Towards Robust Combined Deep Architecture for Speech Recognition : Experiments on TIMIT

Hinda DRIDI[1], Kais OUNI[2]
Research Laboratory Smart Electricity & ICT, SE&ICT Lab, LR18ES44
National Engineering School of Carthage, ENICarthage
University of Carthage, Tunis, Tunisia

***Abstract*—Over the last years, many researchers have engaged in improving accuracies on Automatic Speech Recognition (ASR) task by using deep learning. In state-of-the-art speech recognizers, both Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) based Reccurent Neural Network (RNN) have achieved improved performances compared to Convolutional Neural Network (CNN) and Deep Neural Network (DNN). Due to the strong complementarity of CNN, LSTM-RNN and DNN, they may be combined in one architecture called Convolutional Long Short-Term Memory, Deep Neural Network (CLDNN). Similarly we propose to combine CNN, GRU-RNN and DNN in a single deep architecture called Convolutional Gated Recurrent Unit, Deep Neural Network (CGDNN). In this paper, we present our experiments for phoneme recognition task tested on TIMIT data set. A phone error rate of 15.72% has been reached using the proposed CGDNN model. The achieved result confirms the superiority of CGDNN over all their baselines networks used alone and also over the CLDNN architecture.**

Keywords—Automatic speech recognition; deep learning; phoneme recognition; convolutional neural network; long short-term memory; gated recurrent unit; deep neural network; recurrent neural network; CLDNN; CGDNN; TIMIT

I. Introduction

Speech has always been regarded as the most common mode of communication between humans. With the recent development, this mode of speech communication has also been used for human-machine interaction. Current technology allows machines to process and respond reliably to basic human speech. Using speech as input in the human-machine dialogue puts technology within the reach of all, while the classic manual input techniques cannot really satisfy the needs of people with physical disabilities.

Understanding speech is a difficult task for a machine. Just like the human brain, a machine must first recognize speech. Automatic Speech Recognition (ASR) is a primary step that allows machine to understand the oral information given by a human user. It consists of using several matching techniques to compare an inputted utterance to a set of samples. A common ASR application is to transcribe human speech into a textual representation, which can be further exploited in many applications such as language identification, information extraction, retrieval technology, archiving, etc.

In spite of the huge progress in signal processing, computational resources and algorithms, we are far from getting ideal ASR systems. Thus, we should open the most relevant research topics to attain the principal goal of Automatic Speech Recognition. In fact, recognizing isolated words is not a hard task, but recognizing continuous speech is a real challenge. Any ASR system has two parts: the language model and the acoustic model. For small vocabularies, modeling acoustics of individual words is may be easily done and we may get good speech recognition rates. However, for large vocabularies, developing successful ASR systems with good speech recognition rates has become an active issue for researchers. As vocabulary size grows, we will not be able to get enough spoken samples of all words and we should model acoustics at a lower level. The ASR systems may use sub-word units of words, called phonemes, for both training and decoding process [1],[9].

From last five decades, HMM is considered as the leader to model temporal variability in speech signals. Each phoneme is modeled by a set of HMM states, currently a left-to-right HMM topology with three emitting states is used. Using HMM alone for training a speech recognition system did not bring promising results. To increase the performance researchers introduced several new or hybrid classifiers. And GMM became an integral part of HMM to model the acoustic characteristics of speech at each phonetic state using an n-gram language model [1],[2],[4].

Still there is a vast scope for improving speech recognition systems, since GMM is not efficient to model data on or near to nonlinear manifold in existing data space. An efficient generative modelling technique is so required to solve this problem. Among the recent researches, attention may be attracted to deep learning, which has been shown able to solve complex real-world problems. Deep learning has been successfully used for Automatic Speech Recognition (ASR) to reach better gains. Many researchers have proposed to replace GMM with a deep model to estimate the posterior probabilities of each HMM state. Deep neural models combined with hidden Markov models became the most dominant approach for speech recognition replacing the classic (GMM-HMMs) approach [4]-[6],[8]-[9].

In the last few years, deep learning has been progressively evolved, offering more efficient architectures. Earlier works exploiting deep learning for speech recognition were based on classic multilayer perceptron (MLP). Recently, with the advent of graphical processing unit (GPU) that is able to provide interesting processing power and huge memory to handle ample amount of datum, more advanced architectures have been proposed like, Deep Neural Network (DNN), Convolutional Neural Network (CNN), both Long-Short Term Memory network (LSTM) and Gated Reccurent Unit (GRU) based Recurrent Neural Network (RNN). Of all these architectures, recurrent neural networks based models have shown very strong comparative performance [1],[2].

This paper is organized as follows: in Section 2, we present the related works to continuous speech recognition and in particular to the phone recognition task. Principally, we present the most promising deep architectures; namely, CNN, LSTM, GRU and DNN. In Section 3, we describe our proposed deep combined CLDNN and CGDNN structures. Section 4 presents our experimental setup and results for the TIMIT data set. Finally, we draw some conclusions and we outline our future works in Section 5.

II. Related Terminologies

This section aims to provide an overview on the most performing deep learning architectures used for speech recognition and to discuss the nuances between these models.

A. *Deep Neural Network (DNN)*

As shown in Fig. 1, a Deep Neural Network (DNN) is a conventional Multi-Layer Perceptron (MLP) containing several layers of hidden units stacked on top of each other between the input and output layers. The hidden units of each layer are connected to those of the next layer using unidirectional connections. The DNN is so able to extract more robust and significant features of the input data via these different non-linear hidden layers [2]-[3].

Generally, the logistic function is used by each hidden unit to map its total input from the previous layer x_j to the scalar state y_j that will be send to next layer.

$$y_j = \text{logistic}(x_j) = \frac{1}{1+e^{-x_j}} \quad , \quad x_j = b_j + \sum_i y_i w_{ij} \tag{1}$$

where b_j is the bias of hidden unit j, i is an index over the units in previous layer and w_{ij} is the weight of connection between unit j and unit i of previous layer [5]-[6].

The main idea of a DNN is to pass the speech input to a first layer that will extract a new representation from this input and will then pass the output as input to a layer above. The next layer will produce a new higher representation from its input, and so on; these steps will be repeated with the next layers, until reaching the last layer [5], [7].

For multiclass classification, the output unit j uses a "softmax" non-linearity to convert its total input, x_j into a class probability p_j .[5]-[6]

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)} \tag{2}$$

where k is an index over all classes.

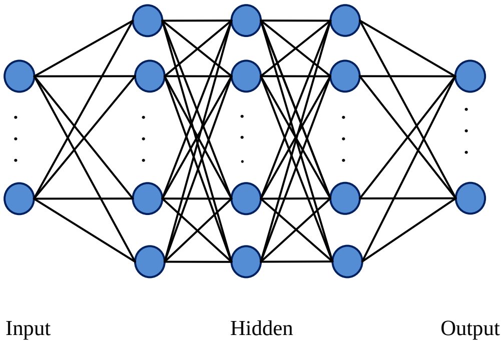


Fig. 1. Fully-Connected DNN Structure.

The DNN may be discriminatively trained in a supervised way by back-propagating the derivatives of a cost function, which computes the error between the network outputs and the desired outputs. This cost function can be defined as a cross-entropy between the target probabilities d and the outputs of the softmax, p: [5]-[7].

$$C = -\sum_j d_j \log p_j \tag{3}$$

where the target probabilities are provided for training the DNN.

In the case of large training sets, it's more interesting to calculate the cost function derivatives on small "mini-batch" randomly taken from the whole training set, before updating the weights in proportion to the gradient. This method called stochastic gradient descent (SGD) may be improved, if we use a "momentum" coefficient $0 \prec m \prec 1$ that smooths the gradient computed for a mini-batch t [5]-[6].

$$\Delta w_{ij}(t) = m\Delta w_{ij}(t-1) - \varepsilon \frac{\partial C}{\partial w_{ij}(\mathrm{t})} \tag{4}$$

The DNN model has been largely used for many speech recognition tasks; with two different training approaches either supervised approach or unsupervised one. This later approach, namely unsupervised training, was proposed by Hinton et al. in [5] and it was done using the weights of a deep belief network (DBN) built by stacking several Restricted Boltzmann Machines (RBMs) on top of each other [5]-[7].

Previous works proposed to combine DNN with Hidden Markov Model to get hybrid system defined as "DNN-HMM". DNN is used to estimate the posterior probabilities for senone (context dependent tied state model) in HMM based speech recognition. Consider on input a feature vector x_t of a context dependent window frame and applies nonlinear transformation on it through many hidden layers of DNN. The senone labels are obtained using forced-alignment through a well-trained "GMM-HMM" system. The DNN uses softmax output layer containing a number of classes (nodes) equals to its number of senones [4], [9].

Earlier, Mohamed et al [6] has successfully implemented a pre-trained DNN for phoneme recognition. In his work a context independent "DNN-HMM" model has brought a phone error recognition rate of 22.4% on TIMIT test set. The obtained result displays significantly the power of DNN over GMM.

B. Convolutional Neural Network (CNN)

Very recently, Convolutional Neural Network (CNN) becomes popular hierarchical deep structure aka deep learning for several tasks of speech processing and recognition. CNN has been shown able to produce highly efficient learning parameters. The efficiency of CNN may be attributed to its ability to exploit translational invariance in speech signals. Compared to other deep learning models, CNN can capture easily the environmental and speaker variability in acoustic features [10]-[15],[18]-[19].

Simply, three extra concepts, i.e., local filters, pooling, and weight sharing make CNN powerful over DNN and contribute to its superior performance. A typical architecture of CNN contains usually several convolutional-pooling layers, followed by a certain number of fully-connected layers [10]-[12],[16].

The inputted speech utterances for a Convolutional Neural Network, presented by the feature vectors, must be transformed into feature maps, where each feature map define the values of one feature vector for different locations. Speech recognition systems require features organized along frequency or time (or both). A convolution operation will be then applied on these feature maps in order to get outputs from small local regions, which represent the features of a limited frequency range. The neurons of this convolution layer are organized into feature maps, where the neurons belonging to one feature map will share the same weights, called also filters or kernels [12]-[16].

Considering the input to CNN is defined by $v = [v_1\ v_2\ ...\ v_B] \in \mathbb{R}^{A \times B}$, where A is the number of features corresponding to an input frequency band, B is the number of input frequency bands and v_b is the feature vector corresponding to band b. The activations of the convolutional layer will be calculated as: [12]-[13]

$$h_{k,j} = \sigma(\sum_{b=1}^{s-1} w_{b,j} v_{b+k}^T + a_j) \quad (5)$$

where $h_{k,j}$ is the activation corresponding to the jth feature map of the kth convolution layer band, s is the filter size, $w_{b,j}$ is the weight vector corresponding to the bth band of the jth filter , a_j is the bias representing the jth feature map and $\sigma(x)$ is the activation function .

Local filters and weight-sharing are two interesting concepts making the success of CNN for speech recognition. In fact, different phonemes have different energy concentrations in different local bands along frequency axis. To distinguish different phonemes the small local energy concentrations are processed by a set of local filters in the convolutional layer [12]-[13].

Generally, a pooling layer will be added after the convolution one. A pooling layer is also arranged into feature maps of a number equal to those of the convolution layer, but with smaller maps. In the pooling layer, a sub-sampling will be done on the activations of the convolution layer to obtain new representations with a reduced resolution. Pooling layer leads to more efficient training by reducing the total number of trainable parameters. Different pooling functions may be used as, max-pooling, stochastic-pooling and average-pooling. For a max-pooling function, it provides the maximum value of a feature map along the corresponding frequency bands. The activations of the max-pooling layer can be calculated as [12]:

$$p_{m,j} = \max_{k=1}^{r}(h_{(m-1)\times n+k,j}) \quad (6)$$

where $p_{m,j}$ is the activation corresponding to the jth feature map and the mth pooling layer band, r is the pooling size, and $n \in \{1,...,r\}$ is the sub-sampling factor [12]-[13].

Finally, a certain number of fully connected layers will be added on top of these convolution-pooling layers to achieve the building of CNN [12]-[13].

Convolutional Neural Networks have achieved an impressive performance in phone recognition task. A CNN with limited-weight-sharing scheme and with frequency convolution has been successfully used in the work of Abdel-Hamid et al [12]. The phone recognition accuracy obtained using this hybrid “CNN-HMM” model was interesting. A phone error rate of 20.36% was achieved on TIMIT test set, which is outperforming the performance achieved using a “DNN-HMM” model. Loth [17] has also presented another “CNN-HMM” model with convolution over both time and frequency. Better performance was achieved in this work, the lowest phone error rate achieved on TIMIT test set was of 16.7%.

Recurrent Neural Network (RNN) based architectures

Despite providing interesting performances, DNNs and CNNs are able to model only a limited temporal dependency. Consequently, they are not efficient to model speech, which is inherently a sequential signal. To handle this weakness, Recurrent Neural Networks (RNNs) are used in many speech recognition applications. RNNs are a class of neural networks, which include recurrent connections from the previous time step as inputs. This structure lets them more efficient for sequence modeling than the traditional neural networks. The interest of RNNs lies in their capability to dynamically model the long-term dependencies by using an internal state and updating it at each time step based on its previous state and current input [20],[25]-[26].

A conventional RNN computes a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $y = (y_1, ..., y_T)$ by calculating the sequence of hidden activations vector $h = (h_1, h_2, ..., h_T)$ using the following equations iteratively from $t = 1$ to T : [20]

$$h_t = H(W_{xh} x_t + W_{hh} h_{t-1} + b_h) \quad (7)$$

$$y_t = W_{hy} h_t + b_y \quad (8)$$

where, W terms are the weight matrices, b is the bias vector and $H(.)$ is the recurrent hidden layer activation function.

Unfortunately, RNN training may be complicated due to the classical vanishing and exploding gradients problem. To address properly this problem, previous studies proposed more sophisticated variant of RNN called “gated RNN”. The core idea of this architecture is using a gating mechanism to control efficiently the flow of information through various time-steps. Two of the most successful gated RNN are Long Short-Term Memory (LSTM) and Gated Recuurrent Unit (GRU) models, which often conduct to state-of-the art recognition accuracies for various machine learning tasks [20]-[21],[26].

1) *Long-Short Term Memory (LSTM)*

Among different implementations of gated RNNs, an alternative variant called Long Short Term Memory (LSTM) has been introduced. The LSTM network has the ability to memorize sequences with long range temporal dependencies. In a conventional LSTM network, each hidden layer is composed by a certain number of recurrently connected units called “memory blocks”. Each memory block contains one or more self-connected memory cells for storing the contextual information and three multiplicative gates called input, output and forget gate for controlling the flow of information from previous steps to the current ones. These three gates try to remember when and how much the information in the memory cell should be updated. This gate mechanism makes LSTM architectures well suited for sequence modeling and has improved robustness [22]-[24].

As shown in Fig. 2, given an input sequence $x = (x_1, ..., x_T)$ a conventional LSTM calculates the network unit activations by iterating the following equations from $t = 1 \text{to} T$: [21],[22]

$$g_t = \phi(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \quad (9)$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i) \quad (10)$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_f) \quad (11)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (12)$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + W_{oc}c_t + b_o) \quad (13)$$

$$h_t = o_t \odot \phi(c_t) \quad (14)$$

where W_{ix}, W_{fx}, W_{gx} and W_{ox} are the weights connected to the LSTM inputs, W_{ih}, W_{fh}, W_{oh} and W_{gh} are the weights connected to the LSTM activations, W_{ic}, W_{fc}, W_{oc} are diagonal weight matrices for peephole connections, b terms are the biases , i, f, o and c are respectively the input gate, forget gate, output gate and cell activation vectors, $\odot$ is the element-wise product of vectors, σ is the sigmoid function and ϕ is the hyperbolic tangent function.

In speech, a phoneme is usually influenced by its past and future dependencies due to co-articulation and linguistic tendency of a word. To take into account this phenomenon, a bidirectional variant of LSTM (BLSTM) has been proposed to further ameliorate the recognition accuracy compared to the unidirectional LSTM. The motivation of a bidirectional LSTM is to exploit the bidirectional contextual information (past and future context) to improve predictions. For each depth, a classic BLSTM model has two layers; a forward layer for processing the inputted utterance in the forward direction and a backward layer for processing the inputted utterance in the backward direction. The final output is resulted by concatenating the outputs of forward and backward layers [22]-[24], [27]-[28].

To bring more improvements to these single-layer architectures, either unidirectional or bidirectional, their deep alternatives may be used. Inspired by DNNs, the deep LSTM-RNNs are built by stacking several LSTM layers on top of each other. When input features propagate through the recurrent layers, the output features at each time step incorporate the history of temporal features from previous time steps. Compared to a shallow LSTM, a deep LSTM gives an improved learning and achieves better generalization. Since the inputs are processed with many nonlinear layers, the deep LSTM models are more robust against overfitting [20]-[24].

Similarly, Deep BLSTM (DBLSTM) are able to extract long term high-level representations of historical and future context before aggregating them to capture full range of temporal dependencies. By using more hidden layers, we are aiming to model temporal dependencies at higher timescale [22]-[24],[27].

Several previous works, have applied Long Short Term Memory (LSTM) model for acoustic modeling. An initial work exploiting the use of Long Short Term Memory (LSTM) models for speech recognition was proposed by Graves et al [23]. This work has shown that a bidirectional LSTM (BLSTM) outperforms the unidirectional extension and that the depth (number of layers) is more important than the layer size. On TIMIT test set, a phone error rate of 17.7% was achieved using a deep BLSTM.

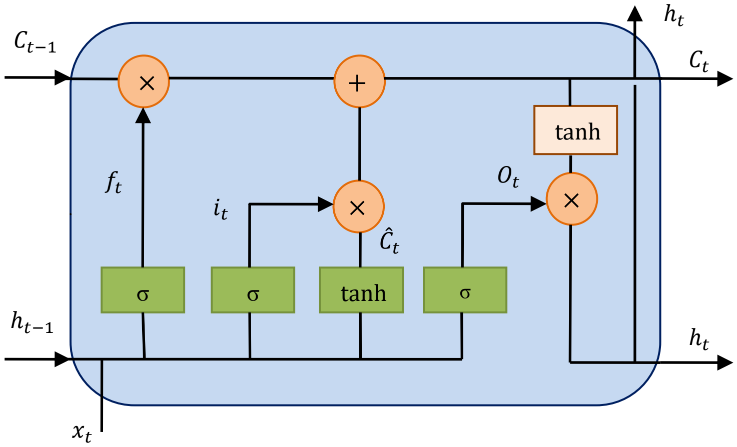


Fig. 2. Diagram for Long Short-Term Memory (LSTM).

2) *Gated Recurrent Unit (GRU)*

Despite the effectiveness of LSTMs, they rely on particular design consisting of a sophisticated gating mechanism that might result in an overly complex model that can be tricky to implement efficiently. To ameliorate the computational efficiency of LSTM some research efforts have proposed a new simplified model called Gated Recurrent Unit (GRU). A GRU is an advanced variant of RNN, which allows solving the gradient-vanishing problem like LSTM, but with a less number of weights. The main reason for the popularity of GRU is the computational cost and simplicity of the model [29].

As shown in Fig. 3, and Compared to LSTM, Gated Recurrent Unit (GRU) is based only on two multiplicative gates; update and reset gates, where the "update gate" is obtained by combining the forget gate and the input gate. The update gate decides how much the units will update their activations [29].

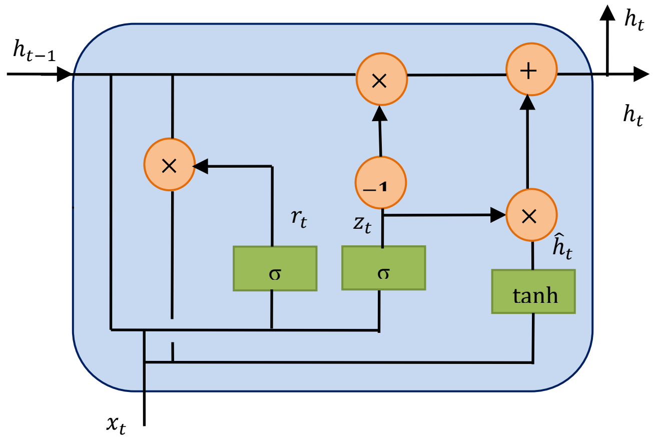


Fig. 3. Diagram for Gated Recurrent Unit (GRU).

In contrast to LSTM, the GRU exposes the whole state at each timestep and computes a linear sum between the existing state and the newly computed state. The GRU reset gate r_t is computed as: [29]

$$r_t = \sigma\ (W_{rx}\ x_t + U_{rh} h_{t-1} + b_r) \qquad (15)$$

where σ is a sigmoid function, x_t and h_{t-1} are the input to GRU and the previous output of GRU. W_{rx} , U_{rh} and b_r are forward matrices, recurrent matrices, and biases for reset gate, respectively.

The update gate z_t controls update value of the activation, defined as:

$$z_t = \sigma\ (W_{zx}\ x_t + U_{zh} h_{t-1} + b_z) \qquad (16)$$

where the parameters are as above.

The candidate activation is defined as:

$$\hat{h}_t = \phi\ (W\ x_t + U\ \ (r_t \circ h_{t-1}) + b\ ) \qquad (17)$$

where ϕ is the hyperbolic tangent function and $\circ$ denotes element-wise multiplication.

The output of the GRU is computed as:

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \hat{h}_t \qquad (18)$$

Using the gating mechanism in both GRU and LSTM architectures has shown strong ability for controlling the flow of information and for creating shortcut paths across many temporal steps. As like the forget gate in LSTM, the update gate in GRU allows capturing long term dependencies. And the reset gate helps GRU to reset whenever the detected feature is not necessary anymore [29].

The principal difference between LSTM and GRU is that there is no output gate in a GRU. Intuitively, coupling the reset gate and the update gate for GRU makes the use of an output gate less valuable and avoids the problem that the output may be unbounded, which may hurts performance significantly. Further, eliminating the output gate in GRU helps to reduce the number of weights compared to LSTM, which makes GRU more robust against overfitting [29].

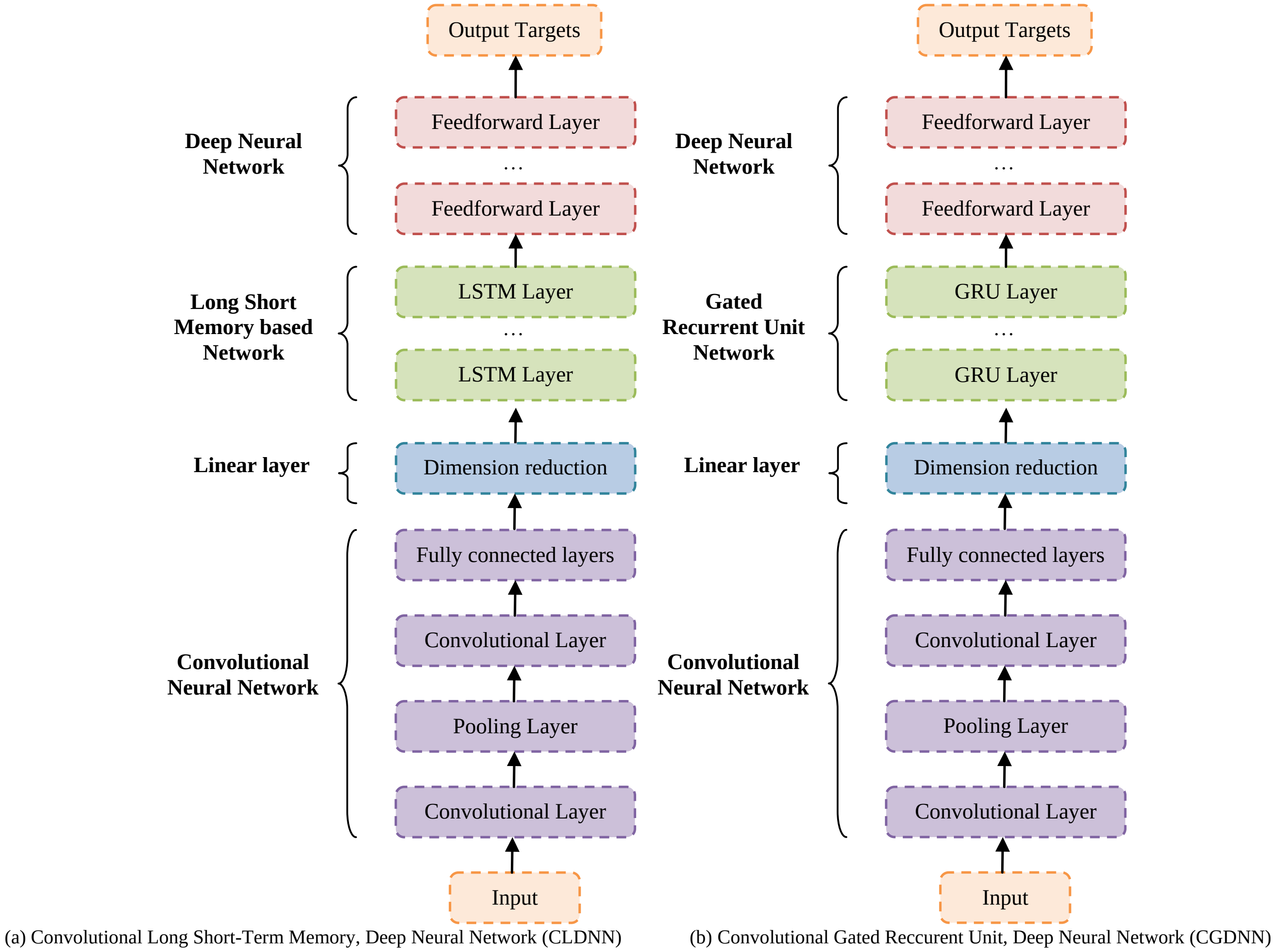
III. Robust Combined Deep Architectures

All deep learning models presented in previous sections have shown promising accuracies for many speech recognition tasks. Nevertheless, they have all some strengths and weaknesses. The individual shortcomings of these different deep learning architectures have motivated many works to combine them in a single architecture to achieve greater performance [30]-[34].

Very recently, a model combining CNN, unidirectional LSTM and DNN called Convolutional Long Short-Term Memory, Deep Neural Network (CLDNN) has been proposed in [31], and achieved greater accuracies over any single model. The CLDNN model has achieved considerable success in a wide range of tasks: speech recognition [31], voice-activity detection (VAD) [32], acoustic scene classification [34].

Later, several works have proposed alternative architectures to achieve additional gains over the CLDNN model. In [33] addressing the task of endpoint detection for streaming speech recognition, the convolution layer in the CLDNN has been replaced with a grid LSTM layer to model both spectral and temporal variations. In [35], the CLDNN model has been extended by introducing a highway connection between LSTM layers.

The focus of this paper, in first step, is to justify the choice of combining CNN, LSTM and DNN into one unified architecture that is trained jointly. Next, similar to previous works, our contribution will be to extend the CLDNN model by replacing the unidirectional LSTM layers with GRU layers (both unidirectional and bidirectional). We refer to this proposed architecture as Convolutional Gated Reccurent Unit, Deep Neural Network (CGDNN) and it's designed by combining this time CNN, GRU and DNN. The idea behind introducing the GRU layers is solving the gradient-vanishing problem like LSTM but with a reduced number of weights, which will make the proposed CGDNN model more robust against overfitting and less sophisticated than the conventional CLDNN model. And motivated by the fact that GRU model brings better recognition accuracies than the LSTM we expect that the proposed CGDNN architecture may show more significant improvements over CLDNN architecture.



(a) Convolutional Long Short-Term Memory, Deep Neural Network (CLDNN)

(b) Convolutional Gated Reccurent Unit, Deep Neural Network (CGDNN)

Fig. 4. An Illustration of the Two Combined Deep Architectures.

Subsequently, our contribution will be to exploit the advantage of a deeper CGDNN structure by increasing the number of GRU layers. In literature, deeper network architectures tend to perform better than shallower models, but the major difficulty with building a very deep structure is the computational cost that has been a shortcoming for the experimentation with more hidden layers or units in deep structures.

In Fig. 4, we illustrate the CLDNN structure experimented in this paper and the proposed CGDNN architecture. These two deep combined architectures are able to achieve further improvements for many tasks of speech recognition. The CLDNN and CGDNN architectures may compensate the problem of spectral variations using a frequency convolution, the problem of long-term temporal dynamics using either LSTM or GRU layers and may reduce the final class discrimination using several DNN layers.

The performing of the CLDNN and the proposed CGDNN architecture may be resumed in three main steps. In first step, the inputted features will be passed into a CNN in order to reduce the spectral variations and hence to address the speaker normalization issues. We used 40-dimensional FBANK features, computed using a 25ms window every 10ms. The CNN model used is only with convolution along frequency and is composed by two convolution layers. A max-pooling layer will be added after the first convolution layer, while no pooling layer is added after the second convolution layer. Next, several fully connected layers will be added; each of them contains 1024 hidden units and with sigmoid activation function. The dimension of the last layer in this CNN architecture is very large; for that a linear layer is added after these CNN layers. This linear layer reduces the number of parameters without deteriorating the recognition accuracy. In second step, the output of this linear layer will be fed into several LSTM layers for the CLDNN model and to several GRU layers for the CGDNN model, which are both efficient for long-term temporal modeling in speech signals.

In last step and after achieving frequency and temporal modeling, the output of the final LSTM or GRU layer for respectively the CLDNN or the proposed CGDNN model will be passed into a DNN containing several fully-connected feed-forward layers. These top DNN layers are promising to get a higher-order feature representation for an easy separation into the different classes that we want to discriminate [31].

The outputs of both CLDNN and CGDNN models are a probability distribution over the possible labels of the central frame. To estimate the phones sequence these probabilities will be divided by the HMM states obtained by the top DNN layer, and will be then passed to a Viterbi decoder.

IV. Analysis and Results

A. Speech Database

The TIMIT data corpus contains 6,300 sentences recorded by 630 speakers of 8 major dialects of American English. After removing all the SA sentences (two sentences recorded by all speakers), we obtain a training set with 3,696 sentences from 462 speakers. A test set containing 192 sentences from 24 speakers. A development (dev) set, composed by a random 10% of the training set, is used for validating our results and adjusting the network parameters.

In all experiments presented in this work, we have used a bigram language obtained from the training set. The training labels are obtained through forced alignment using a well-trained "GMM-HMM" model, with tied context dependent HMM states. We pass the final phoneme label outputs to the usual set of 39 labels. The open speech recognition toolkit Kaldi [36] was used for feature extraction, decoding, and training of the "GMM-HMM" model and all the baselines neural networks exploited in this work.

B. Experimental Results

1) Network architectures

The CNN model is composed by two convolution layers with 128 and 256 filters, respectively and four fully connected layers each of them with 1024 hidden units. A max-pooling layer, with a pooling size of 6 and a sub-sampling factor of 2, is added after the first convolution layer. No pooling layer is added after the second convolution layer.

The stochastic gradient decent (SGD) based back-propagation algorithm is used to train the CNN model. For fine-tuning, we have chosen an initial learning rate of 0.0004 and it will be divided by two for each increasing in cross-validation frame accuracy in a single epoch less than 0.5%.

We have explored the efficiency of four commonly used LSTM and GRU architectures: deep unidirectional LSTM (DLSTM), deep unidirectional GRU (DGRU), deep bidirectional LSTM (DBLSTM) and deep bidirectional GRU (DBGRU). The choice of number of units per LSTM and GRU layers is based on our previous works. In all experiments done along this paper the specification of these architectures is kept as follows:

- Deep LSTM (DLSTM) size of1024 per hidden layer
- Deep GRU (DGRU) size of1024 per hidden layer
- Deep Bidirectional LSTM (DBLSTM) size of 1024 per hidden layer (512 per each forward and backward direction)
- Deep bidirectional GRU (DBGRU)size of 1024 per hidden layer (512 per each forward and backward direction)

TABLE I. Experiments with DNN, CNN and DLSTM models

Method	**PER % (dev core)**	**PER % (test core)**
DNN (6 layers)	20.45	21.18
CNN	17.43	18.83
DLSTM (2 layers)	17.76	18.97

The truncated back-propagation though time (TBPTT) algorithm was used to train the deep unidirectional LSTM and GRU models. Each utterance is divided into short subsequences with a fixed length of T_{bptt} . These subsequences are processed in their original order. For each subsequence, the activations are first calculated and forward-propagated using the LSTM and GRU input and the previous activations, then the cross-entropy gradients are calculated and back-propagated. For efficient computation, N subsequences from different utterances may be operated in parallel by one GPU at a time. After updating the parameters, the GPU continues with the following N subsequences.

To train the deep bidirectional LSTM and GRU models, we used the context sensitive-chunk BPTT (CSC-BPTT) learning algorithm. Firstly, each utterance is divided into chunks of a fixed length N_c . Then N_l previous frames and N_r future frames are concatenated before and after each chunk to give information about, respectively right and left context. Since each trunk can be independently trained, a several number of trunks may be stacked to obtain large minibatches, which leads to faster training.

The DNN model is composed by a few number of fully-connected feed-forward layers, trained in a supervised way. Each layer contains 1024 hidden units and with sigmoid activation function. To train this DNN model, the stochastic gradient decent SGD algorithm is used. The SGD algorithm is using mini-batches of 256 frames. For fine-tuning, we fixed the initial rate to 0.008.

In all experiments, the networks take on input 25 ms frames of 40-dimensional filterbank features (FBANK features), within their first and second temporal derivatives, calculated every 10 ms. The phone error rates (PERs) for the baseline CNN, DNN and DLSTM models are as shown in Table I.

A CNN may bring more improved accuracies than a DNN. The efficiency of CNNs may be attributed to their invariance to small frequency shifts. As consequent, CNNs are more powerful to tolerate speaker variations than DNNs. A DLSTM may lead to phone recognition rates close to those obtained using a CNN. To improve more the performances we must increase the number of LSTM layers.

2) The combined CLDNN and CGDNN architectures

As initial step, we present some experiments to justify the benefit of combining CNN, DLSTM and DNN in a single deep architecture called Convolutional Long Short-Term Memory, Deep Neural Network (CLDNN). First, a deep LSTM (DLSTM) with two LSTM layers is added after the CNN model and we denote this combination as "CNN-DLSTM". Next, this DLSTM model is added after a DNN model with three fully-connected layers and we denote this combination as

"DNN-DLSTM". As shown in Table II, we observe that the power of CNN over DNN still existing even when it is combined with the DLSTM model.

In the following, we show the impact of adding DNN layers after the DLSTM model (composed by 2 LSTM layers). The achieved performances according to the number of DNN layers are so reported in Table III.

Adding several DNN layers after the DLSTM model helps to further improve the recognition accuracies; while adding more than 3 layers will not reduce more the phone error rates. Using DNN layers after achieving the temporal modeling within the DLSTM model helps to map the output of the top LSTM layer to a more discriminative space and to predict easily the targets.

Now, we present the efficiency of the combined deep architecture; namely Convolutional Long Short-Term Memory, Deep Neural Network (CLDNN).The performing of this architecture can be resumed in three steps; in first step the inputted features are passed to a CNN model, in second step the output of the CNN is passed to a DLSM model (composed by 2 LSTM layers) and in last step, the output of the top LSTM layer is passed to 3 DNN layers. Table IV shows the PER for the DLSTM, CNN-DLSTM, DLSTM-DNN and finally the combined CLDNN model.

The CLDNN architecture is very efficient, it may bring up to 0.99% improvement over the DLSTM model used alone for the dev set, and up to 0.87% for the test set.

TABLE II. Phone error recognition rates with CNN- DLSTM vs DNN- DLSTM

Method	PER % (dev core)	PER % (test core)
DNN-DLSTM	20.13	20.82
CNN-DLSTM	17.05	18.41

TABLE III. Phone error recognition rates with DLSTM-DNN

DNN layers	PER % (dev core)	PER % (test core)
0 (DLSTM)	17.76	18.97
1	17.62	18.83
2	17.54	18.71
3	17.48	18.66
4	17.63	18.79

TABLE IV. Phone error recognition rates with CLDNN architecture

Method	PER % (dev core)	PER % (test core)
DLSTM	17.76	18.97
CNN-DLSTM	17.05	18.41
DLSTM-DNN	17.48	18.66
CLDNN	16.77	18.10

TABLE V. Phone error recognition rates with CGDNN architecture

Method	PER % (dev core)	PER % (test core)
DGRU (2 layers)	17.52	18.85
CGDNN	16.55	17.96

Considering the promising phone recognition rates achieved by the CLDNN architecture and motivated by the fact that GRU model may bring better accuracies than the LSTM one we propose another alternative architecture called Convolutional Gated Recurrent Unit, Deep Neural Network (CGDNN) by combining this time CNN, GRU and DNN. Just like the CLDNN architecture, the performing of the CGDNN architecture can be resumed in three steps; in first step the inputted features are passed to a CNN model, in second step the output of the CNN is passed to a deep GRU (DGRU) model and in last step, the output of the top GRU layer is passed to 3 DNN layers. In first step, we used a DGRU model with two GRU layers. The configurations of CNN and DNN models are kept the same as described previously. Table V shows the PER for the DGRU and the proposed CGDNN model.

A deep GRU model (DGRU) is outperforming CNN, DNN and a little more powerful than a DLSTM model. The performances obtained by the proposed CGDNN architecture can bring up to 0.97% improvement in the recognition rates over a DGRU model used alone for the dev set, and up to 0.89% for the test set.

The interesting accuracies obtained by the CLDNN and CGDNN architectures are not surprising because they take advantage from the strong complementarity of the individual modeling capacities of their three deep sub-models, respectively (CNN, DLSTM, DNN) and (CNN, DGRU, DNN).

Using GRU instead of LSTM in the proposed CGDNN architecture has significantly improved the accuracies, while having less number of parameters. The CGDNN architecture can bring up to 0.22% improvement in the recognition rates over the CLDNN architecture for the dev set, and up to 0.14% for the test set.

To show how the depth (number of GRU layers) may affect the overall performance of the proposed CGDNN architecture; a set of experiments is done using respectively 2, 3 and 4 GRU layers. Table VI shows the PER for the proposed CGDNN model according to the number of GRU layers.

TABLE VI. Phone error recognition rates with deeper GRU and CGDNNarchitectures

Method	PER % (dev core)		PER % (test core)	
	DGRU	CGDNN	DGRU	CGDNN
GRU –2 layers	17.52	16.55	18.85	17.96
GRU –3 layers	17.17	16.21	18.49	17.65
GRU –4 layers	16.64	15.77	17.90	17.19

TABLE VII. PHONE ERROR RECOGNITION RATES WITH DEEPER BGRU AND CGDNN ARCHITECTURES

Method	PER % (dev core)		PER % (test core)	
	DBGRU	CGDNN	DBGRU	CGDNN
BGRU –2 layers	16.94	16.03	17.87	17.15
BGRU – 3 layers	16.48	15.67	17.42	16.56
BGRU –4 layers	16.04	15.21	17.10	16.19

The objective of testing different layers was to analyze whether the performance of the proposed CGDNN architecture may be affected by adding the hierarchical depth. It has been shown, that a deeper CGDNN architecture may bring further improvements in the phone recognition accuracies. The lowest error rates are obtained using 4 GRU layers, however increasing the number of GRU layers beyond that makes the training hard and seems to complicate the training without bringing consistent improvements.

The proposed CGDNN architecture using unidirectional GRU layers either shallow or deep has shown very interesting phone recognition rates. In next experiments we propose to use bidirectional GRU (BGRU) model instead of unidirectional GRU model to further improve the performances. A set of experiments with different number of BGRU layers was so done. The achieved performances according to the number of BGRU layers are so reported in Table VII.

The robust CGDNN architecture using Bidirectional GRU (BGRU) layers may bring further improvements over the one using unidirectional GRU layers. This efficiency is not surprising; because the bidirectional GRU (BGRU) layers are able to exploit the bidirectional contextual information (previous and future context), contrariwise to unidirectional GRU layers that can exploit only the past history.

A deeper CGDNN architecture allows an efficient modeling of the long-range history and the non-linear relationship structures. By increasing the number of BGRU layers in the CGDNN architecture the phone error recognition (PER) rates will be further reduced. Nevertheless, adding more than four bidirectional layers will not bring more significant improvements and the performances will be saturated. Theoretically, increasing the number of layers may not harm, while practically that will let the convergence more slow and the network may broke after few epochs.

In last step of our work, we have tested the proposed CGDNN architecture with four BGRU layers by using different type of features. The used features are 39 dimensional MFCC features, 40 dimensional filter-bank (FBANK) features and the LDA+STC+FMLLR features. These later features are obtained by splicing 11 frames (5 on left and right of the current frame) of 13 dimensional MFCCs; then we apply a linear discriminant analysis LDA to reduce the dimension to 40. The MFCCs are normalized with cepstral mean-variance normalization (CMVN). After that, the semi-tied covariance (STC) transform is applied on the previous features. Finally, we apply on these features speaker adaptation using the feature-space maximum likelihood linear regression (FMLLR).

TABLE VIII. PHONE ERROR RATES WITH CGDNN ARCHITECTURE USING DIFFERENT FEATURES TYPES

Features	PER % (dev core)	PER % (test core)
MFCC	15.63	16.58
FBANK	15.21	16.19
FMLLR	14.69	15.72

As shown in Table VIII we notice that using adapted FMLLR features leads to a phone error rate of 15.72% for the TIMIT test set which is the most promising and performing result obtained in this paper. Compared to CLDNN, the proposed CGDNN architecture brings the highest phone recognition rates and achieves more improved performances.

V. CONCLUSION

In this paper, we presented two combined architectures, namely Convolutional Long Short-Term Memory, Deep Neural Network (CLDNN) and Convolutional Gated Reccurent Unit, Deep Neural Network (CGDNN). The first architecture was designed by combining (CNN, LSTM and DNN) and the second architecture was designed by combining (CNN, LSTM and GRU). An overview of the performance gain brought by the deep CGDNN architecture is outlined and compared to all its sub-networks used alone so as to the CLDNN architecture. The proposed CGDNN architecture using deep GRU model (DGRU) achieves a 0.89% relative improvement over the DGRU model used alone and 0.14% over the CLDNN model using a DLSTM, for the TIMIT test set. And a CGDNN architecture using DBGRU model achieves a 0.91% relative improvement over the DBGRU model used alone. A phone error rate of 15.72% has been obtained using the proposed CGDNN architecture with four BGRU layers and using FMLLR features, which has been shown to give state-of-the-art performance for the TIMIT phone recognition task.

From this work we will open several future research issues. The combined CGDNN architecture investigated in this study is found very efficient. Future researches can be conducted by stacking layers with some optimization algorithms to get better performance.

REFERENCES

[1] H. Bourlard, N. Morgan, "Connectionist speech recognition. A hybrid approach", The Kluwer International Series in Engineering and Computer Science, vol.247,1993.

[2] G. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMs," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (2011).

[3] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," Neural Comput., vol. 18, pp. 1527–1554, 2006.

[4] D. Yu, L. Deng, and G. Dahl, "Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition," in NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2010).

[5] A. Mohamed, G. Hinton and G. Penn, "Understanding how deep belief networks perform acoustic modeling," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp.4273-4276 (2012).

[6] A. Mohamed, T. Sainath, G. Dahl, B. Ramabhadran, G. Hinton and M. Picheny, "Deep Belief Networks Using Discriminative Features for Phone Recognition," in Proceedings of the International Conference on

Acoustics, Speech, and Signal Processing (ICASSP), pp. 5060-5063 (2011).

[7] G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton, "Phone recognition with the mean-covariance restricted Boltzmann machine," in Advances in Neural Information Processing Systems.

[8] G. E. Dahl, D. Yu, L. Deng, and Al. Acero,"Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition," in IEEE Transactions on Audio, Speech, and Language Processing, pp. 30-42 (2012).

[9] A. Mohamed, "Deep Neural Network acoustic models for ASR," Ph.D. dissertation, Computer science. Dept., Toronto Univ., Toronto, U.K., 2014.

[10] D. Povey, "Discriminative training for large vocabulary speech recognition," Ph.D. dissertation, Eng. Dept., Cambridge Univ., Cambridge, U.K., 2003.

[11] N. Jaitly, P. Nguyen, AW. Senior, and V. Vanhoucken, "Application of pretrained deep neural networks to large vocabulary speech recognition," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH) (2012).

[12] O. A. Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying Convolutional Neural Network Concepts to Hybrid NN-HMM Model for Speech Recognition," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 4277-4280 (2012).

[13] O. A. Hamid, L. Deng, and D. Yu, "Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH) (2013).

[14] T. N. Sainath, A. Mohamed, B. Kingshury, and B. Ramabhadran, "Deep Convolutional Neural Networks for LVCSR," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 8614-8618 (2013).

[15] D. Palaz, R. Collobert, and M. Magimai.-Doss, "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), pp. 1766-1770 (2013).

[16] D. Palaz, R. Collobert, and M. Magimai. -Doss, "End-to-end Phoneme Sequence Recognition using Convolutional Neural Networks," ArXiv e-prints, Dec. 2013.

[17] L. Tôth. "Combining time and frequency domain convolution in convolutional neural network-based phone recognition," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 190-194 (2014).

[18] S. Basalamah, S.D. Khan, H.Ullah, "Scale Driven Convolutional Neural Network Model For People Counting and Localization in Crowd Scenes," in IEEE access,2019.

[19] F. Saeed, A. Paul, P. Karthigaikumar and A. Nayyar, "Convolutional neural network based early fire detection," in Multimedia Tools and Applications, 1-17, 2019.

[20] D. Palaz, R. Collobert, and M. Magimai. -Doss, "End-to-end Phoneme Sequence Recognition using Convolutional Neural Networks," ArXiv e-prints, Dec. 2013.

[21] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH) (2014).

[22] H. Sak, O. Vinyals, G. Heigold, A. Senior, E. McDermott, R. Monga, and M. Mao, "Sequence discriminative distributed training of long short-term memory recurrent neural networks," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (2014).

[23] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks, vol. 12, pp. 5–6, 2005.

[24] F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context free and context sensitive languages", in IEEE Transactions on Neural Networks, vol. 12, no. 6, pp. 1333–1340, 2001.

[25] D.Bahdanau, J.Chorowski, D.Serdyuk, P.Brakel and Y.Bengio "End-to-end attention-based large vocabulary speech recognition", in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 4945-4949 (2016).

[26] A.Graves, A. Mohammed and G. Hinton. "Speech recognition with deep recurrent neural networks," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 6645-6649 (2013).

[27] M.Ullah, H.Ullah, S.D. Khan and F.A. Cheikh, "Stacked Lstm Network for Human Activity Recognition Using Smartphone Data," in IEEE, EUVIP 2019.

[28] A. Kumar, S.R. Sangwan, A. Arora, A. Nayyar and M. Abdel-Basset, "Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network," in IEEE Access, 7, 23319-23328, 2019.

[29] J. Chung, C.Gulçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in Proceeding of NIPS, 2014.

[30] L. Deng and J. Platt, "Ensemble Deep Learning for Speech Recognition," in Proceedings of the 15th Annual Conference of International Speech Communication Association (INTERSPEECH), pp. 1915-1919 (2014).

[31] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 4580-4584 (2015).

[32] R. Zazo, T. N. Sainath, G. Simko and C. Parada, "Feature learning with raw-waveform CLDNNs for Voice Activity Detection," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), pp. 3668-3672 (2016).

[33] S.Y. Chang, B. Li, T.N . Sainath, G. Simko and C. Parada, "Endpoint Detection using Grid Long Short-Term Memory Networks for Streaming Speech Recognition," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), (2017).

[34] J. Guo, N. Xu, L.J. Li, A. Alwan, "Attention based CLDNNs for short-duration acoustic scene classification," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), (2017).

[35] W. Hsu, Y. Zhang, A. Lee, and J. Glass, "Exploiting depth and highway connections in convolutional recurrent deep neural networks for speech recognition," in Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH), (2016).

[36] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit", in IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society, 2011.