Machine Learning Engineer Nanodegree

Capstone Project
A-Z Handwritten Character Recognizer

Mohammed Salah El-Ghrably

December 1st, 2019

Definition

Project Overview

My project's aim is to program the computer to identify hand-written alphabets via matrix operations. Each alphabet image contains 28*28 pixels, and we create a matrix using these pixels. By multiply the matrix to several sample matrixes, the pixels are converted into a deep neural network. And finally, we Adam method so that the computer can predict the highest possibility of the alphabet written.

Keyword: deep learning, ANN, Feature Extraction, CNN, English, Machine Recognition, natural and physical sciences, image data, image processing

Domain Background

Character recognition is one of the most important research fields of image processing and pattern recognition. Character recognition is generally known as Optical Character Recognition (OCR).OCR is the process of electronic translation of handwritten images or typewritten text into machine editable text. It becomes very difficult if there are lots of paper based information on companies and offices. Because they want to manage a huge volume of documents and records. Computers can work much faster and more efficiently than human. It is used to perform many of the tasks required for efficient document and content management. But computer knows only alphanumeric characters as ASCII code. So computer cannot distinguish character or a word from a scanned image. In order to use the computer for document management, it is required to retrieve alphanumeric information from a scanned image. There are so many methods which are currently used for OCR and are based on different languages. The existing method like Artificial Neural Network (ANN) based on English Handwritten character recognition needs the features to be extracted and also the performance level is low. So a Convolutional Neural Network (CNN) based English handwritten character recognition method is used. It's a deep machine learning method for which it doesn't want to extract the features and also a fast method for character recognition. My personal motivation is that I Have faced a problem, where I wanted to transform a paper

document into a digital one, but i had to type it character by character. I have been trying to find an easy way to solve the problem, simply The purpose of our project is to recognize hand-written alphabets, so the computer can automatically identify the characters without any manual input.

The link to my data source is:

https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format

Problem Statement

The main objective of this research is to find a new solution for handwritten text recognition of different fonts and styles by improving the design structure of the traditional Artificial Neural Network (ANN). ANNs have been successfully applied to pattern recognition, association and classification, forecast studies, and control applications, to name a few. The recognition results of such text or handwritten materials are then fed into Optical Character Recognition (OCR) as an electronic translation of images of handwritten, typewritten or printed text into machine-editable text. OCR is a field of research that is fully developed and has been quite useful in pattern recognition, artificial intelligence and machine vision. Consequently, typewritten text recognition that is void of any distortions is now considered largely a solved problem. However, the direct use of OCR on handwritten characters remains a very difficult problem to resolve, yielding extremely low reading accuracy. handwritten document recognition is currently a difficult problem; as different people have different handwriting styles. Scanning, segmentation and classification are the general processes that are being used to recognize handwritten documents. ANNs have proven to be excellent recognizers of printed characters and 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.

It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

	Actual = Yes	Actual = No
Predicted = Yes	TP	FP
Predicted = No	FN	TN

Analysis

Data Exploration

The A_Z Handwritten Data contains capitalized handwritten alphabet images (A-Z) in size of 28x28 pixels. Each alphabet in the image is centered at 20x20 pixel box. There are 372451 images in total, or approximately 14325 images for each of the alphabet, in the data file. The

dataset contains 2 .csv files with information necessary to make a prediction. The images are taken from NIST

(https://www.nist.gov/srd/nist-special-database-19)

```
print("shape:",X.shape)
print("culoms count:",len(X.iloc[1]))
print("784 = 28X28")

X.head()
```

```
shape: (372450, 784)
culoms count: 784
784 = 28X28
```

|	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	...	0.639	0.640	0.641	0.642	0.643	0.644	0.645	0.646	0.647	0.648
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 784 columns

Exploratory Visualization

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).

```
In [4]: from sklearn.utils import shuffle

        X_shuffle = shuffle(X)

        plt.figure(figsize = (12,10))
        row, colums = 4, 4
        for i in range(16):
            plt.subplot(colums, row, i+1)
            plt.imshow(X_shuffle.iloc[i].values.reshape(28,28),interpolation='nearest', cmap='Greys')
        plt.show()
```
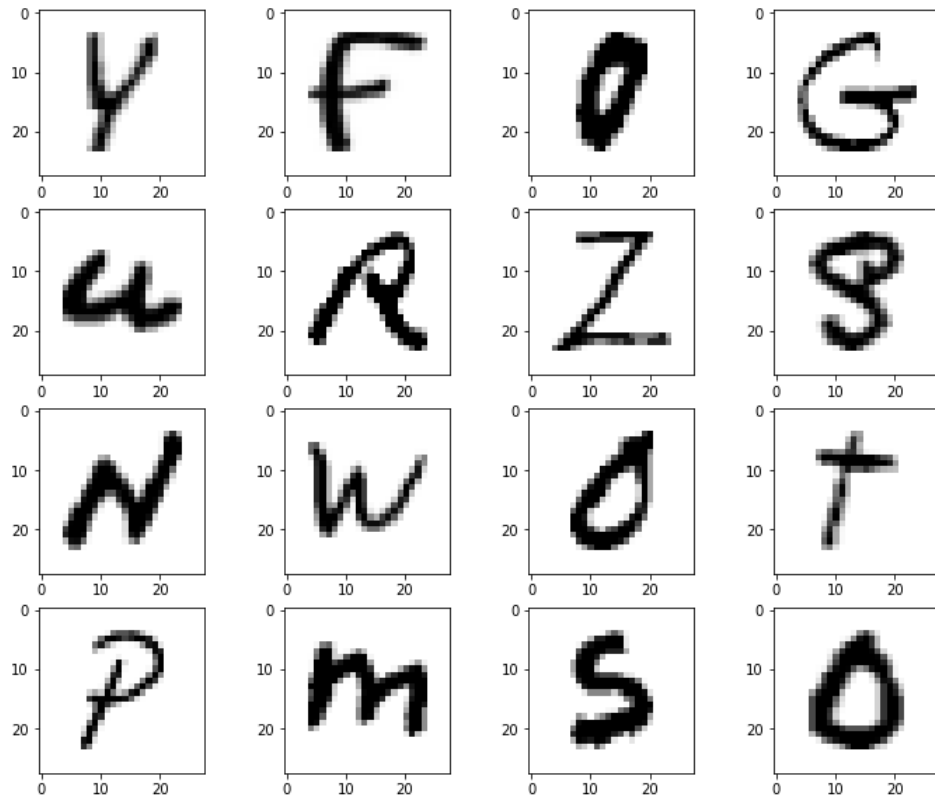
Algorithms and Techniques

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:

I. 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)
II. Neural network architecture
    - Number of layers
    - Layer type (convolutional, fully connected, or pooling)

Benchmark

As a kaggle data set, my benchmark model showed that the results were promising with a 98% classification accuracy rate on testing images. I plan to work on improving the performance of handwritten character recognition. https://www.kaggle.com/chandanshinde/character-recognition-using-keras

Methodology

Data Preprocessing

In order to achieve maximum efficiency, we need to preprocess the data before we feed it into the neural network. first, I split the data into training and test data then scale it; it's important to scale the data not to have a dominate variables as we know from the Euclidean distance concept.

Once we got explore the dataset we preprocess it by:

1. Step 1: Import Libraries. First step is usually importing the libraries that will be needed in the program. ...
2. Step 2: Import the Dataset. ...
3. Step 3: Taking care of Missing Data in Dataset. ...
4. Step 4: Encoding categorical data. ...
5. Step 5: Splitting the Dataset into Training set and Test Set. ...
6. Step 6: Feature Scaling.

in the code below **keras.utils.to_categorical()** is used to convert the number, range from 0-25, into the one-hot array

```
In [9]: X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
        X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')

        y_train = np_utils.to_categorical(y_train)
        y_test = np_utils.to_categorical(y_test)
```

In the code above, we first use the shuffle() method to randomize the data (the original one is sorted from A-Z, and it is not machine-friendly). Then, we extract the label, the number indicating which letter is this row representing, from the raw data. keras.utils.to_categorical() is used to convert the number, range from 0-25, into the one-hot array.

Implementation

1. Import libraries

```
In [1]:  import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import seaborn as sns
         import os
         from sklearn.preprocessing import MinMaxScaler
         import tensorflow as tf
         from keras.models import Sequential
         from keras.layers import Dense
         from keras.layers import Dropout
         from keras.layers import Flatten
         from keras.layers.convolutional import Conv2D
         from keras.layers.convolutional import MaxPooling2D
         from keras import backend as K
         from keras.utils import np_utils
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import confusion_matrix

         Using TensorFlow backend.
```

2. Importing Dataset

well, The Dataset contains capitalized handwritten alphabet images (A-Z) in size of 28x28 pixels. Each alphabet in the image is centered at 20x20 pixel box. There are 372451 images in total, or approximately 14325 images for each of the alphabet, in the data file.

```
In [2]:  dataset = pd.read_csv("A_Z Handwritten Data.csv").astype('float32')
         print ("tamam") #don't worry it means okay in my mother tounge

         dataset.rename(columns={'0':'label'}, inplace=True)

         #Splite data the X - Our data , and y - the prdict label
         X = dataset.drop('label',axis = 1)
         y = dataset['label']
         print("tamam")

         tamam
         tamam
```

3.Data Exploration

```
In [3]: print("shape:",X.shape)
        print("culoms count:",len(X.iloc[1]))
        print("784 = 28X28")

        X.head()

        shape: (372450, 784)
        culoms count: 784
        784 = 28X28
```

Out[3]:

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	...	0.639	0.640	0.641	0.642	0.643	0.644	0.645	0.646	0.647	0.648
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 784 columns

- Data Image visualization
- changing labels to alphabets
  I'm filtering the data frame by label frequencies or characters now, using the famous groupby() method .

```
In [5]: print("Amount of each labels")

        # Change Label to alphabets
        alphabets_mapper = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'J',10:'K',11:'L',12:'M',1
        3:'N',14:'O',15:'P',16:'Q',17:'R',18:'S',19:'T',20:'U',21:'V',22:'W',23:'X',24:'Y',25:'Z'}
        dataset_alphabets = dataset.copy()
        dataset['label'] = dataset['label'].map(alphabets_mapper)

        label_size = dataset.groupby('label').size()
        label_size.plot.barh(figsize=(10,10))
        plt.show()
```

Until we got a conclusion: -

```
In [6]: print("We have very low observations for I and F ")
        print("I count:", label_size['I'])
        print("F count:", label_size['F'])

        We have very low observations for I and F
        I count: 1120
        F count: 1163
```

4.Data preprocessing

5.Building and Training the model

First, we build the model. Then, we use the fit() method to train the network.

Optimizer for the CNN: Adam, with 18 epoch

```
In [10]: cls = Sequential()
         cls.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1), activation='relu'))
         cls.add(MaxPooling2D(pool_size=(2, 2)))
         cls.add(Dropout(0.3))
         cls.add(Flatten())
         cls.add(Dense(128, activation='relu'))
         cls.add(Dense(len(y.unique()), activation='softmax'))

         cls.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
         history = cls.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=18, batch_size=200, verbose=
         2)
```

and here is what we got:

```
Train on 279337 samples, validate on 93113 samples
Epoch 1/18
 - 31s - loss: 0.2370 - acc: 0.9342 - val_loss: 0.0952 - val_acc: 0.9742
Epoch 2/18
 - 27s - loss: 0.0857 - acc: 0.9759 - val_loss: 0.0755 - val_acc: 0.9784
Epoch 3/18
 - 27s - loss: 0.0652 - acc: 0.9814 - val_loss: 0.0571 - val_acc: 0.9840
Epoch 4/18
 - 27s - loss: 0.0526 - acc: 0.9848 - val_loss: 0.0479 - val_acc: 0.9869
Epoch 5/18
 - 27s - loss: 0.0440 - acc: 0.9869 - val_loss: 0.0435 - val_acc: 0.9881
Epoch 6/18
 - 28s - loss: 0.0372 - acc: 0.9888 - val_loss: 0.0424 - val_acc: 0.9884
Epoch 7/18
 - 27s - loss: 0.0316 - acc: 0.9904 - val_loss: 0.0433 - val_acc: 0.9883
Epoch 8/18
 - 27s - loss: 0.0270 - acc: 0.9913 - val_loss: 0.0383 - val_acc: 0.9905
Epoch 9/18
 - 27s - loss: 0.0235 - acc: 0.9925 - val_loss: 0.0359 - val_acc: 0.9906
Epoch 10/18
 - 27s - loss: 0.0215 - acc: 0.9930 - val_loss: 0.0365 - val_acc: 0.9906
Epoch 11/18
 - 27s - loss: 0.0188 - acc: 0.9939 - val_loss: 0.0346 - val_acc: 0.9915
Epoch 12/18
 - 28s - loss: 0.0179 - acc: 0.9941 - val_loss: 0.0363 - val_acc: 0.9911
Epoch 13/18
 - 27s - loss: 0.0158 - acc: 0.9947 - val_loss: 0.0335 - val_acc: 0.9925
Epoch 14/18
 - 27s - loss: 0.0148 - acc: 0.9950 - val_loss: 0.0377 - val_acc: 0.9915
Epoch 15/18
 - 27s - loss: 0.0137 - acc: 0.9954 - val_loss: 0.0337 - val_acc: 0.9925
Epoch 16/18
 - 27s - loss: 0.0131 - acc: 0.9956 - val_loss: 0.0349 - val_acc: 0.9924
Epoch 17/18
 - 27s - loss: 0.0121 - acc: 0.9959 - val_loss: 0.0339 - val_acc: 0.9931
Epoch 18/18
 - 27s - loss: 0.0116 - acc: 0.9960 - val_loss: 0.0329 - val_acc: 0.9934
```

Refinement

As a kaggle data set, my benchmark model showed that the results were promising with a 98% classification accuracy rate on testing images. I plan to work on improving the performance of handwritten character recognition.

but Increasing the number of CNN layers and use convolutional, using max pooling and flatten, using relu and softmax activation function and Optimizing the CNN using Adam, all together did improve the accuracy

```
In [11]: scores = cls.evaluate(X_test,y_test, verbose=0)
         print("CNN Score:",scores[1])

         CNN Score: 0.993373642772
```

booom, we got a 99.4% acc - let's visualize it

```
In [12]: plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('Model loss')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Test'], loc='upper left')
         plt.show()
```
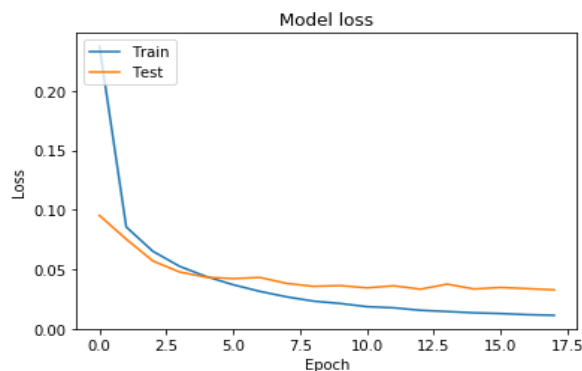

Results

Model Evaluation and Validation

The final architecture is chosen because they performed 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 0.3.
- The first two dense layers have 128 units and use a relu activation function.
- The last dense layers have28 units and use a softmax activation function.

This model can detect Latin character in 28*28 photos
- The model generalizes well to unseen data it's predicted the label perfectly.
- The model didn't affect with small changes in the data, or to outliers because of scaling of the data between values 0 to 1.
- We can trust in the model because of it us a high accuracy after fitting the Neural Network.

Justification

After I train the Network I have an accuracy of 99.4% which is greater than the accuracy presented in the Benchmark model (98%).

When I create the first model I get bad accuracy and try several times with different parameters, so I increased the epochs from 10 t0 18 also I increased the layer to make it deeper so that's enough to have a good model. I changed the optimizer from the descent to Adam to get more high accuracy

This architecture solves the proposed problem

Conclusion

Free-Form Visualization

```
In [1]: from keras.models import load_model
        new_model_2 = load_model('my_model.h5')
        new_model_2.summary()
```

```
Using TensorFlow backend.

_____
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 32)        832
_____
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_____
dropout_1 (Dropout)          (None, 12, 12, 32)        0
_____
flatten_1 (Flatten)          (None, 4608)              0
_____
dense_1 (Dense)              (None, 128)               589952
_____
dense_2 (Dense)              (None, 26)                3354
=================================================================
Total params: 594,138
Trainable params: 594,138
Non-trainable params: 0
_____
```

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)

Improvement

It will be more useful to classify words or even paragraphs rather than characters (e.g. classify email spams)