Machine Learning Engineer Nanodegree

Capstone Project

Quran Reciter Identification

Waleed Algadhi

December 31st, 2050

I. Definition

Project Overview

Quran, the holy book of Islam, is recited by an enormous number of reciters around the world, each having their own unique voice and way of reciting. Naturally,
listeners to Quran prefer some reciters over others. And often, when listening to Quran radio station or to a Quran recitation clip posted on social networks, the reciter is not referenced. We might like his recitation and would like to listen to more of his recitations, but we do not know who he is.

To tackle this issue, I developed this model which is designed to take a five seconds recording of any recitation and tell who is the reciter.

The algorithm was tested on recitations of 50 different reciters, which were downloaded from Quran2y website[1], and proved its ability to predict the correct reciter more than 86% of the time.

Problem Statement

In this project, I will build a model that identifies a Quran reciter's name given a five seconds audio clip from his recitation. The model is supposed to run live on a server receiving queries in real-time and therefore time efficiency is a concern.

To build such a model, I will follow these steps:

1. Data acquisition: finding adequate sound clips for numerous Quran reciters and unify each

reciter’s label.
2. Data preprocessing: segment all sound clips into 25 milliseconds. Since there is going to be a huge amount of small segmented clips for each reciter, a random sample is going to be taken for each reciter.
3. Feature extraction: Mel-frequency cepstrum algorithm is going to be used to extract sound features from each 25 milliseconds segmented clip to feed them into the classifier.
4. Training: different types and architectures of artificial neural networks are going to be used for training to explore which ones are going to produce better results. A validation set is going to be provided to monitor the performance of the algorithm and make sure it does not overfit.
5. Testing: last step is to apply the chosen algorithm on the testing set and measure its performance using the metric specified below.

An attempt to tackle the same problem was conducted before (Asda et al., 2016)[2]. However, it was tested on a small scale (5 reciters) while here I am going to test my model on a bigger dataset (50 reciters).

Metrics

The model will be evaluated on the accuracy metric which is going to be applied on the testing set predictions. Accuracy score can be mathematically defined as:

Accuracy = Number of true predictions / Number of all predictions

To evaluate the model, every sound clip is labeled with its reciter name and those labels are going to be compared to the labels predicted by the model to compute the accuracy score.
Note that accuracy is applicable to this model because labels are balanced in distribution.

II. Analysis

Data Exploration

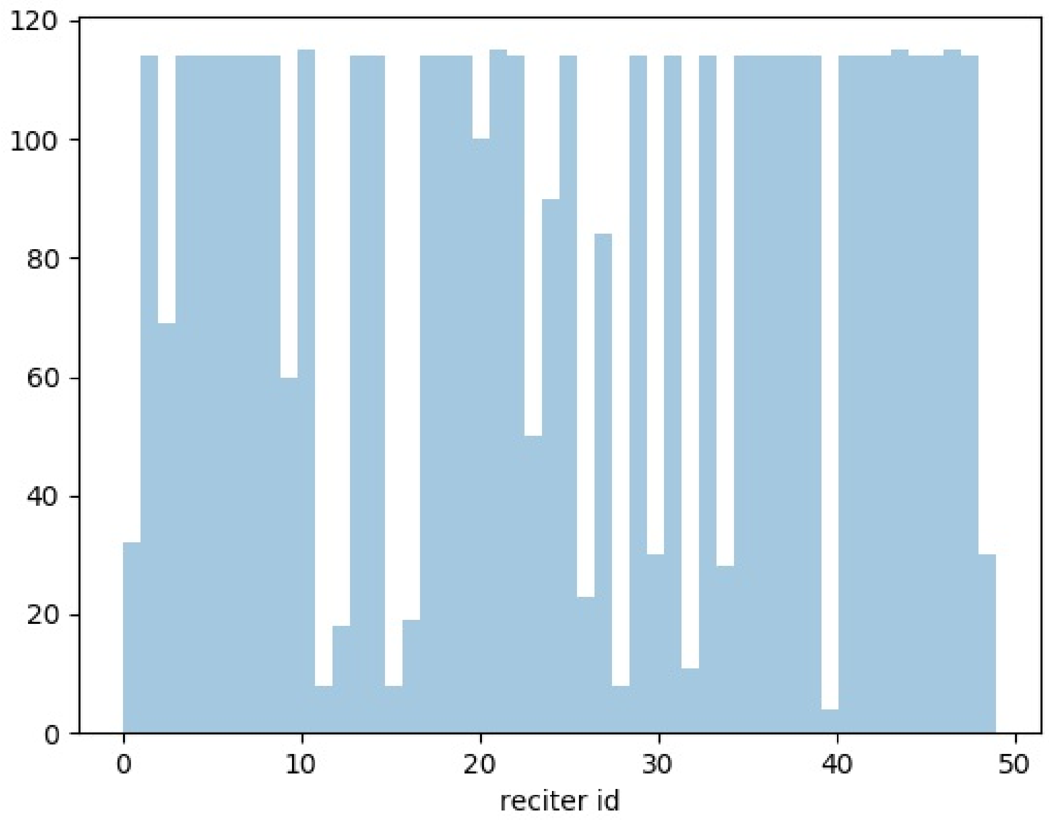
Following are general information and statistics about the dataset:

- The dataset consists of 50 folders downloaded from Quran2y website[1]. Those folders' names represent the target labels.
- Each file contains numerous recitation clips of the same reciter.
- Recitation clips differ in terms of signal rate and vary between 16kHz and 44kHz.
- All recitation clips are in mp3 format.
- No abnormalities have been found in the dataset. However, file naming is not consistent and comes in multiple languages, an issue that needs resolved in preprocessing.
- Descriptive statistics for the dataset are in the table below

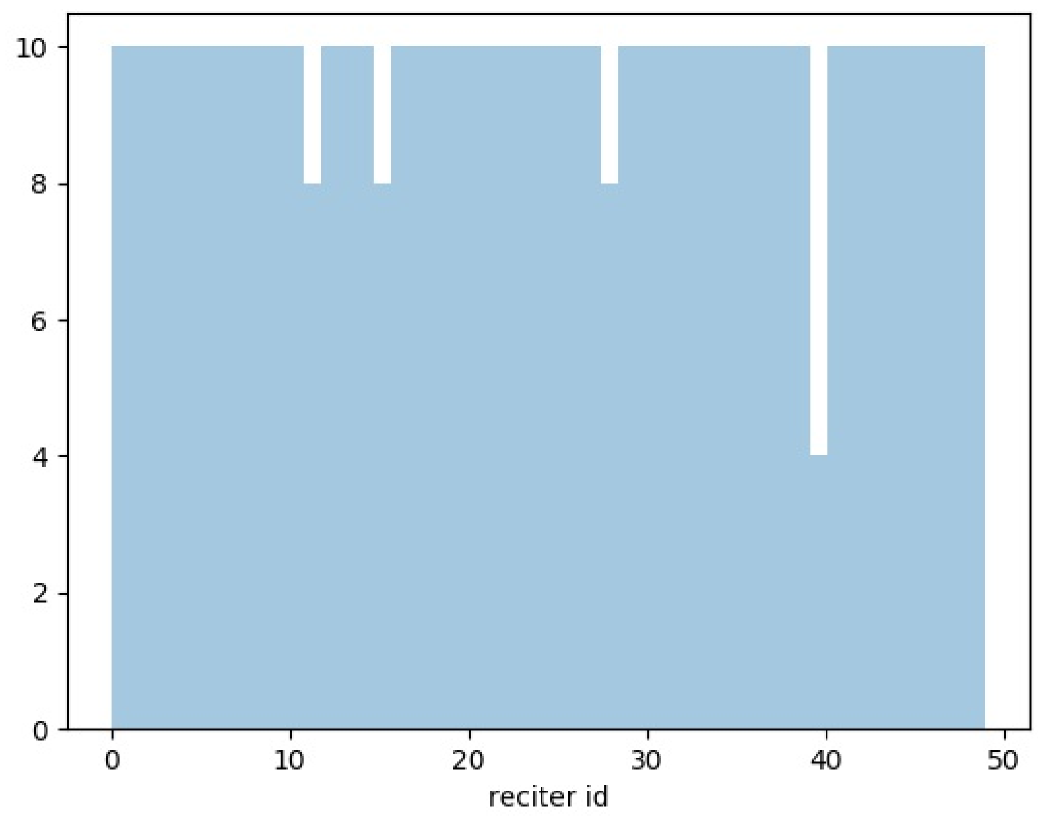
Total number of reciters	50
Total number of recitations	4438
Max number of recitations per reciter	115
Min number of recitations per reciter	4
Mean number of recitations per reciter	86.48
Median number of recitations per reciter	114
Max length of recitation clips (in minutes)	123.43
Min length of recitation clips (in minutes)	0.23
Mean length of recitation clips (in minutes)	13.20
Median length of recitation clips (in minutes)	6.36

Exploratory Visualization

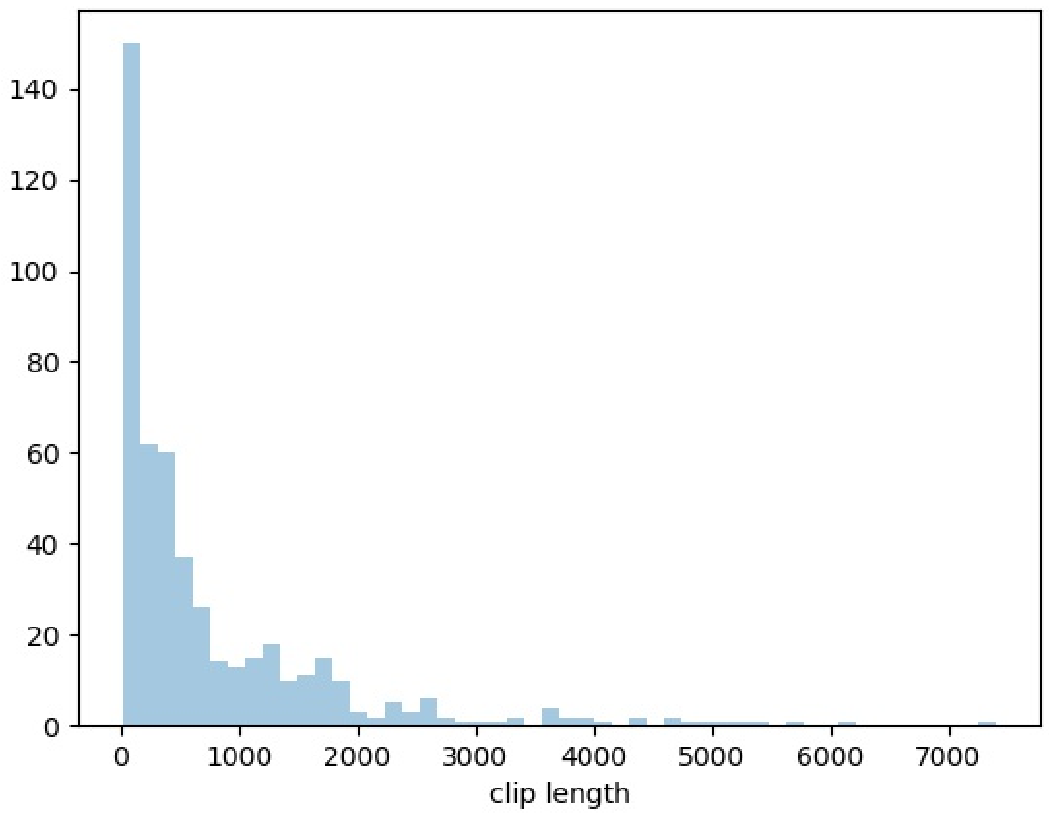
The dataset includes number of recitations available for each reciter. The following figure shows a histogram for the number of recitations available for each reciter.



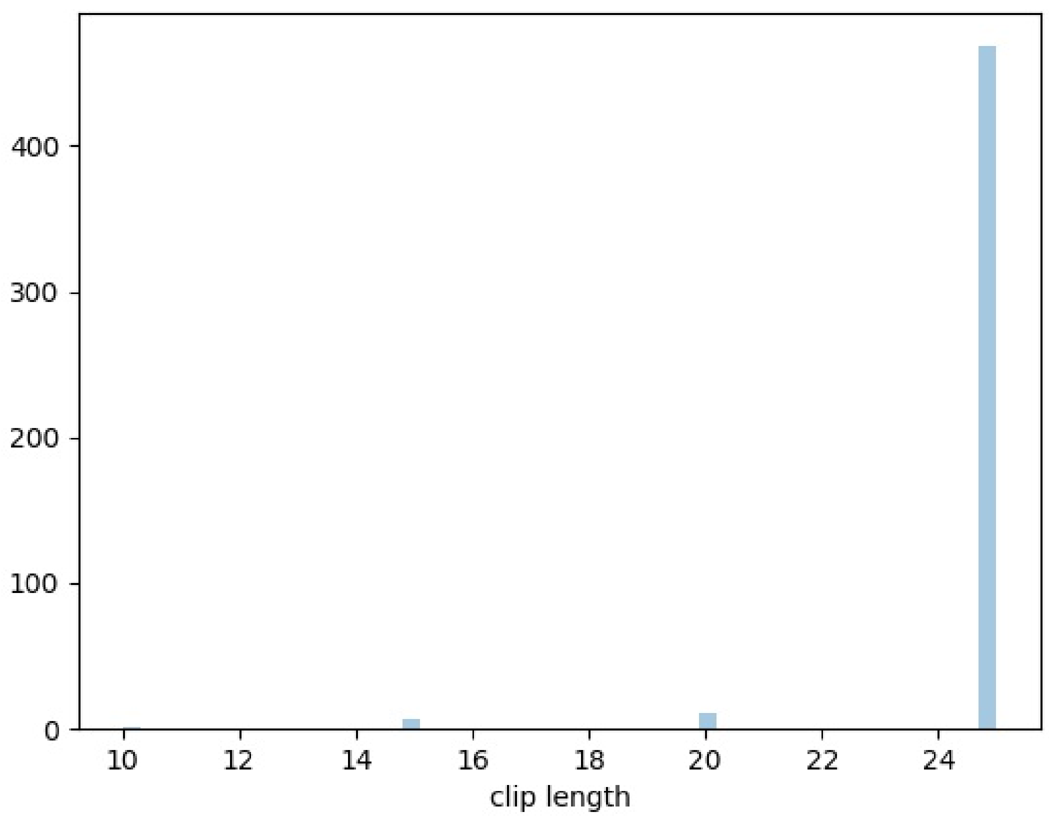
It is apparent that data is not equally distributed among reciters and, consequently, a benchmark metric like accuracy is not the best fit for this kind of imbalanced datasets. However, the algorithm does not need to train on the whole dataset. Taking only a sample of maximum 10 recitations for each reciter is sufficient for the algorithm to produce remarkable results and will bring balance into the dataset. The following figure shows a histogram for the number of recitation samples taken from each reciter.



Also, not all recitation clips are of the same length which will, again, introduce some imbalance to the dataset. The following figure shows a histogram for clip lengths (in seconds) in the dataset.



However, and for the same reason above, a random sample of 5 5-seconds short clips taken from each recitation clip is enough to capture the acoustic fingerprint of the reciter's voice in that clip. The following figure shows a histogram for clip lengths (in seconds) after sampling them.



Algorithms and Techniques

Two model will be include two stages in order to infer the reciter from the recitation clip.

- Feature extraction: in this stage, Mel-frequency cepstrum algorithm *(MFCC)* will be used to extract sound features from recitation clips. Mel-frequency cepstrum algorithm works as following[3]:

  i. Frames the sound clip into short frames (default: 25 milliseconds).
  ii. Compute Discrete Fourier Transform *(DFT)* for each frame to transform the signal from time domain to frequency domain.
  iii. Compute the Mel-spaced filterbank. It is a set of filter bins (default: 26) that tell how much energy exists in various frequency regions. This is motivated by human hearing, human cochlea can not discern the difference between two closely spaced frequencies
  iv. Compute the logarithm of all filterbank energies. This is also motivated by human hearing: we don't hear loudness on a linear scale. Generally to double the perceived volume of a sound we need to put 8 times as much energy into it.
  v. Compute the Discrete Cosine Transform *(DCT)*. Because filterbanks are all overlapping, the filterbank energies are quite correlated with each other. Discrete Cosine Transform decorrelates the energies and produces 26 coefficients, only some of them are kept

(default: 13).

vi. Coefficients from different 25 milliseconds frames will we stacked and returned in a 2D array.

- Classification: in this stage, sequences of MFCC coefficients will be fed to a Gated Recurrent Unit *(GRU)* along with their labels (reciters names). Gated recurrent units are a gating mechanism in recurrent neural networks that have additional parameters that control when and how their memory is updated to capture both long and short term dependencies in sequences. The main components that differentiate Gated Recurrent Neural Networks from ordinary Recurrent Neural Networks are the following[4]:

  i. Update gate: the input at each unit is passed to an update gate that helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. That is powerful because the model can decide to maintain all the information from the past.
  ii. Reset gate: in each unit, this gate is used by the model to decide how much of the past information to forget.

  I chose Recurrent Neural Network, for the classification task because it is one of few algorithms that can learn from temporal inputs competently and catch the relation between the preceding and following time steps in a sequence.

Benchmark

An attempt to tackle the same problem was conducted before (Asda et al., 2016)[2]. Authors report they achieved an accuracy of 91% using a dataset that contains sound clips for 5 different reciters.

However, in section 4.2 it seems that their evaluation was on training, validation and testing set all together (?!).

Consequently, results from this paper cannot be an accurate benchmark for my model. Therefore, I used a simple logistic regression with the default parameters of sklearn library (0.19.1) as my benchmark model on the same training and testing sets. the Accuracy score on this model was 0.53 and this is going to be my benchmark model.

III. Methodology

Data Preprocessing

As mentioned above, the dataset consists of 50 different folders, one for each reciter, and each contains recitation clips for that reciters. The preprocessing stages for those files are the following:

- Renaming all folders and clips numerically. This step is important because file naming is inconsistent in the files and involves multiple languages.
- No need for format conversion since Pydub library can read mp3 format.
- Divide the dataset into training (70%), validation (10%) and testing (20%) sets.
- Segment each recitation clip into 5 seconds clips and choose a random sample of only 10 from them. This is to reduce the number of training records and therefore reduce the model training time.
- Apply Mel-frequency cepstrum algorithm on each 5 seconds clip to extract its acoustic features.
- Convert target labels' array to a one-hot encoding format.
- Finally, MFCC features, along with their target labels, are in the right form to be fed into the classifying algorithm.

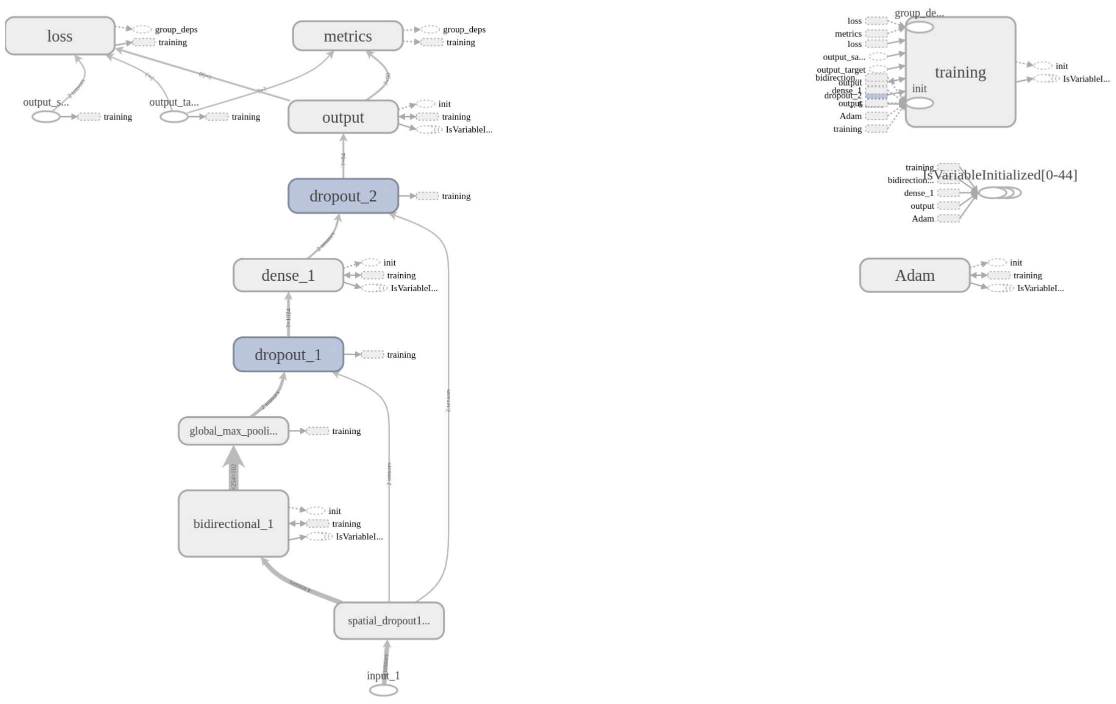
Implementation

The details of algorithm implementation are the following:

- Python programming language was used throughout the whole project.

- Using *python speech features* library, MFCC features extracted from each 5 seconds clip were taken as input along with their labels (reciters' names). Input shape = (number of short frames taken from a 5 second clip, number of MFCC features extracted from each short frame).

- Using *Keras* library, Gated Recurrent Neural Network was used with the following architecture to learn from the input:



- Spatial dropout layer (rate: 0.05): drops some sets of the *MFCC* features to avoid overfiting. The main difference between spatial dropout and regular dropout is that spatial dropout drops entire 1D feature maps instead of random individual elements from the input. This is desirable given that MFCC features within a short frame are strongly correlated while MFCC sets from different short frames should be independent.
- Bidirectional *GRU* layer (nodes: 512): bidirectional wrapper is used so that features sequence can be fed in its original and reversed form to increase the amount of input information available to the network.
- Global max pooling layer: linearizes the output of the GRU layer by taking the maximum of each filter so it can be fed to the dense layer.
- Dropout layer (rate: 0.2): to avoid overfiting.
- Dense (nodes: 256, activation: relu): to link features from different filters.
- Dropout layer (rate: 0.2): to avoid overfiting.
- Dense (nodes: 50, activation: softmax): Classification layer.

- The network was compiled with *binary crossentropy* loss function and *Adam* optimizer.

- The algorithm was trained for 50 epochs on the training set with a batch size of 128.

- *Early stopping* callback was used to make sure the model does not overfit. This is done by computing the accuracy on the validation set after each epoch and stopping whenever the score does not improve for 3 epochs.

- *checkpoint* callback was used to save the model with the best accuracy score on the validation set to use it later for the prediction task.

- The saved model was used to perform the prediction task on the testing set with a batch size of 128.

- The prediction results were compared to their true labels to compute the accuracy score.

Challenges

The only complication I faced is that MFCC algorithm parameters are related to each other and I had no prior experience in dealing with them. Therefore, the main challenge was how to tune the MFCC parameters to suit the input and not to contradict each other. Further research about the algorithm gave me a better understanding and resolved the issue.

Refinement

The details of my initial implementation are the following:

- Setting MFCC algorithm to return only the first 13 cepstral coefficients for every frame from the input clip and discard the rest.
- MFCC features are passed to a Gated Recurrent Neural Network with the following architecture and settings
    - *GRU* layer (nodes: 128).
    - *Global Max Pooling* layer.
    - *Dropout* layer (rate: 0.2).
    - *Dense* (nodes: 64, activation:tanh).
    - *Dropout* layer (rate: 0.2).
    - *Dense* (nodes: 50): Classification layer.
    - *binary crossentropy* loss function and *Adam* optimizer
    - 10 training epochs.

The above implementation resulted in an accuracy score of 67.7 on the testing set.

Afterwards, several experiments on neural network architecture and parameter tuning were made. Below are some of those experiments:

- Changing the activation function in the first dense layer to *RELU* improved the accuracy to 71.9.
- Applying a *Bidirectional* wrapper on GRU layer to process the original input and its reverse improved the accuracy to 74.2.
- Increasing the number of units in GRU layer to 512 and number of neurons in dense layer to 256 improved the accuracy to 84.3.
- Adding a *Global Average pooling* layer after GRU layer and concatenate it with *Global Max pooling* layer to feed them into the next layer did not improve the accuracy score.

- Taking all MFCC cepstral coefficients rather than taking only the first 13 (default) improved the accuracy score to 84.9
- Increasing the training epochs to 50 while applying early stopping with patience of 5 epochs and saving the model with best score on validation set improved accuracy to 87.6.

IV. Results

Model Evaluation, Validation and Justification

I chose Mel-frequency cepstrum algorithm because they have been the state-of-the-art feature extraction method for automatic speech and speaker recognition since 1980's[3].

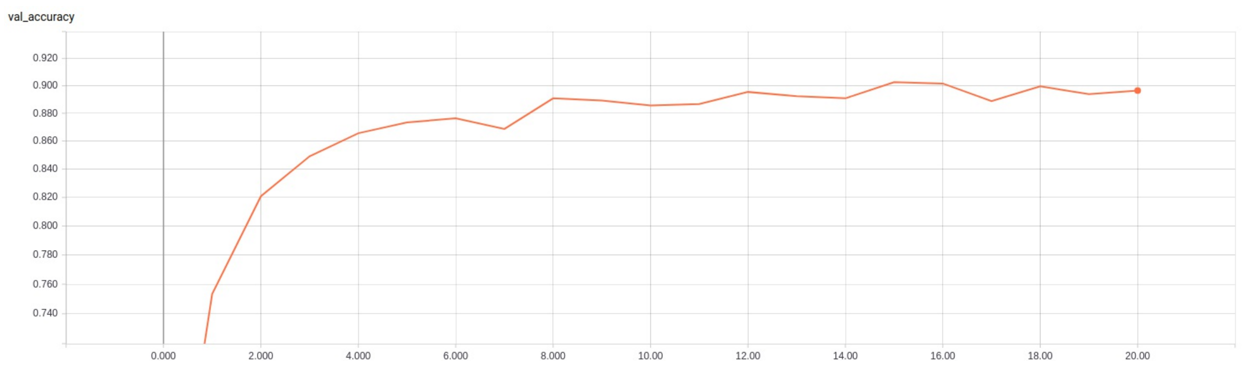
Also, I chose Recurrent Neural Network, for the classification task because it is one of few algorithms that can learn from temporal inputs competently and catch the relation between the preceding and following time steps in a sequence.

All in all, the results proved that my intuition was right. The model's predictive ability exceeds the benchmark model's by far. It scores 87.6 accuracy on the testing set while the benchmark model scores only 53.
The model is robust to change as well. Dropping the latter half of the 26 coefficient extracted from MFCC results in a decrease in the accuracy score only by 2.7%.

V. Conclusion

Free-Form Visualization

I imagined the features to be difficult to learn from, so I set the algorithm to learn for 50 epochs. But it seems to be capturing the acoustic patterns for the reciters after the eights epoch. The following figure illustrate this by plotting epoch number against accuracy on the validation set after that epoch.



This implies that the algorithm is powerful and learns well from the training data in a relatively short time.

Reflection

The project can be summarized by the the following sequence of steps:

1. Searching for a dataset that includes an abundant number of Quran reciters with plenty of recitations for each reciter.
2. Inspecting the dataset, correcting file naming and delete irrelevant files.
3. Preprocessing the data by only taking samples from the recitation clips provided in the dataset. This step is important to make sure that target labels are equally represented.
4. Extracting Mel frequency cepstral coefficient from all acquired samples.
5. Train the neural networks algorithm on the cepstral coefficient.
6. Tuning parameters and experimenting different architectures.

Having dealt with machine learning algorithms and their required preprocessing steps before made my work go smoothly. However, I found one of the steps to be notably challenging: step number four.

It was the first time I deal with sound data as input and therefore feature extraction step, particularly MFCC algorithm and its parameters, required extra research efforts.

All in all, it is evident that MFCC and Neural Networks are a good fit for this problem and deliver exemplary results.

Possible improvements

Following are some improvements that can be introduced to the current model that can improve its predictive ability or time efficiency:

- Passing the input into a convolution layer in addition to the recurrent layer then combining their output and feed it to the next layer could improve the model's accuracy.
- This model is supposed to be live on a server answering real time queries and accepting new reciters into the dataset. In its current form, the model needs to be retrained, or at least its output layer, every time a new reciter is to be added to the dataset. An easier solution would be training the network once on a sufficient amount of data, chop the classification layer and use the network only to extract deep features. Then, the output is fitted into a *K-nearest neighbors* algorithm *(KNN)* to do the classification task. The main advantage here is that refitting KNN when a new reciter is to be added to the dataset is much more time efficient that retraining the whole neural network.

References

1. https://quran2y.blogspot.com/p/full-quran-mp3-download-zip.html
2. Asda, Tayseer, et al. Development of Quran Reciter Identification System Using MFCC and Neural Network. TELKOMNIKA Indonesian Journal of Electrical Engineering, Jan. 2016.
3. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
4. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be