Machine Learning Engineer Nanodegree

Handwritten Arabic character recognition

Shenouda Farouk Wahba
June 6th, 2019

I. Definition

1.1 Project Overview

Optical character recognition (OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). It is widely used as a form of information entry from printed paper data records, whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining.

OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

The field of optical character recognition (OCR) is very important, especially for offline handwritten recognition systems. Offline handwritten recognition systems are different from online handwritten recognition systems. The ability to deal with large amounts of script data in certain contexts will be invaluable. One example of these applications is the automation of the text transcription process applied on ancient documents considering the complex and irregular nature of writing. Arabic optical text recognition is experiencing slow development compared to other languages.

Handwritten Arabic character recognition (HACR) has attracted considerable attention in recent decades. Researchers have made significant breakthroughs in this field with the rapid development of deep learning algorithms.[1] [2] Arabic is a kind of the Semitic language used in countries of the Middle East as a mother language of millions people.

Arabic handwriting fonts is an ancient art, and there are a lot of old books and scripts need to digitize. That motivate researchers to build algorithms for this job to save ancient Arabic culture.

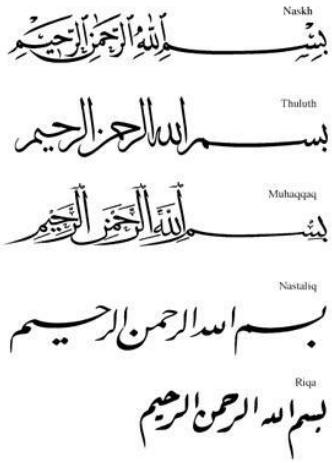
The image shows an old Arabic script.

1.2 Problem Statement

Generally, the **Arabic alphabet characters** consist of **twenty-eight** alphabet characters that illustrated in the following table:

ص	ش	س	ز	ر	ذ	د	خ	ح	ج	ث	ت	ب	أ
sad	sheen	seen	zay	raa	thal	dal	khaa	haa	geem	thaa	taa	baa	alef
ي	و	ه	ن	م	ل	ك	ق	ف	غ	ع	ظ	ط	ض
yaa	waw	haa	noon	meem	lam	kaf	qaf	faa	ghain	ain	zaa	ttaa	dad

There is also a big variety of Arabic handwriting fonts Arabs still use for now. Most of educated Arab people write two fonts **Naskh** and **Reqaa** and sometimes by mistake and fast writing they write a mix of both.



The image shows the same Arabic sentence written in different fonts

Also Some characters are very confusable with similar character stroke that shown in following table:

Master Stroke	ب	ح	د	ر	س	ص	ط	ع	ف	ل
Similar Characters	ب,ت,ث	ج,ح,خ	د,ذ	ر,ز	س,ش	ص,ض	ط,ظ	ع,غ	ف,ق	ل,ك

The tiny distinction of stroke structure bring challenges for some similar character pairs, such as sad and dad. The difference of sad and dad is the dot that above character dad.

1.2.1 Datasets and Inputs

- We will use Arabic Handwritten Characters Dataset from Kaggle you can find it here.
- The dataset is composed of **16,800** characters written by 60 participants, the age range is between 19 to 40 years, and 90% of participants are right-hand. Each participant wrote each character (from ’alef’ to ’yaa’) ten times.
- The database is partitioned into two sets: a **training set** (13,440 characters to 480 images per class) and a **test set** (3,360 characters to 120 images per class).

- Writers of training set and test set are exclusive.
- Ordering of including writers to test set are randomized to make sure that writers of test set are not from a single institution (to ensure variability of the test set).

1.2.2 Solution Statement

It is clear from dataset description and the problem statement we have a labeled data with 28 classes (Arabic alphabet letters). So:

- we intend to build Handwritten Arabic character recognition system using Classification algorithms and Deep Learning algorithms like CNN to build a strong model able to recognize an Arabic letter from image with high accuracy.

- My solution will use some Classification algorithms and CNN which performs better with images classification problems as they can successfully boil down a given image into a highly abstracted representation which make the prediction process more accurate.

- There are some related kernels which discuss this problem on Kaggle **here**

- To Conclude our input of the model will be an image and the output will be one of the Arabic letters which represents the image.

1.3 Metrics

According to dataset description test set is randomized, shuffled and has a good variability. So we will use accuracy score on test set to evaluate the classifier.

- Accuracy Score is a suitable metric for this dataset because the **test set** is balanced (3,360 characters to 120 images per class). So, no need for more complex metrics.
- The accuracy score function provided by `sklearn.metrics.accuracy_score` .
- The accuracy_score function computes the accuracy, either the fraction (default) or the count (normalize=False) of correct predictions.
- In multi-label classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.
- If $\hat{y}_i$ is the predicted value of the i-th sample and y_i is the corresponding true value, then the fraction of correct predictions over $n_{samples}$ is defined as:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(y = \hat{y})$$ where $1(x)$ is the indicator function.

- For MLP and CNN we will use the same function provided in Keras model API evaluate which Returns the loss value & metrics (accuracy) values for the model in test mode.

II. Analysis

2.1 Data Exploration

The datasets provided as 4 CSV files:

- Train Images : "csvTrainImages 13440x1024.csv"

  13440 Images every image 1024 pixels in gray scale with Max. = 255 and Min. = 0

- Train labels : "csvTrainLabel 13440x1.csv"

  13440 label between 1 and 28 represent Arabic alphabet letters.

- Test Images : "csvTestImages 3360x1024.csv"

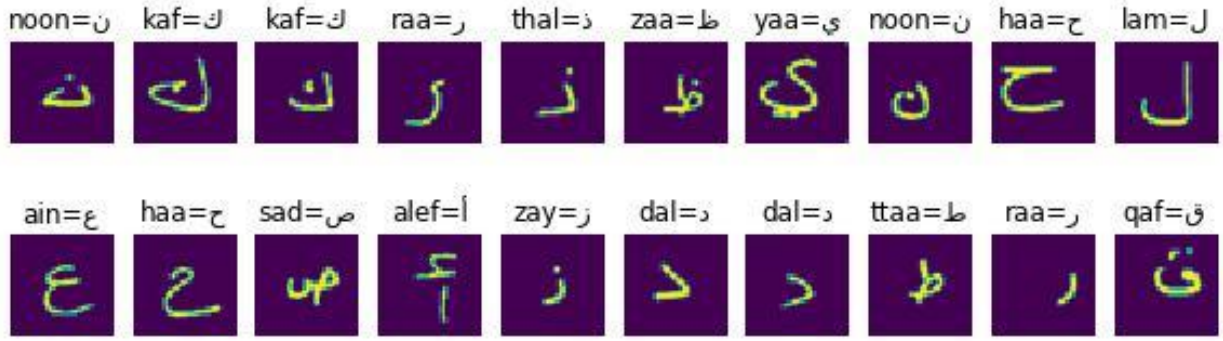
  3360 Images every image 1024 pixels in gray scale with Max. = 255 and Min. = 0

- Test labels : "csvTestLabel 3360x1.csv"

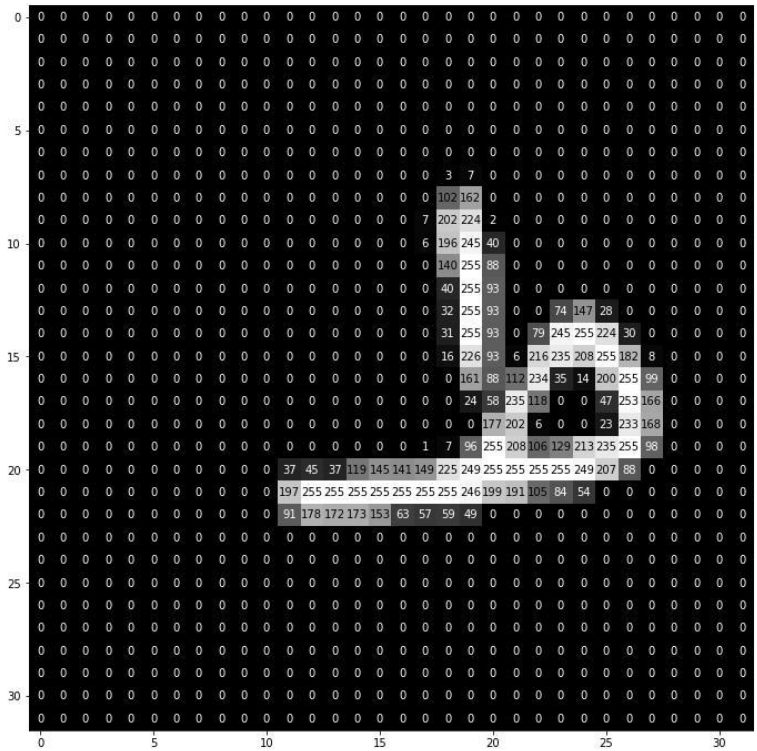
  3360 label between 1 and 28 represent Arabic alphabet letters.

2.2 Exploratory Visualization

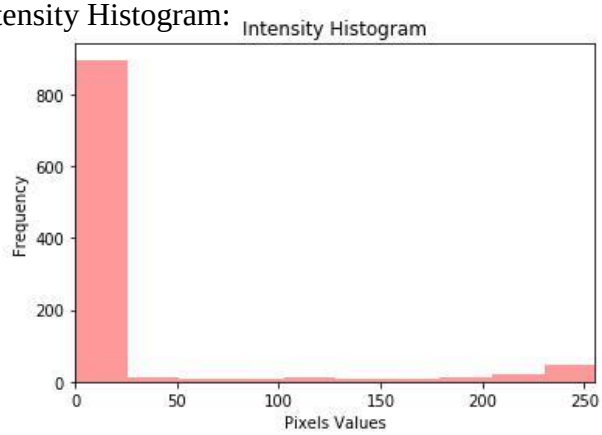
We show random 20 images from Train images after reshaping them to 32 x 32 pixels with showing the letter related to image label in as Arabic letter and how it’s sound using English letters.



Showing an image of them in detail to show the value for each pixel.



Show the image Intensity Histogram:



2.3 Algorithms and Techniques

The project separated to two parts :

- The first part using classic classifiers : Support Vector Machines (SVM), Ensemble Methods (Random Forest, AdaBoost and Decision tree).
- The second part using Deep Learning: MLP and CNN.

2.3.1 Part1:

In this part we discuss the strengths and weaknesses for each algorithm in first part:

1. **Support Vector Machines (SVM)**:
    - SVM constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.
    - The intuition of SVM :
      tries to **maximize the margin** between the hyperplane and the samples. Thanks to that, the decisions it makes, are more accurate. Large distance between the decision boundary and the samples, makes the probability of miss classifying lower. This kind of classifiers are called **large margin classifiers**.
    - Strengths of SVM: can model non-linear decision boundaries with wide margin, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.
    - Weaknesses: memory intensive, trickier to tune due to the importance of picking the right kernel, and don't scale well to larger datasets.
    - The algorithm provided in `sklearn.svm`.SVC
2. **Random Forest Classifier:**

- **Random forests** or **random decision forests** are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (will discuss below) at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set
- Strengths: The model is easy to use. It can very easily handle categorical variables that do not expect linear features or even features that interact linearly. The model also handles high dimensional spaces very well, as well as large numbers of training examples. Finally, it's less likely to overfit than a decision tree.
- Weaknesses: it's more difficult to interpret a Random Forest than a Decision Tree.
- The algorithm provided in `sklearn.ensemble`.RandomForestClassifier

3. **Adaboost**:
   - AdaBoost or **Adaptive Boosting** is a type of "Ensemble Learning" where multiple learners are employed to build a stronger learning algorithm. AdaBoost works by choosing a base algorithm (e.g. decision trees) and iteratively improving it by accounting for the incorrectly classified examples in the training set.
   - We assign equal weights to all the training examples and choose a base algorithm. At each step of iteration, we apply the base algorithm to the training set and increase the weights of the incorrectly classified examples. We iterate n times, each time applying base learner on the training set with updated weights. The final model is the weighted sum of the n learners.
   - Strengths of Adaboost is a powerful classification algorithm that tends to be very adaptive. It can capture very complex decision boundaries. Another advantage, it doesn't require to tweak lot of parameters.
   - Weaknesses: sensitive to noisy data and outliers. It can also be slow to train.
   - The algorithm provided in `sklearn.ensemble`.AdaBoostClassifier
4. **Decision Trees:**
   - **Decision tree learning** uses a decision tree to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).
   - Tree models where the target variable can take a discrete set of values are called **classification trees**; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.
   - **Decision Tree Classifier** breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. The topmost decision node in a tree which corresponds to the best predictor called **root node**.
   - Strengths: perform very well in practice. They are robust to outliers, scalable, and able to naturally model non-linear decision boundaries.

- Weaknesses: if a complex model is used as the base classifier, this can lead to overfitting.
- The algorithm provided in `sklearn.tree`.DecisionTreeClassifier

➢ From previous details it is clear that all four algorithms suitable to work on our data. But using dimensionality reduction will be very useful to increase algorithms speed and performance So we will use PCA as a preprocessing step.

➢ Also to tune the hyper-parameters for the algorithms we will run Grid Search algorithm.

2.3.2 Part two:

In this part we will discuss Deep Learning techniques which are very promised in computer vision field:

➢ Deep Neural networks consist of input layer and multiple nonlinear hidden layers and output layer, so the number of connections and trainable parameters are very large.

➢ The deep neural network needs very large set of examples to prevent over fitting.

➢ One class type of Deep Neural Network with comparatively smaller set of parameters and easier to train is Convolution Neural Network (CNN).

➢ CNN is a multi-layer feed-forward neural network that extract features and properties from the input data (images or sounds). CNN trained with neural network back-propagation algorithm.

➢ CNN have the ability to learn from high-dimensional complex inputs, nonlinear mappings from very large number of data (images or sounds).

➢ The advantage of CNN is that it automatically extracts the salient features which are invariant and a certain degree to shift and shape distortions of the input characters. Another major advantage of CNN is the use of shared weight in convolution layers, which means that the same filter is used for each input in the layer. The share weight reduces parameter number and improves performance.

➢ **Pooling Layer** It is common to periodically insert a Pooling layer in-between successive Convolution layers in a CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

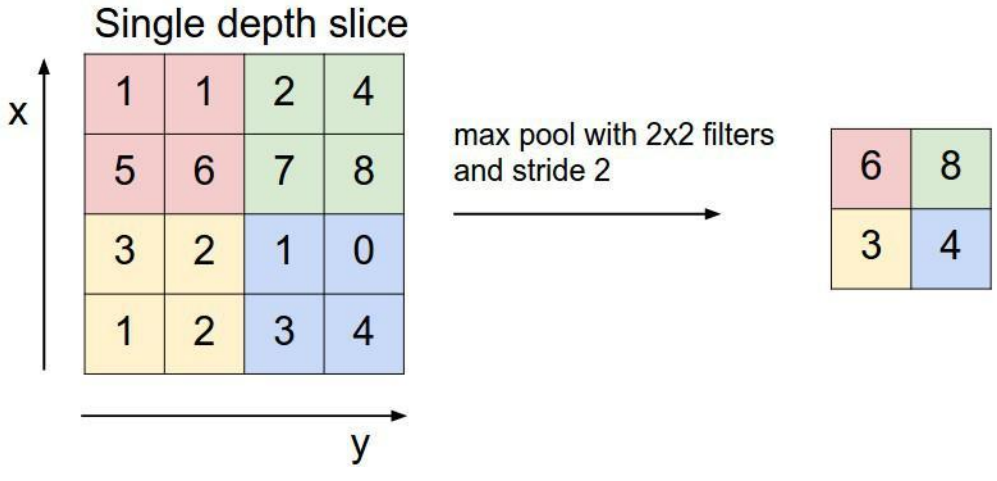


Image show Max Pooling operation

- ➢ Deep Neural Networks will be implemented using Keras The Python Deep Learning library.

2.4 Benchmark

- As mentioned in Metrics we will use accuracy score on test set to evaluate the algorithms.
- The dataset is based on research paper which achieved 94.9% accuracy which also mentioned some **related work** achieved accuracy 77.25%, 93.92%, 73.4%, 93.8%, 93.3% and 90.73%.
- So, we well choose the lowest accuracy mentioned in this paper **73.4%** as a benchmark.

II. Methodology

3.1 Data Preprocessing

As mentioned in Data Exploration the images are in gray scale with pixels value between 0-255.

We have proceeded preprocessing steps to improve quality of the algorithms:

1. **Data scaling**: Dividing every pixel value by 255.0 to change the range between 0-1.
2. **Data Normalization**: to set values with zero mean.
3. **Dimensionality Reduction**: using Principal component analysis (PCA) to improve classic algorithms speed and quality.(This step will skipped with Deep Learning).
4. Encode categorical integer labels using a **One-Hot Encoding** to work with with output layer of Neural Networks.
5. Reshape Input Data from 1024 to 23 x 32 to work with NN input.

3.2 Implementation

All python libraries used in the implementation of the algorithms mentioned in Algorithms and Techniques with links to libraries documentation.

3.2.1 The Classic Algorithms:

We Used grid Search for each algorithm and used the best estimator found by grid search as algorithm model, the hyper-parameters tuned for algorithms are:

1. **SVM: C** Penalty parameter and **Gamma** Kernel coefficient for rbf.

hyper-parameter	Range
C	1e3, 5e3, 1e4, 5e4, 1e5
Gamma	0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1

   Grid search takes much time but the results was great.

2. **Random Forest**: **n_estimators** number of estimators, **max_depth** Trees maximum depth.

hyper-parameter	Range
n_estimators	From 10 to 40 with step 2
max_depth	From 3 to 30 with step 1

   Grid search takes a little bit much time but less than SVM also accuracy was less.

3. **AdaBoost**: **n_estimators** number of estimators.

hyper-parameter	Range
n_estimators	From 10 to 40 with step 2

4. **Decision Tree**: **max_depth** Trees maximum depth.

hyper-parameter	Range
max_depth	From 3 to 30 with step 1

3.2.2 Get Accuracy Function:

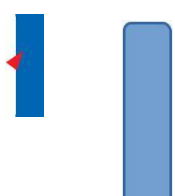
1. get_accuracy function accept 5 attributes Classifier, Training features, Training labels, Testing features, Testing labels.
2. The function fits the classifier on training data, predict labels for testing data and print accuracy score for testing data as percent value and return it.

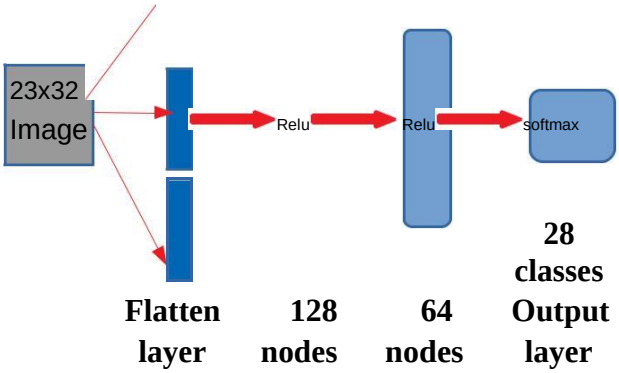
3.2.3 Deep Learning:

In this part we build two deep neural networks MLP and CNN:

MLP Architecture:

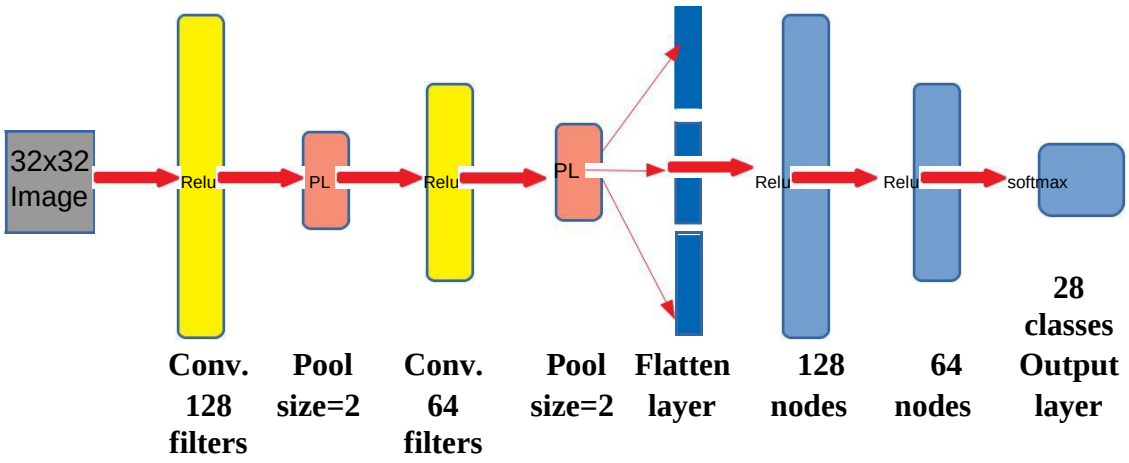
1. **Input layer** followed with flatten layer to set image pixels as a row.
2. **Two hidden layers** the first one with **128 node** and the second with **64 node**.
3. Every layer of the hidden layers with **Relu activation** function and followed by **drop out layer** to prevent overfitting.
4. **Output layer** with **softmax activation** function with **28 output class**.





CNN Architecture:

1. **Input layer** is a **convolution layer** with **128 filters** with **kernel size = 3** and **valid padding** with **Relu** activation function.
2. The second **convolution layer** with **64 filters** with **kernel size = 3** and **same padding** with **Relu** activation function.
3. Every convolution layer followed by **Max Pooling layer** and **Batch Normalization.**
4. The first **convolution layer** followed by **drop out** layer to prevent overfitting which may caused by complexity of many filters.
5. **Fully connected layers** we used the same architecture as in the previous MLP network.



Some complications I had when building the model.:

- How to recognize text in my samples/dataset?
- How to recognize text in lines/sentences?
- How to compute a confidence score for the recognized text?

3.3 Refinement

1. For classic algorithms using grid search give a good results but only **SVM** gives accuracy more than the benchmark with **74% accuracy**.
2. For Deep learning we use many trials of architectures to get a good results and **CNN** model got accuracy more than the benchmark with **83%** accuracy.
3. Adding **Batch Normalization** improved **CNN** accuracy from **78%** to **82%**.
4. Changing **Dropout** ratio from 0.2 to 0.5 improved the accuracy for **86%.**

IV. Results

Model Evaluation and Validation

To evaluate the model:

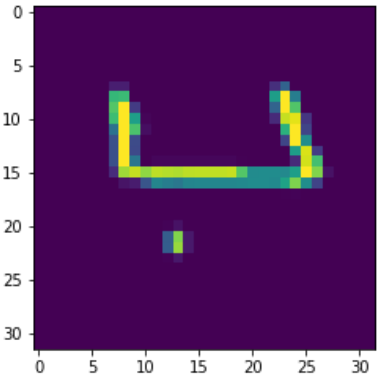
we choose a random 6 images from test set to predict their labels and the model predicted 4 labels correctly.

Another step we create 6 character images using paint and the model predicted 3 labels correctly.

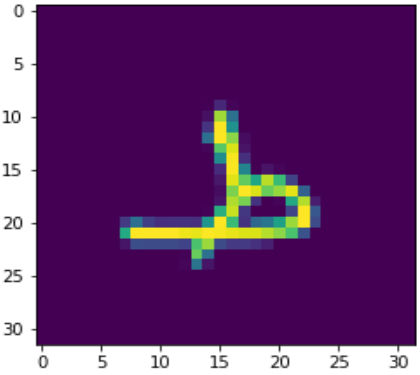
1. As a result, for all trials and testing the CNN model shows the best accuracy with 86.4% even we use a simple architecture.
2. The model passed the benchmark metric.

3. The model could predict 50% of very unseen data correctly.
4. The model is robust enough as personal project but it can be improved to work as a real-world application.

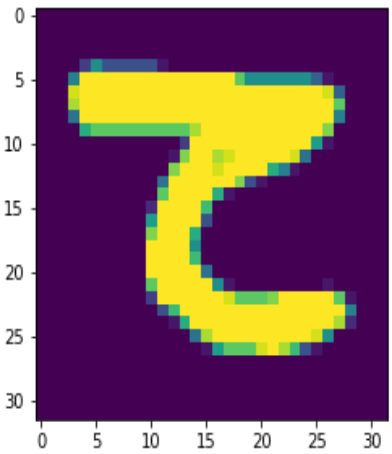
Some images I've used to evaluate my model:

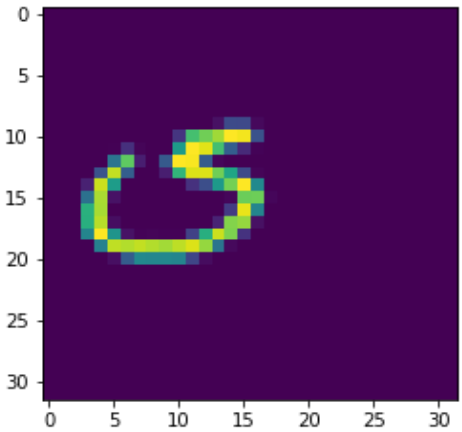


```
This image predicted as
letter  baa  =  ب
And that is right
```



```
This image predicted as
letter noon  =  ن
But sorry that is wrong
```



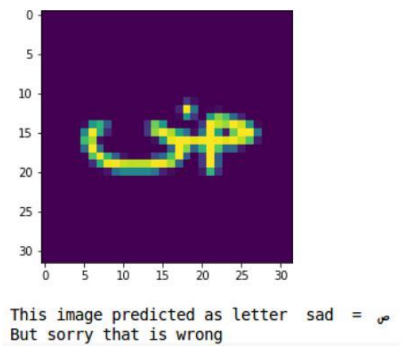


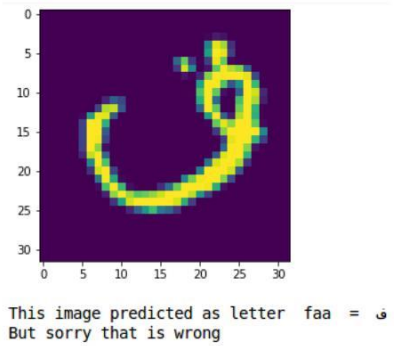
V. Conclusion

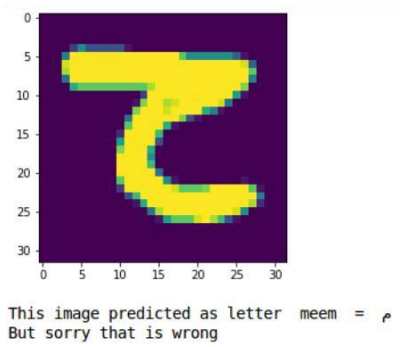
5.1 Free-Form Visualization

To show the model quality we get the images classified incorrectly in test set and result shows the model confused mostly in letters which have the same letter stroke as we mentioned in **Problem Statement**, That may confuse a human if Arabic is not his native language. The result shows in the following table:

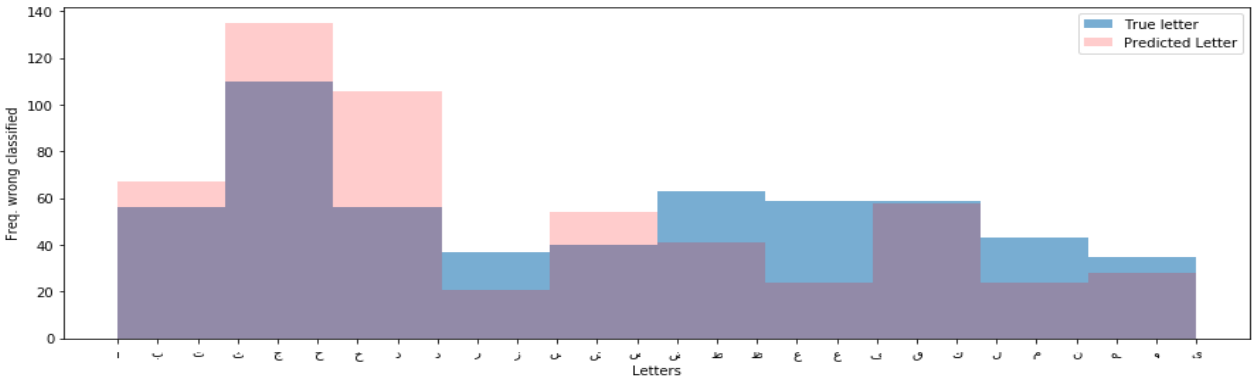
True Label	Predicted Label	True Letter	Predicted letter
Jeem	Haa	ج	ح
Khaa	Haa	خ	ح
Dal	Thal	د	ذ
Raa	Zay	ر	ز
Ttaa	Zaa	ط	ظ
Noon	Lam	ل	ن
Thal	Zay	ذ	ز
Ain	Khaa	ع	خ
Thaa	Taa	ث	ت
Faa	Qaf	ف	ق







Also the following histogram show letters frequently predicted wrong:



5.2 Reflection

Through working on this project we got some of important points to focus:

1. **Data Exploration** and **Data Visualization** are very important steps to start with, That you should to the sea before you dive in.
2. **Preprocessing data** is an important step to improve the model speed and quality.
3. Classic ML Algorithms need Dimensionality Reduction to speed up and get reasonable accuracy But still not very suitable for image classification.
4. **The Deep Convolution Neural Networks** are very promised in Computer Vision and Handwritten Arabic OCR fields. Also Support Vector Machines algorithm can used in this fields as a cheaper solution with reasonable accuracy.
5. Deep Learning is often more art than science and there a lot of tips to deal with network training issues.
6. Handwritten Arabic character recognition systems face several **challenges**, including the unlimited variation in human handwriting and the unavailability of large public databases of handwritten characters and words. The use of synthetic data for training and testing handwritten character recognition systems is one of the possible solutions to provide several variations for these characters and to overcome the lack of large databases. While this can be using arbitrary distortions, such as image noise and randomized affine transformations, such distortions are not realistic. In this work, we model real distortions in handwriting using real handwritten Arabic character examples and then use these distortion models to synthesize handwritten examples that are more realistic.

5.3 Improvement

1. The CNN model can be improved using the power of GPUs (I don't have) by using **augmentation**, **regularization** and **transfer learning**.
2. We could eventually, keep improving our model by adding regularization, tweaking parameters, or applying more advanced CNN but the goal was to prove the concept that TensorFlow can easily be used to program a DNN model to quickly recognize the handwritten digits and convert it to clear digit (class number) with a small effort and high accuracy and without having to extract features. It shows the incredible value of DNN in coming up with model solution and proves the capability of TensorFlow as convenient and fast development framework for deep learning..