COMMUNICATIONS IN
SCIENCE AND TECHNOLOGY
Homepage: cst.kipmi.or.id

Al-Quran recitation verification for memorization test using Siamese LSTM network

Rian Adam Rajagede[a,*], Rochana Prih Hastuti[b]

[a]*Department of Informatics, Faculty of Industrial Technology, Universitas Islam Indonesia, Sleman 55584, Indonesia*
[b]*Independent Scholar, Klaten 57417, Indonesia*

Article history:
Received: 2 April 2021 / Received in revised form: 2 June 2021 / Accepted: 3 June 2021

Abstract

In the process of verifying Al-Quran memorization, a person is usually asked to recite verses without looking at the text. This process is generally done together with a partner to verify the reading. This paper proposes a model using Siamese LSTM (Long Short-Term Memory) Network to help users to check their Al-Quran memorization alone. Siamese LSTM network will verify the recitation by matching the input with existing data for a read verse. This study evaluated two Siamese LSTM architectures: the Manhattan LSTM and the Siamese-Classifier. The Manhattan LSTM output a single numerical value representing the similarity, while the Siamese-Classifier used a binary classification approach. We also compared Mel-frequency Cepstral Coefficient (MFCC), Mel-Frequency Spectral Coefficient (MFSC), and delta features against model performance. We used the public dataset from Every Ayah website and provided the usage information for future comparison. Our best model, using MFCC with delta and Manhattan LSTM, produced an F1-score of 77.35%.

Keywords: Siamese Network; Long Short-Term Memory Network; Al-Quran Recitation

1. Introduction

Al-Quran is the holy book of Muslims, and memorizing it is a form of worship for Muslims. To test their Al-Quran memorization, a person is usually asked to recite one or more verses without looking at the text. A *Hafizh* (a person who memorizes Al-Quran) sometimes is not aware if he or she has made a mistake because he do not see the texts being read. That is why a partner is typically required in the process to give a feedback on whether the recitation is correct. When someone does not have a partner or has difficulty to meet their partner, verifying Al-Quran memorization might be challenging. To overcome this difficulty, it needs a system to check someone's recitation's correctness without human assistance. This system is required to detect whether the verse read in the memorization test is correct.

Creating a system to test whether someone's speech is correct can be done with several approaches. The first is by using a speech-to-text approach, where the features from the audio file are converted using a machine learning to obtain text or transcripts from the speech. After receiving the text, it can go to check the similarity of the text. Today's popular speech-to-text methods use a number of deep learning approaches such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), or a combination of both. [1-3]. Nevertheless, the disadvantage of this method is that the resulting model is relatively large. In the last Deep Speech model [2], released in late 2020, the pre-trained model size reached 180 MB for English. Previously in [4], a Deep Speech model was trained using only Al-Quran recitation data. However, the resulting Deep Speech model was more than 45 MB. A large model can reduce the flexibility of using the model on some platforms.

Another approach is by directly mapping an audio file to a class without extracting the text first. It directly classifies that the received audio signal is a speech of a particular digit, number, or letter [5,6]. In the case of speech verification, this means that the model will classify the speech as valid or not. This method is used in the small portion of speech-to-text to recognize the spoken phonemes before extracting the text [1]. Therefore, this method is usually used for speech recognition with a relatively short audio length, for example, a number, word, or command [5-7]. Both approaches have their advantages and disadvantages dependent upon what case they are used.

Even though the Al-Quran uses Arabic, there are several differences in speech recognition for Al-Quran compared to Arabic speech recognition. The difference lies in the rules for reading the Al-Quran or called *tajweed*, such as the tempo, reading style, and the accuracy of the reading letters' pronunciation. Several previous studies have specifically discussed these problems. In [8] proposed the Mel-Frequency Cepstral Coefficient (MFCC) and Hidden Markov Model (HMM) features to detect short lengths and the reading styles. The MFCC features is also used in [9] to detect one of the tajweed rules in combination with Vector Quantization. To detect the reading accuracy of difficult letters, in [6,10] proposed the use of a neural network, especially CNN.

* Corresponding author.
Email: rian.adam@uii.ac.id
https://doi.org/10.21924/cst.6.1.2021.344

This research proposed a different perspective for Al-Quran recitation verification system. We proposed it by using a combination of Siamese and Long Short-Term Memory (LSTM) Network to verify whether the Al-Quran recitation is correct without performing speech-to-text extraction process. Siamese LSTM Network has previously been used to verify the data similarity in the form of sequences, such as text data [11], fragments of images [12], or speech data for speaker verification [13]. Al-Quran recitation data can also be represented as a sequence that can be processed using the LSTM Network. Due to its specific task, we believed that this method is more efficient than the speech-to-text approach. The use of Siamese Network will change the representation of the problem to binary classification, whether the reading that is read matches the answer rather than classifying what verse is being read. Siamese network is also suitable for use in cases where the amount of training set for each class is small [14]. This will be very helpful considering that the speech features are quite large to hinder creating models with limited resources. Combining Siamese with LSTM network is done previously to handle data in sequences such as speech or text data [11,15]. In this study we used a publicly available dataset from Every Ayah (https://everyayah.com/) to facilitate comparison in future research.

2. Materials and Methods

2.1. Dataset

This study used publicly available datasets to facilitate comparisons in future research. The dataset consisted of five reciters who read the last ten Surahs of Al-Quran from Surah number 105 to 114 or specifically, there were 48 verses. The reciters are shown in table 1. The reciter code in table 1 was used on Every Ayah website to choose the specific reciters with specific audio quality. The audio files were downloaded using url http://www.everyayah.com/data<reciter_code>.

Of the five reciters, we used four reciters for training and one reciter for testing. In the training process, 30% of the train set samples were used as a validation set to avoid overfitting. We only used the testing set at the end of the experiment after obtaining the best models to avoid information leaks. In this study, we used the testing set from a different reciter from the training set. It made the evaluation process carried out to unobserved data commonly occurred in speech recognition evaluation.

Table 1. Reciter information

Reciter Code	Data Usage
Alafasy_64kbps	Training
Hani_Rifai_64kbps	Training
Maher_AlMuaiqly_64kbps	Training
Muhammad_Ayyoub_64kbps	Training
Khalefa_al_tunaiji_64kbps	Testing

The Siamese network dataset was built by pairing two audio files and then labeled as 1 if both verses read were the same and 0 if the verses were different. By pairing verses between the reciters, the combination resulted in an imbalance class with many labeled 0 verse pairs. Training the model with imbalance class was deemed necessary to make the model more robust in determining the correct answer. However, to avoid a significant gap between both classes, we downsampled the majority class by randomly sampling the data so that the distribution of training set and test set appeared as shown in table 2.

Table 2. Class distribution

Dataset	Correct	Incorrect
Train set	288	9006
Test set	192	253

Apart from pairing the two audio files, we also created a small data set named the inference set. The data consisted of 36 audio files randomly chosen from 12 verses from reciter 5. The chosen verses are shown in table 3. The data were also used to evaluate the model but differently from the test set. The inference set was used when simulating the model, checking whether the recitation was correct with the expected verse. The dataset consisted of 36 samples: 12 correct samples; 12 samples modified from the original recitation by cropping 30% of the end of the audio; and 12 samples that did not match the verse that the reciter should read.

Table 3. Verses used in inference set (surah/ayah)

105/004	106/004	107/002	107/007
108/001	109/003	110/002	111/002
112/004	113/001	113/004	114/006

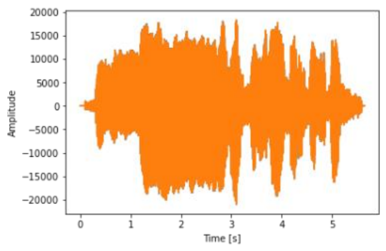
2.2. Feature extraction

All data used were turned into a mono channel and set to have a frame rate of 16 kHz. Two feature extraction methods were used separately: Mel-frequency Cepstral Coefficient (MFCC) and Mel-frequency Spectral Coefficient (MFSC). Feature extraction was carried out using Python Speech Features (https://.github.com/jameslyons/python_speech_features). MFCC is a commonly used method as in [10,16], while the MFSC method was originally proposed for the CNN architecture in [1] and it was also used for Arabic datasets in [6]. The MFSC method ignored the Discrete Cosine Transform (DCT) step in MFCC so it only used spectral values instead of its inverse, cepstral.MFCC produced 13 coefficients for each frame, while the MFSC produced 26 coefficients, without using the energy features. In this study, the frame width of 30 milliseconds was used. Each data then had a different number of frames dependent on the original length of the audio. Figure 1 shows the plot of MFCC features for an audio file.

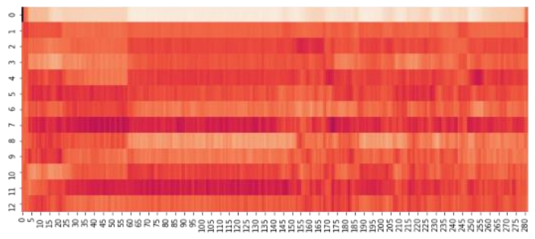
The features of MFCC and MFSC were normalized using Cepstral Mean Normalization (CMN) before feeding it to the model. It was used to reduce channel distortion in the data caused by different recording environments and devices. CMN was calculated on each coefficient between frames using (1) and (2). In that equation, o_i^t refers to the value of the i-th coefficient in the t-th frame, where T is the number of frames in each sample. The standard deviation of the i-th coefficient, σ_i, calculated on the same dimension as the mean value μ_i.

$$\mu_i = \frac{1}{T}\sum_{t=1}^{T} o_i^t \quad (1)$$

$$\overline{o_i^t} = \frac{o_i^t - \mu_i}{\sigma_i} \quad (2)$$

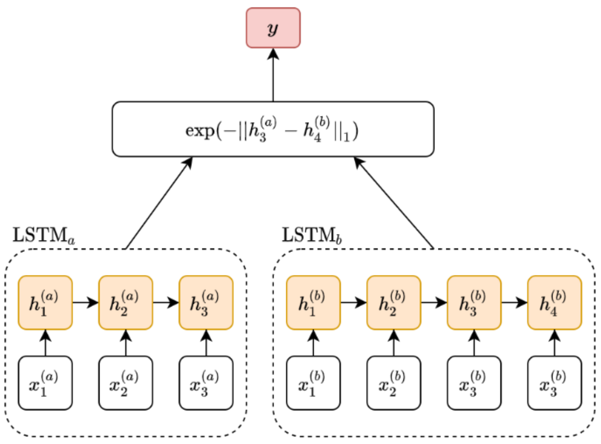


(a) original audio



(b) mel-frequency cepstral coefficients

Fig. 1. (a) Original audio plotted in time domain (b) Extracted MFCC features from original audio, 281 frames and 13 features



y
$\exp(-\|h_3^{(a)} - h_4^{(b)}\|_1)$
LSTM$_a$
LSTM$_b$
$h_1^{(a)}$ $h_2^{(a)}$ $h_3^{(a)}$
$x_1^{(a)}$ $x_2^{(a)}$ $x_3^{(a)}$
$h_1^{(b)}$ $h_2^{(b)}$ $h_3^{(b)}$ $h_4^{(b)}$
$x_1^{(b)}$ $x_2^{(b)}$ $x_3^{(b)}$ $x_3^{(b)}$

Fig. 2. Illustration of Manhattan LSTM architecture

In addition, we explored the usage of delta features from MFCC, as used in [1], [6]. Delta features of the t-th frame were computed using (3), where c_{t+1} and c_{t-1} were the coefficients of MFCC from previous and next frame. We also normalized the delta features using CMN. When a model used delta features, we concatenated MFCC and delta features to double the feature vector length.

$$d_t = \frac{c_{t+1} - c_{t-1}}{2} \quad (3)$$

2.3. Siamese LSTM models

Siamese network [17] is a type of neural network that is used to verify the similarity of two samples. The Siamese network architecture consists of two models with the same weight and parameters but receiving different inputs. Both inputs were then combined into one. The output was usually a single value that represented the similarity of the two data. Due to the data in the form of sequences, this study combined Long Short-Term Memory (LSTM) network with Siamese architecture.

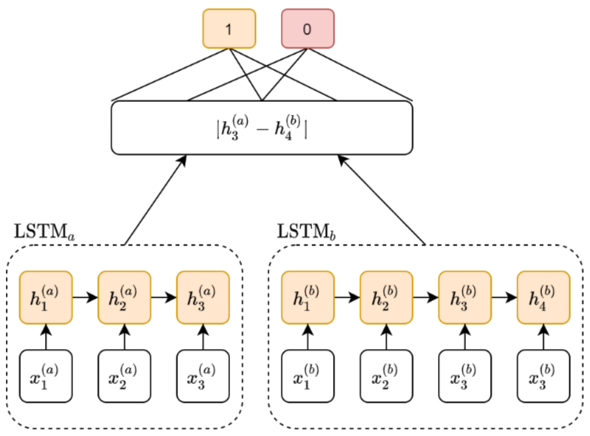
LSTM is a type of neural network architecture designed for data in sequences [18]. It improved the Recurrent Neural Network (RNN), which excelled in preserving information in long sequences. In this study, the hidden state of the last sequence, h_{last}, was used as the LSTM model output and then compared with other inputs in the Siamese network. The hidden state value at time t was calculated using (4).

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad (4)$$

In Equation (4), x_t is the input at time t. Meanwhile, i_t, f_t, g_t, o_t are LSTM gates, which differentiate LSTM with RNN. Sigmoid function, σ, used as non-linear activation function. This research used multilayer LSTM. In the next layer, the input of the model at time t was obtained from the hidden state of the previous layer at time t.

We evaluated two types of Siamese-LSTM networks. The first type was the Manhattan LSTM (MaLSTM) Network proposed in [11]. MaLSTM used two LSTM networks with the same weights to process data in the form of sequences. The illustration is shown in figure 2. The input $x_i^{(a)}$ is the coefficient vector in i-th frame for data a. The final output of each LSTM was combined and then the similarity value was calculated using a function based on Manhattan distance. The single value as the result of this difference was the output of the MaLSTM model.

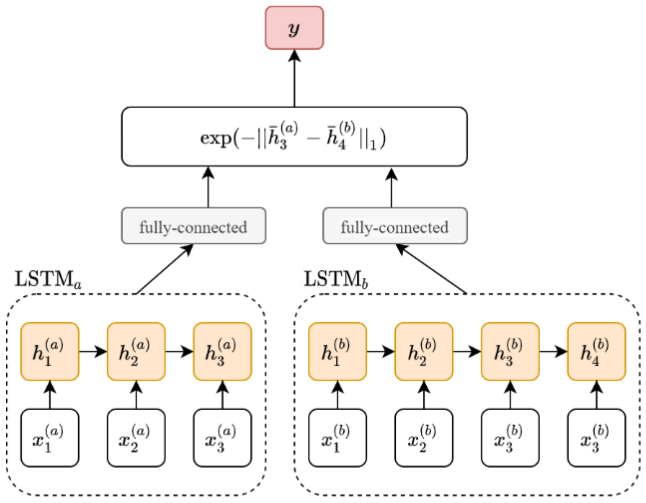
For the second type, the Siamese-LSTM network produces output in binary class, correct or not. The absolute difference of the two LSTM networks' output is calculated and then fed as input to the fully connected layer with the two outputs as illustrated in figure 3. This architecture is similar with the one used in [19]. Henceforth, the first model will be referred to as MaLSTM, while the second model is called Siamese-Classifier.



1
0
$|h_3^{(a)} - h_4^{(b)}|$
LSTM$_a$
LSTM$_b$
$h_1^{(a)}$ $h_2^{(a)}$ $h_3^{(a)}$
$x_1^{(a)}$ $x_2^{(a)}$ $x_3^{(a)}$
$h_1^{(b)}$ $h_2^{(b)}$ $h_3^{(b)}$ $h_4^{(b)}$
$x_1^{(b)}$ $x_2^{(b)}$ $x_3^{(b)}$ $x_3^{(b)}$

Fig. 3. Illustration of Siamese-Classifier architecture

We also explored the addition of as fully connected layer with 200 neurons before the similarity function in MaLSTM, as shown in figure 4. We added a fully connected layer that received output from the LSTM to add to the complexity of the model as is done in a typical LSTM model. We used $\bar{h}$ to represent the outputs from additional fully connected layer in the figure.



y
$\exp(-\|\bar{h}_3^{(a)} - \bar{h}_4^{(b)}\|_1)$
fully-connected
fully-connected
LSTM_a
LSTM_b
$h_1^{(a)}$ $h_2^{(a)}$ $h_3^{(a)}$
$h_1^{(b)}$ $h_2^{(b)}$ $h_3^{(b)}$ $h_4^{(b)}$
$x_1^{(a)}$ $x_2^{(a)}$ $x_3^{(a)}$
$x_1^{(b)}$ $x_2^{(b)}$ $x_3^{(b)}$ $x_4^{(b)}$

Fig. 4. MaLSTM with additional fully-connected layer

Both models were trained using Adam optimizer [20] and implemented using PyTorch framework [21]. We initialized the weight and bias model with Glorot Uniform [22], as shown in (5). This method initialized weights and biases by randomly select a value between $-\frac{1}{n}$ to $\frac{1}{n}$, where n is the hidden size in the LSTM model or the number of input neurons in the fully connected layer.

$$W, b \sim U\left(-\frac{1}{n}, \frac{1}{n}\right) \tag{5}$$

The cost function used in the MaLSTM model is the sum of the squared error for handling imbalance class. Meanwhile, the Siamese-Classifier uses a weighted cross-entropy with a weight ratio of 1:4. Weighted cross-entropy is calculated using (6) where x is the output vector, c is the prediction class, and w is the class's weight vector.

$$Cost(x, c) = w[c]\left(-x[c] + \log\left(\textstyle\sum_j exp(x[j])\right)\right) \tag{6}$$

2.4. Experimental stages

During the training process, the model was evaluated using the validation set to find the best parameters. We attempted to find some parameters such as architecture, learning rate, and the number of training epochs. When finding parameters, we only used the dataset with MFCC features. After finding the best parameters in the validation set, the model was trained with all training sets, including the validation set, and was evaluated using the test set and the inference set.

In the test set, the Precision, Recall, and F1-score (7) were used to evaluate the model. We used the F1-score as an evaluation metric because the data had class imbalance. The number of pairs with label 0 was more than those with label 1. The use of F1-score for the imbalance class was considered better than using accuracy because of its ability to measure the model's performance on the smaller class [23]. Besides the F1-score, the model precision also needed to be considered. Models with high precision values can avoid false positives where the model confirms recitations that are false. Precision is calculated using (8).

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

$$\text{Precision} = \frac{\textit{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{8}$$

3. Results and Discussion

3.1. Hyperparameters search

After several trials using validation set, we selected five architectures that performed reasonably well. The selected architectures can be seen in table 4. Model that uses additional layer as shown in Fig. 4 is indicated in the Additional Layer column in table 4.

The learning rate, number of epochs, and threshold values used are shown in table 5. The number of epochs was selected when the validation loss obtained during training has not decreased or tended to start to increase (overfitting). MaLSTM model has an additional hyperparameter, threshold, to determine whether two input data are the same. The threshold parameter in MaLSTM is one of the drawbacks because finding the best value requires an additional search process. The threshold that produces the best score can differ depending on the data, model, and even other hyperparameters such as learning rate. This is not experienced by Siamese-Classifier, which immediately classifies whether two audio files read the same verse.

Table 4. Models architecture

Model	Type	LSTM hidden neuron	LSTM layers	Add. layer
A	Siamese Classifier	200	3	No
B	MaLSTM	200	3	No
C	MaLSTM	200	3	Yes
D	Siamese Classifier	300	3	No
E	Siamese Classifier	300	2	No

Table 5. Model hyperparameter

Model	Epoch	Learning rate	Threshold
A	45	0.00002	-
B	190	0.002	0.005
C	305	0.001	0.018
D	50	0.00002	-
E	50	0.00002	-

3.2. Evaluation on test set

After the training process, the five models with the best hyperparameters were evaluated with the test set. The results obtained are shown in table 6 showing that Model B had the best performance on the F1-score than other four models. Model B was the original MaLSTM model that the output was a similarity score. Model C, which used MaLSTM model with an additional layer, had the second-best performance in the F1-score evaluation. The highest Precision value was owned by Model D, but this model had a low F1-score. In contrast, model B with the highest F1-score value, had the lowest Precision value. The robust model we expected had a similar high value both in F1-score and precision.

Table 6. Models performance on test set

Model	Precision	Recall	F1-Score
A	71.72	54.16	61.72
B	67.61	**86.97**	**76.10**
C	70.77	80.73	75.43
D	**75.69**	56.77	64.88
E	71.97	58.85	64.76

3.3. Features comparison

As shown in table 7, we compared the model performance on a dataset using the MFSC feature. It can be seen that the best model with MFCC features, i.e. model B, outperformed all models with MFSC features. Even though it was reported [1], [6] that MFSC helps to produce better model performance because it maintains locality features, we assumed that it depends on the architecture used in the study. In this study, as we used LSTM instead of Convolutional Neural Network (CNN) used in [1,6], which processes data as a sequence, it appeared that the MFCC feature is more recommended.

Table 7. Results comparison between MFCC and MFSC

Model	Features	F1-score
B	MFCC	**76.10**
A	MFSC	52.96
B	MFSC	70.24
C	MFSC	74.24
D	MFSC	60.26
E	MFSC	56.03

Table 8. Results comparison after using delta features on test set and inference set

Model	Precision	Recall	Test set F1-score
B	67.61	**86.97**	76.10
B + Delta	70.23	78.65	74.2
C	80.73	80.73	75.43
C + Delta	**75.62**	79.17	**77.35**

After MFCC seemed to outperform MFSC, we explored the use of additional delta features of MFSC. We used MaLSTM models B and C, and then tested them on the test set and inference set. The results are shown in table 8 in which it can be seen that the delta features succeeded in increasing the F1-score on the test set especially for model C.

As shown in table 8, all models used MFCC as features, except models with "+ Delta" that concatenated delta features on the feature vector. It can be seen that delta features not only increased the F1-score, but also made model C to have the highest precision value compared to the other four models and slightly lower than model D as depicted in table 6.

3.4. Evaluation on inference set

The next stage of evaluation was carried out using the inference set. At this step, we made a prototype using the best model to verify whether a recitation for the specific ayah was correct. The audio file, along with the verse number read, was input into the system, and then the system would conduct inference four times by pairing the input data with four reciters that read the same verse. The four reciters were those used in the training set as shown in table 1.

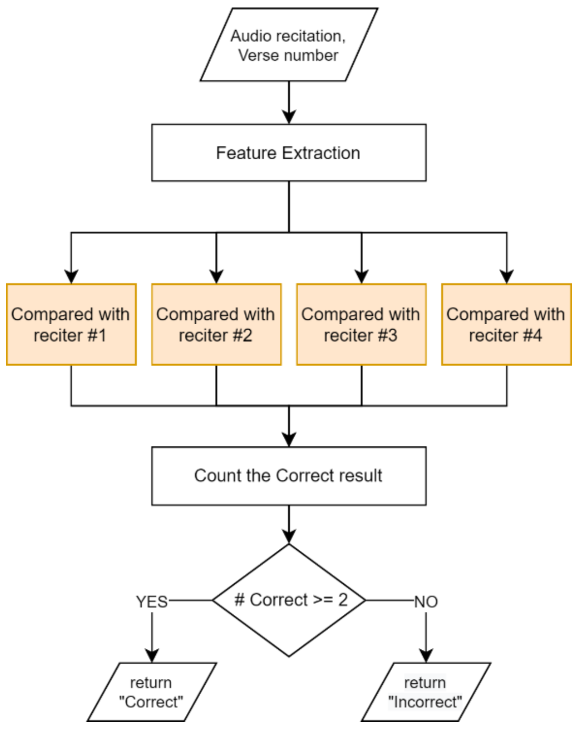


Fig. 5. Evaluation flow on inference set second sub-figure

We stored the records of the four reciters in a features matrix. When the system received a recording, the system sought the corresponding verse from the four reciters on the features matrix and then performed the inference process using the model four times. The process is shown in figure 5. To determine the system output, we used a rule: if at least two of the four models return a "correct" value, then the audio will be labeled as correct or match the verse read.

The F1-scores we obtained for the evaluation using the inference set are shown in table 9. As seen in the table, model C with delta features consistently outperformed other models both on F1-score and Precision value.

3.5. Evaluation on model size

We saved the features matrix using Pickle, and the resulting file size was 10.4 MB. Model C was saved using PyTorch serialization, resulting in 3.4 MB in size. It can be seen that, the total storage required for the inference process was only 13.8 MB, quite efficient compared to the general pre-trained model mentioned before.

Table 9. Models performance on inference set

Model	Precision	Recall	F1-score
A	53.33	66.67	59.26
B	62.5	83.33	71.43
C	62.5	83.33	71.43
D	58.82	83.33	68.97
E	58.82	83.33	68.97
B + Delta	71.43	83.33	76.92
C + Delta	90.00	75.00	**81.82**

4. Conclusion

In this study, we proposed the use of the Siamese LSTM network to verify the Al-Quran recitation for memorization test. We evaluated two types of models, i.e. the MaLSTM and Siamese-Classifier, which used two different approaches. In addition, we experimented with some feature extraction methods, namely MFCC, MFSC, and Delta. The model was trained using data from four reciters who read 48 verses from the last ten Surahs of Al-Quran and tested using different reciters. We selected the best model based on the F1-score in the test set. Our best model using MaLSTM with additional fully connected layer and using MFCC and delta features got an F1-score of 77.35%. Compared to the pre-trained speech-to-text model, our model only required small storage for inference. For the future research, it is suggested to use more complex model, for example a deeper Siamese network or using attention-based model. Train the model with more data, different speakers, or fine tune from the available pre-trained model may achieve a better performance in Al-Quran recitation verification.

References

1. O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, *Convolutional neural networks for speech recognition*, IEEE/ACM Trans. Audio, Speech, Lang. Process. 22 (2014) 1533–1545.
2. A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, et al., *Deep speech: scaling up end-to-end speech recognition*, arXiv Prepr. arXiv1412.5567, 2014.
3. D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, et al., *Deep speech 2: end-to-end speech recognition in english and mandarin, Int. Conf. Mach. Learn., New York City, NY, USA, 2016, pp. 173–182*.
4. E. Tareek, *Project DeepSpeech Quran*, Github repository, https://github.com/tarekeldeeb/DeepSpeech-Quran (accessed 26 May 2021).
5. N. Hammami and M. Sellam, *Tree distribution classifier for automatic spoken arabic digit recognition, Int. Conf. Internet Technol. Secur. Trans., London, UK, 2009, pp. 1–4*.
6. R. A. Rajagede, C. K. Dewa, and Afiahayati, *Recognizing arabic letter utterance using convolutional neural network, 18th IEEE/ACIS Int. Conf. SNPD, Kanazawa, Japan, 2017, pp. 181–186*.
7. H. A. Elharati, M. Alshaari, and V. Z. Këpuska, *Arabic speech recognition system based on MFCC and HMMs*, J. Comput. Commun. 8 (2020) 28–34
8. B. Yousfi, A. M. Zeki, and A. Haji, *Holy qur'an speech recognition system distinguishing the type of prolongation*, Sukkur IBA J. Comput. Math. Sci. 2.1 (2018) 36-43.
9. A. Ismail, M. Yamani, I. Idris, N. M. Noor, Z. Razak, and Z. Yusoff, *MFCC-VQ approach for qalqalah tajweed rule checking*, Malaysian J. Comput. Sci. 27 (2014) 275-293.
10. E. S. Wahyuni, *Arabic speech recognition using MFCC feature extraction and ANN classification, 2nd Int. Conf. Inf. Technol. Inf. Syst. Elect. Eng., Yogyakarta, Indonesia, 2017, pp. 22–25*.
11. J. Mueller and A. Thyagarajan, *Siamese recurrent architectures for learning sentence similarity, 30th AAAI Conf. Artif. Intell., Phoenix, AZ, USA, 2016, pp. 2786-2792*.
12. R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang, *A siamese long short-term memory architecture for human re-identification, Comput. Vis. ECCV, Amsterdam, The Netherlands, 2016, pp. 135-153*.
13. K. Sriskandaraja, V. Sethu, and E. Ambikairajah, *Deep siamese architecture based replay detection for secure voice biometric, INTERSPEECH, Hyderabad, India, 2018, pp. 671-675*.
14. J. Zhang, X. Jin, Y. Liu, A. K. Sangaiah, and J. Wang, *Small sample face recognition algorithm based on novel siamese network*, J. Inf. Process. Syst., 14 (2018) 1464-1479.
15. P. Neculoiu, M. Versteegh, M. Rotaru, and T. B. V Amsterdam, *Learning text similarity with siamese recurrent networks, RepL4NLP-2016, Berlin, Germany, 2016, pp. 148-157*.
16. M. Bezoui, A. Elmoutaouakkil, and A. Beni-Hssane, *Feature extraction of some quranic recitation using mel-frequency cepstral coeficients (MFCC), 5th Int. Conf. Multimedia Comput. Syst., Marrakech, Morocco, 2016, pp. 127–131*.
17. J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore et al., *Signature verification using a 'siamese' time delay neural network*, Int. J. Pattern Recognit. Artif. Intell. 7 (1993) 669–688.
18. S. Hochreiter and J. Schmidhuberx, *Long short-term memory*, Neural Comput. 9 (1997) 1735–1780.
19. J. Wang, Y. Qin, Z. Peng and T. Lee, *Child speech disorder detection with siamese recurrent network using speech attribute features, INTERSPEECH, Graz, Austria, 2019, pp. 3885-3889*.
20. D. P. Kingma and J. L. Ba, *Adam: A method for stochastic optimization, 3rd Int. Conf. Learn. Representations, San Diego, CA, USA, 2015*.
21. A. Paszke, et al., *PyTorch: an imperative style, high-performance deep learning library*, Adv. Neural Inf. Process. Syst. 32 (2019) 8026–8037.
22. X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, J. Mach. Learn. Res. 9 (2010) 249–256.
23. G. Forman and M. Scholz, *Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement*, SIGKDD Explor. 12 (2010) 49-57.