**Media Engineering and Technology Faculty**
**German University in Cairo**

# Image Classification for handwritten Arabic Letters

**Bachelor Thesis**

| | |
|---|---|
| Author: | Hosain Ghoraba |
| Supervisor: | Assoc. Prof. Dr Wael Zakaria Abdallah |
| Co-Supervisor: | Asst. Prof. Dr Ahmed Ashry Abdelfattah |
| Submission Date: | 19 May, 2024 |

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Hosain Ghoraba
19 May, 2024

# Acknowledgments

I would like to express my sincere gratitude to Dr Wael Zakaria, for his invaluable guidance, support, and feedback throughout this research project. His expertise and insights were instrumental in shaping this thesis.

# Abstract

There is a growing trend in Arabic letters recognition systems. There is also a growing demand to digitize ancient records. Academics also require it to make it easier for them to convert their handwritten manuscripts into databases so they can keep track of students' grades. While there might be systems that partially extract text from Arabic images, they undoubtedly have a variety of deficiencies and drawbacks, in particular the limits imposed on free usage of the applications or the unavoidable mistakes and difficulties that need more investigation and development by individuals who understand Arabic. This paper focused on building a model that can classify handwritten Arabic letters images (with 99.999 accuracy). It was also a study to find which letters are considered "similar" by the model, by dividing them into clusters using an unsupervised machine learning approach, which should to be further studied to understand more about similarities between Arabic letters, and the key differences between each cluster, leading to the improvement of any future Arabic letters recognition systems.

# Contents

**Acknowledgments** **III**

**1 Introduction** **1**
- 1.1 Overview . . . . . 1
- 1.2 Limitations & Challenges . . . . . 1
- 1.3 Tequniques . . . . . 2

**2 Background** **3**

**3 Literature Review** **5**

**4 Methodology** **6**
- 4.1 Model training . . . . . 6
- 4.2 Clustering . . . . . 8

**5 Results** **9**

**6 Conclusion** **20**

**7 Future Work** **21**

**Appendix** **22**

**References** **25**

# Chapter 1

# Introduction

## 1.1 Overview

In recent years , the advancement of optical character recognition (OCR) that is based on neural networks & deep learning has led to the development of many digitalization systems that can convert scanned text into digital form. This has been a great help for many , especially for people who are visually impaired , who are now finally able to listen to a scanned document via using OCR to convert the document to text , then passing that text to TTS (Text To Speech) readers. It has also helped many to digitize scanned documents , to perform all kind of operations where computers have the advantage over humnas , (e.g., search , analysis..etc) , also for academics who need to convert their handwritten manuscripts into databases so they can keep track of students' grades, and many more.

This project focused on building a ML model that can classify arabic letters images with accuracies of (90&, 93%, & 99.9% , according to the dataset used). It also used transfer learning by leveraging the VGG16 model to exract features from the images, to use these features in clustering the dataset, to understand more about similarities between handwritten Arabic letters, and the key differences between each cluster.

## 1.2 Limitations & Challenges

However, the majority of these systems are designed to recognize English text, and there are few systems that can recognize Arabic. This is due several factors including the scarcity of good Arabic datasets, the complexity of the Arabic language (e.g., each letter can take up to 4 different forms depending on its position in the word , and the presence of up to 5 diacritics like : fatha , damma , kasra , sokoon & shadda) that can make it more difficult to recognize the letter with its diacritic (if any !) , even the existing OCR systems that can recognize Arabic text are mainly designed to recognize printed text, and there are few systems that can recognize handwritten Arabic text. Which themselves have their own limitations and drawbacks such as the limits imposed on free usage of the applications or the unavoidable mistakes and difficulties that need more investigation and development by individuals who understand Arabic.

## 1.3 Tequniques

### Common Tequniques

When working with any machine learning model , one has to experiment with many known factors to get the best classification accuracy , including but not limited to : the size that the input image has to be resized to before passing it to the model , batch size , preprocessing steps , number of epochs , layers architechture & many more.

### Clustering Tequnique

Another less-used approach that can be used is clustering , which comes in handy especially in datasets that contain a large number of classes (28 arabic letters in our case , assuming each letter will not be sub-divided according to its position) , the idea is that when there are lots of classes that the model has to differentiate between , the learning process becomes much harder for the model. So , the clustering tequnique works by grouping images of similar letters together in one set (e.g., cluster) , then for each cluster, a model is trained to differentiate between the images of that cluster only , which is easier than differentiating between all the images of all the classes. Then , when a new image is passed to the OCR model , the model will first predict the cluster that the image belongs to , then pass the image to the model that was trained on that cluster to get the final prediction. The process of identifying the cluster that the image belongs to can itself be considered an image classification problem , where we consider all the images in one cluster as one class , regardless of the real number of classes in that cluster. So in case we divide our dataset into 4 clusters for example , we will train a model (call it the "outer model") and pass to it all the images belonging to cluster 1 as one class , all the images belonging to cluster 2 as another class , and so on.

# Chapter 2

# Background

Image classification is the process of categorizing images into similar classes or categories. It is used as one of the main steps of building an optical character recognition system (OCR) , which can be used to digitalize documents , enabling us to perform all kind of machine-superiority operations in a document such as search and analysis. The emergence of deep learning and convolutional neural networks (CNN) has made this task relatively easy for documents originally written by a computer , due to the consistency of each letter shape across the whole document , it is even easier for languages where each character can take only 1 or 2 forms (e.g., 'capital' and 'small' forms for english letters), but It is a much challenging task for handwritten documents , due to the incosistency of each letter shape across the document , and even more challenging for languages where each letter can be in 4 different forms (e.g., start , middle , end & isolated positions for arabic letters).

In order to improve the accuracy of a CNN model for image classification , One has to experiment with many factors, such as the input image dimensions , batch size , data augmentation , layers architecture , ...etc

Another approach used to improve the classification accuracy is clustering, the use of this approach comes in handy in datasets where there is a high number of classes to diffirentiate between, which makes it more complex for the model to differentiate between each class , what clustering does is grouping similar classes in a single set (or cluster) , so for example if we have 28 classes to differentiate between (one for each letter in the arabic alphabet) , we can group similar letters into a separate set (e.g., cluster) , where each cluster contains letters that are so similar to each other. For example we will put the letters (ب) , (ت) , and (ث) in one cluster , and the letters (ج) , (ح) , and (خ)

in another one , and so on. We will end up with 4 clusters for example , then instead of training just one model to differentiate between all the 28 classes , we will train a separate model on each cluster , so we will end up with 4 models , each one is responsible for classifying the letters belonging to this cluster only , this will make the classification task easier for the model , and will improve its accuracy. What remains is , given an input image , we need to know which model to use to classify it , this task itself can be considered as an image classification task & can be done using another model , lets call it the "meta-model" , this model will take the input image and will predict which cluster this image belongs to , then it will use the model of that cluster to classify the image.

# Chapter 3

# Literature Review

In their study [1], the authors proposed a system for Arabic handwriting letter recognition. Their approach involved initially converting images to grayscale, followed by smoothing and noise removal through filtering. Classification was accomplished using a CNN network, achieving an accuracy of 94%. However, this system only considered 28 alphabetical characters and did not address variations in letter shapes based on their positions.

The authors in [2] introduced a model that is capable of recognizing handwritten Arabic characters based on deep learning, which uses Convolutional Neural Networks to divide the Arabic characters into 28 subclasses (but without accounting for each letter position) to impove the classification phase in the OCR process, the model was tested on the Handwritten Arabic Characters Database (HACDB) dataset and it gave 98% of classification accuracy.

In [3], a deep learning architecture is used to address the issue of classifying Arabic letters written by hand, and the method based on a convolutional neural network (CNN) architecture as a self-extractor and classifier gives a remarkable accuracy of 99.02%.

Also in [4], the authors used transfer learning to classify Arabic, Hijja, and AHCD handwriting datasets and achieved a 0.0001 learning rate for each of the three datasets, while using data augmentation and 50% dropout on a fully connected layer to reduce overfitting.

Another authors in [5] used a Convolutional Neural Network for handwritten characters' recognition. The model has been trained on a dataset of 16,800 images of handwritten Arabic characters with different shapes to perform classification. The proposed model achieved high recognition accuracy of 97.2%, outperforming other state-of-art models. When applying data augmentation, the model achieved better results and accuracy of 97.7%

# Chapter 4

# Methodology

## 4.1 Model training

### Architechture

The first step done was to build a model that can distinguish between 2 classes only , the code of such a model was found at kaggle here , the model starts first by splitting the data into 3 parts, 80% for training, 10% for validation and 10% for testing, and the percentage of each class was the same in the 3 parts.

The architecture of the model is a Convolutional Neural Network (CNN) with four blocks, each containing a convolutional layer followed by a batch normalization layer, a max pooling layer, and a dropout layer. The number of filters in the convolutional layers doubles with each block, starting from 32 in the first block. After these blocks, the model has a flatten layer which is followed by a fully connected (dense) layer with 512 units and a ReLU activation function. This is followed by another batch normalization layer and a dropout layer. The output layer of the model is a dense layer with a number of units equal to the number of classes found in the dataset, and uses a sigmoid activation function. The full architecture of the neural network is shown in figure 1 in the appendix.

Batch size was set to 32, "Adam" optimizer & and a binary-crossentropy loss function were used. Early stopping was also used to stop training if the validation loss did not improve for 3 consecutive epochs. The learning rate was reduced by a factor of 0.5 if the validation accuracy did not improve for 2 consecutive epochs.

After that, several modifications were done to the model to enable it to distinguish between any number of classes, not just 2 , key changes includes changing the output layer to have a number of units equal to the discovered number of classes (or folders) in the dataset, not just 2. Also the "sigmoid" activation function was replaced with a "softmax" one, then the loss function was also changed to categorial-crossentropy instead of binary-crossentropy

## Datasets

The model was trained on 3 datasets separately. The first dataset contains 28 folders, each one contains about 2400 images of one of the 28 Arabic letters, regardless of its position (start, middle, end or isolated). The size of each image is about 20 x 20 pixels.

The second dataset has the same structure as the first one, but each folder contains 900 images of one letter, and the size of each image is about 300 x 300 pixels.

The third dataset has the same structure as the previous 2, but each folder contains 1000 images of one letter, and the size of each image is about 32 x 32 pixels.

## Image size

The model was trained on each one of the 3 datasets separately. For each dataset, the model was trained 6 times, each time with a different image size. Image sizes used were 16 x 16, 32 x 32, 64 x 64, 128 x 128, 256 x 256, and 300 x 300 pixels. The overall accuracy of the model on both the validation & testing sets, as well the accuracy of each class, was calculated for each image size.

## Testing

After training the model on each dataset with the 6 different image sizes separately, we ended up with 18 models (3 datasets x 6 image sizes), the overall accuracy of each model was tested on 100% of each one of the 3 datasets separately.

## Training on 3 classes only

After that, we wanted to test whether training the model only on some selected classes of the second dataset would improve the classification accuracy of those classes compared to the case when training the model on the whole dataset or not. So we selected 3 (visually) similar classes , which are (ب) , (ت) , and (ث) , and trained the model on images belonging to these 3 classes only, then compared the classification accuracy of each class with the classification accuracy of those classes when the model when trained on all the 28 classes.

## 4.2 Clustering

The next step was to cluster the dataset with the help of the VGG16 model. The VGG16 model is a pre-trained model that is used to extract various features from images and group them in an array called the "feature vector". The features in the feature vector represent various aspects of the image, such as the edges, colors, and textures. These features can be used to group similar images together, which is the main idea behind clustering. Feature vectors were extracted from the images in the dataset using the VGG16 model, then these feature vectors were used to cluster the dataset into a pre-defined number of clusters using the MiniBatchKMeans[1] algorithm, which is a variant of the KMeans algorithm[2] that is faster and more efficient for clustering large datasets. Datasets used

In a try to find the optimal number of clusters, the "Elbow Method"[3] was used, and the optimal number of clusters was "2" according to that method, which is not a realistic number (the 28 Arabic letters cannot be divided into 2 clusters where each cluster have letters that looks very similar to each other), so for each one of the datasets, the number of clusters was set to 6, which is a more realistic number.

The clustering was done on 2 datasets, the first one is the same first dataset used in the model training, the second one is also the same as the second dataset used in model training, but in the 2nd dataset only, the images of each letter were separated according to the letter position (start, middle, end, isolated), so the number of classes in the 2nd dataset used here was 103 instead of 28.[4]

After each clustering trail, several metrics were calculated and plotted, including the size of each cluster with respect to the total number of images in the dataset, the classes included in each cluster, and for each class in each cluster, the percentage of that class photos in the cluster with respect to the total number of photos of that class in the whole dataset was also plotted.

[1]In fact, the MiniBatchKMeans algorithm is a less-accurate version of the standard KMeans algorithm, but it is much faster and more efficient for large datasets. It also solves the problem of memory overflow that may occur when using the KMeans algorithm with a large dataset, which indeed happened when trying to use the standard KMeans algorithm , despite using 30 GB of RAM on Kaggle GPU !

[2]The KMeans algorithm is a clustering algorithm that groups data points into a number of clusters based on their similarity. The algorithm works by iteratively assigning data points to the nearest cluster center, then updating the cluster centers based on the new data points assigned to them. The algorithm converges when the cluster centers no longer change significantly between iterations.

[3]The Elbow Method is a technique used to find the optimal number of clusters in a dataset. The method works by plotting the sum of squared distances between data points and their cluster centers for different numbers of clusters, then looking for the "elbow" point in the plot, which is the point where the rate of decrease in the sum of squared distances starts to slow down. This point ususally represents a good estimate of the optimal number of clusters.

[4]Not 112 classes (28 letter x 4 positions), since some letters images didnot include all the 4 positions.

# Chapter 5

# Results

## Model training results

These are the model accuracies on the 3 datasets, separated according to the image size used for training. The results are shown in the tables below.

| Dataset 1 | | |
|---|---|---|
| Image size | Testing | Validation |
| 16 | 97.5 | 97.1 |
| 32 | 98.2 | 97.9 |
| 64 | 99.9 | 99.9 |
| 128 | 99.9 | 99.6 |
| 256 | 99.9 | 99.3 |
| 300 | 98 | 97.5 |

Table 5.1: Model accuracies (%) on the 1st dataset (Dataset original images size = 20x20 pixels)

| Dataset 2 | | |
|---|---|---|
| Image size | Testing | Validation |
| 16 | 27.5 | 25.8 |
| 32 | 68.6 | 65.6 |
| 64 | 91.4 | 85.9 |
| 128 | 73.1 | 65.8 |
| 256 | 60.3 | 51.8 |
| 300 | 9.1 | 8.2 |

Table 5.2: Model accuracies (%) on the 2nd dataset (Dataset original images size = 300x300 pixels)

| Dataset 3 | | |
|---|---|---|
| Image size | Testing | Validation |
| 16 | 70.6 | 66.2 |
| 32 | 78.9 | 75.3 |
| 64 | 94.8 | 86.9 |
| 128 | 81.3 | 74.7 |
| 256 | 56.3 | 52.3 |
| 300 | 93.7 | 74.8 |

Table 5.3: Model accuracies (%) on the 3rd dataset (Dataset original images size = 32x32 pixels)

As we can see in the validation columns, in the first dataset, the image size of 64x64 pixels gives the best results (99.9%), although other sizes also gives close accuracies. In the second one, the image size of 64x64 pixels also gives the best results (91.4%), while the image size of 300x300 pixels gave very bad accuracy (9.1%). In the third dataset, again the image size of 64x64 pixels was the best , and the image size of 300x300 pixels gives good accuracy also (93.7%), but other sizes are not comparable to 64 and 300 sizes. One can notice the huge effect of the image size on the model accuracy, and that the image size of 64x64 pixels is the best for all the datasets.

## Model testing results

Each one of the 18 models above (3 datasets x 6 image sizes) was tested on 100% of each one of the 3 datasets separately, and the results are shown in the tables below.

| Dataset 1 Models | | | |
|---|---|---|---|
| Model input images size | Dataset 1 | Dataset 2 | Dataset 3 |
| 16 | 97.5 | 7.6 | 16.9 |
| 32 | 98.1 | 10.4 | 18.3 |
| 64 | 99.9 | 7.9 | 19.9 |
| 128 | 99.9 | 10.4 | 17.1 |
| 256 | 99.8 | 9.1 | 13.7 |
| 300 | 97.9 | 7.7 | 9.3 |

Table 5.4: Accuracies (%) when testing models that was trained on the 1st dataset on the 3 datasets

| Dataset 2 Models | | | |
|---|---|---|---|
| Model input images size | Dataset 1 | Dataset 2 | Dataset 3 |
| 16 | 17.8 | 27.3 | 25.5 |
| 32 | 27.4 | 68.1 | 42.8 |
| 64 | 22.2 | 90.4 | 49.6 |
| 128 | 8.3 | 71.9 | 25.9 |
| 256 | 6.8 | 58.6 | 18.0 |
| 300 | 7.1 | 9.0 | 8.8 |

Table 5.5: Accuracies (%) when testing models that was trained on the 2nd dataset on the 3 datasets

| Dataset 3 Models | | | |
|---|---|---|---|
| Model input images size | Dataset 1 | Dataset 2 | Dataset 3 |
| 16 | 27.9 | 21.6 | 69.8 |
| 32 | 32.7 | 47.7 | 78.2 |
| 64 | 38.8 | 64.1 | 93.3 |
| 128 | 23.4 | 47.4 | 80.2 |
| 256 | 6.4 | 22.3 | 55.5 |
| 300 | 12.2 | 47.1 | 89.8 |

Table 5.6: Accuracies (%) when testing models that was trained on the 3rd dataset on the 3 datasets

As we can see in each one of the tables above, each model performed well on the dataset it was trained on, and the performance significantly decreased when testing on the other datasets.

## Testing the "3 classes only" model

As previously mentioned in the methodology chapter, we wanted to test whether training the model only on some selected classes of the second dataset would improve the classification accuracy of those classes compared to the case when training the model on the whole dataset or not. Here are the results of this test, the accuracies of the 3 classes when the model was trained on them only, and when the model was trained on all the 28 classes. Note that the dataset used in this test is the second dataset, and the used image size was 128x128 pixels.

| Class | All 28 classes | 3 classes only |
|---|---|---|
| (ب) | 95.7 | 100 |
| (ت) | 74.8 | 95.7 |
| (ث) | 70.7 | 88.3 |

Table 5.7: Accuracies (%) of classes (ب) , (ت) , and (ث) when the model was trained on them only, compared to the case when the model was trained on all the 28 classes

As we can see in the table above, the model performed better on the 3 classes when it was trained on them only, compared to the case when it was trained on all the 28 classes. The accuracy of class (ب) increased from 95.7% to 100%, the accuracy of class (ت) increased from 74.8% to 95.7%, and the accuracy of class (ث) increased from 70.7% to 88.3%.

## Clustering results

As mentioned in the methodology chapter, the clustering was done on 2 datasets, with the number of clusters set to 6 for each one of them. The results are shown in the figures below. Note that in the plots of the classes included in each cluster, if a class was included in a certain cluster, but the percentage of that class photos in the cluster with respect to the total number of photos of that class in the whole dataset was less than a certain threshhold , the class was not shown in the plot. That threshhold was set to: 1 / number of clusters, so if the number of clusters was 6, then the threshhold was set to 1 / 6 = 16.7%. This threshhold was set to improve the readability of the plots, and also to avoid showing classes that were included in a cluster by a very small percentage, which most likely were included in that cluster because of some random noise in the dataset, or due to the fact that Kmeans algorithm (or specifically the MiniBatchKMeans algorithm used here) is not 100% accurate.

## 1st dataset clustering results

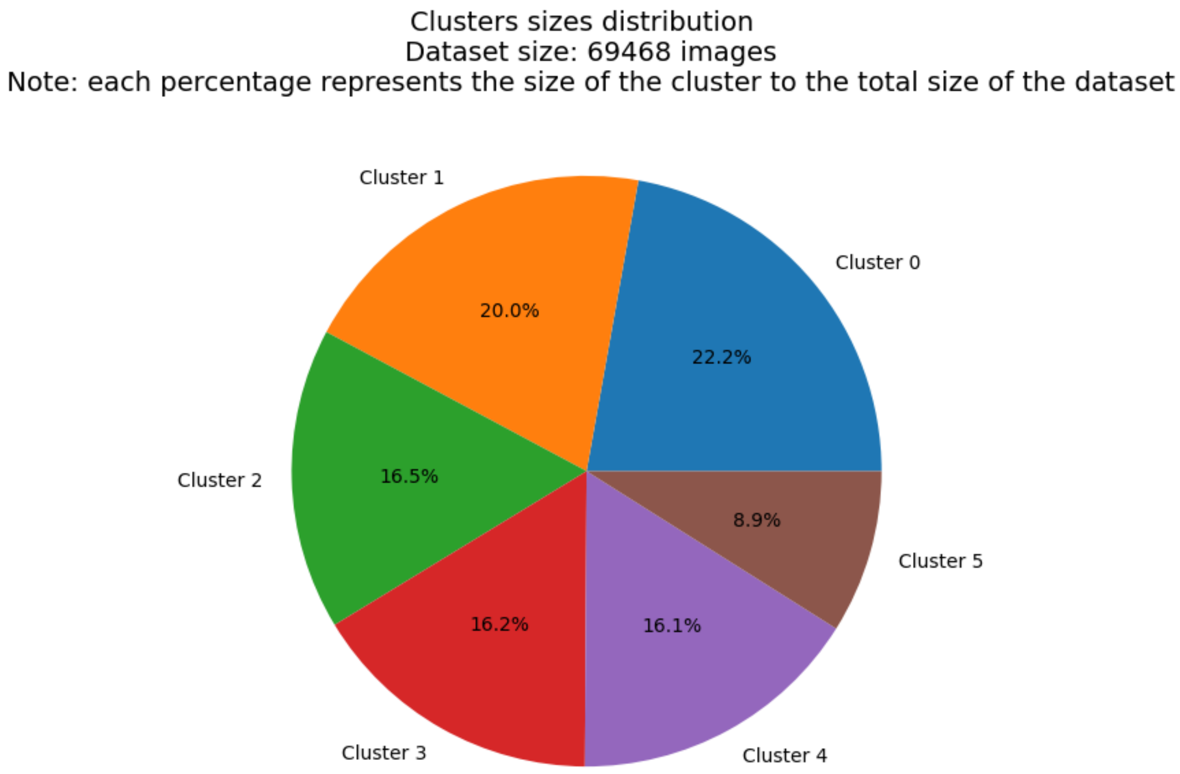


Figure 5.1: Clusters sizes for dataset 1

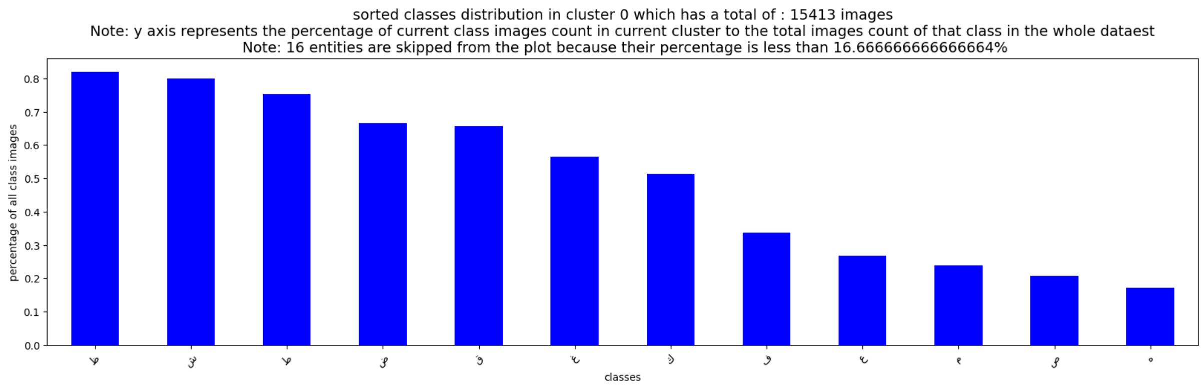


Figure 5.2: Cluster 0 letters in dataset 1

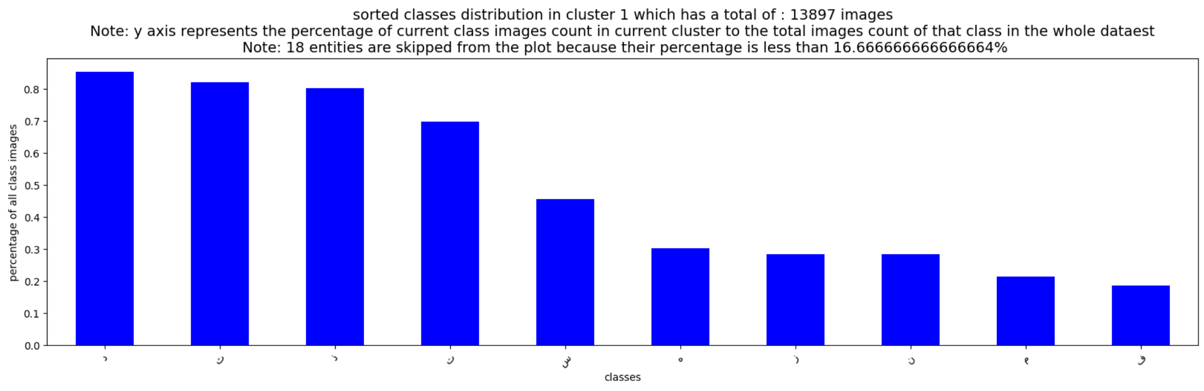


Figure 5.3: Cluster 1 letters in dataset 1

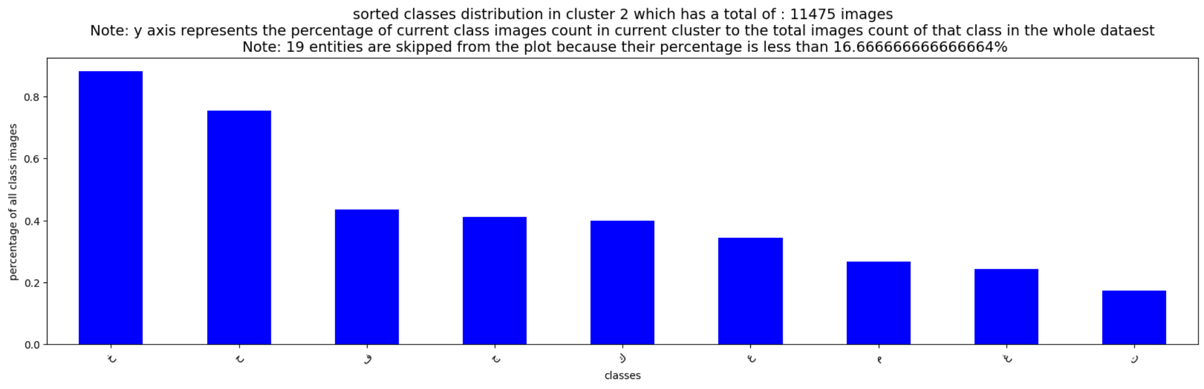
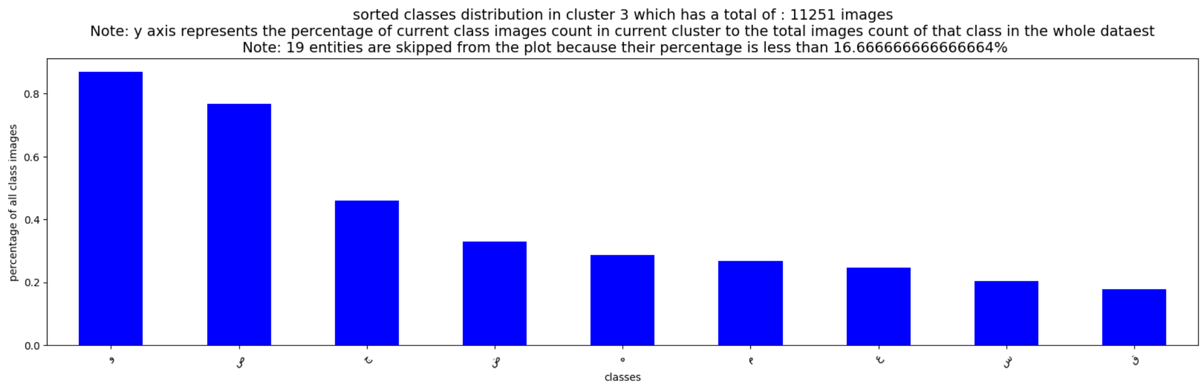
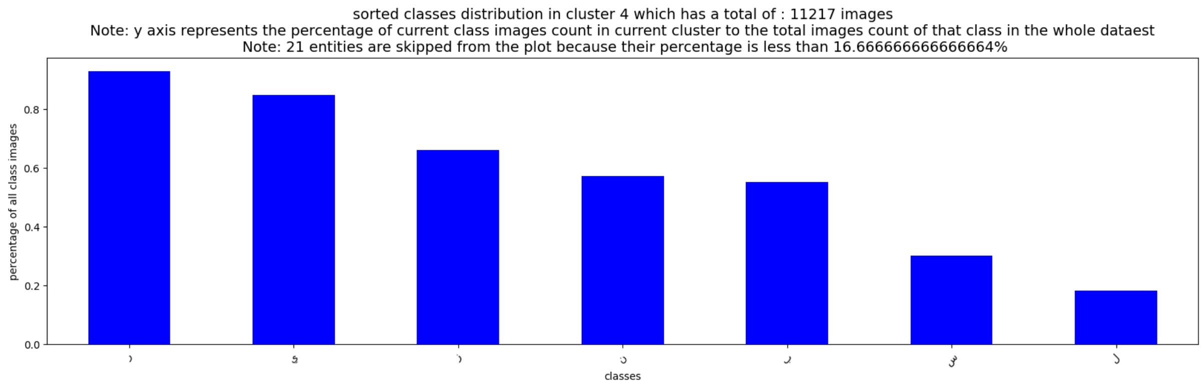


Figure 5.4: Cluster 2 letters in dataset 1



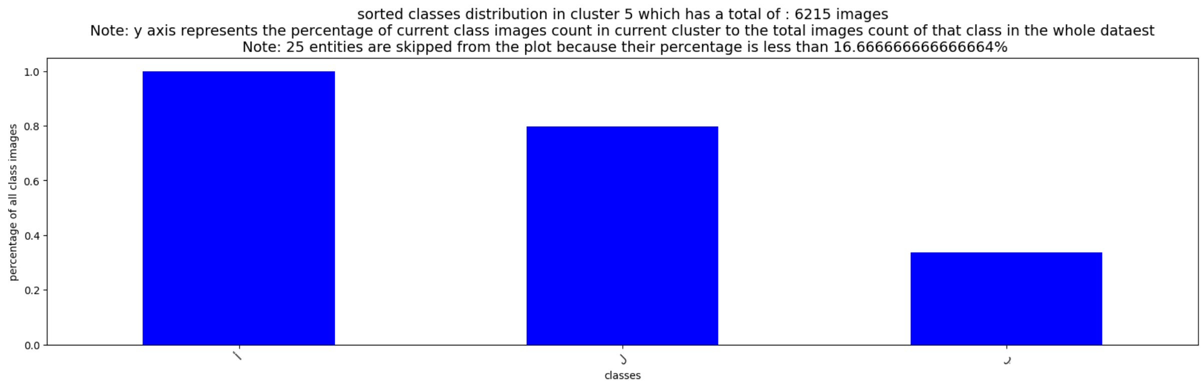
sorted classes distribution in cluster 3 which has a total of : 11251 images
Note: y axis represents the percentage of current class images count in current cluster to the total images count of that class in the whole dataest
Note: 19 entities are skipped from the plot because their percentage is less than 16.666666666666664%

Figure 5.5: Cluster 3 letters in dataset 1



sorted classes distribution in cluster 4 which has a total of : 11217 images
Note: y axis represents the percentage of current class images count in current cluster to the total images count of that class in the whole dataest
Note: 21 entities are skipped from the plot because their percentage is less than 16.666666666666664%

Figure 5.6: Cluster 4 letters in dataset 1



sorted classes distribution in cluster 5 which has a total of : 6215 images
Note: y axis represents the percentage of current class images count in current cluster to the total images count of that class in the whole dataest
Note: 25 entities are skipped from the plot because their percentage is less than 16.666666666666664%

Figure 5.7: Cluster 5 letters in dataset 1

## 2nd dataset clustering results

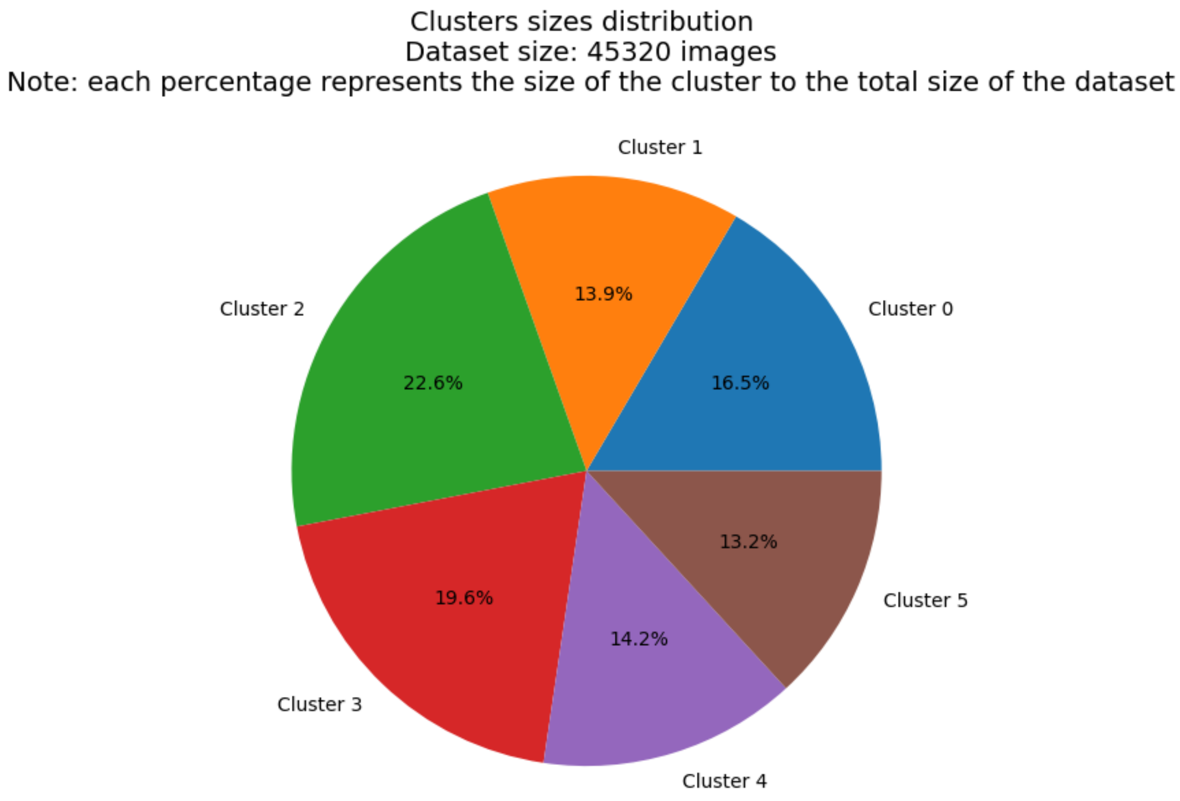


Figure 5.8: Clusters sizes distribution for dataset 2

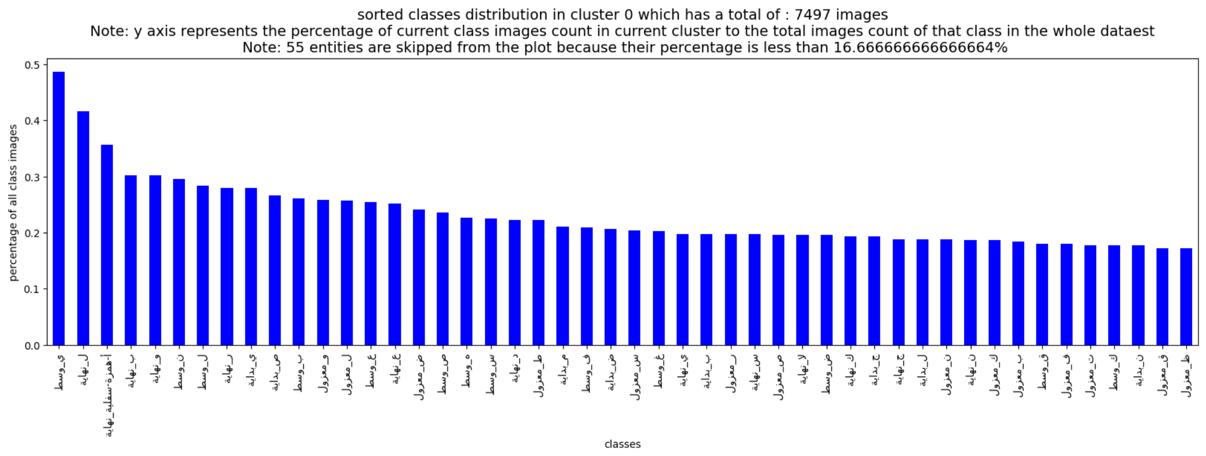


Figure 5.9: Cluster 0 letters in dataset 2

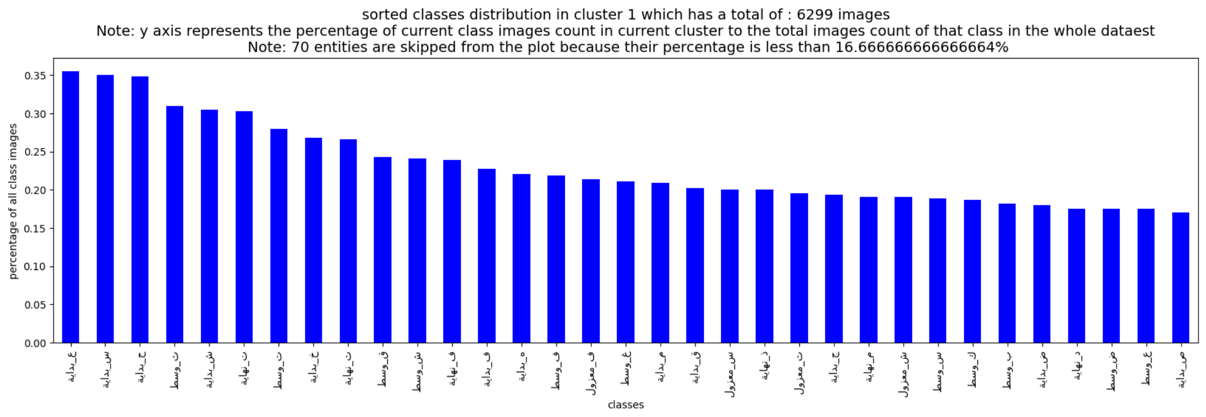


Figure 5.10: Cluster 1 letters in dataset 2

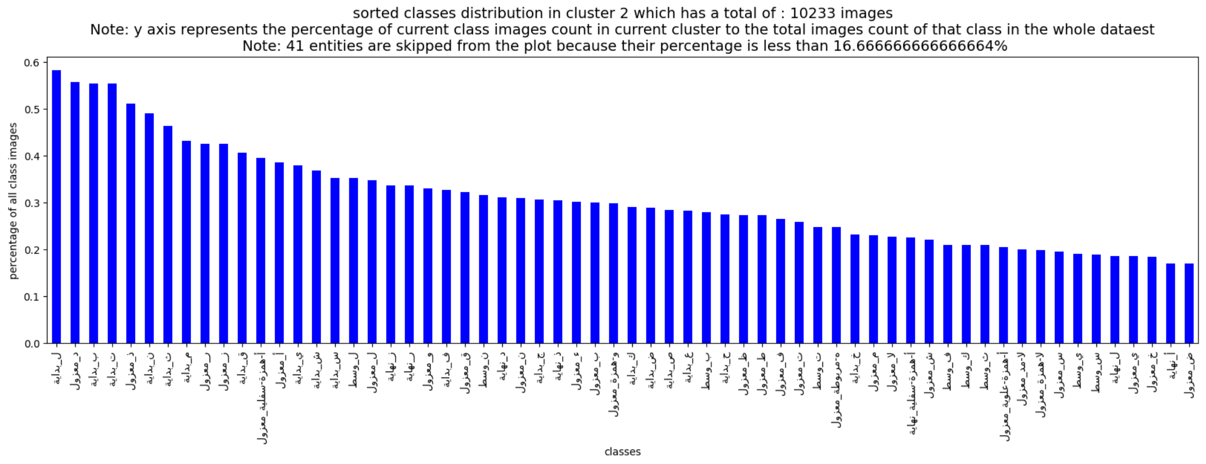


Figure 5.11: Cluster 2 letters in dataset 2

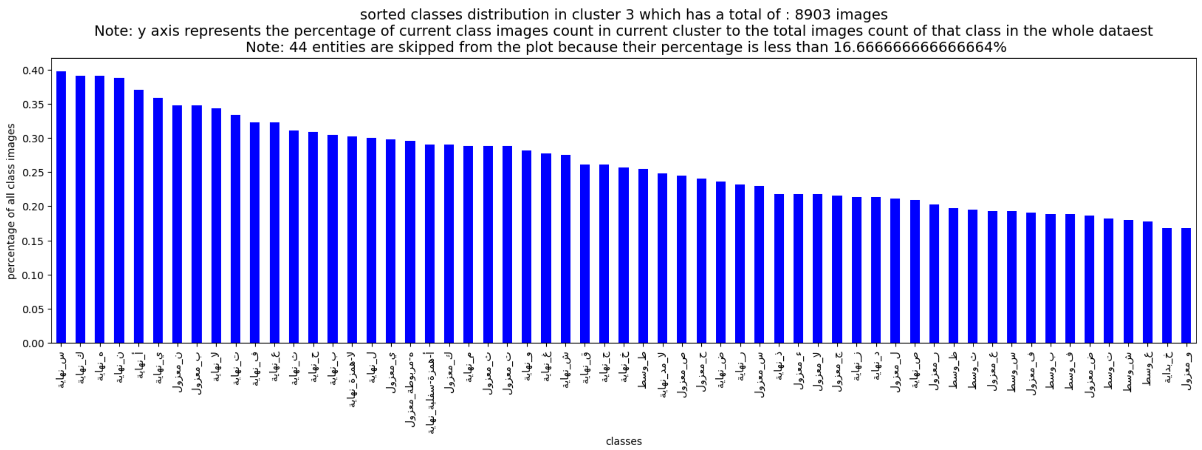


Figure 5.12: Cluster 3 letters in dataset 2

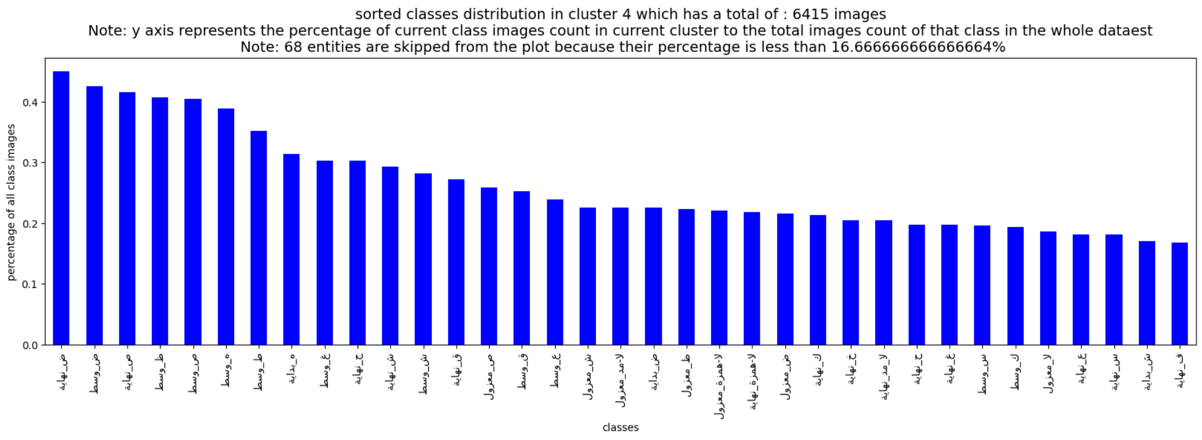


Figure 5.13: Cluster 4 letters in dataset 2

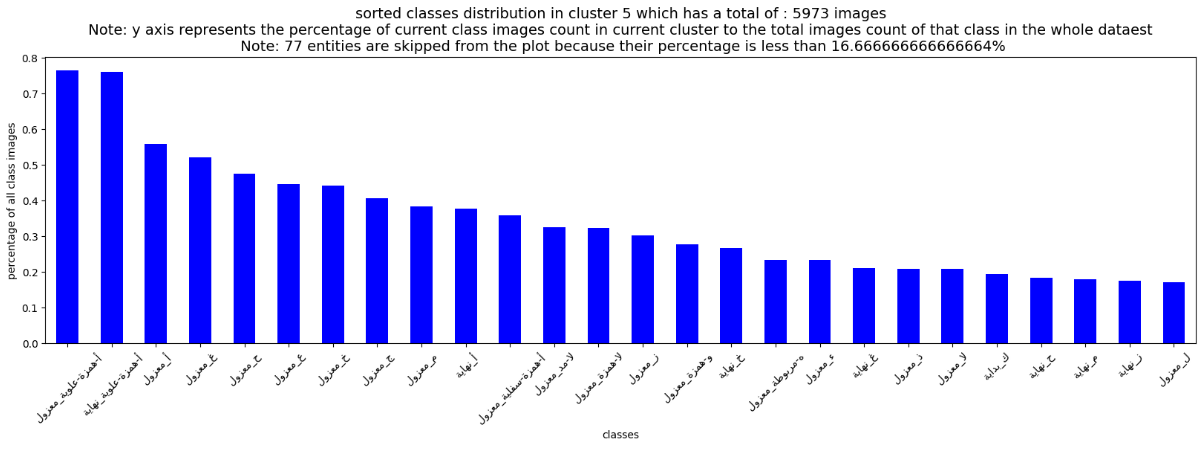


Figure 5.14: Cluster 5 letters in dataset 2

# Chapter 6

# Conclusion

In this study, we developed a model to classify Arabic letters and evaluated its performance on three datasets with different image sizes. Our findings indicate that an image size of 64x64 pixels yields the best classification results across all datasets. When testing the model on the three datasets, we observed that the model performed well on the dataset it was trained on, but its performance significantly decreased when tested on the other datasets. This suggests that the model's generalizability is limited and highlights the importance of training on diverse data.

Additionally, we investigated whether training the model on a subset of selected classes would improve the classification accuracy for those classes. The results confirmed that the model achieved better performance on the selected classes when trained exclusively on them. This implies that specialized training can enhance model accuracy for specific subsets of data.

We also employed the MiniBatchKMeans algorithm to cluster two of the datasets into six clusters. The clustering results were reasonable, particularly for the first dataset, though some classes were included in clusters by a small percentage. This minor misclassification is likely due to random noise in the dataset or the inherent limitations of the MiniBatchKMeans algorithm.

While our study provides valuable insights into the classification of Arabic letters, it has several limitations. The decrease in performance on unseen datasets underscores the need for further research to improve model generalizability. Future work could explore more advanced techniques to enhance cross-dataset performance, such as transfer learning or data augmentation. Additionally, investigating other clustering algorithms might yield more accurate groupings.

In conclusion, our study demonstrates the efficacy of using 64x64 pixel images for Arabic letter classification and highlights the benefits of targeted training for specific classes. These findings contribute to the field of machine learning for Arabic script recognition and suggest avenues for future research to address the identified limitations and enhance model robustness and accuracy.

# Chapter 7

# Future Work

Next steps in this project mainly include the following:

1. Improving the accuracy of each model on other datasets that was not exposed to the model during the training phase, which is the main deficiency of the current model , by using data augmentation techniques and transfer learning to improve the generalization of the model.

2. Using the outcomes of the clustering process done in this paper to build the "meta-model" discussed , that is reponsible for predicting the cluster of an input image , to determine which model to use for the recognition process, and compare the accuracy for each letter with the current approach , note that the accuracy of the model in the new approach will be :

$$\sum_{l=\text{أ}}^{\text{ي}} M_l * C_l$$

   where $M_l$ is the accuracy of the "meta-model" while trying to classify the letter i into its right cluster , and $C_l$ is the classification accuracy of the model responsible for recognizing the letter i. Also note that $C_l = 0$ if $M_l \neq 1$ , because the wrong cluster will be chosen and the model will not be able to recognize the letter since it does have letter class as a possible output.

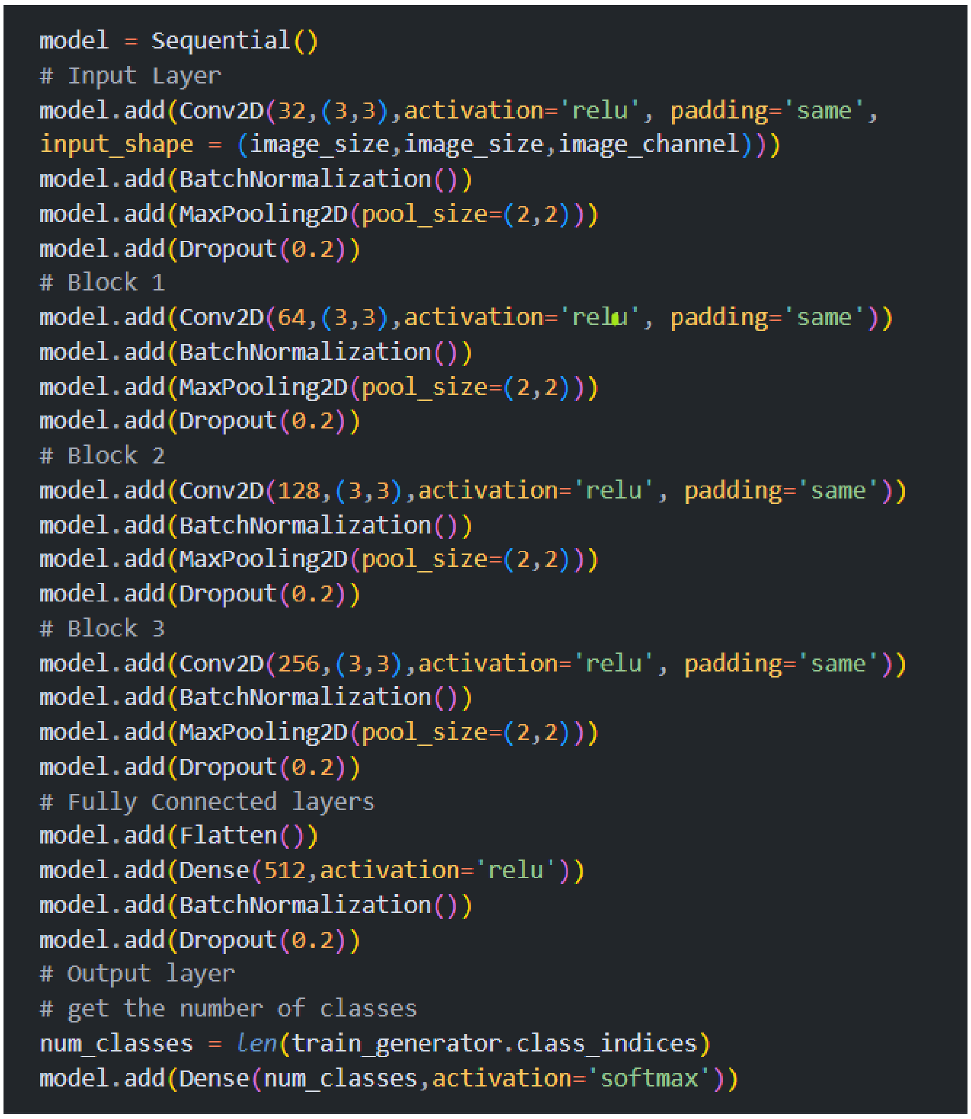
3. Investigating other clustering algorithms that might yield more accurate groupings, which can help further undersatding the similarities seen by the computer between the Arabic letters, and how to group them in a more accurate way to iprove the performance of future models.

# Appendix

# List of Figures

5.1 Clusters sizes for dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . . 14
5.2 Cluster 0 letters in dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . 15
5.3 Cluster 1 letters in dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . 15
5.4 Cluster 2 letters in dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . 15
5.5 Cluster 3 letters in dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . 16
5.6 Cluster 4 letters in dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . 16
5.7 Cluster 5 letters in dataset 1 . . . . . . . . . . . . . . . . . . . . . . . . . 16
5.8 Clusters sizes distribution for dataset 2 . . . . . . . . . . . . . . . . . . . 17
5.9 Cluster 0 letters in dataset 2 . . . . . . . . . . . . . . . . . . . . . . . . . 18
5.10 Cluster 1 letters in dataset 2 . . . . . . . . . . . . . . . . . . . . . . . . . 18
5.11 Cluster 2 letters in dataset 2 . . . . . . . . . . . . . . . . . . . . . . . . . 18
5.12 Cluster 3 letters in dataset 2 . . . . . . . . . . . . . . . . . . . . . . . . . 19
5.13 Cluster 4 letters in dataset 2 . . . . . . . . . . . . . . . . . . . . . . . . . 19
5.14 Cluster 5 letters in dataset 2 . . . . . . . . . . . . . . . . . . . . . . . . . 19

1 CNN Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24



```python
model = Sequential()
# Input Layer
model.add(Conv2D(32,(3,3),activation='relu', padding='same',
input_shape = (image_size,image_size,image_channel)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
# Block 1
model.add(Conv2D(64,(3,3),activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
# Block 2
model.add(Conv2D(128,(3,3),activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
# Block 3
model.add(Conv2D(256,(3,3),activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
# Fully Connected layers
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
# Output layer
# get the number of classes
num_classes = len(train_generator.class_indices)
model.add(Dense(num_classes,activation='softmax'))
```

Figure 1: CNN Architecture

# Bibliography

[1] A. El-Sawy, M. Loey, and H. El-Bakry. Arabic handwritten characters recognition using convolutional neural network. *WSEAS Transactions on Computer Research*, 5(1):11–19, 2017.

[2] Guided classification for arabic characters handwritten recognition. 2022.

[3] Mouhssine El Atillah, K. El fazazy, and Jamal Riffi. Classification of arabic alphabets using a combination of a convolutional neural network and the morphological gradient method. *Baghdad Science Journal*, 2023.

[4] Siti Ummi Masruroh, Asep Taufik Muharram, and Rizka Amalia Putri. Deep convolutional neural networks transfer learning comparison on arabic handwriting recognition system. *JOIV : International Journal on Informatics Visualization*, 7(2):330–330, 2023.

[5] Mohammed N AlJarrah, Moath Mohmmad Zyout, and Rehab Duwairi. Arabic handwritten characters recognition using convolutional neural network. 2021.