artificial Intelligence, Random Forest, feature extraction, recognize handwritten Arabic letters, Python.

Résumé

L'objectif principal de notre travail est d'exploiter l'efficacité des méthodes d'intelligence artificielle pour reconnaître les lettres arabes manuscrites. Dans ce travail, pour l'ensemble de données, nous utilisons l'ensemble de données AHCD pour former et tester notre modèle, et nous utilisons le Random Forest comme modèle pour former et tester. Nous nous appuyons sur deux modèles de Random Forest, le premier prend en entrée 16 caractéristiques extraites d'une image ; d'autre part, le second prend une image en entrée. Nous utilisons Python comme langage de programmation. Finalement, nous avons atteint une précision d'entraînement de 97,14 % et une précision de test de 73,48 %.

Mots clés : intelligence artificielle, Random Forest, caractéristiques extraites, reconnaître les lettres arabes manuscrites, Python.