IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AN OVERVIEW OF MODERN ERA SPEECH RECOGNITION MODEL

Chhayarani Ram Kinkar[1], Yogendra Kumar Jain[2]

[1]Research Scholar Electronics & Telecommunication Department, R.G.P.V., Bhopal, M.P. India

[2]Electronics & Instrumentation Department, Smarat Ashok Technological Institute, Civil Lines, Vidisha, M.P. India

Abstract: In the modern era, the state of the art approach for interaction with digital devices is replaced by the more natural form that is by using natural language speech commands. The natural language speech commands are recognized by the digital devices with the help of the speech recognition model. The presented paper gives an overview of two more popular speech recognition models employed in modern era digital devices. Moreover the presented paper also gives an overview of the implementation of different layers of the model on the popular Tensorflow platform. This paper helps the beginner to understand the basics.

Keywords: Speech recognition; Tensorflow; Convolutional neural network; Recurrent neural network.

I. INTRODUCTION

The definition of speech recognition according to award winning Macmillan Dictionary for advanced learner [1] is “a system where you speak to a computer to make it do things, for example instead of using a keyboard”. This definition is published in the year 2002 by Macmillan education and it is very true for the current era in which applications of speech recognition are rocking. Also Mr. Bill Gates, co-founder of Microsoft Corporation, in his book “The Road Ahead” [2] described speech recognition as one of the most important innovation. His company “Microsoft” [2] launched ‘window 10’ on July 15 which is based on the speech recognition framework [3]. ‘Window 10’ consists of personal digital assistance that communicates with users through speech. The name of this assistance is “ Crotona” [2]and it is based on speech recognition and synthesis. Besides “Google” launched “Google now” [5] and “Apple” launched “Apple Siri” [5]digital assistant which communicates with users through speech commands [3].

All these modern era speech-based assistance are developed using different forms of artificial intelligence[6] and personalization and communicate with users through speech for performing the web search, entertainment, weather information, travel assistants, reminders,[5] etc. The development of “Google now”, “Apple Siri”, “Crotona” etc. is a complex task and involves various phases and complicated algorithms. However, for novel engineering applications, researchers and scientists developed much simple speech interaction based digital assistance that runs locally and performs specific tasks according to application. The simple speech interaction based digital assistance developed by researcher and scientists employ end to end speech recognition model [7] instead of traditional speech recognition model that is composed of acoustic model, language model and decoder,[3],[4] etc. In the presented paper an overview of the speech recognition model employed in modern era speech interaction based digital assistance is discussed. Also, the presented paper will focus on the Tensorflow platform [12] for implementing the modern era speech recognition model.

2. MATERIAL AND METHOD

2.1Method

In the modern era, the trade of communication between human and non-human devices is replaced by a more natural form that is in the same way as the human communicates with each other. Nowadays it becomes fairly common that humans control intelligent appliance at their home, personal assistant build in their smartphone, or automatic dialogue systems [5] in the call center with the help of natural language speech as illustrate in figure1.

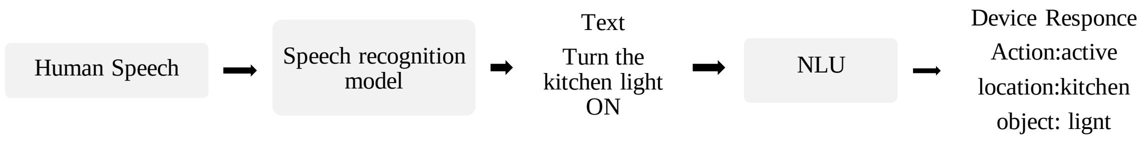


Fig1: Speech based interaction between user and smart home appliance in the form of block diagram

In the modern era, the replacement of communication devices (keyboard, mouse, touch screen, etc.) with speech is possible with the introduction of advanced machine learning algorithms [8], and neural networks [5], etc. Moreover, the scientist and researchers developed many simple end to end speech recognition frameworks. Of course, the development of speech recognition framework for modern era speech interaction based digital assistance is not a simple process it involves various subsystems, for example, front end subsystem[4] and back end subsystem[4] as illustrated in figure 2.

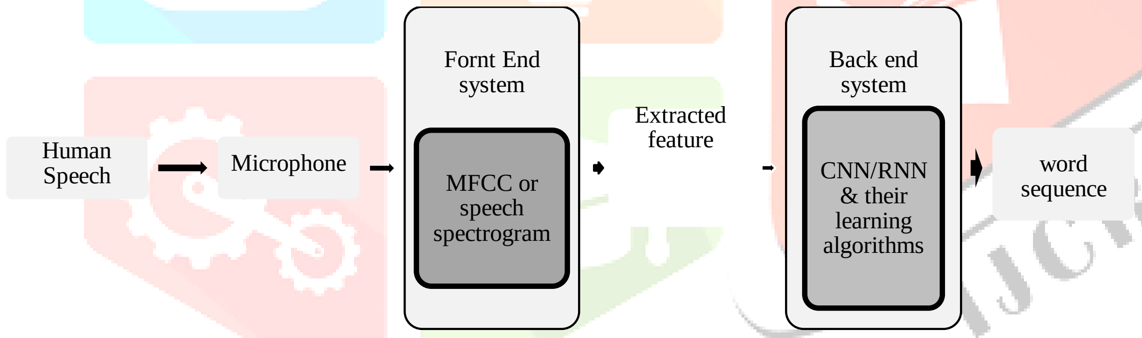


Fig.2: Modern era speech recognition process using RNN or CNN

As illustrated in figure 2 speech recognition process in modern era speech interaction base digital assistance begins with capturing human speech with the help of an electronic device –microphone. Next, the received speech is processed in a way similar to human visualization, analysis, and characterization of speech with the help of an expert system (neural networks) [6]. The expert system integrates acoustic, phonemic, lexical, syntactic, semantic knowledge [4] and recognizes the speech fast, more accurately as compare state of the art model [7].

The expert system in modern era speech interaction based digital assistance is build-up by using either Recurrent Neural Network (RNN) [10],[11] or Convolutional Neural Network (CNN) [5],[6],[7]. A detailed description of the speech recognition process employing an expert system is given in the following text.

2.1.1 Speech recognition employing convolutional neural network

Convolutional neural networks are introduced in 1990 and originally used in image processing applications [3]. In the year 2012 first time convolutional neural networks are used for speech recognition in a large vocabulary continuous speech recognition task [7]. From the year 2012 till today convolutional neural networks has achieved many state of the art results in various speech recognition tasks. The detailed architecture of the speech recognition model employing convolutional neural networks is illustrated in figure 3.

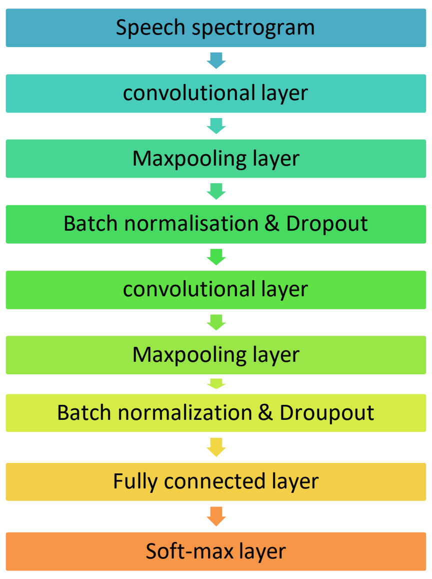


Fig 3: Speech recognition model employing CNN

As illustrated in figure3 the speech recognition model employing convolutional neural networks learns from speech spectrograms [3] and performs convolution, nonlinearity, pooling, classification, [6]operation on it for recognition purposes. Optionally the model will include batch normalization [8] and dropout operation [9] by stacking them with other operations.

Convolutional operation: The convolution operation on the input spectrogram is performed in the convolutional layer. The convolution operation will be either 1-dimensional or 2-dimensional [3]. To perform the convolution operation the convolutional layer consists of learnable filters [6]. The size of each learnable filter is small so that all the crucial details of the input spectrogram are learned deeply [7]. For learning the learnable filters strides over the entire input independently and generate output feature maps [3] for the filters of the next layer. The number of learnable filters in each convolutional layer and the number of convolutional layers in the CNN based speech recognition model will depend on the required recognition accuracy [5].

Non linearity operation: The learned information by filters of the convolutional layer is passed to the next layer through the nonlinearity operation. The nonlinearity operation is performed using the activation function. Maximum speech recognition model employing CNN uses Rectified Linear unit (Relu) activation function to perform nonlinearity operation [6] because this function has more training efficiency as compared to other function (for example sigmoid and tanh). Relu takes as input x and returns $\max(0, x)$ as an output [15].

Pooling operation: The pooling operation is performed on input received through an activation function. This operation will merge semantically similar features learned during the convolutional operation [5]. While merging the features this operation will keep maximum (max-pooling) [5] or average value (average pooling) [6] within each patch. Also, this operation reduces shifts and distortions in the input features by reducing the size. The pooling operation is performed in the max-pooling layer (most the CNN speech recognition model use max-pooling operation researcher may applying average pooling operation and compare the result) of CNN and this operation makes the speech recognition model employing CNN deeper [5].

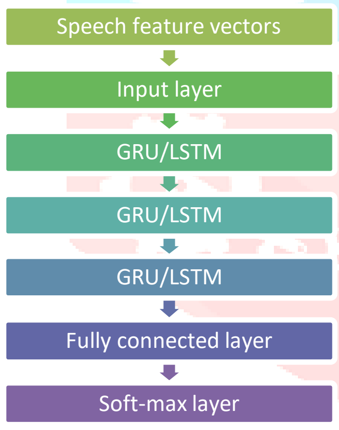
Batch normalization operation: This operation is optional. Normally this operation is included in the speech recognition model employing CNN to improve the training of the model. During training of the model network parameters are updated and distribution of weight activation changes i.e. internal covariance shift[9]. The internal covariance shift makes the training process slow and introduces many constraints on parameter initialization to maintain an acceptable learning rate of the model. The batch normalization operation solves the problem of low learning rate and careful parameter initialization by introducing two trainable parameters ‘gamma’ and ‘beta’[8]. Both the parameters normalize the output from previous layers and accelerate the learning rate.

Dropout operation: This operation is also optional. The dropout operation is performed in the speech recognition model employing convolutional neural networks for regularization [8],[9]. The dropout operation forces convolutional neural networks to learn more general representation rather than a more specific representation from the input received from the preceding layers [8]. This operation will reduce the training time.

Classification operation: The classification operation is performed at last with the help of a fully connected layer and "softmax" function [25]. This operation allows the matching of the output size of the convolutional neural network to the desired output size of the speech recognition model as per application [11]. Normally in the speech recognition model employing convolutional neural networks fully connected layers are placed after the last convolutional layer. The fully connected layer receives the output in the form of feature maps or a fatten output of the last max-pooling layer [10]. The classification operation in a fully connected layer passes the received information through the "softmax" function in the form of a probability representation for the prediction.

2.1.2 Speech recognition employing recurrent neural network

Recurrent neural networks are introduced in the year 1997 and used for sequential data processing [10]. Recurrent neural networks are the variants of feed forward neural networks containing a feedback loop. The feedback loop feedbacks activation to the next layer as well as to the current layer in the next time steps [10]. This design of recurrent neural networks enables the employment of recurrent neural networks architecture in the number of speech recognition applications for example both the variants of recurrent neural networks i.e. Bidirectional Gated Recurrent Unit (BGRU) and Bidirectional Long Short term memory (BLSTM) are heavily applied in Google's "Google now", Amazon's "Alexa", Microsoft's "Cortana" [3],[5] etc. The detailed architecture of the speech recognition model
employing variants of recurrent neural networks is illustrated in figure 4.



.

Fig 4: Speech recognition model employing RNN

As illustrated in figure 4 in large vocabulary continues speech recognition task deep recurrent neural networks architecture for recognition purpose is formed by stacking multiple layers of LSTM cells [10] or GRU cells[11] together. Each LSTM/GRU cell for recognition purposes scans the input feature vector from left to right and computes hidden vector sequence for feedback loop and output vector sequence for the next layer [4]. The output vector sequence is passed to the next layer using either of the Relu or sigmoid or hyperbolic tangent function [15]. The output of the last LSTM/GRU layer is fattened to 1-dimension shape and feed to a fully connected layer for the generation of output word sequence using the 'softmax' activation function. The speech recognition model employing recurrent neural networks is trained using Error Back Propagation Through Time (EBPTT) algorithm [11]. The EBPTT algorithm converts the LSTM/GRU cells into a purely feed forward system by folding the network overtime during training.

2.2 Material

Modern era speech recognition models are implemented on advance natural language interface platforms (NLIP) [14] for example Tensorflow, Theano [13], etc. The advanced NLIP contains the number of deep learning libraries [12] and helps in understanding complex sentence structure. Although the number of platforms available for implementing the layers of modern era speech recognition model the unbeatable leader preferred by the researcher community is Tensorflow [12].

Tensorflow is developed by the Google brain team and it is a symbolic math library based on dataflow [19]. The Tensorflow library contains multiple high level application program interfaces (API) [12] for implementing the speech recognition models by using either variant of neural network i.e. CNN or RNN. The Tensorflow APIs are available in multiple programming languages for example Python, Java, C++,[12] etc., and the most popular Tensorflow API for implementing layers of speech recognition model is Keras [18].

Keras is a python interface and provides support for implementing CNN or RNN layers as well as other supporting layers for example input layer[17], zero padding layer[17] etc. The implementation of CNN or RNN layer in Keras along with other supporting layers required for speech recognition is discussed in the following text.

2.2.1 Convolutional layer implementation

On the Tensorflow platform using Keras interface, learnable filters of the convolutional layer are implemented by defining the arguments [20] illustrated in figure 5. The illustrated arguments i.e. the size of filter, its striding rate, activation function [21] for passing the information are defined by the developer of the model according to the application and required model size. Besides if it is necessary to shift the output then a bias is added to the output through use_bias [21] function. The developer will add the bias in the output by setting use_bias function as a true value. Moreover, the developer will set the padding in the convolutional layer either 'valid' or 'same'[20] according to application requirements. The rest of the parameters while implementing convolutional layer are set according to the class definition [21]. The implemented learnable filters as per the illustration of figure 5 scan the input spectrogram to produces an output tensor for the next max-pooling layer.

```
import tensorflow as tf:
   tf.keras.layers.Conv2d(filter,
      Kernel_size=# to be define by developer of the model according to application,
      strides= # to be define by developer of the model according to application,
      padding=# to be set either "valid" or "same" as per the application,
      dilation_rate=# set according to stride,
      activation=# to be select by developer of the model according to application,
      use_bias=# to be set either "True" or "Flase",
      kernel_constrains=# set by developer as application specific
      kernel_initializer="glorot_uniform"(according to class definition),
      bias_initializer="zeros",
      kernel_regularizer= None,
      bias_regularizer= None,
      bias_constraint= None,)
```
Fig.5 Convolutional layer implementation on Tensorflow

2.2.2 Max-pooling layer implementation

On the Tensorflow platform using Keras interface, a 2D-max-pooling layer for the reduction in dimension is implemented as illustrated in figure 6. While implementing this layer the developer of the model needs to define only pooling size. The implemented max-pooling layer will reduce the dimension of input received from convolutional layer according to defined pool size as - (((size of input shape)-(pool size) +1)) / stride) for valid padding setting [22].

```
import tensorflow as tf:
   tf.keras.layers.Maxpooling2D(
      pool_size= # to be define by developer of the model,
      strides=None,
      padding="valid",)
```
Fig6: Max pooling layer implementation on Tensorflow.

2.2.3 Batch normalization and Dropout layer implementation

The optional batch normalization and dropout layer for speeding up the model training are implemented as illustrated in figure 7. While implementing batch normalization layer the developer of the model needs to define the batch size according to the application. The batch normalization layer will normalize the output of each batch by keeping mean of the output close to 0 and standard deviation of the output close to 1. After the normalization the layer returns a batch of size (((batch-mean(batch))/(var(batch)+epsilon)* gamma+ beta) [23].

Moreover while implementing dropout layer the developer of the model needs to define only dropout rate [24] according to the application to prevent overfitting of the model during training. The dropout layer will drop a fraction of the input unit according to the assign value of argument 'rate' in the dropout layer and speed up the training process [24].

```
import tensorflow as tf:
    tf.keras.layers.BatchNormalization(
        axis= -1,
        momentum= 0.99, (as per class definition)
        epsilon=# a constant scaler to be set by developer of the model,
        center=True,
        scale= True,
        beta_initilizer=# initially set as "zero" & learn during the training
        gamma_initializer =# initially set as "ones" & learn during training
        moving_mean_initializer="zero",
        moving_variance_initializer ="ones",
        beta_constraint= None,
        gamma_constraint= None,
        trainable= True,
        virtual_batch_size= None
        adjustment= None,)
    tf.keras.layers.Droupout(
        rate=# to be define by developer of the model,
        noise_shape = None,
        seed= None,)
```
Figure 7: Batch normalization and dropout layer implementation in Tensorflow

2.2.4 Fully connected and soft-max layer implementation

The final fully connected layer in both the modern era speech recognition model i.e. build-up by employing CNN architecture or by employing RNN is implemented as illustrated in figure 8. While implementing this layer the developer will define the kernel (by assigning value to arguments- units, activation, use bias) [19] in such a way that the fully connected layer provides an output{(output=activation (dot(input, kernel)+bias) [18]to "softmax" layer for the generation of output sequence. The "softmax" layer in both the modern era speech recognition model is defined as per class definition [16].

```
import tensorflow as tf:
    tf.keras.layers.Dense(
        units= # to be define by developer of the model,
        activation = # to be define by developer of the model,
        use_bias = # to be set either True or false,
        Kernal_initializer =" glorot_uniform",(as per class definition)
        bias_initializer = "Zero"
        kernel_regulizer= None,
         kernel _constraints= None,
        bias_constrains= None,)
    tf.keras.layers.softmax(
        axis=1,)
```
Fig 8: Fully connected layer implementation in Tensorflow

2.2.4 Recurrent layer implementation

On the Tensorflow platform using Keras interface the GRU layer [26] implementation is illustrated in figure 9. While implementing GRU cell developer needs to define unit, recurrent_activation, use_bias arguments. The rest of the parameters are set according to class definition [26]. While implanting modern era speech recognition model using RNN variants there is no need to implement a dropout layer separately. The dropout rate will be defined in the class definition. In case developer is building speech recognition model by employing LSTM variant [25]of RNN then in layer call definition the LSTM cell is called instead of GRU cell. The internal runtime hardware support of Keras will provide a predefined LSTM cell with defined arguments of developer.

```
import tensorflow as tf:
tf.keras.layers.GRU(
        units=# to be define by developer,
        recuurent_activation=#to bedefine by developer,
        use_bias=# to be set either "true" or "false"
        kernel_initializer="glorot_uniform"( according to class definition)
        recuurent_initilaizer="biderectional",
        bias_initializer="zeros",
        kernel_regularizer= None,
        bias_regularizer= None,
        kernel_constrains=# set by developer as application specific,
        bias_constraint= None,
        dropout=# to be set by developer,
        recurrent_dropout=# to set by developer,
        return_seuqence= False,
        return_state=False,
        go_backword=True,
        tateful=False,
        time_major=False,)
```
Fig9: RNN implementation on Tensorflow

3. CONCLUSION

The presented paper discusses the architecture of the speech recognition model employed in modern era speech interaction based digital assistance. The discussion will help the beginner to understand the basic layers employed in the model, organization of the layers inside the model, and performed operations by a particular layer. In addition the presented paper helps the beginner to understand the control provided by the leading platform TensorFlow to them for implementing the modern era speech recognition model. The presented paper provides ease to the beginner for customization of their modern era speech recognition model. The future work in the mention direction will include a discussion of customize layer implementation so that the beginners will implement their research ideas along with the exiting layer of Tensorflow.

Conflict of interest
On behalf of all authors, the corresponding author states that there is no conflict of interest.

REFERENCE

[1] https://www.macmillandictionary.com/dictionary/british/download_1
[2] https://www.gatesnotes.com/About-Bill-Gates/The-Road-Ahead-after-25-years
[3] Sanjay Krishna Gouda, Salil Kanetkar, David Harrison, Manfred K Warmuth "Speech Recognition: KeyWord Spotting through Image Recognition" Machine learning, DOI:arXiv.1803.03759.
[4] Rabiner LR, Juang B-H. Fundamentals of Speech Recognition. Englewood Cliffs: PTR Prentice Hall 1993
[5]Neil Zeghidour, Qiantong Xu, Vitaliy Liptchinsky, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert "Fully Convolutional Speech Recognition" computation and language, DOI: arXiv.1812.06864
[6] O. Abdel-Hamid, A. rahman Mohamed, H. Jiang, L. Deng,G. Penn, and D. Yu, "Convolutional neural networks for Speech recognition," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, pp. 1533–1545, 2014 DOI: 10.1109/TASLP.2014.2339736
[7] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in Acoustics, Speech and Signal Processing (ICASSP),2015 IEEE International Conference on IEEE, 2015, pp. 4295–4299 DOI:10.1109/ICASSP.2015.7178781
[8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research,vol. 15, no. 1, pp. 1929–1958, 2014, Corpus ID:6844431
[9] Hye-Woo Lee , Noo-ri Kim , Jee-Hyong Lee "Deep Neural Network Self-training Based on Unsupervised Learning and Dropout" 2017,International Journal of Fuzzy Logic and Intelligent Systems Vol17, No1,pp. 1-9
[10] Parnia Bahar Albert Zeyer Ralf Schluter Hermann Ney "On Using 2D Sequence-To-Sequence Models For Speech Recognition", 2019, IEEE International Conference on Acoustics, Speech and Signal Processing pp2131-2142
[11] Jinyu Li, Rui Zhao, Hu Hu ,Yifan Gong "Improving RNN Transducer Modeling For end-to-end Speech Recognition" Speech and Language Group, Microsoft https://www.microsoft.com/en- us/research/uploads/prod/2019/10/RNNT.pdf
[12] William Grant Hatcher and Wei Yu "A survey of deep learning: platforms, applications and emerging research trends",2018,IEEE accesses,special section on human-centered smart systems and technologiespp24411-24438
David Foster Genrative Deep Learning, 2019, O'Reilly media Inc.
[13] https://www.tensorflow.org/api_docs/python/tf/keras/layers
[14] https://www.tensorflow.org/tutorials/customization/custom_layers
[15] https://keras.io/api/layers/constraints/
[16] https://keras.io/api/layers/initializers/
[17] https://keras.io/api/layers/
[18] https://www.tensorflow.org/tutorials/images/cnn
[19] https://www.tensorflow.org/guide/keras/rnn
[20] https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
[21] https://keras.io/api/layers/#convolution-layers
[22] https://keras.io/api/layers/convolution_layers/convolution2d/
[23] https://keras.io/api/layers/pooling_layers/max_pooling2d/
[24] https://keras.io/api/layers/normalization_layers/batch_normalization/
[25] https://keras.io/api/layers/regularization_layers/dropout/
[26] https://www.tensorflow.org/api_docs/python/tf/keras/layers/GRU