© https://fti-tn.net/publications Future Technologies and Innovations (FTI) Proceedings: 4th international conference on computer applications and information security (iccais'2021) / March 19 / 2021/ Tunisia: https://fti-tn.net/iccais-2021-list-of-papers

Automatic Qurʾān Tutor to Assist Qurʾānic Recitation with Neural Network

1st Noura Abdulrahman Alawwad
College of Computer Science and Information
Al-Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh, Saudi Arabia
noura.a.alawwad@gmail.com

2nd Abdulaziz Almohimeed
College of Computer Science and Information
Al-Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh, Saudi Arabia
aialmohimeed@imamu.edu.sa

***Abstract*— Islam is the second largest religious group, comprising 24% of the world's population. As the Qurʾān is the holy book in Islam, it can only be recited in Arabic and it is important to recite it in a clear and accurate way. The native language of the majority of Muslims is not Arabic, which can lead them to make mistakes in reciting the Qurʾān. Unfortunately, relatively little speech recognition research has focused on Arabic compared to other languages. In addition, research is rare regarding how technology can be used in the context of the Holy Qurʾān. In this research, we aim to build a system that help users recite Qurʾān by simulating the functionality of a Qurʾān tutor. The system can listen to the user, dynamically judge their memorization and pronunciation, instantaneously help them by voice if they forget words or get stuck and warn them if they make a mistake in Qurʾān recitation. The creation of the system consists of six major stages: data preparation; pre- processing and feature extraction; training; validation; testing; and integrating the model into a mobile application. The model was built with the DeepSpeech platform. The acoustic model for our Qurʾān tutor system achieved 80% validation accuracy and 77% testing accuracy, the model overall achieved a BLEU score of 0.681 and a PER score of 0.261. Human testing indicated that our system was able to detect errors and correct them dynamically with remarkable speed.**

Keywords—Artificial Intelligence (AI), Neural Network (NN), Automatic Speech Recognition (ASR), Natural Language Processing (NLP), Deep Neural Network (DNN)

I. Introduction

Speech is the most effective and easiest method of communication between humans. Since we are in the era of technology, improving interactions between humans and machines is a primary goal for achieving the great benefits of it. Artificial intelligence (AI) systems play a significant role in the field of human-to-machine communications. Machine learning (ML) is a form of AI that can predict future output based on past data or big data. Recent research has contributed to the development of automatic speech recognition (ASR) systems, which have become increasingly embedded in our everyday lives.

Arabic is currently considered one of the most widely spoken languages in the world, with an estimated 313 million speakers and 270 million speakers of Arabic as a second language [1]. It is ranked as the fifth after Mandarin, English, Hindustani, and Spanish. While Arabic is currently one of the most widely spoken languages in the world, however, there has been little speech recognition research on Arabic compared to other languages. Moreover, very little research has been performed on how these technologies can be applied in the context of the Holy Qurʾān [2]. The relationship between Arabic and Islam is very strong, the most sacred book in Islam is the Holy Qurʾān. It contains 114 chapters, each of which is divided into a number of ayat, or verses. The Qurʾān can only be recited in Arabic. According to the Central Intelligence Agency (CIA), Islam is the second largest religious group. The total population of Islam is more than 1.8 billion, of whom 24.1% are Muslims, while only 3.12% of the total population can speak the Arabic language [3]. Since not all Muslims' native language is Arabic, they may make mistakes when reciting Qurʾān.

This research develops a tutor for the Holy Qurʾān using one of the most recent ML approaches, namely artificial neural networks (ANNs). The system simulates the functionality of a real human Qurʾān tutor; the system can listen to the user's recitation, judge their recitation, dynamically help them by voice if they forget words or get stuck, or warn them if they make a mistake in Qurʾān recitation. This research is significant because the produced system makes it easier for more than 24% of the world's population to recite and memorize the Qurʾān.

In this paper, we will specify the purpose of this research and the aims and objectives of it. Then provide an overview of the state of the art in existing Qurʾānic ASR systems. And present the methodology used. Next, describe the system design and implementation. Finally, we present the model testing and discuss our results.

II. Purpose and Aim

Although Islam is the second largest religious group in the world, there are few Muslims whose native language is Arabic [4]. As such, when they read the Holy Qurʾān, they are more likely to make mistakes, as the Qurʾān can only be recited in the Arabic language. Because it is important to recite the Qurʾān in a clear and correct way, the need for a tutor is clear. Furthermore, Islam is considered the fastest-growing religion, the number of Muslims is expected to increase by 70% from 1.8 billion in 2015 to nearly 3 billion in 2060, making this work even more important [3].

Moreover, in many places, a Qurʾān tutor may not be available or it is hard to contact tutors because of location or time constraints. Even the Qurʾān tutor needs another tutor who can listen to and validate his or her recitation. It is important to find a fast and accessible self-learning method to recite the Qurʾān that improves the learning process, decreases and optimizes study time, and can be used from anywhere.

In addition, while ANNs have proven to be more efficient than other ML models, there are still few resources for Qurʾānic research and applications using ANNs. We hope this work will facilitate the future development of Qurʾānic research.

In this research, we investigate the feasibility of a speech recognition and correction system for Qurʾānic recitation using an ANN. We present our methodology for building a speech recognition and correction system suitable for the Holy Qurʾān based on ANN. The system shows an ayah, then listens to the user's recitation, judges their memorization and pronunciation, instantaneously and dynamically helps them by voice if they forget words or get stuck, and warns them if they make a mistake. Finally, we evaluate the model and its ability to assist users in reciting the Qurʾān.

This research addresses the following research questions:

- How can we build a system using neural networks that can dynamically correct a user's recitation of the first five chapters of the Holy Qurʾān?

- Will using an ANN model to dynamically correct the user's recitation of the first five chapters of the Holy Qurʾān is going to be convenience for the user in reciting the Qurʾān?

III. State-of-the-Art

A. History of Automatic Speech Recognition

The science of speech recognition has evolved a great deal since the 1920s. The first speech recognition system was Radio Rex in 1922, which was a toy ("a hidden dog inside a box"). When someone called Rex, the dog would come out of the box. However, the system could only recognize one word (i.e., Rex) and could not detect a variety of voices (e.g., a child's voice or a woman's voice).

In 1952, a system was developed at Bell Laboratories that could recognize the digits 0 to 9 with 90% accuracy. However, the system could only recognize these numbers when they were spoken by its inventor. IBM's first speech recognition machine, Shoebox, was created in 1962. The machine was able to recognize 16 isolated words and digits from 0 to 9, so long as many pauses were given. The system was able to recognize different types of voices. In 1976, the HARPY system was developed by ARPHA. The project had a three-million-dollar budget and could recognize continuous speech with limited vocabulary (1,000 words, including some phrases) with different pronunciations. However, it did not use any statistical method.

Statistical model–based systems, which became popular in the 1980s, led to many breakthroughs. Vocabulary size grew larger, nearing 10,000 words. The hidden Markov model (HMM), which used a mathematical approach to analyze sound waves, was the most used model in ASR systems at the time. The first ASR systems used Mel-frequency cepstral coefficients (MFCCs) or relative spectral-perceptual linear prediction (RASTA-PLP) for feature extraction and HMMs and Gaussian mixture models (GMMs) for training and recognition, as these were the state of the art at the time. Later, in the 1990s, the maximum likelihood training criterion was used for training GMM-HMMs. In the 2000s, the minimum phone error (MPE) and minimum classification error (MCE) were proposed for training along with some improvements to ASR accuracy.

In recent years, thanks to improvements in computation power and the ability to train models on large datasets, deep learning models—particularly deep neural networks (DNNs)—have become very popular due to their ability to reduce error rate and improve accuracy. Deep learning models have been successful in the last few years and are a promising area of ML for future tasks in this field [5].

B. Structure of ASR

The design and development of a speech recognition system are dependent on various components. *Figure1* visualizes the basic structure of an ASR system. Yu and Deng [6] indicated that each ASR system has the following four main components.

- Signal processing and feature extraction takes the audio signal as input, then extracts the features that are suitable for the acoustic model.

- The acoustic model (AM) takes the features as input, then generates the AM score for the variable length feature sequence.

- The language model (LM) learns the correlations between words in a training corpus to estimate the probability of a hypothetical word sequence or LM score. It is optional to have an LM.

- Finally, the hypothesis search combines the AM and LM scores, resulting in the feature vector sequence and hypothesized word sequence. The word sequence with

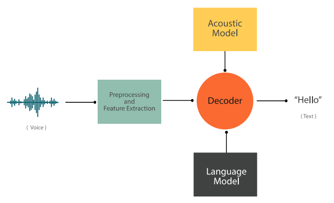


Fig. 1. Basic structure of an automatic speech recognition system

C. Related Work "Qurʾānic Speech Recognition Systems"

In the last few years, interest has grown regarding scientific research on the benefits of technology in Qurʾān education. Some popular techniques used in ASR include ANNs, dynamic time warping, and HMMs. However, most of them used HMMs.

The authors in [7] proposed a speech recognition system that distinguished the types of Madd (meaning "elongated tone or prolongation") and the method of recitation (Hafss or Warsh). The system can recognize and point out any mismatches found. The researchers used MFCCs for feature extraction and HMMs for feature classification. The result was 60% and 50% for the exchange prolongation and 40% and 70% for the greater connective prolongation for Hafss and Warsh, respectively. The system had some drawbacks related to extra noise due to audio file compression and poor quality in the recording process. However, high performance was achieved. In [8] they proposed a system capable of determining which of the Ahkam Al-Tajweed was used in a specific audio recording of a Qurʾānic recitation. The rules they covered included EdgamMeem, EkhfaaMeem, Ahkam Lam, and Edgam Noon, and they considered both correct and incorrect usage for each rule. The system covered the entire Qurʾān. The researchers used several feature extraction techniques and several classifiers. The best results were obtained using MFCC, WPD, HMM-SPL, and CDBN for feature extraction and SVM for classification. A method proposed in [9] for tajweed rule checking in an AASR system using speech recognition technology for self-learning to recite the holy Qurʾān. The system used MFCCs for feature extraction and HMMs for feature classification. The system recognized the user's recitation and pointed out whether a mismatch was found. The paper did not have any results, however; the authors only proposed the method. Also, an open source dataset, called Tarteel [10] They claimed to have the world's first and largest open source dataset of Qurʾānic recitations by ordinary Muslims, with over 25,000 recordings of the Holy Qurʾān. Although this represents an impressive effort to collect a huge number of ordinary people reciting these verses, many of the provided recordings are unfortunately corrupted or do not meet the quality standards for use in feeding networks.

Based on the available research, we can conclude that:

- Most of the work done on Arabic speech is based on the Sphinx engine or HMMs, while some is based on ANNs.
- Little of the available research has focused on Qurʾān recognition for the purpose of learning.
- Little of the available Qurʾānic research used an ANN as a LM.

IV. Methodology

To answer the first question in this research *(how to build a system that can correct the user's recitation using a neural network)*, the developed system needs to show the verse, listen to the user's recitation of the verse while converting that recitation into text, and finally compare the recognized words with the shown words in the verse. After specifying the requirements for the system, we determined its structure.

First, an STT system should listen to the user and recognize their speech. To build such a system, the following steps are necessary:

- Gather an audio dataset;
- Prepare the dataset based on the required format for the STT model;
- Preprocess the data to be ready for the STT model; and
- Train a STT model to generate an AM that can convert the spoken words into text.

The second step is to design a mobile application for comparing the shown words in the verse to the recognized words in the STT model.

There are many difficulties when building an ASR system (especially for Qurʾān recitation) for various reasons, including: variability of speaker, variability of volume, variability of word speed ("recitation speed"), variability of pronunciation ("recitation"), word boundaries (when the user recites a verse without pauses between words), and noises such as background sounds, audience speech.

To overcome these difficulties, we address each of the above problems while describing the structure of the system.

As mentioned above, to build a STT model, it is necessary to gather a dataset of audio content. However, audio should be gathered from different people with different recitation speeds so that the system will be able to recognize words for different speakers and word speeds. This solves the problem of speaker variability and recitation speed variability. After gathering the dataset, we will specify each audio file with its transcription to be prepared for the preprocessing stage. Preprocessing the data involves analyzing the data. First, the audio is converted to a signal that shows the amplitude of the audio over time *(see Figure3).* A single signal actually consists of many simultaneous frequencies, we used Fourier transforms to analyze the signal and break it into components to produce a spectrogram. The spectrogram gives information about amplitude in terms of both frequency and time, where frequency is on the vertical axis plotted against time. The intensity of shading indicates the amplitude of the signal. To reduce the noise and dimensionality of the data, we extracted the features from the spectrogram using MFCC analysis. We used MFCC analysis to identify the essential components of the data, using both Mel-frequency and cepstral analysis. The goal is to obtain the vocal tract frequency response from the speech signal. Only the shape of the sound is needed and not the source, as the latter is unique to individual vocal cords. After data preprocessing, we trained the system to get the AM. In order for our STT model to learn the Arabic alphabet, it first needed to be trained to recognize Arabic speech.

DNNs are constructed from multiple interconnected layers. The first layer is called the input layer and the last layer is called the output layer. The layers in between are the hidden layers, each of which contains multiple nodes or neurons that are in turn connected to each other. The neuron processes and propagates the input signal it receives to the layer above it. The strength of the signal depends on the weight, bias, and activation function. Each layer in a neural network represents a deeper level of knowledge. The more hidden layers exist, the more complex features the system will learn. Learning occurs in two phases:

1) The first phase is "Forward propagation", is when the input data is passed through all the neurons in the network so that it calculates its transformation (prediction) on the information they receive from the neuron in the previous layer, then forward it to the next neuron in the above layer. After the data has crossed all the layers, it will reach the final layer with a result of the prediction (label) of the input data.

2) Then we will calculate the "Loss function" to estimate the errors in the predictions and measure how good the prediction was compared to the correct result.

3) Once the loss function has been calculated, the information will be propagated backward. And this process called "Backpropagation", which is the second phase of the learning. It starts from the output layer to the neurons in the hidden layers. Each neuron will receive a fraction of the loss signal that describes their relative contribution to the total loss. Then the weights of the connection between neurons will *be adjusted by using a technique called "gradient descent", This technique changes the weights in small increments with the help of the calculation of the derivative of the loss function.*

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration or training steps.

After the training is complete, an AM is produced that can convert speech into text. While it would be possible to stop here and build our mobile application based on the AM, for our problem an AM is in fact insufficient. The AM gives a result based on the user's pronunciation, ignoring the shape of the word. However, when reciting the Qurʾān with the Harakat and Madd, the pronunciation of the word may not be the same as the actual shape of the word.

When the user recites " و لم يكن لهٌ كفوا أحد" or " ويل لكل همزة لمزه," the AM will mostly produce "و لم يكن لهو كوفون أحد" and "ويلل لكلل همزتلمزه" as a result. The system needs to be able to recognize the words without duplicating the alphabets, especially when the user's recitation of a verse is not clear. In addition, the system only needs to produce certain specific words; as it is an STT model specialized for Qurʾānic recitation, it only needs to recognize words in the Qurʾān. As such, it is crucial to build an LM for our system in order to handle ambiguities in spelling and context. Injecting a LM into the STT model forces the model to predict only a specific vocabulary and teaches it to predict the probability of the occurrence of a word or word sequence.

AMs and LMs are typically trained separately and then combined at inference using a beam search decoder. Including an LM in decoding can improve the accuracy of our ASR system. After training the system, we built a mobile application to serve as an interface between the user and the STT system and to compare the results, then interact with the user based on them.

A. Dataset collecting

To build the system, a good dataset should be collected. The data to be collected are audio files. Since the STT model encompasses only the first five chapters of the Qurʾān, the vocabulary is small, and the dataset therefore need not be large. This problem differs from building a system for the Arabic language, where there is a large vocabulary, implying the need to collect a large dataset to achieve a reasonable accuracy rate for the system.

We collected data from both expert reciters and average people. If we only collected audio from professional reciters, the model would learn the correct alphabet features but could not tolerate normal users' background noises, pauses, humming, and the like. We collected the audio files for expert reciters from Alsabbile website and the audio files for average people from Tarteel website. In addition, we recorded some of the audio files ourselves.

The final dataset included around 1,400 audio files: 1,100 from expert reciters and 300 from average people, with around 40 different speakers. Each audio file contained one verse and ranged from 3 to 14 seconds in length. There were on average 18 audio files for each verse.

B. Tools and Services

In this research, we used certain tools and services to implement various functionalities, as described below.

*a) Mozilla DeepSpeech:*We first trained our model using the OpenNMT framework. It achieved 79% validation accuracy and 75% testing accuracy. Unfortunately, however, we discovered that the OpenNMT framework does not support online decoding ("dynamic decoding") and thus would not be able to correct users' recitations. We then built our application using the IBM Watson STT service. However, the accuracy was not as good as with OpenNMT. In addition, we could not improve the accuracy by training the model on our Qurʾānic dataset, since this feature was not available in our country. Next, we used Mozilla's DeepSpeech framework. Mozilla's DeepSpeech is an end-to-end neural system that can be trained quite easily, unlike Kaldi, which requires more domain knowledge, or wav2letter++, which has not yet been widely tested by the community. Mozilla's DeepSpeech framework is based on a TensorFlow implementation of Baidu's end-to- end ASR architecture.

*b) Android Studio:*This is the official integrated development environment (IDE) for Android app development, based on IntelliJ IDEA. It offers powerful developer tools with an excellent code editor and a fast emulator.

c) Google Colaboratory: This tool allows researchers to write and execute Python code in their browser without pre-configuration and provides free access to GPUs.

d) KenLM: This library implements two data structures for efficient LM queries [11].

e) Android DeepSpeech SDK: This is an Android library that contains an API for Mozilla's speech recognition services.

C. Evaluating the system

The system was evaluated using the following tools:

- BLEU Score: This is one of the most popular metrics for evaluating sequence-to-sequence tasks, as it is convenient for researchers as well as fast and easy to calculate. Due to its ubiquity, it is easy for other researchers to compare the model to benchmarks for the same task based on BLEU score.

- Position Independent Word Error Rate (PER): We chose PER over Word Error Rate (WER) and Character Error Rate (CER). CER is not suitable for our system because the system only produces specific words that are specified in the LM. The same is true for WER, which does not take permutations of words into consideration, leading to faulty accuracy results.

- Human Evaluation: Our system was tested by a number of users to evaluate its convenience, which can be measured with three metrics: effectiveness, time, and usability. We prepared a questionnaire consisting of 24 questions that covered the different metrics used to measure convenience using a number of quantifiable metrics.

V. System Design and Implementation

A. Dataset preparation

After collecting the audio dataset, we prepared the audio files for the training process. Audio files were cleaned of whitespace and changed to the following parameters: mono, 16 bits, and 16 kHz. We divided the dataset into three sections: train, dev, and test. In total, 80% of the audio files were fed to the train section, 10% to the dev section, and 10% to the test section. Each audio file can only appear in one section to ensure good model creation and avoid overfitting. To load the data into the DeepSpeech model, we generated a .csv file containing the audio file path, transcription, and file size for each of the three directories. The test.csv, train.csv, and valid.csv files appear as follows:

At this point, the dataset was ready to be loaded into a DeepSpeech model.

B. Pre-processing and Feature Extraction

The data need to be analyzed for preprocessing and feature extraction. We first converted the audio to a signal showing the amplitude of the audio over time, then converted it to a spectrogram with FTT. Next, we extracted the features from the spectrogram using MFCCs.

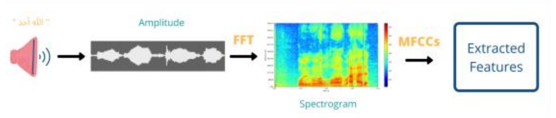


Fig. 3. Flow of the pre-processing and feature extraction processes

C. Training

The DeepSpeech network is composed of five layers of hidden units. First, the audio features are fed into three fully connected and nonrecurrent layers, followed by an LSTM-RNN layer that includes a set of hidden units with forward recurrence. This is followed by another fully connected nonrecurrent layer and finally an output layer, the latter of which is a standard logit that corresponds to the probability of the character at each time step, as shown in *Figure4*.

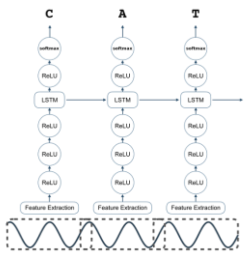


Fig. 4. The structure of the DeepSpeech network. [12]

The first layer is the input layer. The output of the first layer at each time step depends on the MFCCs' frame. The remaining fully connected layers are given an independent data for each time step. The fully connected layers use a rectified linear unit (ReLU) as an activation function. This non-linear activation function has gained popularity in the deep learning domain. The main advantage of the ReLU function over other activation functions is that it does not activate all the neurons at once. The function returns 0 if it receives any negative input but returns x for any positive value x. Thus, its output can range from 0 to infinity.

Finally, the network outputs a matrix of character probabilities. For each time step, the system provides a probability for each character in the alphabet that represents the likelihood of that character corresponding to the audio. The connectionist temporal classification (CTC) loss function is computed to measure prediction error and maximize the probability of correct transcription. We used 2,048 hidden nodes and a learning rate of 0.0001.

D. Language Model

LM includes the knowledge of alphabets, words, and sentence structure for a given language. Including an LM is optional in DeepSpeech speech recognition models. It is referred to as "scorer" file because it computes the likelihood of the sequences of words and characters in the output and then forces the decoder to pick the most likely result based on the score. Therefore, it significantly improves the accuracy of the speech model [12]. The scorer package is composed of two components: a KenLM LM and a trie data structure. To create the scorer package, there are two steps.

- First, to create a KenLM language model we need to feed all the transcriptions of our speech model to the KenLm toolkit to creates a ".arpa" file to store the n-grams. Then converting the ".arpa" file into a language model ".binary" file for binary build.
- Second, use the "generate_scorer_package" binary, that is a part of the native client packages included in DeepSpeech along with the language model ".binary" file and vocabularies and alphabets of our language, to create the final package file including the trie data structure that contain all the words and vocabularies of our model.

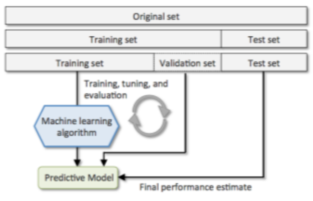
The LM file and AM file were combined at inference using a beam search decoder.

VI. Testing and Result Discussion

A. *Model validation*

Model development generally occurs in two stages. The first stage involves training the model: applying algorithms to the data with the model knowing the correct transcription to find patterns and bind the features of the source data to their targets. The second stage involves testing the model: applying the trained model to new data to return a predicted transcription based on probability scores. It is the gold standard used to evaluate the model.

However, the problem with dividing the data into only training and testing sets is that training may only indicate how well the model performs on the training set. It is possible that the model may overfit to the training data to perform well on these data but cannot generalize to unseen data. Overcoming this problem necessitates an indicator of the model's performance on unseen data during training that can be used for accurate evaluations. One solution is to divide the data into three sections: training, validation, and testing sets. The validation dataset is used during training to evaluate the model and tune the parameters with the aim of finding and optimizing the best model, thereby checking and avoiding overfitting.


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

he

Fig. 5. Data structure [13]

- Validation set: The data used to evaluate a given model and tune the parameters to find and optimize the best model. The model sees the data but never learns from it.
- Test set: The data used to evaluate the final model.

Our Qurʾān tutor model trained for approximately 21 hours on the training set (roughly 100 epochs with a learning rate of 0.0001 and a dropout rate of 0.30). Our AM had 96% training accuracy, 81% validation accuracy, and 77% testing accuracy.

B. *Model testing*

In any system testing situation, there are various methods of evaluation. As mentioned in the previous chapter, we evaluated our Qurʾān tutor model AM and LM with two metrics: BLEU score and PER score.

TABLE I. EVALUATION OF QURʾĀN TUTOR MODEL

Metrics	**Qurʾān Tutor model**
BLEU score	0.681
PER	0.261

C. *Human Testing*

This section answers the second question of the research: *(Will using an ANN model to dynamically correct the user's recitation of the first five chapters of the Holy Qurʾān is going to be convenient for the user in reciting the Qurʾān according to the metrics of accuracy, usability and time?)* System convenience is a multidimensional concept that includes many measurements leading to the fulfillment of a certain task. The convenience of our system can largely be measured using three metrics: the effectiveness of our proposed system, the time, and the usability of our system.

- Effectiveness: This measurement refers to the accuracy of our system— defined as accuracy in finding and correcting mistakes—and the completeness of the user's goal achievement.
- Time: This measurement refers to the amount of time the model takes to find mistakes in recitation and correct them (i.e., the speed of the system) and the time the user takes to complete the desired task.
- Usability: This measurement refers to user satisfaction with the Qurʾān tutor system in terms of user attitude, comfort level, application relevance, and acceptability of use.

To measure effectiveness, time, and usability, we prepared a questionnaire consisting of 24 questions that covered the different metrics used to measure convenience using a number of quantifiable metrics. A total of 20 users completed the questionnaire. After the users finished testing our system, we handed them a questionnaire to judge the convenience of our Qurʾān tutor system. (*Figure 6)* present the human evaluation results for each of the three metrics for the Qurʾān tutor system.

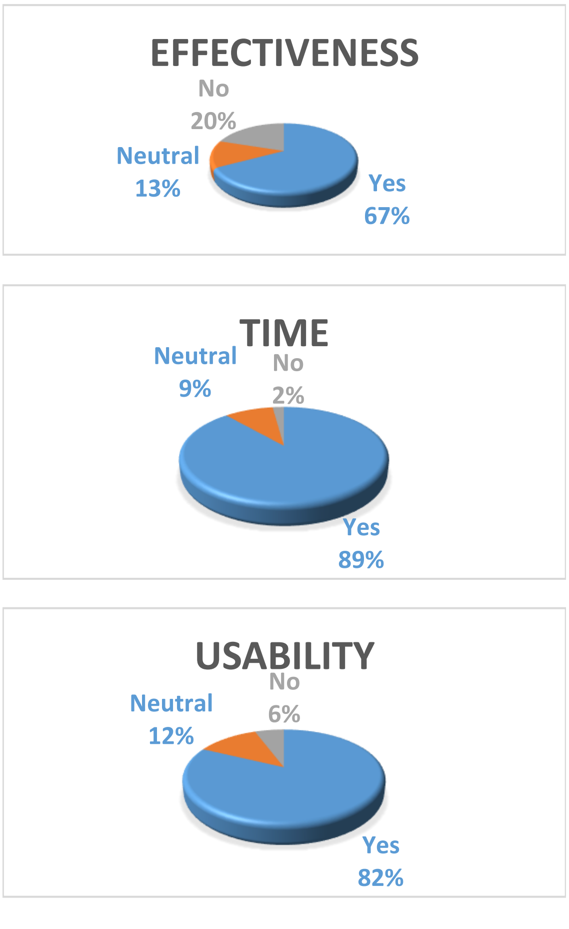


Fig. 6. Human evaluation of effectiveness, Time and Usability for Qurʾān tutor mobile application

D. *Result Discussion*

We emphasize that our Qurʾān tutor ASR model is only as good as the input data used to train it. The model will not achieve the desired level of accuracy or performance if important data inputs are missing or do not meet the quality standards for feeding the network. In addition, the more recordings from average people are in the training dataset, the more accurate the system will be, since such recordings include noise and more closely resemble the actual data that the system will receive during inference.

Furthermore, training the system for multiple surahs will not have a significant impact on the accuracy of the system compared with training it on a single surah. Because the Qurʾān has a limited vocabulary, training the system with more recordings of different users' recitations of a single surah will lead to greater accuracy than training the system to recognize multiple surahs but with fewer recordings for each surah.

The results of the human testing indicate that a system to listen to and correct recitations is highly needed. According to the questionnaire, around 80% of users preferred our app for reading and memorizing Qurʾān over traditional Qurʾān schools based on the accuracy and speed of learning, even though they spoke Arabic and were familiar with Qurʾānic pronunciation and recitation—which leads us to wonder about the global need for such a system.

Our system's speed in detecting and dynamically correcting errors was remarkable: All users checked for the speed feature. However, for one of the effectiveness metric statements ("I felt confident that the app will catch my mistake and correct it"), around 55% of respondents disagreed and stated that they did not feel confident that the application would catch their mistakes and correct them. This may be because it is a new learning method for Qurʾān and users are still unsure about its authenticity, meaning that they would feel more confident in front of a real human. In one questionnaire, a user proposed memorization progress as an application feature that could provide information on users' memorization progress.

Finally, to answer the second question of this research, the answer is yes based on the results of the questionnaire; around 79% of respondents chose "yes."

VII. Conclusion and Future Work

A. *Conclusion*

This research proposed a system that helps users recite the Qurʾān by simulating the functionality of a real human tutor. The system listens to the user's recitation, judges their memorization and pronunciation, instantaneously and dynamically helps them by voice if they forget words or get stuck, and warns them if they make a mistake. Our application improves the learning process, decreases and optimizes study time, enables users to learn from anywhere, and correct the student's recitation in a simple yet effective way to guide and assist the user in reading the Qurʾān. This research is significant because the produced system makes it easier for more than 24% of the world's population to recite and memorize the Qurʾān.

Throughout this research, we investigated the feasibility of building a speech recognition and correction system for Qurʾānic recitation based on ANNs. We accomplished our research objectives by implementing a Qurʾān tutor model that could recognize Qurʾānic recitations and convert the recognized speech into text in order to compare it for correction. We sought to answer the following questions:

- How can we build a system using neural networks that can dynamically correct a user's recitation of the first five chapters of the Holy Qurʾān?
- Will using an ANN model to dynamically correct the user's recitation of the first five chapters of the Holy Qurʾān is going to be convenient for the user in reciting the Qurʾān?

One characteristic that distinguishes our research from other research on the Arabic language is our focus on Qurʾānic recitations and the role DNNs can play in this context.

Multiple processes and stages were involved in building our system architecture. First, we collected the audio dataset, then prepared the audio files for the preprocessing stage to extract the features. We then trained the system with a DeepSpeech network composed of five layers of hidden units to create the AM and finally validated and tested the model. We then trained an LM for our system that included knowledge on alphabets, words, and sentence structure for the Arabic language, which improved the accuracy of the model. After that, we combined the AM and LM at inference using a beam search decoder to produce the corresponding words based on the probability of it. Finally, we built a mobile application to serve as the focal point between the user and STT model, compare the results, and interact with the user based on those results.

The experimental findings were discussed and summarized in the testing and results discussion section. Our model achieved 80% validation accuracy and 77% testing accuracy for the AM, a BLEU score of 0.681, and a PER score of 0.261. Based on the human testing results, our system's speed in detecting errors and correcting them dynamically is remarkable.

B. *Future Work*

The Qurʾān tutor system still needs improvement. As far as future work is concerned, further effort in collecting audio recitations is necessary to continue developing the model to cover the entire Qurʾān. In addition, collecting audio recitations from average people is preferable for raising the accuracy of the model. The mobile application needs to be improved to support multiple languages when alerting the user if a mistake is found to mimic a real Qurʾān tutor. Some interesting research questions for future development include the following:

- Would collecting and using recitations only from normal people (i.e., without expert recitations included) positively affect accuracy?
- Will there be a difference in the model's accuracy if we train the model separately on each surah, then combine the models for all surahs?

ACKNOWLEDGMENT

First and foremost, praises and thanks to Allah, the Almighty, for His showers of blessings throughout my research work to complete the research successfully.

I would like to express my deep and sincere gratitude to *My Family* for their love, prayers, caring, understanding and continuing support to complete this research work. I am extremely grateful to my research supervisor, Dr. Abdulaziz Almohimeed, for providing invaluable guidance throughout this research, it was a great privilege and honor to work and study under his guidance. Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

REFERENCES

[1] Simons,G.and D,C.(2018).Arabic-Ethnologue.[online] Ethnologue.Available at:https://web.archive.org/web/20160105171142/https://www.ethnologue.com/language/ara [Accessed 13 Feb. 2019].

[2] Ahmed, B.H.A., Ghabayen, A.S., (2017). Arabic Automatic Speech Recognition Enhancement, in: 2017 Palestinian International Conference on Information and Communication Technology (PICICT). Presented at the 2017 Palestinian International Conference on Information and Communication Technology (PICICT), IEEE, Gaza, Palestine, pp. 98–102. https://doi.org/10.1109/PICICT.2017.12

[3] Lipka, M. and Hackett, C. (2017). Why Muslims are the world's fastest-growing religious group. Retrieved from: http://www.pewresearch.org/fact- tank/2017/04/06/why-muslims-are-the- worlds- fastest-growing-religious-group/

[4] Pbs.org. (2020). Global Connections. Stereotypes | PBS. Retrieved from:http://www.pbs.org/wgbh/globalconnections/mideast/questions/types/index.html

[5] Juang, B.H., Rabiner, L.R., n.d. Automatic Speech Recognition – A Brief History of the Technology Development 24.

[6] Dong,Yu. Li, Deng., (2014). Automatic speech recognition A Deep Learning Approach. New York: Springer.

[7] Yousfi, B., Zeki, A.M., Haji, A., (2018). Holy Qur'an Speech Recognition System Distinguishing the Type of prolongation 2, 8.

[8] Al-Ayyoub, M., Damer, N. A. and Hmeidi, I. (2018) 'Using Deep Learning for Automatically Determining Correct Application of Basic Quranic Recitation Rules', 15(3), p. 7.

[9] Ahsiah, I., Noor, N.M., Idris, M.Y.I., (2013). Tajweed checking system to support recitation, in: 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS). Presented at the 2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS), IEEE, Sanur Bali, Indonesia, pp. 189–193. https://doi.org/10.1109/ICACSIS.2013.6761574

[10] Tarteel - AI for Perfecting Quran Recitation. n.d. Tarteel. [online] Available at: Tarteel.com

[11] Kenneth, K., n.d. [online] Kheafield.com.Available at: <https://kheafield.com/papers/avenue/kenlm.pdf>[Accessed 1 January 2021].

[12] DeepSpeech Model - DeepSpeech 0.6.1 documentation. (2019). DeepSpeech. https://deepspeech.readthedocs.io/en/v0.6.1/DeepSpeech.html

[13] Patel, S. (2018, October 5). Data Science essentials: Why train-validation-test data? Medium. https://medium.com/datadriveninvestor/data-science-essentials-why-train-validation-test-data-b7f7d472dc1f