



Machine Learning Engineer

Nanodegree Program

Capstone project

Arabic Handwritten Characters Recognition

Mohamed EL-Raghy

Definition

Project Overview

Make a computer to be able to know what is the Arabic characters contained in the photos, This is what I programmed in this project

Problem Statement

Handwritten Arabic character recognition systems face several challenges, including the unlimited variation in human handwriting and large public databases. In this work, we model a deep learning architecture that can be effectively applied to recognizing Arabic handwritten characters.

Metrics

Generating a confusion matrix

for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

Analysis

Data Exploration & Exploratory Visualization

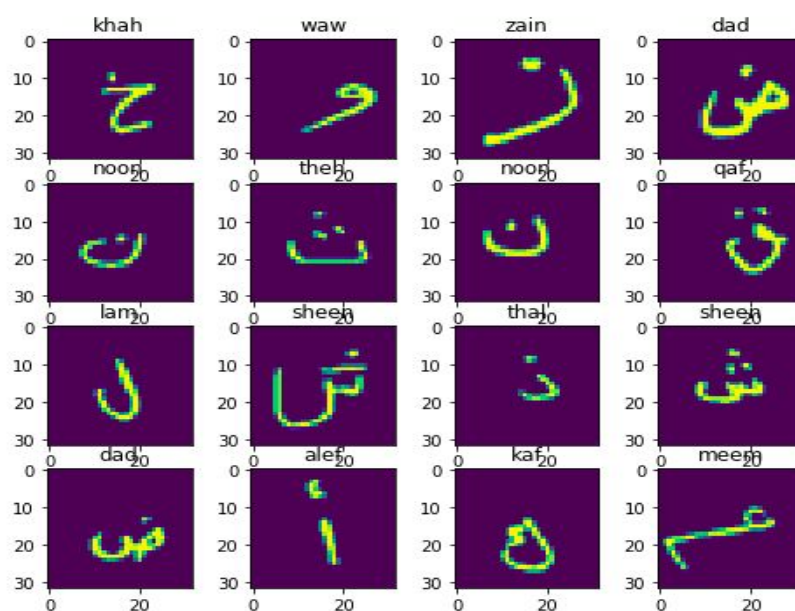
The data-set is composed of 16,800 characters written by 60 participants, the age range is between 19 to 40 years, and 90% of participants are right-hand. Each participant wrote each character (from 'alef' to 'yeh') ten times on two forms.

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

Kaggle is the dataset source: [and we could download it from this link](#)

```
In [55]: display_image(train_data, train_label)
```



Algorithms and Techniques

This classifier is a Convolution Neural Network, which is the state-of-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches, fortunately, the dataset is big enough the algorithm outputs an assigned probability for each class.

The following parameters can be tuned to optimize the classifier:

- Training parameters
 - Training length (number of epochs)
 - Batch size (How many images to look at once during a single training step)
 - Learning rate (how fast to learn, this can be dynamic)
- Neural network architecture
 - Number of layers
 - Layer type (convolutional, fully connected, or pooling)

Benchmark

In an experimental section, we showed that the results were promising with a 94.9% classification accuracy rate on testing images. In future work, we plan to work on improving the performance of handwritten Arabic character recognition.

[Miss-Classification & correct-Classificationrate and number of wrongs and correct recognition](#)

Methodology

Data Preprocessing

- Data reshape
- Encoding categorical variables
- Normalization
- Scaling

Implementation

The implementation process can be divided into:

1. Loading data and preprocessing it
2. Building the CNN and training it

1. Loading data and preprocessing it

The dataset is split into four files:

1. csvTrainImages 13440x1024.csv
2. csvTestImages 3360x1024.csv
3. csvTrainLabel 13440x1.csv
4. csvTestLabel 3360x1.csv

I load it into pandas DataFrame then convert them to numpy arrays

Using Numpy to do some math on the data.

The second stage is preprocessing the data performed by :

- **Reshape the data** to make the data represent a 2D image.
- **Encoding categorical variables:**

There is no meaning in performing operations on a number representing a category. So, categorical encoding needs to be done.

- **Normalization**

Normalization is done to bring the entire data into a well-defined range, preferably between 0 and 1

In neural networks, it is a good idea not just to normalize data but also to scale them. This is intended for faster approaching to global minima at the error surface.

2. Building the CNN and training it

the architecture of the CNN model :

- I will use 4 Conv2D layers the first two with 64 nodes and the second two with 32 nodes I guess it works well
- The activation function we will be using for layers is the ReLU
- In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers
- 'Dense' is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks.

A summary of the CNN

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	832
conv2d_2 (Conv2D)	(None, 32, 32, 32)	25632
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_4 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_2 (Dropout)	(None, 8, 8, 64)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_1 (Dense)	(None, 256)	1048832
dense_2 (Dense)	(None, 256)	65792
dropout_3 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 28)	7196
Total params: 1,203,708		
Trainable params: 1,203,708		
Non-trainable params: 0		

Optimizer for the CNN: The optimizer used here is an RMSprop,

Then fitting the CNN with 30 epoch

Refinement

As I mentioned in Benchmark Model In an experimental section, we showed that the results were promising with a 94.9% classification accuracy rate on testing images, This is improved using the following

- Increasing the number of CNN layers and use convolutional, max pooling and flatten
- Use relu and softmax activation function

- Optimize the CNN using RMSprop

This lead to having an accuracy greater than the mentioned before

Calculating the accuracy

```
In [32]: accuracy = sum(cm[i][i] for i in range(28)) / test_label.shape[0]
          print("accuracy = %.3f%%" %(accuracy * 100))
          accuracy = 97.738%
```

Results

Model Evaluation and Validation

The final architecture is chosen because they preformed the best results among the tried combination (the architecture in jupyter notebook)

The complete description of the final model:

- The kernel size of the first two convolution layers is $5 * 5$
- The first two convolutional layers learned 32 filters and they use a relu activation function
- The max pool layer has a pool size of $2 * 2$
- The dropout rate is 0.25
- The kernel size of the third and the fourth convolution layers is $3 * 3$
- The third and the fourth convolution layers learned 64 filters, using a relu activation function
- The max pool layer has a pool size of $2 * 2$, strides = (2, 2)
- The dropout rate is 0.25

- The first two dense layers have 256 units and use a relu activation function
- The dropout rate is 0.5
- The last dense layers have 28 units and use a softmax activation function

This model can detect Arabic character in 32*32 photos

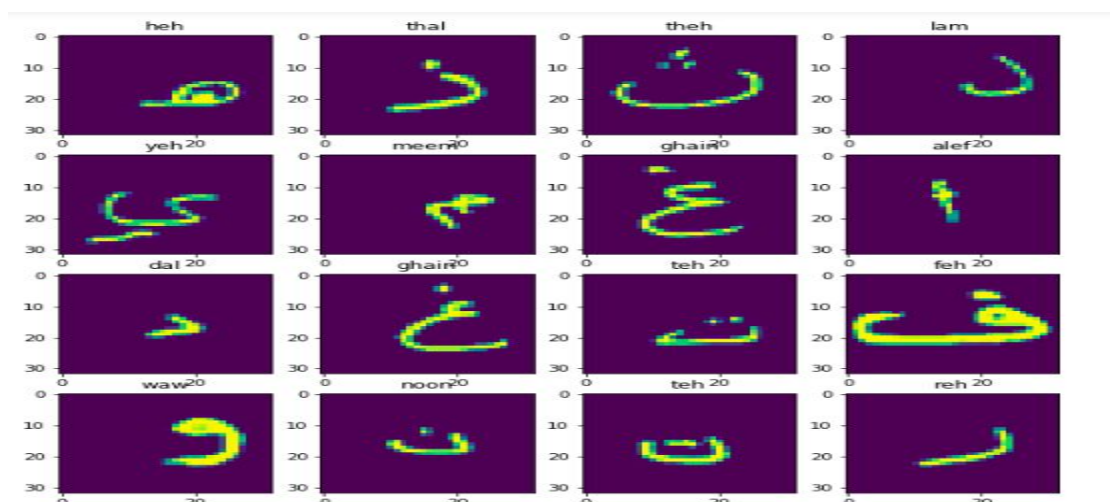
Justification

After I train the Network I have an accuracy of 97.738% which is greater than the accuracy presented in Benchmark model (94.9%)

Conclusion

Free-Form Visualization

The model perfectly predicts the label of the Arabic characters



Reflection


The process used for this project can be summarized using the following steps :

1. An initial problem and relevant, public dataset were found
2. The dataset was downloaded and preprocessing
3. The benchmark model was created for the model
4. The classifier was trained using the data (multiple time, until a good set of parameters were found)

I found step 4 is the most difficult for me and the interesting thing I found the dataset need to be transposed

Improvement

The future work is to use the Neural Language Processing to use text instead of characters



It will be more useful to predict and understand text rather than characters such as:

- classifying the message is spam or not
- Analysis of the posts and comments to the people in social media and build a recommendation system according to this analysis