National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 471: Machine Learning

Project Report

Submitted by:

Sr. no.	Name	Registration no.
1	Shehryar Saqib	347703
2	Aiza Islam	343201

Date: 8th May 2023

Instructor: Dr. Daud Abdullah Asif

Table of Contents

Introduction:

The Quran is being recited globally and incorrect recitation is common due to the complexity of the Arabic language and pronunciation. Currently, there is no widely-available automated system that can detect and correct errors in Quranic recitation.

By referring to the problem statement, in this project, we developed of a high-accuracy system that detects errors in Quranic recitation, using the concepts of machine learning at the core, that provides the learner immediate feedback as to whether their recitation is correct. The project includes the usage of following concepts:

- Speech Recognition.
- Classification; recitation is correct or incorrect.
- Error Detection.

It will be very helpful in learning how to correctly recite Quran and allows learners to practice this at any time and with unlimited repetitions, without the presence of an instructor.

Methodology:

- **Dataset:**
  The Quranic recitation dataset was obtained from two websites:
  https://www.everyayah.com/data
  https://dailyayat.com/
  The recordings were collected from different reciters and contained different surahs and ayahs from the Quran. The Everyayah website contains a collection of recitations from 60 different reciters, while the DailyAyat website contains recitations from 20 different reciters. But this data had to be cleaned before using for our model which reduced it to being from only 66 reciters combined. The audio files were downloaded in MP3 format.

- **Data Preprocessing:**
  To preprocess the data, the audio files were first converted to WAV format and resampled to a sampling rate of 22kHz.
  To increase the size of the dataset and prevent overfitting of the models, data augmentation techniques were applied to the original recordings. Specifically, pitch shifting was used to create new recordings. Pitch shifting was used to increase or decrease the pitch of the original recording without changing its duration. This was achieved by using the **librosa.effects.pitch_shift** function from the **librosa** Python library, which shifts the pitch of the audio signal by a specified number of semitones. Pitch shifting was performed by randomly selecting a semitone value between -3 and 3 and applying it to the original recording.
  Test size was 10%, validation set was 15% and the training set was 75%.

- **Feature Extraction:**

  The feature extraction process involved loading an audio file using the **Librosa** library. Then, the **MFCCs** are computed using the **librosa.feature.mfcc** function, which takes the audio signal and sampling rate as input parameters.

  After the MFCCs are computed, they are reshaped to have the expected shape by creating a new array of zeros with the same number of rows as the original MFCCs and the same number of columns as the expected number of MFCC vectors per segment. Then, the original MFCCs are copied into this new array, starting from the first column.

  Finally, the extracted MFCCs are stored in a JSON file, along with the corresponding label for the audio file. This file is used as input to the machine learning model.

- **Speech-to-Text Conversion:**

  For this task, **Google Speech-to-Text API** was used. The feature extraction and speech recognition are done by the API itself. The API uses a pre-trained neural network to perform speech recognition on the audio input and returns the corresponding text output in Arabic. The audio file is first read using the **pydub** library, and then exported as a WAV file. The Speech-to-Text API is then called with the appropriate configuration, which includes enabling automatic punctuation and setting the language code to Arabic. The response from the API is then processed to extract the transcript of the audio file. Finally, the extracted transcript is appended to a list called 'extracted_features'. After this, some data preprocessing steps on the extracted text are performed to ensure that it is in a uniform format and ready for comparison with the reference text. It joins all the individual transcribed sentences into a single string, removes any leading or trailing white spaces using the strip() method. Next, it removes any periods using the replace() method. Finally, it splits the string into individual words using the split() method and stores them in a list. The resulting list contains the extracted words from the Arabic speech in a uniform format, ready for comparison with the reference text.

- **Model:**

  The deep learning model built in this project is a **Convolutional Neural Network (CNN)** for audio classification. Currently the classification model words for the seven verses of **Surah Fatiha**. The model takes as input 13 **MFCC** coefficients (Mel-frequency cepstral coefficients) extracted from the audio clips. The model consists of several layers:

  - There are four sets of Convolutional and Max Pooling layers with Batch Normalization and ReLU activation functions to extract important features from the input.
  - A Flatten layer to transform the 2D feature maps obtained from the previous layer into a 1D feature vector.
  - Two Dense layers with ReLU activation functions and a Dropout layer to prevent overfitting.

- That the end, an output Dense layer with a softmax activation function to output the probability distribution over the seven classes for the 7 ayats.

The model is compiled with the 'sparse_categorical_crossentropy' loss function and 'Adam' optimizer. It is trained with a batch size of 16 and for 25 epochs. The learning rate of the optimizer is set to 0.0001 for the first 25 epochs, and then reduced to 0.000004 for another 25 epochs. Finally, the model is evaluated on a test set to measure its accuracy on unseen data. The trained model is saved as a .h5 file for future use.

The main code executes the functions in the following order:

- Prepare the training, validation, and test sets using prepare_datasets function.
- Build a CNN model using build_model function.
- Compile the model with an optimizer, loss function, and metrics using model.compile.
- Train the model using model.fit on the training and validation sets.
- Compile and train the model again on the training and validation sets with a different optimizer and learning rate.
- Evaluate the model accuracy on the test set using model.evaluate.
- Save the trained model to a file using model.save.
- Load the saved model from the file using tf.keras.models.load_model.
- Predict the class label of an audio file from the test set using predict function.

➢ **Detection and Correction Phase:**

There are seven different verses from the Quran i.e. **Surah Fatiha**, represented as strings of Arabic text. The correctness of the recitation is being checked. It first extracts the words from a recited text, joining the words into a single string, and then splitting them into a list. It then compares the length and sequence of words of the extracted text to the original text. After extracting and processing the recited verses, the program checks whether the number of words in the recitation is the same as the number of words in the original verse. If they are not the same, then the program prints the message "Incorrect Recitation". Otherwise, the program checks whether the sequence of words in the recited verse is the same as in the original verse. If they are the same, the program prints the message "Correct Recitation". If the words are the same but in a different order, the program prints the message "The words are recited correctly, but in different order." Finally, if the recited verse contains incorrect words, the program prints the message "The words at index i are different.". This also tells what was the incorrect word along with the correct word.

Deliverables (Proposal vs Product):

Proposed Solution	Final Product
Automated System	Achieved
Recognizing Speech	Achieved
Identifying Mistakes	Achieved
Feedback	Achieved
Correct sequence of verse	Achieved
Training on native and non-native reciters	Achieved
RNN and Transformers as possible algorithms	CNN is used as the algorithm

Figure 2: Comparison Table

CNN was used instead of RNN because CNN improved the accuracy. Although RNN was explored but desired accuracy was not achieved. CNNs are computationally more efficient than RNNs and transformers due to their parallel processing architecture.

Challenges:

Speech error detection is arguably one of the hardest tasks in speech processing domain. The lack of sufficient incorrect tagged data seriously impedes the accurate detection of incorrect recitation. Taking into consideration of the complexity of the task, there were a few challenges that were face in the development cycle of this model which are being briefed as follow:

1. Since the dataset included recitation from different reciters the length of audio verses was not same which led to inability of the audio being broken down into segments so it had to be processed all at once. This caused greater complexity in the preprocessing and handling of the data.
2. There was much of a time constraint considering the scope of our project which was quite vast; it involved more than just one model. Due to this, there was a risk of incomplete data preprocessing leading to suboptimal model performance and reduce experimentation.
3. One of the challenges related to developing a Quranic CAPT system was the lack of guidance and similar work in the field. There are very few pronunciation training systems available and almost none for Arabic language, especially for the recitation of the Quran. So, it was challenging to identify the most effective approaches and best practices for developing an accurate and reliable system.
4. One of the core requirements of the system was the ability to accurately detect error achieving high accuracy in error detection can be challenging, especially when dealing with variations in pronunciation due to different accents, dialects, and speaking styles.

Results:

The proposed system was evaluated on a dataset of Quranic recitations consisting of seven different verses, and achieved an overall accuracy of 97%. The results of accuracy are shown in Figure 2.

```
52/52 [==============================] - 15s 280ms/step - loss: 0.0106 - accuracy: 1.0000 - val_loss: 0.1926 - val_accuracy: 0.9519
Epoch 5/25
52/52 [==============================] - 15s 286ms/step - loss: 0.0145 - accuracy: 0.9976 - val_loss: 0.1939 - val_accuracy: 0.9567
Epoch 6/25
52/52 [==============================] - 15s 287ms/step - loss: 0.0126 - accuracy: 0.9988 - val_loss: 0.1919 - val_accuracy: 0.9567
Epoch 7/25
52/52 [==============================] - 15s 289ms/step - loss: 0.0120 - accuracy: 1.0000 - val_loss: 0.1899 - val_accuracy: 0.9567
Epoch 8/25
52/52 [==============================] - 15s 288ms/step - loss: 0.0137 - accuracy: 0.9976 - val_loss: 0.1889 - val_accuracy: 0.9615
Epoch 9/25
52/52 [==============================] - 15s 287ms/step - loss: 0.0124 - accuracy: 0.9988 - val_loss: 0.1837 - val_accuracy: 0.9663
Epoch 10/25
52/52 [==============================] - 16s 300ms/step - loss: 0.0123 - accuracy: 0.9988 - val_loss: 0.1829 - val_accuracy: 0.9615
Epoch 11/25
52/52 [==============================] - 15s 282ms/step - loss: 0.0102 - accuracy: 1.0000 - val_loss: 0.1873 - val_accuracy: 0.9519
Epoch 12/25
52/52 [==============================] - 15s 284ms/step - loss: 0.0139 - accuracy: 0.9964 - val_loss: 0.1870 - val_accuracy: 0.9615
Epoch 13/25
52/52 [==============================] - 15s 287ms/step - loss: 0.0124 - accuracy: 0.9988 - val_loss: 0.1876 - val_accuracy: 0.9615
Epoch 14/25
52/52 [==============================] - 15s 284ms/step - loss: 0.0110 - accuracy: 0.9988 - val_loss: 0.1855 - val_accuracy: 0.9663
Epoch 15/25
52/52 [==============================] - 15s 289ms/step - loss: 0.0141 - accuracy: 0.9976 - val_loss: 0.1883 - val_accuracy: 0.9615
Epoch 16/25
52/52 [==============================] - 15s 289ms/step - loss: 0.0111 - accuracy: 0.9976 - val_loss: 0.1937 - val_accuracy: 0.9615
Epoch 17/25
52/52 [==============================] - 15s 292ms/step - loss: 0.0136 - accuracy: 0.9964 - val_loss: 0.1951 - val_accuracy: 0.9615
Epoch 18/25
52/52 [==============================] - 15s 287ms/step - loss: 0.0104 - accuracy: 0.9976 - val_loss: 0.1980 - val_accuracy: 0.9615
Epoch 19/25
52/52 [==============================] - 15s 283ms/step - loss: 0.0112 - accuracy: 1.0000 - val_loss: 0.1992 - val_accuracy: 0.9615
Epoch 20/25
52/52 [==============================] - 15s 284ms/step - loss: 0.0109 - accuracy: 1.0000 - val_loss: 0.2027 - val_accuracy: 0.9567
Epoch 21/25
52/52 [==============================] - 15s 296ms/step - loss: 0.0092 - accuracy: 0.9988 - val_loss: 0.2068 - val_accuracy: 0.9567
Epoch 22/25
52/52 [==============================] - 15s 293ms/step - loss: 0.0117 - accuracy: 0.9988 - val_loss: 0.2033 - val_accuracy: 0.9567
Epoch 23/25
52/52 [==============================] - 15s 284ms/step - loss: 0.0065 - accuracy: 1.0000 - val_loss: 0.2029 - val_accuracy: 0.9615
Epoch 24/25
52/52 [==============================] - 15s 286ms/step - loss: 0.0085 - accuracy: 1.0000 - val_loss: 0.2009 - val_accuracy: 0.9567
Epoch 25/25
52/52 [==============================] - 15s 289ms/step - loss: 0.0138 - accuracy: 0.9976 - val_loss: 0.1953 - val_accuracy: 0.9663
11/11 [==============================] - 1s 117ms/step - loss: 0.0902 - accuracy: 0.9712
Accuracy on test set is: 0.9711815714836121
```

Figure 2: Accuracy Results

The system gives the feedback as shown in Figure 3. It identifies where the error occurred and correction. VS code terminal does not support Arabic text that is why the output is shown in form of tokens but if we copy and paste the output on notepad then it shows correct word.

```
7/7 [==============================] - 1s 96ms/step - loss: 0.0045 - accuracy: 1.0000
Accuracy on test set is: 1.0
1/1 [==============================] - 0s 146ms/step
Expected: 7 and predicted: [7]
['ayat-7']
['شيراطون', 'الذين', 'انعمت', 'عليهم', 'غير', 'المغضوب', 'عليهم', 'ولا', 'الضالين']
The words at index 0 are different.
The word incorrectly recited is:صراط
```

Figure 3: Output and Feedback

Future Prospects:

There are several future prospects for this program. One potential improvement is to incorporate female recitations as well into the system, allowing users to choose between male and female voices for their recitation. This would involve collecting dataset for female

reciters as well. Additionally, real-time audio input could be integrated into the program, enabling users to recite the verses and receive immediate feedback. The program could be developed into a user-friendly interface, which could be accessed via a website or mobile app, making it more accessible to a wider audience. There is another future prospect for incorporating tajweed rules into the program to improve their recitation. Additionally, audio reconstruction could be used to provide users with the correct pronunciation of any incorrectly recited parts. A user interface for the program could be developed, which would make it more user-friendly and accessible. With these improvements, the program has the potential to become a valuable tool for those studying the Quran and seeking to improve their recitation skills.

Conclusion:

In conclusion, we have developed a system for automatically verifying the correct recitation of the seven verses of **Surah Fatiha** using classification model using **CNN** and text comparison approach. The model was tested using recorded recitations of **Surah Fatiha** and was found to be effective in classification and later identifying incorrect recitations. The system utilizes natural language processing techniques to extract the text from an audio file and compares it with the correct text. Future prospects of the system include extending it to recognize female voice, real-time audio, integration tajweed and a user-friendly interface for wider usage. The development of a Quranic recitation recognition system is a significant contribution to the field of Quranic studies. The system can accurately recognize and verify Quranic recitations, and provides feedback for incorrect recitations. Overall, this project has shown great potential for the integration of technology and Islamic studies, opening up new possibilities for the preservation and dissemination of the Quranic recitation tradition.