[**Abstract**](https://link.springer.com/article/10.1007/s10278-022-00666-z#Abs1)

Today, the Transformer model, which allows parallelization and also has its own internal attention, has been widely used in the field of speech command recognition (SCR). The great advantage of this architecture is the fast-learning speed, and the lack of sequential operation, as with recurrent neural networks. In this work, Transformer models and an end-to-end model based on connectionist temporal classification were considered to build a system for the automatic commands recognition of Arabic speech. It is known that Arabic is part of a number of important languages and has limited data for implementing speech recognition systems. Some studies have shown that the Transformer model improves system performance for low-resource languages. Based on our experiments, it was revealed that the joint use of Transformer and connectionist temporal classification models contributed to improving the performance of the Arabic SCR system

* 1. [**Introduction**](https://link.springer.com/article/10.1007/s10278-022-00666-z#Sec1)

Innovative information and digital technologies are increasingly making their way into the life of a modern person: this applies to deep learning systems like voice recognition, images, speech recognition, and synthesis. Namely, speech technologies are widely used in communications, robotics, and other areas of professional activity. Speech recognition is a way to interact with technology.

from audio. SCR enables users to activate voice assistant systems on devices like smartphones and smart speakers by simply speaking the keyword phrase. For the usability and privacy of users, it is important to deploy an accurate KWS system on the device. A typical approach for SCR especially in the Arabic language is to train a keyword-specific acoustic model to predict a confidence score for each keyword phrase. Earlier works use deep neural networks with a hidden Markov model (HMM), and more recent works use convolutional neural networks (CNNs) and recurrent neural networks.

Currently, the end-to-end (E2E) model has become widespread. The E2E structure presents the system as a single neural network, unlike the traditional one, which has several independent elements4,5. The E2E system provides a direct reflection of acoustic signals in the sequence of labels without intermediate states, without the need to perform subsequent processing at the output, which makes it easy to implement. To increase the performance of E2E systems, it is necessary to solve the main tasks related to the definition of the model architecture, the collection of sufficient speech commands with the appropriate transcription, and the availability of high-performance equipment. Solving these issues ensures the successful implementation of speech recognition systems and other deep learning systems. In addition, E2E systems can significantly improve the quality of recognition from learning large amounts of training data.

It should be noted that the attention mechanism is a common method that greatly improves the quality of the system in machine translation and speech recognition. And the Transformer model uses this attention mechanism to increase the learning rate. This model has its own internal attention, which aligns all positions of the input sequence to find a representation of the set, which does not require alignments. In addition, Transformer does not need to process the end of the text after processing its start.

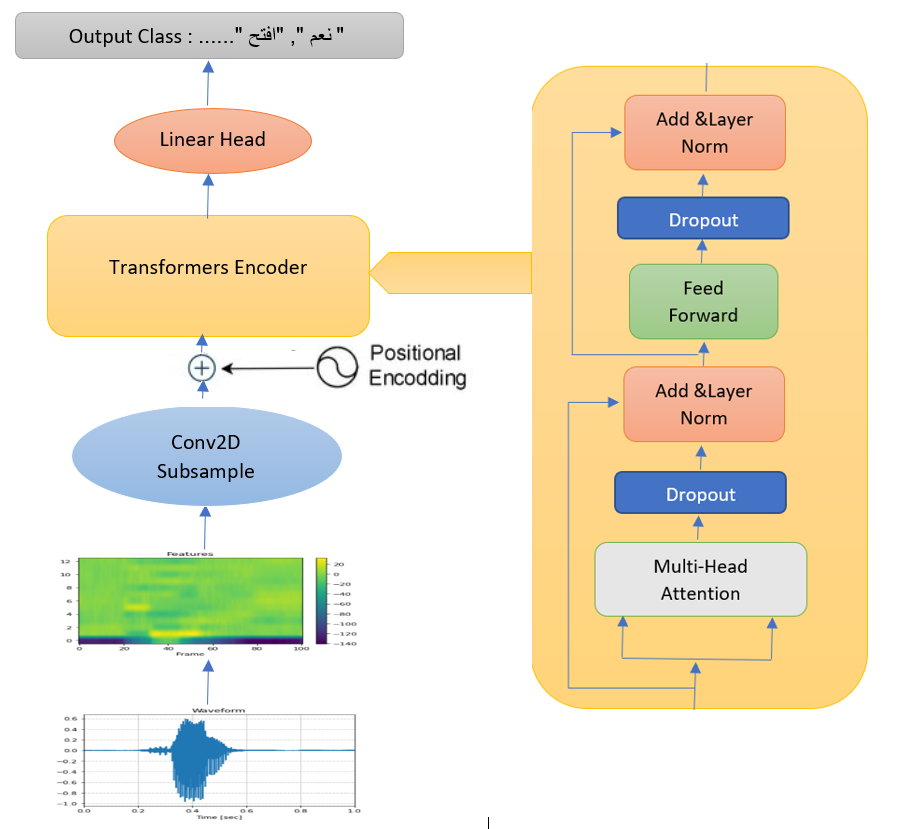
In order to implement such models, a large amount of speech command data are required for training, which is problematic for languages with limited training data, namely for the Arabic language, which is included in the group of agglutinative languages. To date, systems have been developed based on the different models of RNN for recognizing Arabic speech Commands with different sets of training data. The use of Transformers methods and models to improve the accuracy of recognition of Arabic speech is a promising direction to improve the performance of the system and speed up the training. our models are trained on an ASCR dataset, which consists of pairs of audio and corresponding phrase-level labels.

The main goal of our study is to improve the accuracy of the automatic recognition system for Arabic commands speech and speed up the training process by the use of models based on Transformer. Where our work is the first work that adopts the transformers model for SCR in the Arabic language.

* 1. [**Literature Review**](https://link.springer.com/article/10.1007/s10278-022-00666-z#Sec2)**:**

With the advancement in the computer sciences and technology, transfer learning, which is primarily a substantial feature of deep learning, is now become indispensable to many applications as an integral part. It has been used by different fields of research in order to apply it in the field of Speech Recognition, [34,35,36,37]. Speech Emotion recognition[38, 39], and some are applied to commands and keyword recognition. Besides this, Google has approved the related research on Speech Commands [40],

* 1. **Speech Transformer Encoder:**

 The architecture of the Transformer encoder is illustrated in [**Figure 1**](https://www.mdpi.com/2076-3417/12/5/2626#fig_body_display_applsci-12-02626-f001). It consists of a stack of encoder blocks, wherein each block has two sublayers: a multi-head attention layer and a position-wise feed-forward layer. A residual connection was constructed around each of the sublayers. In each sublayer, dropout and layer normalization were added for regularization.

**Figure 1**: The Keyword Transformer architecture. Audio is preprocessed into an MFCC spectrogram, and a positional encoding is added. The output of the class is passed through a linear head and used to make the final class prediction.

*3.1. Positional Encoding*

Since the Transformer encoder does not contain recurrence, the order information in the input data sequence should be added to the model. For this purpose, some information corresponding to the position of the data, called positional encoding, is artificially generated and added to the input data. The positional encoding is given by:

where pos is the position in an input sequence, is the dimension of a sequence, and is the number of attention units.

*3.2. Multi-Head Attention*

The attention mechanism used in the Transformer encoder is referred to as "Scaled Dot-Product Attention" and is defined as follows:

where the attention output is computed as the weighted sum of the values, . The weights assigned to the values are computed by the dot products of the queries, with the corresponding keys, is the dimension of the key. When , as is in the Transformer encoder used in this study, the attention is called "self-attention".

Instead of the single attention output, as in (2), multi-head attention (MHA) that performs multiple attention is used in the Transformer encoder. MHA projects the queries, keys, and values with different linear transformations and then performs the attention function for each of them. The different attention outputs are then concatenated and projected again to obtain the final output. The MHA is defined as follows:

where the learnable projections are , and . In addition, we have .

*3.3. Feed-Forward Networks*

In addition to the MHA layer, a fully connected feed-forward network is applied to each position separately and identically. This is composed of two linear layers and is defined as follows:

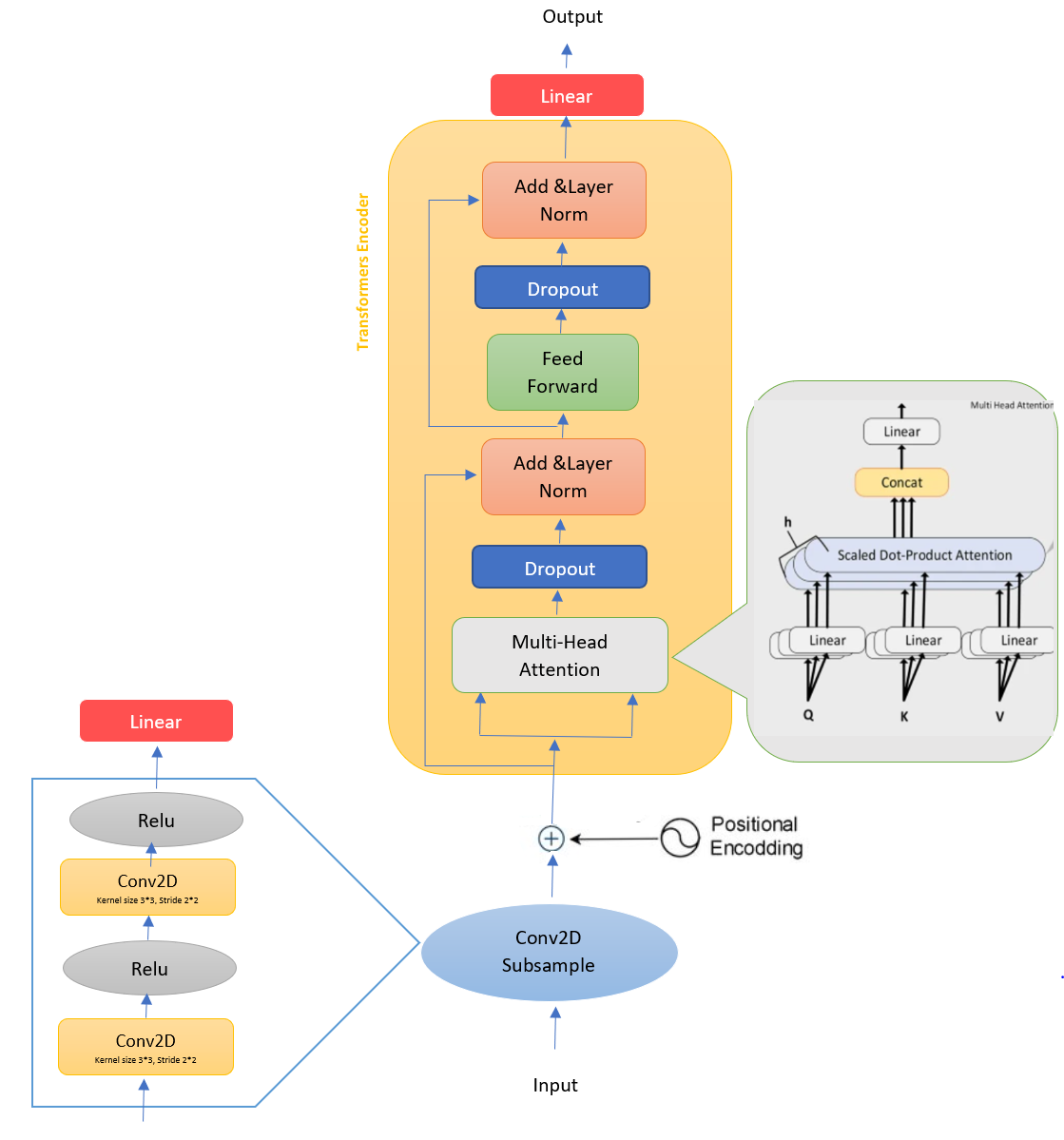
where is the -th frame of the input sequence , and ,

**4) Method**

#### 4.1. Proposed Transformer Network Model

The architecture of the proposed Transformer model is shown in Figure 2. It consists of the baseline Speech-Transformer aims at transforming the words feature sequence to the corresponding character sequence. The feature sequence, commonly a few times longer than the character sequence, can be depicted as 2-dimensional spectrograms with time and frequency axes. Therefore, we choose the convolutional networks to exploit the structure locality of spectrograms and mitigate the length mismatch by striding along time. Based on the above, we present the model architecture of the Speech-Transformer and the details of it encoder as follows:

We first stack two 3×3 CNN layers with stride 2 for both time and frequency dimensions to prevent the GPU memory overflow and produce the approximate hidden representation length with the character length. Then, we perform a linear transformation on the flattened feature map outputs to obtain the vectors of the dimension , which is called input encoding here. Afterwards, in order to enable the model to attend by relative positions, the dimensional positional encoding is added to the input encoding We can obtain the final encoded outputs by inputting the sum of input encoding and positional encoding to a stack of Transformer Encoder, each of them has two sub-blocks: The first is a multi-head attention whose queries, keys and values come from the outputs of the previous block. And the second is position-wise feed-forward networks. Meanwhile, layer normalization and residual connection are introduced to each sub-block for effective training



**Figure 2.** Proposed Transformer Network Model.

*4.2. Feature Extraction*

Feature extraction is also known as speech parameterization, and it was used to characterize

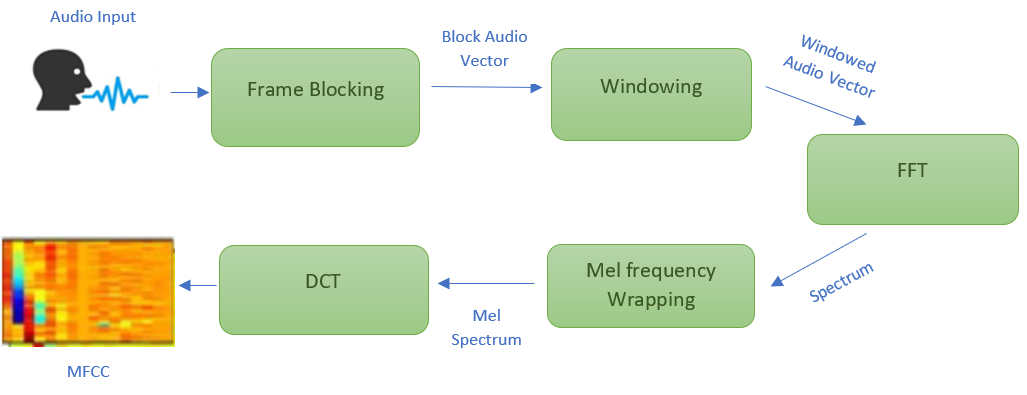
spectral features of an input audio signal in order to facilitate speech decoding. The Mel frequency

cepstral coefﬁcient (MFCC) introduced by Davis et al. is one of the most popular techniques

for feature extraction in speech recognition systems. The reason behind the popularity of MFCC is

its ability to mimic the behaviour of the human ear. Figure 2 shows the key steps involved in the

calculation of MFCC features.



4.2.1. Frame Blocking

The ﬁrst step involved in the process is frame blocking, in which a streaming audio signal is

blocked into frames of 25 ms shifted by 10 ms.

4.2.2. Windowing

After blocking, each frame is multiplied by a window using a windowing function. There are

many windowing functions available in Kaldi, but usually, the Hamming window is used, as shown

in Equation (1):

where Nis the number of samples in a frame, and multiplying every frame by the Hamming window reduces discontinuities at the beginning and end of each frame. This step was also required to do the frequency analysis of each frame.  
4.2.3. Fast Fourier Transform  
Spectral analysis of speech signals showed that different timbres in a signal have different energy distributions over frequency. FFT was applied to each frame of samples to obtain its magnitude frequency response. This process converted signals from the time to the frequency domain. FFT is a fast implementation of the discrete Fourier transform (DFT).

4.2.4. Mel Frequency Warping

In this step, the magnitude frequency response resulting from FFT was multiplied by triangular bypass filters on the Mel scale to get the log energy of each bypass filter. The Mel frequency that is more discriminative at lower frequencies and less discriminative at higher frequencies mimics the non-linear perception of sound by the human ear. The human ear behaves logarithmically towards speech both on the amplitude and frequency scales, i.e., the frequency resolution is better (high) at lower frequencies and low, particularly at high frequencies. Furthermore, in the human inner ear cochlea, sound waves are transduced into electrical impulses that can be interpreted by the brain as individual frequencies of sound [44]. This is similar to how cepstral coefficients are calculated in MFCC. We could convert between Mel frequency and frequency in hertz by using Equations ( 2 ) and (3).

Each filter in the triangular filter bank had a response of one at the centre frequency and decreased linearly towards zero until it reached the centre of the adjacent frequency. In Kaldi, the default number of filters is 23 because it usually gives the best results on speech signals.

4.2.5. Discrete Cosine Transform

After log energy computations, the Mel frequency cepstrum was obtained by applying DCT on filtered results. The coefficients of the Mel frequency cepstrum are called Mel frequency cepstral coefficients (MFCC). Using DCT after FFT transforms the frequency domain into the time-like domain called the quefrency domain. In Kaldi, by default, the first 13 cepstral coefficients are kept as features. MFCC can be directly used as features for speech recognition. However, to get better performance, various transforms are applied to the results of MFCC. One of these transformations is cepstral mean and variance normalization (CMVN) [45]. CMVN is a computationally efficient normalization technique that reduces the effects of noise. Similarly, to add dynamic information to MFCC features,  
first and second order deltas can be calculated. Given a feature vector X, first-order deltas can be calculated by using Equation (4).

where is the regression coefficients and is the window width. Second-order deltas can be derived from first-order deltas by using Equation (5).

After the first- and second-order delta calculation, the combined feature vector becomes:

Other feature transformation techniques used in Kaldi are linear discriminant analysis (LDA) [46], heteroscedastic linear discriminant analysis (HLDA) [47], and maximum likelihood linear transform (MLLT) [48]. These transforms can be applied individually, as well as in multiple combinations to enhance the performance of a speech recognition system. It was observed that by applying diagonalizing MLLT after LDA improved the effect of LDA (LDA + MLLT).

## **5) Experiments & Result**

5.1. Dataset

Our dataset is a list of pairs (x, y), where x is the input speech signal, and y is the corresponding keyword. The final dataset consists of 12000 such pairs, comprising 40 keywords. Each audio file is one-second in length sampled at 16 kHz. We have 30 participants, each of them recorded 10 utterances for each keyword. Therefore, we have 300 audio files for each keyword in total (30 \* 10 \* 40 = 12000), and the total size of all the recorded keywords is ~384 MB. The dataset also contains several background noise recordings we obtained from various natural sources of noise. We saved these audio files in a separate folder with the name background\_noise and a total size of ~49 MB.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Translation | Keyword | Translation | Keyword | Translation | Keyword | Translation | Keyword |
| Zero | صفر | Right | يمين | Enable | تفعيل | Send | إرسال |
| One | واحد | Left | يسار | Disable | تعطيل | Receive | إستقبال |
| Two | اثنان | Up | اعلى | Ok | موافق | Move | تحريك |
| Three | ثلاثة | Down | اسفل | Cancel | الغاء | Rotate | تدوير |
| Four | اربعة | Forward/Font | امام | Open | فتح | Record | تسجيل |
| Five | خمسة | Backward/Back | خلف | Close | اغلاق | Enter | ادخال |
| Six | ستة | Yes | نعم | Zoom in | تكبير | Digit | رقم |
| Seven | سبعة | No | لا | Zoom out | تصغير | Direction | إتجاه |
| Eight | ثمانية | Start | ابدأ | Previous | السابق | Options | خيارات |
| Nine | تسعة | Stop | توقف | Next | التالي | Undo | تراجع |

Table 1: The 40 chosen keywords with Their Translations

Data Split

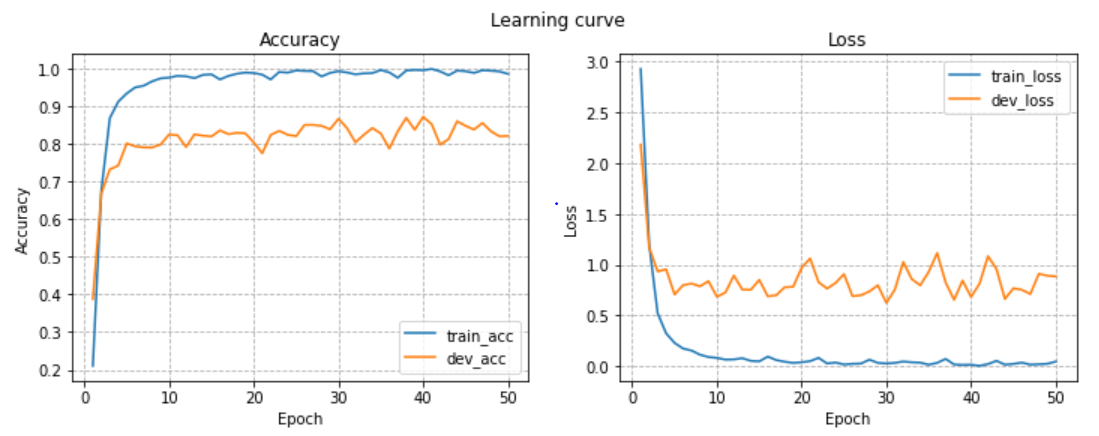
We recommend using the provided CSV files in your experiments. We kept 60% of the dataset for training, 20% for validation, and the remaining 20% for testing. In our split method, we guarantee that all recordings of a certain contributor are within the same subset.

5.2. Experimental Details (Network Training)

The input data of the model were 40-dimensional Fbank features. We performed experiments using Ne=3 Transformer layers for the encoder, respectively, with d=64 and h=2. the window size was 25. Considering the variable length of input data sequences, we used dynamic Batch to improve memory utilization. The batch size is 32. The transformer was trained with cross-entropy loss. We used the Adam optimizer with a learning rate of 0.001 and. The dropout method in both the position encoding and attention matrix was 0.1. PyTorch was used as the backend framework. Moreover, on a GPU-based desktop machine with 128 GB RAM, Nvidia GEFORCE RTX 3050 (32 GB VRAM), and an AMD RYZEN 9 processor, we train networks.

.

Figure 5 shows the learning curves of the transformers-based.



#### 4.3. Comparison Experiments

Arabic Speech Commands is a multiples classification task in which audios are classified as keywords. Through experimentation on a dataset of Arabic Speech Commands, it is revealed that pre-trained transformer transfer learning performs better as compared to other state-of-the-art deep learning models. Performance comparison of Speech transformer and other deep learning models is shown in Table

|  |  |  |
| --- | --- | --- |
|  | Accuracy | Time Take |
| DNN | 81.91 |  |
| LSTM | 83.54 |  |
| CNN | 91.10 |  |
| Transformer based | 86.55 |  |

three diﬀerent NN-based approaches models are outlined below. All of these models use cross-entropy loss with a softmax output layer of 41 units (the number of classes); adam optimizer with 10−3initial learning rate; Rectiﬁed Linear Unit (ReLU) activation function in the hidden layers; 32 mini-batch size and 10−3L2 weight decay; and a learning rate scheduler that multiplies the learning rate by 0.1 when the training loss plateaus for 5 consecutive epochs. These models are trained for 50 epochs.

DNN

4.3.1. DNN

A simple feedforward fully connected neural network (Goodfellow et al., 2016, Ch. 6) of 3 stacked hidden layers of 256 hidden units in each. Each layer is succeeded by a batch normalization layer (Ioﬀe and Szegedy, 2015) (note that Logistic Regression can be viewed as a simple case of DNNs).

CNN

4.3.2. CNN

A convolutional neural network (Goodfellow et al., 2016, Ch. 9) of 4 stacked convolutional layers with

16, 32, 64, and 128 channels (feature maps), respectively. Each layer is followed by batch normalization (Ioﬀe and Szegedy, 2015) and a 2×2max-pooling layer. Finally, these layers are succeeded by a dropout layer (Srivastava et al., 2014) with 0.25 omission probability and a fully connected feedforward layer with 256 hidden units.

LSTM

4.3.3. LSTM

A recurrent neural network (Goodfellow et al., 2016, Ch. 10) with two LSTM layers (Hochreiter and

Schmidhuber, 1997), followed by a dropout layer (Srivastava et al., 2014) with 0.5 omission probability, and ﬁnally succeeded by two Time Distributed fully connected layers with 128 and 64 hidden units.

Conclusion