



# Arabic speech recognition using end-to-end deep learning

Hamzah A. Alsayadi<sup>1,2</sup>  | Abdelaziz A. Abdelhamid<sup>2,3</sup>  | Islam Hegazy<sup>2</sup> | Zaki T. Fayed<sup>2</sup>

<sup>1</sup>Computer Science Department, Faculty of Sciences, Ibb University, Ibb, Yemen

<sup>2</sup>Computer Science Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

<sup>3</sup>College of Computing and Information Technology, Shaqra University, Shaqra, Saudi Arabia

## Correspondence

Hamzah A. Alsayadi, Computer Science Department, Faculty of Sciences, Ibb University, Ibb, Yemen.

Email: [hamzah.sayadi@cis.asu.edu.eg](mailto:hamzah.sayadi@cis.asu.edu.eg)

## Abstract

Arabic automatic speech recognition (ASR) methods with diacritics have the ability to be integrated with other systems better than Arabic ASR methods without diacritics. In this work, the application of state-of-the-art end-to-end deep learning approaches is investigated to build a robust diacritised Arabic ASR. These approaches are based on the Mel-Frequency Cepstral Coefficients and the log Mel-Scale Filter Bank energies as acoustic features. To the best of our knowledge, end-to-end deep learning approach has not been used in the task of diacritised Arabic automatic speech recognition. To fill this gap, this work presents a new CTC-based ASR, CNN-LSTM, and an attention-based end-to-end approach for improving diacritised Arabic ASR. In addition, a word-based language model is employed to achieve better results. The end-to-end approaches applied in this work are based on state-of-the-art frameworks, namely ESPnet and Espresso. Training and testing of these frameworks are performed based on the Standard Arabic Single Speaker Corpus (SASSC), which contains 7 h of modern standard Arabic speech. Experimental results show that the CNN-LSTM with an attention framework outperforms conventional ASR and the Joint CTC-attention ASR framework in the task of Arabic speech recognition. The CNN-LSTM with an attention framework could achieve a word error rate better than conventional ASR and the Joint CTC-attention ASR by 5.24% and 2.62%, respectively.

## 1 | INTRODUCTION

Traditional automatic speech recognition (ASR) systems use an architecture with multiple components, such as lexicon building, language model, and acoustic model. These components are built and processed separately [1]. The traditional ASR approach includes several methods such as Gaussian mixture models (GMMs), hidden Markov models (HMMs), a deep neural network (DNN) and a hybrid HMM-DNN [2]. In contrast, end-to-end approach components can be trained and manipulated in one package using deep learning methods. All the components in this approach may be considered more optimal compared with the traditional components [1].

The data linguistic knowledge is not used in an end-to-end approach. In addition, all the components can be trained using one algorithm [2]. Some researchers have tried to investigate the work-flow processes in this approach using different neural models and properties [2, 3]. However, the models of end-to-

end ASR are still under research [1]. Several methods and technique are included in the end-to-end ASR architecture such as a recurrent neural network (RNN) transducer, connectionist temporal classification (CTC), convolutional neural networks (CNNs), long short-term memory networks (LSTMs), attention-based encoder-decoder, and hybrid models [4].

The CTC method has an intermediate label that allows for the repetition of labels and occurrences of blank labels to recognise a no data label [5]. On the one hand, CTC makes a strong independent assumption between the labels. Thus, it cannot perform well without a strong language model (LM) [6]. In addition, CTC ignores the need for pre-segmented training data [7]. For solving the strong LM problem in the CTC method, attention-based methods are utilised for training a decoder that generates labels depending on the previous ones generated [6]. The attention-based encoder-decoder directly learns mapping from the acoustic frame to the

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2021 The Authors. *IET Signal Processing* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

character sequences [5]. In speech recognition tasks, the alignment between the input features and the output symbols is usually monotonic. CTC uses the so-called latent path to represent this alignment [6]. In addition, each input in ASR is a signal with variable length. So, both the methods address the problem of variable-length input and output sequences [5]. The combined CTC and attention-based end-to-end system decreases the word error rate (WER) more than other approaches [6].

CNN is a suitable method for ASR according to the spectral variations and local correlation features in the speech signal. The pooling layer in CNN includes local filter and weight sharing processes. These processes are used to remove the noise in features and shift the small frequency.

In addition, LSTM is used as a higher-level technique to represent the acoustic data that leads to the enhancement of the recognition rate. LSTM has a memory block that enables it to handle time dependencies in the learning problem resulting from vanishing and exploding gradient processes. Thus, the hybrid CNN-LSTM is considered a state-of-the-art approach in end-to-end systems.

In recent years, an external LM has been utilised in end-to-end ASR for improving the accuracy and performance of the ASR. This technique is called shallow fusion, where an external language model is used with the decoder network by employing log probability during the decoding process [4]. There are several open-source ASR toolkits, such as Kaldi [8], Eesen [9], OpenSeq2Seq [10], and ESPNet [11], based on acoustic modelling in the end-to-end approach using graphemes instead of phonemes. In addition, this model is preferred because of the following reasons: 1) the models based on the end-to-end approach are not complex; 2) the accuracy is enhanced compared to traditional systems [12].

Although Arabic is one of the most common languages in the world and considered as the fifth language worldwide [13, 14], there are very little research studies on speech recognition of this language [13, 14]. Arabic ASR is a difficult task due to several challenges like sparsity of language data, lexical variety [15], the presence of different dialects that are spoken around the Arab world [16], and the predominance of non-diacritised text material [13]. In addition, the Arabic language is challenging for the speech research community as well because of its morphological complexity [17] and large-vocabulary ASR.

There are three forms of the Arabic language: 1) Classical Arabic is the language that was used by Arabic people about 10 centuries ago, which is considered the language of the Muslim holy book and old Arabic poetry; 2) modern standard Arabic (MSA), which uses the form and characteristics of classical Arabic without some features such as syntax structure and diacritics. MSA is used in formal communication, news, modern books, and newspapers; 3) dialectal Arabic, which is used as a non-formal form in daily life for communication between people. Every country or region in the Arab world uses a different dialect. Each type of Arabic dialect has different grammatical, lexical, and morphological standards. It is considered the language of social media nowadays. The

difficulties in building Arabic ASR applications arise from the varieties in dialectal Arabic [18].

On the other hand, Arabic language can be written with or without diacritics, i.e. it has a diacritised and non-diacritised text. Diacritics are marks that play an important role for understanding the meaning of a word. Each word in Arabic has a different meaning according to its diacritics. Arab speakers and readers can read and understand a non-diacritised text. They use the context of the word in a sentence to understand its meaning. It is highly challenging for machines and non-native speakers to understand the intended meaning of a non-diacritised Arabic text [19, 20].

In this work, we introduce the application of the most recent methods in end-to-end speech recognition for the task of Arabic ASR. To the best of our knowledge, this work is the first study to propose these methods for the task of diacritised Arabic ASR. For this purpose, this work proposes a new method, which combines CTC-based ASR with an attention-based the end-to-end ASR in the system. Furthermore, this paper hybridises CNN-LSTM with the attention-based encoder-decoder as another new method for diacritised Arabic ASR. Based on the conducted experiments, significant conclusions are obtained regarding this approach, including the following:

- End-to-end systems show better accuracy and performance than other Arabic ASR systems.
- End-to-end state-of-the-art technologies with a diacritics version show promising performance, which makes them suitable for building robust Arabic ASR models.
- Diacritised Arabic ASR systems perform better than non-diacritised ones.

The rest of this work is organised as follows: In Sections 2, we describe works related to Arabic ASR using some techniques such as HMM-based, neural network, recurrent neural network, deep learning etc. and also include previous works in Arabic ASR that are developed using the end-to-end approach. In Section 3, we illustrate the research methodology. Section 4 includes the experimental results. Finally, Section 5 contains the conclusion and future perspectives.

## 2 | RELATED WORK

Diacritised Arabic ASR has an advantage and disadvantage compared with non-diacritised Arabic ASR. The advantage lies in its ability to integrate the diacritised version with other systems, such as voice-enabled translation, better than the non-diacritised version. Its disadvantage is that the diacritised version will decrease the accuracy compared with the non-diacritised version. So most researchers prefer to use the non-diacritised version in order to increase the accuracy. Most works that are presented in this section are built using the non-diacritised version. Special attention will be on Arabic ASR using conventional ASR and end-to-end ASR approaches in this section.

## 2.1 | Conventional Arabic ASR

Khatatneh et al. [21] presented a system for Arabic phoneme recognition. They used many techniques for enhancing the result. In the pre-processing stage, the Gaussian low pass filtering algorithm was used with the neural network. Then, for recognising the speech signal, the neural network was used to train the system. The proposed work included sampling, catching a signal, setting energy, and quantisation for recognising the phoneme. They concluded that the neural network improves the results.

Ahmed et al. [22] have introduced different approaches for improving Arabic ASR: 1) The decision tree used a variant pronunciation generation system to build the punctuation modelling, 2) the native acoustic model was adapted with another model using a hybrid approach, and 3) the processed text was used to enhance the language model. In the results, WERs were decreased by 1% using the pronunciation model and 1.2% by the acoustic modelling, while the language model reduced the WER by 1.9%.

Hamed and Allen [23] suggested biologically inspired methods for building the Arabic ASR. The system includes different structures for evaluating performance. The main parts were used were as follows: firstly, the features were generated by MFCC steps. Then, they adapted dataset normalisation for model training and testing. A single Multi-Layer Perceptron (MLP) identification system achieved 47.52% accuracy and the accuracy of the category-based phonemes recognition system was 44.58%, while the last system, the individual phoneme classifier system achieved 46.63% accuracy.

Bouchakour and Debyeche [24] presented methods for enhancing the Arabic ASR's performance in mobile communication systems. They used a front-end module to extract the features and a back-end module to build the recognition phase. In the front-end module, Multi-taper Frequency Cepstral Coefficients features (MFCC-MT) and Gabor features (GF-MFCC) were used to build the acoustic features, while the back-end module trained the acoustic model using Continuous Hidden Markov Models (CHMM), DNN, and the HMMDNN hybrid. The authors reported that the HMMDNN method resulted in the best accuracy. It enhanced the accuracy by 8%, 13%, and 8.5% for clean speech, the AMR-NB coder, and the DSR coders, respectively.

Mousa et al. [25] presented a novel approach for Egyptian dialect Arabic ASR. They integrated the best feature-rich modelling with the DNN-LMs and morpheme-based LMs to represent the features. They merged the words and morphemes with their features. The results showed an accuracy better than the traditional word-based LMs.

AlHanai et al. [26] have developed an ASR system for Arabic using a 1200-h speech corpus. The system was built using a deep neural network (DNN), where the DNN structure included many techniques: feed-forward, time delay, convolutional, Highway LSTM (HLSTM), Recurrent (LSTM), and Grid LSTM (GLSTM). The evaluation achieved the best result, with 18.3% as the WER using a trained GLSTM model on the selected corpus.

Cardinal et al. [16] described the comparison for several of the state-of-the-art ASR techniques. These systems used a news channel "Al Jazeera" corpus (contain 50-h of transcription audio) for training. The broadcast report (BR), broadcast conversations (BC), and combined data were used for evaluation. The hybrid DNN/HMM approach used the MPE (Minimum Phone Error) criterion for training using a sequential DNN. The sequential DNN achieved the best result. The results obtained were 17.86%, 29.85%, and 25.6% of WER for BR, BC, and combined, respectively.

Ahmed et al. [27] developed a speech recognition system for Arabic news. They wrote the KALDI recipe to build this system. The acoustic model was trained using the broadcast news system with 200 h. The BR, BC, and combined were used for evaluation. The system has included the details of text normalisation, vowelisation, and language models. The results achieved were 15.81%, 32.21%, and 26.95% of WER for BR, BC, and combined, respectively.

Tomashenko et al. [28] described the framework of the 2016 Multi-Genre Broadcast (MGB-2) Challenge of Arabic ASR. The main goal of this system is building an acoustic model for training the GMM-derived features using DNN. This acoustic model was integrated with time-delay neural networks to recognise Arabic words. The integrated system achieved the best result which was 9% of WER for the best single LIUM ASR system.

Ettaouil et al. [29] suggested a hybrid model, ANN/HMM, for building the Arabic digit system. They used the Self-Organising Maps method to generate the optimal codebook and the optimal class based on the optimal neural network. The results show the effects of the dictionary size on the classification. The codebook vector with size 34 had a recognition rate of 84%, the one with size 36 had 85%, and the one with size 48 had 86% recognition rate.

Wahyuni [30] presented a system to recognise spoken Arabic letters, which is a challenging issue for Indonesian speakers. The proposed methods were built using MFCC and ANN to differentiate between the pronounce of (tsa, sa, and sya) letters. The accuracy of this work was 92.42%.

AbdAlmisreb et al. [31] proposed the DNN method to develop Arabic ASR. DNN consists of 3 hidden layers, 500 Maxout units and two neurons for each unit. The features were extracted using the MFCC steps. The approach of this work used a corpus, which includes 20 Malay speakers of Arabic phonemes, for training and testing. They used 5 waveforms for the trained set and 15 waveforms for the tested set. In the results, the Maxout-based deep structure achieved accuracy and performance better than other deep networks such as the CNN, Restricted Boltzmann Machine, the conventional feed-forward neural network, Deep Belief Network, and the Convolutional Auto-Encoder.

## 2.2 | End-to-end Arabic ASR

Zerari et al. [32] presented a general end-to-end approach for isolated Arabic word recognition based on the Arabic spoken

digit spelling. This approach used MFCC and the relevant features from the natural speech signal as its features. The extracted features were presented by a DDN to adapt the non-uniformity of the sequences' length in the speech utterances. They firstly used LSTM with three gates to decode the sequences of the features as fixed vectors. Then a multi-layer perceptron network received these vectors to produce the classification. The spoken Arabic digit dataset was used in this work, which consists of 8800 tokens by 88 individual Arabic native speakers. The accuracy of this approach achieved 98.77% of overall f-measure.

Smit et al. [33] evaluated the character-based modelling using a state-of-the-art ASR in the end-to-end approach for Arabic. This system used the Kaldi toolkit to present an effective character-based ASR. This system uses words, characters, and statistical morphs to evaluate the models. In this paper, the word, morphs, and character n-gram models were trained using a projection layer, a highway layer, and an LSTM layer. In addition, Recurrent Neural Network Language Models (RNN-LMs) were used for enhancing the model. An acoustic model used the MGB-2 corpus (consists of 1200 h) for training.

Ahmed et al. [7] introduced the first recipe of the end-to-end approach for the Arabic ASR system using BDRNNs. This system includes a character-based decoder without the lexicon. In addition, a CTC was used to represent the objective function. The output of character sequences are used to represent the acoustic features. Those features will be maximised using the objective function. The system includes three parts: 1- a BDRNNs acoustic model, 2- a language model, and 3- a character-based decoder. In addition, the training and decoding processes were based on Arabic grapheme. The objective function was used to train BDRNNs to remove the repeated characters in pre-segmented acoustic observations. In experimental results, the test dataset was evaluated on both the character and word levels in order to validate the results with other word-based models. The recipe used the 1200 h corpus, which was collected from Aljazeera programs for evaluating the system. On the development set, the WER was obtained by 12.03% for non-overlapped speech. The morph-based models achieved the best recognition results for both lower-resourced and well-resourced tasks; however, the performance of the character-based ASR using the lower-resource is better than the word-based models.

Zhou et al. [34], have developed a framework for multilingual speech recognition based on low-resource languages. This work used sequence-to-sequence based and attention-based models and a single transformer for recognition. Subwords were utilised as the multilingual modelling unit without the pronunciation lexicon. The ASR transformer represents the main architecture of this model. They employed a language-specific softmax layer instead of a softmax layer to solve the problem of a few training data on low-resource languages. The CALLHOME datasets with six languages, Arabic, Mandarin, Japanese, English, German, and Spanish, were used in the experiments. The result of the Arabic dataset was 13.5% average WER.

Belinkov et al. [1] in this work, analysed an end-to-end ASR model in terms of Arabic graphemic and phonetic representations as well as different articulatory features. In this work, the traditional ASR systems and components of grapheme/phoneme modelling were employed to run forced alignment and obtain annotated data in order to use them in the classifier. In addition, the lexicon used the phonetic lexicon for the phoneme system. Convolutional layers and 5 LSTMs were used for encoding and training with CTC. The system consists of two steps: 1) training the end-to-end model in a normal fashion on pairs of transcriptions and utterances, 2) running the trained model on the dataset using frame-level annotations. The representation quality in this model was analysed with the help of the above steps. The MGB-2 corpus, which has 1200 h from the Al Jazeera Arabic TV channel, was used for training and testing. Experiments on the corpus yielded 12.94% and 10.60% WER for phonemes and graphemes, respectively.

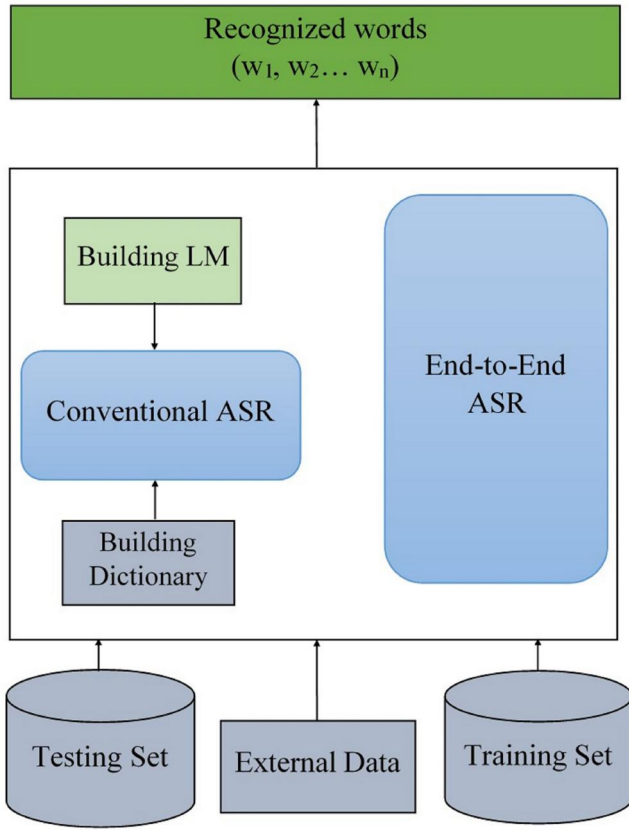
Zerari et al. [35] proposed a system to recognise the isolated Arabic utterances using two ASR applications: a) TV spoken command recognition, b) spoken digit recognition. This system consists of several steps: first, pertinent features were extracted from the natural speech (static and dynamic) MFCC features and the Filter Banks (FB) coefficients. Second, the extracted features were packed for working with the non-uniformity of the sequences length. Third, the RLSTM or GRU (Gated Recurrent Unit) represented the deep architecture in order to represent the sequences of MFCC/FB features as a fixed size vector. Finally, the Multi-Layer Perceptron network (MLP) used the vectors as input to perform the recognition (classification). The RNN was used to manipulate the lengths of features (MFCCs, FBs, and delta-delta) in the spoken digits/commands. Two datasets were used in this work: 1) the spoken Arabic digit, which contains 8.8 K tokens from 88 native Arabic speakers for spoken digit recognition, 2) speaker-independent mode, which includes 100 Arabic native speakers (males and females). The accuracy achieved was 98.77% and 96% for the first dataset and the second dataset, respectively.

### 3 | METHODOLOGY

#### 3.1 | The proposed systems

Two types of speech recognition approaches are used in this work: a) Conventional ASR approach and b) end-to-end ASR approach. The architecture of these systems is shown in Figure 1. In the first system, various acoustic models are trained on the speech corpus, while the second system trains two acoustic models on the same corpus. All the details of these systems will be described in the next sections. All the acoustic models are built using three different toolkits: KALDI, ESPnet, and Espresso. For building these models, we employed the hardware specifications depicted in Table 1.





**FIGURE 1** Overall architecture of the proposed conventional and end-to-end automatic speech recognition systems. LM, language model

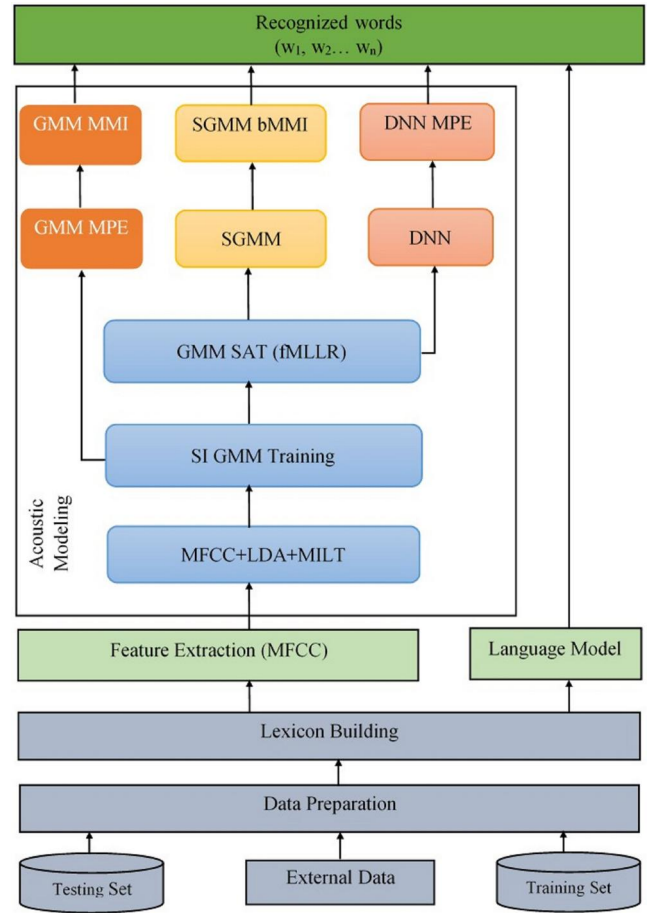
### 3.1.1 | Conventional ASR

The conventional ASR is built using the Kaldi toolkit. Figure 2 shows the steps of the conventional ASR which are described in the following:

1. Data Preparation: The Kaldi-format data directories and lexicons are employed in this step. The CMUSphinx (CMUCLMTK) toolkit is used for building the basic pronunciation dictionary. In addition, Kaldi is used to create the training, testing, wav.scp, and utt2spk files.
2. Lexicon Building: The lexicon collected from many websites that are mostly MSA without diacritisation. We use the Mishkal toolkit<sup>1</sup> to convert raw Arabic data into vocalised text. All words in the lexicon occur more than once in the sentence. The lexicon has 245K unique words.
3. Feature extraction: MFCC-based features are extracted and presented for data preparation and training steps. The presented models use the standard 13-dimensional cepstral mean-variance normalised (CMVN) features for building the acoustic models.
4. Language Model: The language model is built using the training and collected data. The LM is built using the CMUCLMTK toolkit based on the 3-g method.

**TABLE 1** Machine hardware specifications

CPU	Intel i7-8750H
RAM	16 GB
GPU	GeForce GTX 1060 6 GB
CUDA version	10.0
OS	Ubuntu 18.04



**FIGURE 2** Steps of conventional automatic speech recognition. DNN, deep neural network; MPE, minimum phone error; SGMM, Subspace Gaussian mixture models

5. Acoustic Modelling: The KALDI toolkit is used for building acoustic models using a recipe similar to [27]. The training data were first trained using CPU to create six different models that are as follows: GMM-SI (Speaker-Independent), GMM + feature-space Maximum Likelihood Linear Regression (fMLLR), GMM + Minimum Phone Error (MPE), GMM + Maximum Mutual Information (MMI), subspace GMM (SGMM), and SGMM + boosted Maximum Mutual Information (bMMI). The MFCC + Discriminative Analysis (LDA) + Maximum Likelihood Linear Transform (MLLT) are used to build the GMM-SI model, then MPE and MMI methods are used to build the GMM. Finally, fMLLR features are used to build the SGMM model. In addition,

<sup>1</sup><https://github.com/linuxscout/mishkal>

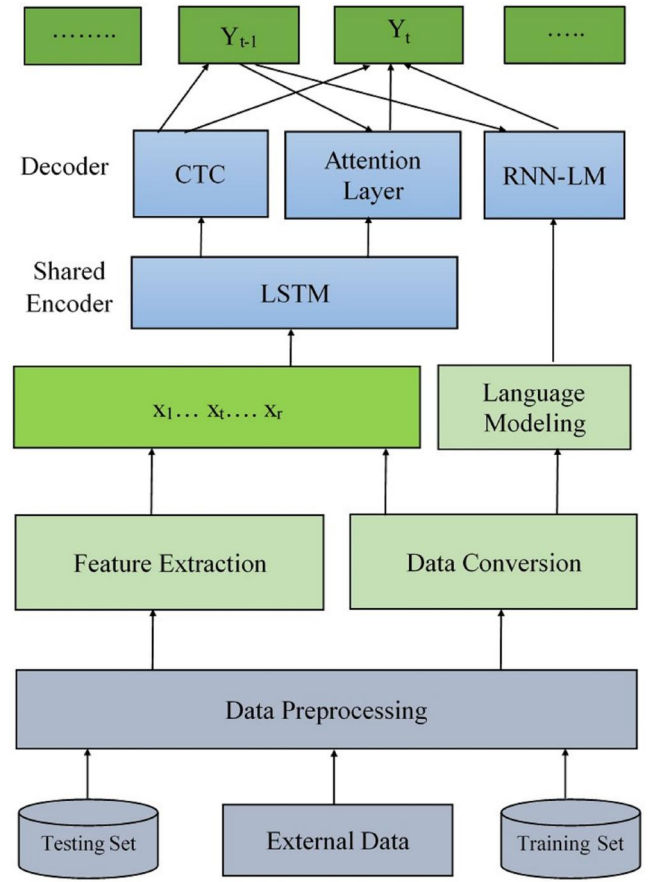
GPU is used to create two different models: DNN and DNN-MPE. The fMLLR features are used to build the SGMM model. The models in [27] are trained with the classic 13-dimensional MFCC features, whereas in each frame was included  $\pm 4$  neighbouring frames to splice the MFCC. Then the linear LDA is applied to reduce the dimension to 40. The dimensional reduction is followed by MLLT [8]. The GMM systems use 40 dimensions as the optimal input dimensions for DNN [36]. We selected the larger dimensions for DNN and used these dimensions for all our models. For DNN, we used five hidden layers as each layer has 2048 nodes, and the learning rate is 0.008 similar to [27]. fMLLR is also used for speaker adaptation as in [37]. The GMM models contain 512k Gaussians with 8k states and the SGMM has 5k states similar to [27].

### 3.1.2 | End-to-end ASR using joint CTC-attention

The first end-to-end model is built using Joint CTC attention based on the ESPnet toolkit which uses PyTorch as the underlying deep learning engine. Figure 3 shows the steps of the proposed model. It starts with preparing the corpus data, lexicon, and language model. The next step involves extracting the features and converting the information into files. Then, RNN-LM is used for building the language modelling. Finally, the joint CTC-attention is used for the encoder-decoder. The following subsections describe the details of each step:

1. Data pre-processing: ESPnet uses the Kaldi-format to prepare data for directories, lexicons, and language models. The external data are collected from Internet resources for building dictionaries, lexicons, and language model data [11]. The data of dictionaries, lexicons, and language models are built using the training and collected data mentioned above.
2. Feature extraction: In this step, the Kaldi recipe is used for feature extraction. This recipe is used to generate pitch features and the 80-dimensional Log Mel-filterbank (totally 83 dimensions). It is used to prepare the training data for 3.5k utterances.
3. Data conversion: The Kaldi data directory process was used to convert all information to one file (data.json). This Json file does not include the input features. The list of characters is created in this step.
4. Language model: The attention-based technique in end-to-end ASR needs a large amount of text data for training the language model (LM) [11]. So, the LM is trained using training and collected data-based RNN-LM. In addition, this LM is integrated with the log probability  $p_{lm}$  of RNN-LM as follows:

$$\log p(y_n | y_{1:n-1}, h_{1:T'}) = \log p^{yb}(y_n | y_{1:n-1}, h_{1:T'}) + \beta \log p^{lm}(y_n | y_{1:n-1}) \quad (1)$$



**FIGURE 3** Steps of joint connectionist temporal classification-attention automatic speech recognition system

where  $y_n$  denotes the hypothesis of the output label at position  $n$ ,  $y_{1:n-1}$  represents a history of labels,  $y_{1:T'}$  represents the output of the encoder,  $p^{att}$  denotes the probability of attention and  $p^{ctt}$  represents the probability of CTC.

This method is considered a shallow fusion method which is used in a decoder network and the RNN-LM. The shallow fusion is presented in neural machine translation [38]. This method is used for end-to-end ASR for improving the performance [39].

In this work, the character-based RNN-LM is trained using PyTorch and combined with the decoder network in the decoder (recognition) step.

5. End-to-end model training: The network training is trained using a hybrid CTC/attention-based encoder-decoder. A bidirectional LSTM (BLSTM) represents the encoder network with sub-sampling [40] given the  $T$ -length of the speech feature sequence  $o_{1:T}$ . This method is used to generate the high-level feature sequence  $h_{1:T_0}$  as follows:

$$h_{1:T_0} = BLSTM(o_{1:T}) \quad (2)$$

where  $T_0 < T$ , in general, due to the sub-sampling. A sequence of input features  $X$  is converted to represent the hidden states

$h_t$  using frame-wise operations. The input features can be modelled by a BLSTM as follows:

$$\text{Encoder}(X) = \text{BLSTM}(X) \quad (3)$$

The BLSTM produces the outputs for reducing the computational cost. So in ASR, the input length is not equal to the output length [40]. The attention-based method calculates the posterior  $p(Y|X)$  where  $X = x_1, x_2, \dots, x_T$  represents the input features' sequence,  $Y = y_1, y_2, \dots, y_L$  represents an output character's sequence [41]. The  $p(Y|X)$  uses the probabilistic chain rule as factors as follows:

$$p(Y|X) = \prod_{l=1}^L p(y_l | y_{1:l-1}, X) \quad (4)$$

where  $y_{1:l-1}$  represents a subsequence  $y_1, y_2, \dots, y_{l-1}$  and  $p(y_l | y_{1:l-1}, X)$  is estimated as follows:

$$a_{lt} = \begin{cases} \text{Dotproduct}_{\text{Attention}}(q_{l-1}, h_t) \\ \text{Location}_{\text{Attention}}(q_{l-1}, h_t, a_{l-1}) \end{cases} \quad (5)$$

$$r_l = \sum_{t=1}^T a_{lt} h_t \quad (6)$$

$$p(y_l | y_{1:l-1}, X) = \text{Decoder}(r_l, q_{l-1}, y_{l-1}) \quad (7)$$

Equations (2) and (3) represent the encoder, Equation (7) represents the decoder networks where the attention weight is represented by  $a_{lt}$ , the attention weight vectors  $a_{l0}, a_{l1}, \dots, a_{lT}$  is represented by  $a_l$ , the subsequence of the attention vectors  $a_1, a_2, \dots, a_{l-1}$  is represented by  $a_{l:l-1}$ , the hidden states of the encoder is represented by  $h_t$ , the decoder networks are represented by  $q_l$ , and a letter-wise hidden vector is represented by  $r_l$ . The letter-wise hidden vector represents the summarisation of the attention weight vector with the weight of the hidden vectors. In Equation (5)  $a_{lt}$  represents the attention weight, which represents an alignment between the hidden states  $h_t$  and each atom of the output sequence  $c_l$  in the encoder. The simplest attention represents the dot-product attention such that in Equation (5) [42], the dot product is calculated as follows:

$$e_{lt} = q_{l-1}^T W_a h_t \quad (8)$$

$$a_l = \text{Softmax}(e_l) \quad (9)$$

where  $W_a$  denotes the trained matrix parameters and  $e_l$  represents the energies  $e_{l1}, e_{l2}, \dots, e_{lT}$ . The default attention is

location-aware attention as in Equation (5) [43] and the energy in Equation (8) is calculated by the following:

$$F_l = K * a_{l-1} \quad (10)$$

$$e_{lt} = g^T \tanh(W_q q_{l-1} + W_h h_t + W_f f_{lt} + b) \quad (11)$$

where  $F_l$  includes the  $f_{l1}, f_{l2}, \dots, f_{lT}$  vectors and  $K$  represents the filters of trained one-dimensional convolution.

The presented model used the attention-based and the location-aware attention mechanism as in [43]. In addition, this model supports dot-product attention [42]. The default attention (location-aware) achieves better performance, while the computational cost term runs faster in the dot-product attention. However, PyTorch back-end is used in the presented model to support the multi-head attention [44], coverage mechanism [45], and additive attention [46] functions.

The hybrid CTC/attention end-to-end ASR [47] uses training and decoding advantages for enhancing the performance. The multi-objective learning is utilised during the training step. It merges the CTC ( $L_{\text{ctc}}$ ) and attention-based cross entropy ( $L_{\text{att}}$ ) to achieve fast convergence and enhance the robustness as follows:

$$\ell = \alpha \ell^{\text{ctc}} + (1 - \alpha) \ell^{\text{att}} \quad (12)$$

The output model is integrated with attention decoder networks and CTC for enhancing the recognition time.

During training, label-smoothing techniques are used to handle the problems of overfitting. In those techniques, the target distribution should be smoothed; therefore, the probability mass of the correct label and the other labels will be divided in a specific ratio.

CTC is considered one of the most affected parts in the computation training time. The warp CTC library [48] is used for the PyTorch backend to enhance the time speed in the training by 5%–10%. In this work, we train the model using a hybrid CTC/attention-based encoder-decoder network that is developed in [47].

6. Recognition: In the decoding step, a one-pass beam search algorithm is utilised to merge CTC and attention-based scores for removing irregular alignments. Let  $y_n$  represent a hypothesis of output label  $a_t$  in each position, given the encoder output  $h_{1:T_0}$  and a history  $y_{1:n-1}$ . In the beam search, the score log probabilities of the hybrid CTC  $p_{\text{ctc}}$  and attention  $p_{\text{att}}$  are calculated as follows:

$$\begin{aligned} \log p^{\text{hyb}}(y_n | y_{1:n-1}, h_{1:T'}) &= \alpha \log p^{\text{ctc}}(y_n | y_{1:n-1}, h_{1:T'}) \\ &\quad + (1 - \alpha) \log p^{\text{att}}(y_n | y_{1:n-1}, h_{1:T'}) \end{aligned} \quad (13)$$

The parameter  $\alpha$  is assigned by  $\alpha = 0.5$  for tuning to inset the objective functions.

The speech recognition in this work is evaluated based on the RNN-LM and end-to-end ASR model, respectively, that were introduced earlier. The recipe in the ESPnet toolkit has been modified in order to accommodate our methods, data and language model.

### 3.1.3 | End-to-end ASR using CNN-LSTM with attention

The end-to-end model is built based on CNN-LSTM with the attention method using the Espresso toolkit. It uses PyTorch as its underlying deep-learning engine. Figure 4 shows the model steps, which starts with pre-processing the corpus data, lexicon, and language modelling. The next step involves feature extraction, followed by training and building the language modelling. After that, CNN and LSTM are used for the encoder. Finally, the LSTM with attention is used for the decoder.

For pre-processing and feature extraction, we used the same data pre-processing and feature extraction steps as in the joint-CTC ASR.

- End-to-end model training

The training and collected data are used to train and build the external language model (LM) in this model. The external LM contains about 1.8 m words and 245k unique words. The LM model has 3 LSTM layers with 12k hidden size and around 2.7 million parameters in total. The OOV of the LM is 6.25%. Shallow fusion [38] is used as an LM integration technique for enhancing speech recognition [49, 50]. The LSTM decoder is integrated with shallow fusion to calculate the sum of weights for two posterior distributions using sub-word units. As in [4], a look-ahead word-based LM is used to produce the trained character-based and word-based LMs. In each decoding step, the probabilities of a look-ahead word-based LM are calculated based on the word prefix decoded. The prefix trees are used to convert the word-based LM into a character-based LM [4].

The encoder-decoder model which represents network training and recognition has been built. In the encoder side, the hybrid CNN-LSTM method is used for building the acoustic model. The CNN layers are used to prepare the input sequences as vectors  $v_1, v_2, \dots, v_n$  and send them into the LSTM [51]. The 4-layer uses 2-dimensional convolution and kernel size (3; 3) for the feature and time frame axis [11, 52]. In order to obtain the results in one-fourth time frames after convolution, we used 2 $\times$ -downsampling at layers 1 and 3. Then 3 layers of bidirectional LSTMs [53] are put upon the output of the stacked convolution layers. Using one word at one time, the word is encoded based on the stacked CNN and current status. After that, the status is updated and presented for the next steps. On the other hand, the encoder yields a set of encoded vectors  $h_{1:n}$  with the number of words  $l$  in the input sentences.  $N$  epochs are used in training and we trained the first epochs with gold labels,

while the remaining epochs support scheduled sampling [54]. The training process scheduled the probability  $p$  and used the smaller probability to motivate the model. In order to enhance accuracy, we utilise label smoothing [55] during the loss calculation process (cross-entropy). For the label smoothing process, temporal smoothing [56] is combined with a distribution probability mass for producing the neighbouring token unit in the transcript.

In the decoder, the LSTM with attention [42, 46] receives the encoder output and its hidden state to compute the current hidden states  $v_1, v_2, \dots, v_n$ . The attention mechanism uses the states  $s_1, s_2, \dots, s_n$  as the query, the hidden states  $h_{1:n}$  of encoder sides as the keys, values, and outputs the final hidden representation for representing the sum of values weighted. The architecture of the Google Neural Machine Translation (GNMT) system [57] is used in this work, where the 3 layers in the decoding LSTM receive the generated vectors using the attention mechanism. Residual connections [58] are used to link the decoder layers together. The decoder depends on the trained external LM to improve and enhance the end-to-end ASR performance. The external LM is utilised to enhance the performance of the end-to-end ASR [4, 49]. We used shallow fusion to integrate the external LM and the decoder as follows:

$$y_t^* = \arg \max_{y_t \in V} \log p_{ASR}(y_t | X, y_{1:t-1}) + \lambda \log p_{LM}(y_t | y_{1:t-1}) \quad (14)$$

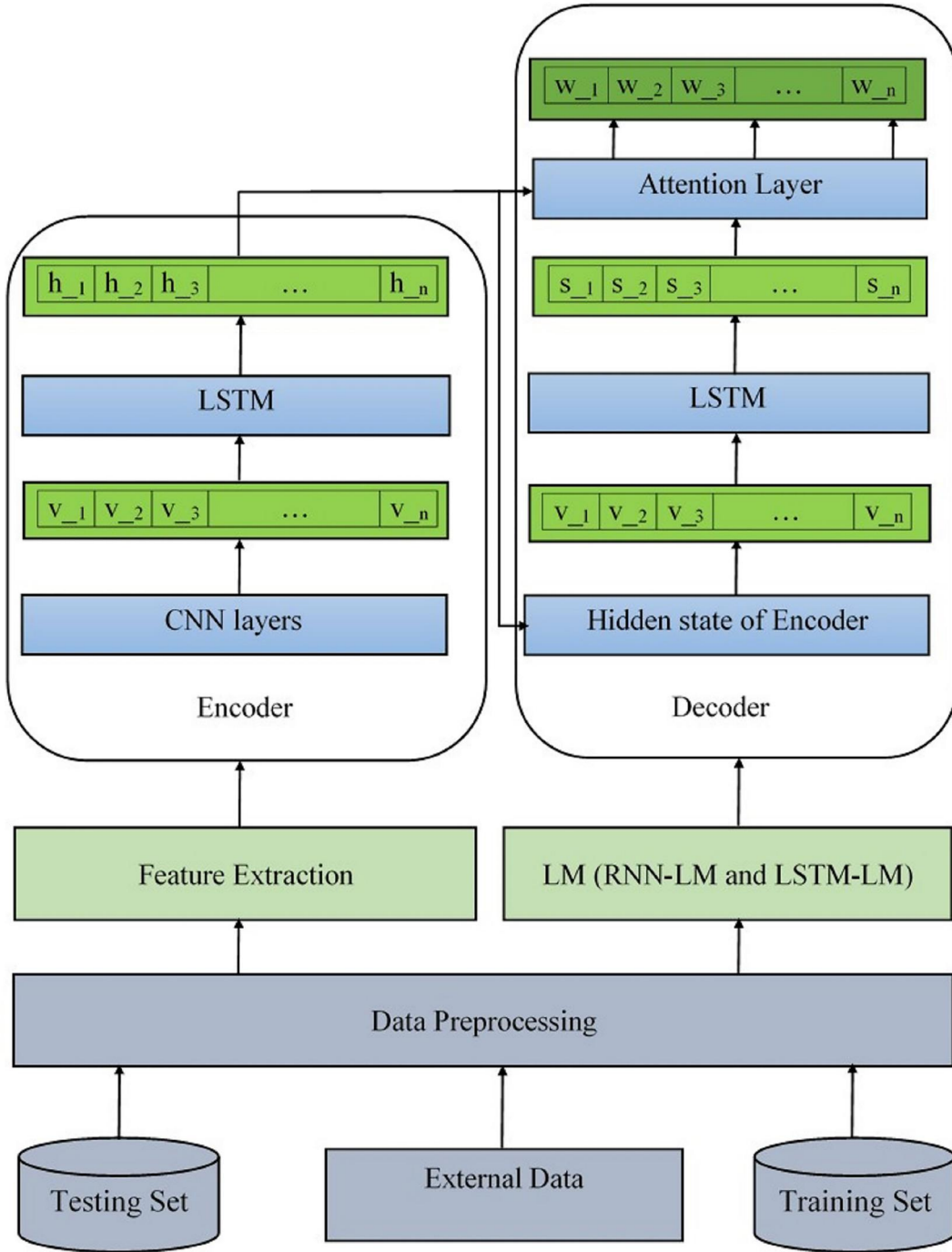
At each time stamp  $t$ , the prediction,  $y_t^*$ , is generated by interpolating the distribution over the vocabulary ( $V$ ) given by the seq2seq ASR model ( $\log p_{ASR}(y_t | x, y_{1:t-1})$ ) and the same distribution given by the external LM ( $\log p_{LM}(y_t | y_{1:t-1})$ ). The external LM weight,  $\lambda$ , is tuned as a hyper parameter. The LM with the fusion is used to enable the decoder to delete the errors when the weight of the LM is larger [59]. The scalar vales, namely coverage term, are used for each hypothesis in the beam to encourage the long transcripts [56]. This coverage is also utilised to remove the attention from  $n$ -grams decoding over the utterance by shallow fusion:

$$coverage = \sum_j 1 \left[ \sum_i a_{ij} > \tau \right] \quad (15)$$

where  $a_{ij}$  is the weight of attention for the encoder frame  $j$  and decoder step  $i$ , a tunable hyper-parameter is represented by  $\tau > 0$ , and the indicator function is represented by  $1(\cdot)$ . The authors [60] modified the coverage term in order to punish the repeating attentions more perfectly:

$$coverage = \sum_j 1 \left[ \sum_i a_{ij} > \tau_1 \right] - 1 \left[ \sum_i a_{ij} > \tau_2 \right] \cdot \left( c + \left[ \sum_i a_{ij} - \tau_2 \right] \right) \quad (16)$$





**FIGURE 4** Steps of the convolutional neural networks-long short-term memory networks with attention system

where the tunable hyper-parameters are represented by  $\tau_2 > \tau_1 > 0$  and  $c > 0$ . The first term in Equation (16) is similar to that in Equation (15), while the second term is the hypothesis score where the final sum of attention on the encoder frame  $j$  is denoted by  $\tau_2$ . The modified coverage term supports the employed model for reducing the repeating n-grams and deleting errors. The end-of-sentence threshold (EOS) technique [61] is taken into consideration in this work. This technique is utilised to bias the decoder from transcriptions when the decoder used a fused LM.

EOS tokens will be produced if the output token candidate during beam search is less than the probability of EOS as follows:

$$\log P(< eos > |h) > \gamma \cdot \max_{t \in V} \log P(t|h) \quad (17)$$

where  $V$  represents the set of vocabulary and  $h$  denotes the word history.

## 4 | EXPERIMENTAL RESULTS

The Standard Arabic Single Speaker Corpus (SASSC) [62] used in our work includes 51 thousand words that required 7 h of recording via a single young male speaker. The 4372 utterances (audio + diacritised text) represent this corpus recorded at a 96 kHz sampling rate and across nine different categories. In the employed models, we used data from 3500 utterances (80%) for training and 872 utterances (20%) for the purpose of testing purpose in the conventional ASR. In the end-to-end ASR, we used 80% data for both the training and development set (3400 utterances for training and 100 utterances for the development set) and 20% data (872 utterances) for the purpose of testing.

Several ASR models are trained on the pre-defined training set and evaluated on the specified testing set. As we mentioned above, the ASR systems are trained in terms of two ASR approaches: (i) Conventional ASR and (ii) end-to-end ASR. So, we will show the results of these systems in the following:

### 4.1 | Results of conventional ASR

Word error rate (WER) is reported for 8 models, all models are trained on the same training data. After training all 8 models, each model was evaluated separately. The results in Table 2 show the WER for all the models of the conventional ASR. From Table 2, we conclude that the GMM-SI model achieved 39.75% WER. After applying MPE and bMMI on GMM-SI, GMM + MPE, and GMM + bMMI (using 0.5 boosting factor) WER is decreased further by 1.34% and 1.23%, respectively, while the GMM + fMLLR model achieved 37.02% WER. In addition, SGMM + fMLLR and SGMM + bMMI models (with 0.1 boosting factor) achieved 1.2% and 0.73% gain on top of the GMM + fMLLR model. The result of the deep neural network model is 34.43% WER. Thus, DNN achieved an excellent gain of WER, by 1.39%, compared with the best model in the SGMM. The sequential training of the DNN (DNN + MPE) model attained the best results with an overall WER of 33.72%. In general, the DNN method improved the performance better than the GMM and SGMM methods. After applying the adaptation techniques (fMLLR, MPE, and bMMI) on the GMM and SGMM methods, we noted that fMLLR decreased the WER better than MPE and bMMI techniques.

The results are also visually presented in Figure 5. What stands out in Table 2 is the significant enhancement of the WER after applying the recent methods in the conventional ASR. It also shows the effect of the adaptation techniques on WER.

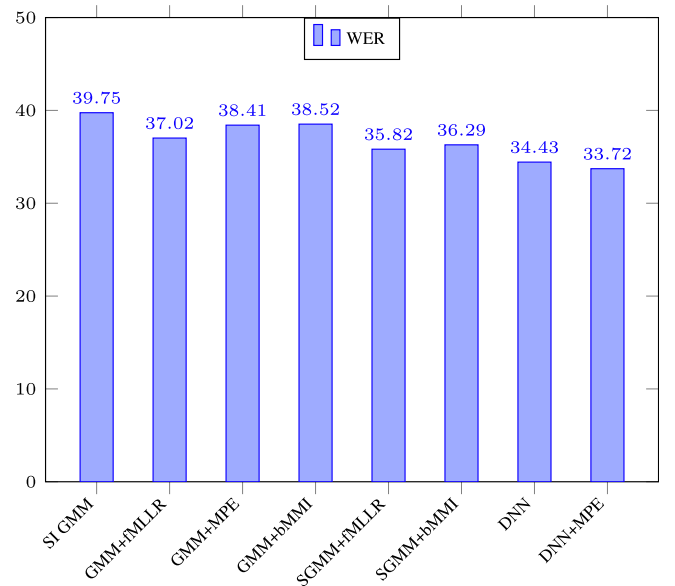
### 4.2 | Results for end-to-end ASR

We report the character error rate (CER) and the word error rate (WER) for the systems based on joint CTC-attention ASR in Table 3 and CNN-LSTM with attention technique in

**TABLE 2** Word error rate (WER) for conventional ASR

Model	WER(%)
GMM-SI	39.75
GMM + fMLLR	37.02
GMM + MPE	38.41
GMM + bMMI(0.5)	38.52
SGMM + fMLLR	35.82
SGMM + bMMI (0.1)	36.29
DNN	34.43
DNN + MPE	33.72

Abbreviations: ASR, automatic speech recognition; DNN, deep neural network; GMM, Gaussian mixture models; MPE, minimum phone error; SGMM, Subspace Gaussian mixture models.



**FIGURE 5** Word error rate (WER) for conventional automatic speech recognition methods

**TABLE 3** Word error rate (WER) for end-to-end ASR using joint CTC-attention

	CER(%)	WER(%)
Joint CTC-attention ASR	7.76	31.1

Abbreviation: ASR, automatic speech recognition; CER, character error rate; CTC, connectionist temporal classification.

Table 4. The experimental results of joint CTC-attention ASR use a deeper encoder network. This network is an integration of character-based LSTM-LM and joint CTC/attention decoding techniques. We used LSTM-weight = 0.1 as label smoothing in training and decoding, the size of the beam is 10, and the LM weight is 1.0. Table 3 shows that using the joint CTC-attention ASR method resulted in decreasing WER, by almost 2.62% absolute, compared to the best WER of the

**TABLE 4** Word error rate (WER) for end-to-end ASR using CNN-LSTM with attention

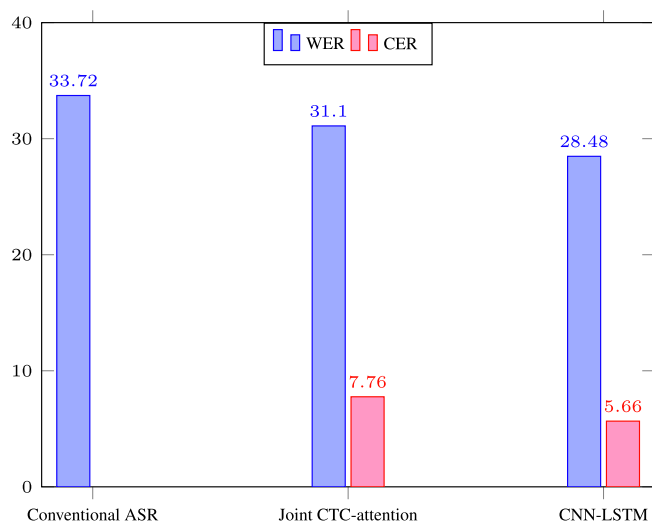
LSTM + Look-ahead Word LM + Improved Coverage + EOS Threshold		
#Epochs	CER(%)	WER(%)
200	6.04	30.80
300	6.06	29.18
400	5.84	28.60
600	5.66	28.48

Abbreviation: ASR, automatic speech recognition; CER, character error rate; CNN-LSTM, convolutional neural networks-long short-term memory networks; EOS, end-of-sentence threshold.

**TABLE 5** The best results achieved by the proposed models

System	WER(%)	CER(%)
Conventional ASR	33.72	N/A
End-to-end ASR using joint CTC-attention	31.10	7.76
End-to-end ASR using CNN-LSTM with attention	28.48	5.66

Abbreviation: ASR, automatic speech recognition; CER, character error rate; CNN-LSTM, convolutional neural networks-long short-term memory networks; WER, word error rate.

**FIGURE 6** The best results achieved by the proposed models

conventional ASR. The LM in joint CTC-attention ASR resulted 6.25% as out-of-vocabulary (OOV).

The results of the model based on CNN-LSTM with attention technique is shown in Table 4. To achieve these experimental results, we use  $p = 0.05$  as temporal label smoothing and  $p = 0.5$  as scheduled sampling and use the epoch #6 for the starting adaptation. The LM fusion weight is assigned by 0.25 as an optimal parameter and integrated with the look-ahead word-based LM. In addition, we use a beam width of 50. In addition, we also trained the suggested model on a variable number of epochs (200, 300, 400, and 600). For optimising the model, we increased the epochs to 600. Table 4 shows the results for all the epochs and the best WER and CER are (28.48 as WER and 5.66

**TABLE 6** Comparison with other Arabic ASR systems for the SASSC corpus

System	Model	WER (%)	Wacc (%)
Azim et al. [63]	SMO, Naïve Bayes and J48	36.00	64.00
Ali et al. [27]	GMM	39.75	60.25
Ali et al. [27]	SGMM	36.29	63.71
Ali et al. [27]	DNN	34.43	65.57
Azim et al. [64]	HMM	31.43	68.57
<b>The proposed model</b>	<b>CNN-LSTM</b>	<b>28.48</b>	<b>71.52</b>

Note: We use bold font for distinguishing and showing our results.

Abbreviation: ASR, automatic speech recognition; DNN, deep neural network; GMM, Gaussian mixture models; HMM, hidden Markov models; SGMM, Subspace Gaussian mixture models; WER, word error rate.

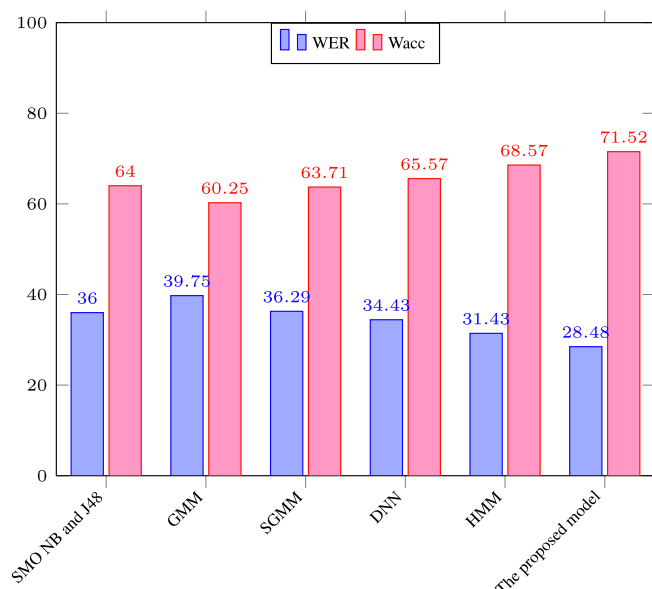
as CER) for epoch #600. In addition, Table 4 shows that using CNN-LSTM with attention technique resulted in decrease better than other models and achieved better WER and CER than the end-to-end model based on joint CTC-attention ASR.

Table 5 shows the best results achieved by conventional ASR, joint CTC-attention, and CNN-LSTM with attention. From Table 5, we conclude that the end-to-end ASR performed better than conventional ASR. In end-to-end ASR, the model using an CNN-LSTM encoder and LSTM decoder with attention achieved better performance and accuracy than the model which is built using hybrid CTC/attention-based encoder-decoder. Table 5 shows that using the joint CTC-attention ASR method resulted in decreasing WER, by 2.62%, and CER, by 2.10% absolute, compared to the best WER of joint CTC-attention ASR.

These results are also visually presented in Figure 6. This figure shows the significant reduction of the WER after applying end-to-end methods. It also shows that the state-of-the-art CNN-LSTM with attention-based method, achieved better accuracy than other end-to-end methods.

We used the diacritised data version of the Arabic language in our work. There are few works in the literature related to diacritised data version. So, we targeted making a comparison with the best methods for the diacritised Arabic ASR for the SASSC corpus.

We compared our best results with previous works by Azim et al. [63, 64] and Ali et al. [27] using WER and word



**FIGURE 7** Comparison with other Arabic automatic speech recognition systems for the SASSC corpus

accuracy (Wacc). Our results achieved WER and Wacc better than their system as in Table 6.

The chart in Figure 7 is used to visually represent the results in Table 6. This figure shows a comparison of the different methods for the diacritised Arabic ASR. It also shows the significant enhancement of WER and Wacc after applying the CNN-LSTM with attention-based encoder-decoder method.

## 5 | CONCLUSION AND FUTURE WORK

In this work, we proposed two new end-to-end systems for the diacritised Arabic ASR: i) conventional ASR and (ii) end-to-end ASR. In addition, we implemented a conventional ASR system as a benchmark approach. The LM is built using the collected data from many Internet resources in this work. All models are trained and tested on the SASSC corpus with diacritised text data, where MFCCs and FBANK are used for feature extraction. In the conventional ASR system, eight models are considered: 4 models use GMM, 2 models use SGMM, while the last 2 models use DNN. These models were built using the KALDI toolkit. CMUCLMTK was used for language modelling. Across these eight models, the best achieved WER was 33.72% using DNN-MPE. On the other hand, an end-to-end approach is presented using new methods for the diacritised Arabic ASR. These methods are built based on joint CTC-attention and CNN-LSTM with attention as the state-of-the-art methodology using ESPnet and Espresso, respectively. In our experiments, the achieved results are 7.76% CER and 31.10% WER using the joint CTC-attention method. For CNN-LSTM with attention method, the best results are obtained from this model: 5.66% and 28.48% as CER and WER, respectively. This method reduced WER by 5.24% and 2.62% when compared with the conventional ASR and joint

CTC-attention method, respectively. In future work, we plan to examine the applicability of the presented models on other large datasets. We also recommend updating the external LM in order to enhance the models' accuracy and performance.

## ACKNOWLEDGEMENTS

There is no fund supporting the work of this manuscript.

## ORCID

Hamzah A. Alsayadi <https://orcid.org/0000-0002-6062-0899>

Abdelaziz A. Abdelhamid <https://orcid.org/0000-0001-7080-1979>

## REFERENCES

1. Belinkov, Y., Ali, A., Glass, J.: Analysing Phonetic and Graphemic Representations in End-to-End Automatic Speech Recognition. arXiv preprint arXiv:190704224 (2019)
2. Nagamine, T., Seltzer, M.L., Mesgarani, N.: Exploring how deep neural networks form phonemic categories. In: Sixteenth Annual Conference of the International Speech Communication Association (2015)
3. Nagamine, T., Seltzer, M.L., Mesgarani, N.: On the role of nonlinear transformations in deep neural network acoustic models. In: Interspeech, pp. 803–807 (2016)
4. Hori, T., Cho, J., Watanabe, S.: End-to-end speech recognition with word-based RNN language models. In: 2018 IEEE Spoken Language Technology Workshop (SLT), pp. 389–396. IEEE (2018)
5. Kim, S., Hori, T., Watanabe, S.: Joint ctc-attention based end-to-end speech recognition using multi-task learning. In: 2017 IEEE International Conference on Acoustics, Speech And Signal Processing (ICASSP), pp. 4835–4839. IEEE (2017)
6. Xiao, Z., et al.: Hybrid ctc-attention based end-to-end speech recognition using subword units. In: 2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP), pp. 146–150. IEEE (2018)
7. Ahmed, A., et al.: End-to-end lexicon free Arabic speech recognition using recurrent neural networks. In: Computational Linguistics, Speech and Image Processing for Arabic Language. 4, 231 (2018)
8. Povey, D., et al.: The Kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. CONF. IEEE Signal Processing Society (2011)
9. Miao, Y., Gowayyed, M., Metze, F.: EESSEN: end-to-end speech recognition using deep RNN models and WFST-based decoding. In: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 167–174. IEEE (2015)
10. Kuchaiev, O., et al.: Openseq2seq: extensible toolkit for distributed and mixed precision training of sequence-to-sequence models. In: Proceedings of Workshop for NLP Open Source Software (NLP-OSS), pp. 41–46 (2018)
11. Watanabe, S., et al.: Espnet: end-to-end speech processing toolkit. arXiv preprint arXiv:180400015 (2018)
12. Pratap, V., et al.: wav2letter++: the fastest open-source speech recognition system. arXiv preprint arXiv:181207625 (2018)
13. Satori, H., et al.: Investigation Arabic speech recognition using CMU sphinx system. Int Arab J Inf Technol. 6(2) (2009)
14. Vergyri, D., et al.: Morphology-based language modelling for Arabic speech recognition. SRI International Menlo Park United States (2004)
15. Ali, A., Mubarak, H., Vogel, S.: Advances in dialectal Arabic speech recognition: a study using twitter to improve egyptian asr. In: International Workshop on Spoken Language Translation. IWSLT (2014)
16. Cardinal, P., et al.: Recent advances in ASR applied to an Arabic transcription system for Al-Jazeera. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
17. Diehl, F., et al.: Morphological decomposition in Arabic ASR systems. In: Comput Speech Lang. 26(4), 229–243 (2012)



18. Alghibab, W., et al.: Arabic speech recognition with deep learning: a review. In: International Conference on Human-Computer Interaction, pp. 15–31. Springer (2019)
19. Hamed, O., Zesch, T.: A survey and comparative study of Arabic diacritization tools. *J Lang Technol Comput Linguistics*. 32(1), 27–47 (2017)
20. Hadjir, I., Abbache, M., Belkredim, F.Z.: An approach for Arabic diacritization. In: International Conference on Applications of Natural Language to Information Systems, pp. 337–344. Springer (2019)
21. Khatatneh, K., et al.: A novel Arabic speech recognition method using neural networks and Gaussian filtering. *Int J Electr Electron Comput Syst*. 19(1) (2014)
22. Ahmed, B.H., Ghabayen, A.S.: Arabic automatic speech recognition enhancement. In: 2017 Palestinian International Conference on Information and Communication Technology (PICICT), pp. 98–102. IEEE (2017)
23. Hmad, N., Allen, T.: Biologically inspired continuous Arabic speech recognition. In: International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 245–258. Springer (2012)
24. Bouchakour, L., Debyeche, M.: Improving continuous Arabic speech recognition over mobile networks DSR and NSR using MFCCS features transformed (2018)
25. Mousa, A.E.D., et al.: Morpheme-based feature-rich language models using deep neural networks for LVCSR of Egyptian Arabic. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8435–8439. IEEE (2013)
26. AlHanai, T., Hsu, W.N., Glass, J.: Development of the MIT ASR system for the 2016 Arabic multi-genre broadcast challenge. In: 2016 IEEE Spoken Language Technology Workshop (SLT), pp. 299–304. IEEE (2016)
27. Ali, A., et al.: A complete Kaldi recipe for building Arabic speech recognition systems. In: 2014 IEEE Spoken Language Technology Workshop (SLT), pp. 525–529. IEEE (2014)
28. Tomashenko, N., et al.: Lium ASR systems for the 2016 multi-genre broadcast Arabic challenge. In: 2016 IEEE Spoken Language Technology Workshop (SLT), pp. 285–291. IEEE (2016)
29. Ettaouil, M., Lazaar, M., En.Naimani, Z.: A hybrid ANN/HMM models for Arabic speech recognition using optimal codebook. In: 2013 8th International Conference on Intelligent Systems: theories and Applications (SITA), pp. 1–5. IEEE (2013)
30. Wahyuni, E.S.: Arabic speech recognition using MFCC feature extraction and ANN classification. In: 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), pp. 22–25. IEEE (2017)
31. AbdAlmisreb, A., Abidin, A.F., Tahir, N.M.: Maxout based deep neural networks for Arabic phonemes recognition. In: 2015 IEEE 11th International Colloquium on Signal Processing & its Applications (CSPA), pp. 192–197. IEEE (2015)
32. Zerari, N., et al.: Bi-directional recurrent end-to-end neural network classifier for spoken Arab digit recognition. In: 2nd International Conference on Natural Language and Speech processing (ICNLSP), pp. 1–6. IEEE (2018)
33. Smit, P., et al.: Character-based units for unlimited vocabulary continuous speech recognition. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 149–156. IEEE (2017)
34. Zhou, S., Xu, S., Xu, B.: Multilingual end-to-end speech recognition with a single transformer on low-resource languages. *arXiv preprint arXiv:180605059* (2018)
35. Zerari, N., et al.: Bidirectional deep architecture for Arabic speech recognition. *Open Computer Science*. 9(1), 92–102 (2019)
36. Rath, S.P., et al.: Improved feature processing for deep neural networks. In: *Interspeech*, pp. 109–113 (2013)
37. Povey, D., Saon, G.: Feature and model space speaker adaptation with full covariance Gaussians. In: Ninth International Conference on Spoken Language Processing (2006)
38. Gulcehre, C., et al.: On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:150303535* (2015)
39. Hori, T., et al.: Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. *arXiv preprint arXiv:170602737* (2017)
40. Chan, W., et al.: Listen, attend and spell: a neural network for large vocabulary conversational speech recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4960–4964. IEEE (2016)
41. Hayashi, T., et al.: Multi-head decoder for end-to-end speech recognition. *arXiv preprint arXiv:180408050* (2018)
42. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:150804025* (2015)
43. Chorowski, J.K., et al.: Attention-based models for speech recognition. In: *Advances in Neural Information Processing Systems*, pp. 577–585 (2015)
44. Vaswani, A., et al.: Attention is all you need. In: *Advances in Neural Information Processing Systems*, pp. 5998–6008 (2017)
45. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarisation with pointer-generator networks. *arXiv preprint arXiv:170404368* (2017)
46. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:14090473* (2014)
47. Watanabe, S., et al.: Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*. 11(8), 1240–1253 (2017)
48. Amodei, D., et al.: Deep speech 2: end-to-end speech recognition in English and Mandarin. In: International Conference on Machine Learning, pp. 173–182 (2016)
49. Kannan, A., et al.: An analysis of incorporating an external language model into a sequence-to-sequence model. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5828. IEEE (2018)
50. Toshniwal, S., et al.: A comparison of techniques for language model integration in encoder-decoder speech recognition. In: 2018 IEEE Spoken Language Technology Workshop (SLT), pp. 369–375. IEEE (2018)
51. Chorowski, J., et al.: End-to-end continuous speech recognition using attention-based recurrent nn: first results. *arXiv preprint arXiv:14121602* (2014)
52. Zhang, Y., Chan, W., Jaitly, N.: Very deep convolutional networks for end-to-end speech recognition. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4845–4849. IEEE (2017)
53. Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional LSTM networks for improved phoneme classification and recognition. In: International Conference on Artificial Neural Networks, pp. 799–804. Springer (2005)
54. Bengio, S., et al.: Scheduled sampling for sequence prediction with recurrent neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1171–1179 (2015)
55. Szegedy, C., et al.: Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826 (2016)
56. Chorowski, J., Jaitly, N.: Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:161202695* (2016)
57. Wu, Y., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:160908144* (2016)
58. He, K., et al.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
59. Bahdanau, D., et al.: End-to-end attention-based large vocabulary speech recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4945–4949. IEEE (2016)

60. Wang, Y., et al.: Espresso: a fast end-to-end neural speech recognition toolkit. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 136–143. IEEE (2019)
61. Hannun, A., et al.: Sequence-to-sequence speech recognition with time-depth separable convolutions. arXiv preprint arXiv:1904.02619 (2019)
62. Almosallam, I., et al.: SASSC: a standard Arabic single speaker corpus. In: Eighth ISCA Workshop on Speech Synthesis (2013)
63. Azim, M.A., Badr, N.L., Tolba, M.F.: An enhanced Arabic phonemes classification approach. In: Proceedings of the 10th International Conference on Informatics and Systems, pp. 210–214 (2016)
64. Azim, M., et al.: Large vocabulary Arabic continuous speech recognition using tied states acoustic models. Asian J Inf Technol. 18(2), 49–56 (2019)

**How to cite this article:** Alsayadi, H.A., et al.: Arabic speech recognition using end-to-end deep learning. IET Signal Process. 15(8), 521–534 (2021). <https://doi.org/10.1049/sil2.12057>