Vishal Passricha and Rajesh Kumar Aggarwal*

A Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition

https://doi.org/10.1515/jisys-2018-0372
Received September 10, 2018; previously published online March 5, 2019.

Abstract: Deep neural networks (DNNs) have been playing a significant role in acoustic modeling. Convolutional neural networks (CNNs) are the advanced version of DNNs that achieve 4–12% relative gain in the word error rate (WER) over DNNs. Existence of spectral variations and local correlations in speech signal makes CNNs more capable of speech recognition. Recently, it has been demonstrated that bidirectional long short-term memory (BLSTM) produces higher recognition rate in acoustic modeling because they are adequate to reinforce higher-level representations of acoustic data. Spatial and temporal properties of the speech signal are essential for high recognition rate, so the concept of combining two different networks came into mind. In this paper, a hybrid architecture of CNN-BLSTM is proposed to appropriately use these properties and to improve the continuous speech recognition task. Further, we explore different methods like weight sharing, the appropriate number of hidden units, and ideal pooling strategy for CNN to achieve a high recognition rate. Specifically, the focus is also on how many BLSTM layers are effective. This paper also attempts to overcome another shortcoming of CNN, i.e. speaker-adapted features, which are not possible to be directly modeled in CNN. Next, various non-linearities with or without dropout are analyzed for speech tasks. Experiments indicate that proposed hybrid architecture with speaker-adapted features and maxout non-linearity with dropout idea shows 5.8% and 10% relative decrease in WER over the CNN and DNN systems, respectively.

Keywords: ASR, CNN, CNN-BLSTM, DNN, LSTM-RNN.

1 Introduction

Deep neural network (DNN)-based acoustic models have almost displaced Gaussian mixture models (GMMs) from automatic speech recognition (ASR) systems [18]. Due to fully connected nature of DNN, structural locality from the feature space is not grabbed, which is a fundamental drawback. Second, DNN faces vanishing gradient problem at the time of stochastic gradient descent (SGD) training [11]. Although convolutional neural networks (CNNs) successfully model the structural locality from the feature space [24]. They also reduce the translational variance and take care of disturbances and small shifts in the feature space because they adopt the pooling at a local frequency region. They can utilize the long-time dependencies among the speech frames by exploiting prior knowledge of speech signal. However, Soltau et al. [35] claimed that CNNs do not take stress for semi-clean data in the ASR systems; as a result the performance of the system deteriorates. On the other side, recurrent neural networks (RNNs) offer higher recognition accuracy by capturing long contexts especially for noise robust tasks [27]. However, vanishing and exploding gradient problem bound the capability of RNNs to learn time dependencies [3]. To tackle these problems, long short-term memory (LSTM) was introduced that controls the flow of information by a special unit called memory block [31]. LSTM-RNNs are sensitive to the static data, so target delays with respect to features arise. Low latency between inputs and corresponding outputs is the preferred choice for acoustic modeling. Therefore, a special architecture that operates the input sequence in both directions to make decisions came into existence and is called bidirectional LSTM (BLSTM) [33].

***Corresponding author: Rajesh Kumar Aggarwal,** Computer Engineering Department, National Institute of Technology Kurukshetra, Haryana, India, e-mail: rka15969@gmail.com
Vishal Passricha: Computer Engineering Department, National Institute of Technology Kurukshetra, Haryana, India

Open Access. © 2020 Walter de Gruyter GmbH, Berlin/Boston. This work is licensed under the Creative Commons Attribution 4.0 Public License.

In this paper, we aim to design a deep hybrid structure for acoustic modeling that resolves the problem of vanishing gradient and makes use of prior knowledge. After analyzing the input feature set, various non-linearities, and CNN architectures, a hybrid of CNN and BLSTM structure is proposed as a solution in which CNN controls the translational variance and BLSTM resolves the vanishing gradient problem. The information from speech frames is captured by CNN, and BLSTM layers process this input in both directions. Abdel-Hamid et al. [1] used a single limited weight sharing (LWS) layer for CNN in speech. The benefit of LWS is that it allows each local weight to focus on the most confusable parts of the signal. Sainath et al. [29] investigated multiple full weight sharing (FWS) convolutional layers and argued that these layers are more beneficial as compared to a single LWS convolutional layer. In this paper, the analysis of multiple layers of LWS and FWS is done. Various pooling strategies like L_p pooling [34] and stochastic pooling [44] are offering better generalization in vision task than max pooling. Hence, the effectiveness of these pooling methods is explored for continuous speech recognition tasks. Locality in frequency and time must be present in the features for CNN. Feature space maximum likelihood linear regression (fMLLR) features [10] improve the performance for DNNs. Sainath et al. [29] made an attempt to apply fMLLR transformation directly on log-mel features. In their work, log-mel features were transformed into an uncorrelated space, and fMLLR transformation was applied to them then these new features were again transformed into correlated space. Unfortunately, no improvement was observed in recognition rate. Therefore, a new methodology in which fMLLR features are directly fed to a fully connected (FC) layer is introduced. In the next FC layer, CNN-BLSTM features are joined with fMLLR features as in [34]. This method is found more fruitful as compared to creating fMLLR transformed log-mel features. Hessian-free (HF) sequence training [21] offers 10–15% relative gain over cross-entropy (CE) trained DNN. Therefore, various non-linearities and dropout are investigated for HF sequence training.

The performance of hybrid architecture is explored for continuous speech recognition task on a Hindi dataset. After experiments, it is observed that LWS and FWS are almost the same with multiple layers for continuous speech recognition. It is also noted that there is not much difference in various pooling strategies. Although the max pooling is found best, fMLLR features improve the input feature set, and as a result, the word error rate (WER) reduces 1.1% relatively. Dropout + HF training also improves the recognition rate. A relative improvement of 5.8% and 10% is obtained over best-performing CNN and DNN, respectively.

The subsequent sections of the paper are organized as follows. In Section 2, the basic CNN, LSTM-RNN, and BLSTM acoustic models are given. Section 3 provides details about the proposed architecture and its experimental setup. In Section 4, an exploration of various pooling strategies is presented. Analysis of input features is discussed in Section 5. In Section 6, the various non-linear functions and dropout are explained. In Section 7, experimental results are compared with other existing and competitive techniques. Finally, Section 8 presents the conclusion of the paper.

2 Baseline CNN and LSTM Framework

This section introduces the CNN, LSTM-RNN, and BLSTM acoustic models that are employed as baseline systems.

2.1 CNN Acoustic Model

The main aim of CNN, the advanced version of DNN is to discover local structure in input data. CNN successfully reduces the spectral variations and adequately models the spectral correlations in acoustic features. Earlier, the convolution was applied only on frequency axis which made the ASR system more robust against variations generated from different speaking styles and speakers [1, 9, 29]. Later, Toth [37] and Waibel et al. [41] applied convolution along the time axis. In many computer vision tasks, researchers applied convolution in both frequency and time [22, 34]. CNN introduces three new concepts over the DNN: local filters, weight sharing, and pooling. Local filters are imposed on a subset region of the previous layer to capture a specific kind of local patterns known as structural locality. Stride of local filters is the necessary element of CNN that reduces the temporal resolution. Weight sharing is used with local filters to reduce translational variance.

To utilize the characteristics of different spectral bands, LWS and FWS are convolutional weights that are applied to convolutional layers [2]. They combine the information received from the neurons and assign the collected information to various spectral bands. Abdel-Hamid et al. [1] favor LWS by presenting different spectral regions for different spectral phenomena. On the other hand, Sainath et al. [28] favor FWS by showing the same performance as LWS. FWS applies all the neurons across all spectral regions which is much simpler. Pooling layer regularly follows the convolutional layer. Pooling provides additional translational and rotational invariances. It also provides a reduction in dimensionality of feature map and parameters [19]. CNN-based acoustic models have performed better than state-of-the-art DNN systems in ASR by selecting more adequate features for different classes like speaker, gender, and phone [1, 2, 28, 29, 38]. Toth [39] argues that CNN improves the mean of the per-speaker recognition rate and reduces their variance by ≈5.7% when compared to DNN.

2.2 LSTM-RNN Acoustic Model

LSTM-RNN introduces the idea of gate functions which determine the value of the neuron's activation to transform or just pass through [42]. This gating function forces the layer's inputs and outputs to be of the same size. LSTM-RNN successfully models the long-time dependencies. It models the input sequence unidirectionally and hence cannot retain structural locality, and it is more prone to overfitting. It has hidden state $\overrightarrow{h}$ in forward direction that processes the input from left to right by using the left context of the current input. Deep LSTM-RNNs are made by arraying LSTM-RNN layers. The benefit of deep LSTM-RNNs over conventional LSTM-RNNs is that it optimally uses its parameters by distributing them over the space through multiple layers. Deep LSTM-RNNs have given good results in large vocabulary speech recognition tasks [15, 31].

2.3 Bidirectional LSTM

LSTM is unidirectional; hence, it accesses a small amount of right context which deteriorates the recognition rate. The way to improve recognition rate without increasing latency is to process the input bidirectionally. BLSTM processes the input in both directions forward as well as backward by the use of separate layers. It has hidden states with forward sequence $\overrightarrow{h}$ for the left context and backward sequence $\overleftarrow{h}$ for the right context. It should be ensured that every next layer must be fed from both forward as well as backward layer. Graves et al. used deep BLSTM to perform framewise phoneme classification [13] and hybrid speech recognition [14] on the TIMIT database.

3 Hybrid CNN-BLSTM Architecture

In this section, hybrid CNN-BLSTM architecture and its experimental setup are explained.

3.1 Proposed Architecture

The proposed architecture incorporates three different models like CNN, BLSTM, and fully connected layers, as shown in Figure 1. Initially, a few convolutional layers are located to reduce the frequency variance present in input signal. Initially, two convolutional layers having 256 feature maps in each CNN layer are taken. The reason behind this is that the feature dimension for speech is small (i.e. 40). High- and low-frequency regions show quite different behaviors. After processing by two convolutional layers, the feature map is reduced into much smaller size near 16. Therefore, there is no further need to model locality and remove invariance. Sainath et al. [30] argue that 9×9 frequency-time filter for first convolutional layer and 4×3 frequency-time filter for second convolutional layer cover the entire frequency-time space. Therefore, the first convolution layer uses 9×9 frequency-time filter, and the second layer uses 4×3 frequency-time

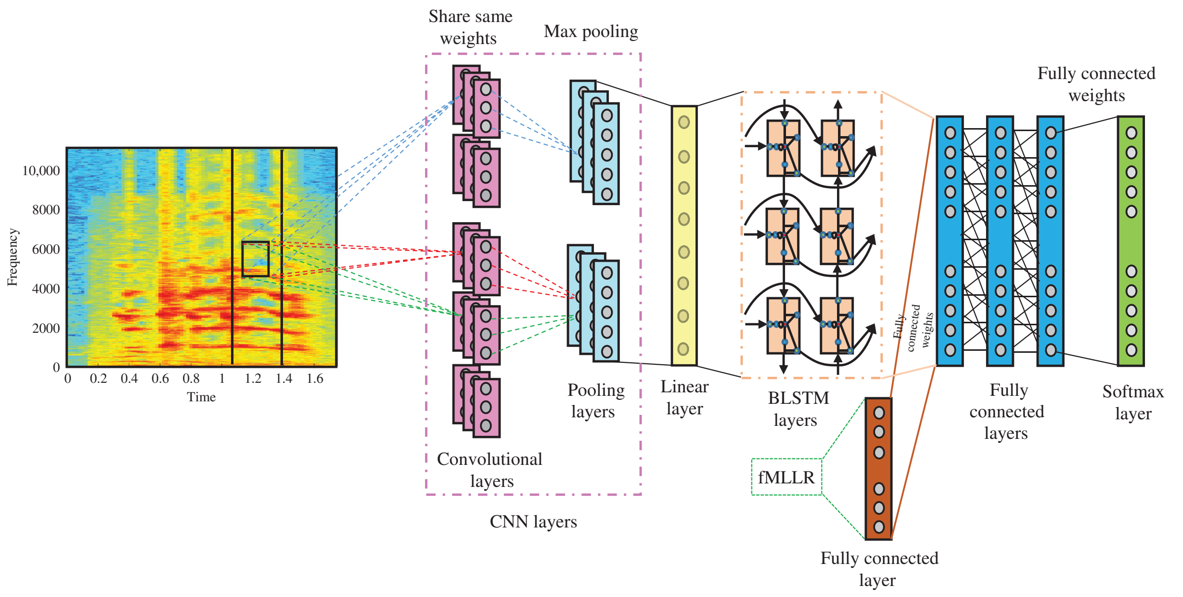

Figure 1: An Illustration of the CNN-BLSTM Architecture Applied to Log-Mel FB Features of the Speech Signal.

filter. Initially, max pooling is used in our model, and pooling is applied only in the frequency domain. For both layers, the pooling size used is 2, and the value of stride is also 2.

In CNN, the dimension of the layer is a function of the number of feature maps $\times$ time $\times$ frequency, resulting in the larger dimension of the next layer. Thus, there is a need to reduce the feature dimensions. After CNN layers, a linear layer is applied for a reduction in dimension without any loss in accuracy, as shown in Figure 1. Two hundred fifty-six outputs from linear layer are appropriate so the dimensionality is reduced in a similar fashion, and this process is called frequency modeling.

After frequency modeling, CNNs output is passed to BLSTM layer which is well-suited for modeling the signal in time. Here, two BLSTM and three FC layers are preferred, but according to the nature of the experiment, the number of layers can vary. Each BLSTM layer contains 832 cells and 512 units (256 LSTM units per direction) projection layer for dimensionality reduction. The BLSTM is pre-trained for 20 time steps with truncated backpropagation through time.

After frequency and temporal modeling, the output of BLSTM layers is passed to FC layers. These layers are suitable for generating higher-order feature representation that is easily discriminable into the different classes. All fully connected layers contain 1024 hidden units. Different speakers have different accent, loudness, etc., that create variability in speech. This variability can be reduced by a popular speaker adaptation technique, fMLLR [10]. CNN cannot be directly used for modeling fMLLR features. Earlier, fMLLR transformation was applied to log-mel features [29]. Unfortunately, no improvement was shown by this method. In our work, a new methodology is introduced to integrate fMLLR features with log-mel features effectively. Specifically, CNN layer is fed by log-mel filter bank (FB) $+ \Delta + \Delta\Delta$ features, and FC layer is fed by fMLLR features. The output of both fMLLR FC layer and log-mel FB CNN-BLSTM layers are combined in the next FC layer in the same way as given in [34]. Most CNN work makes use of FC layers to do class discrimination based on local information learned. Our framework is a flexible joint architecture in which CNN-BLSTM module is used for frequency and temporal modeling, DNN module is used for joining fMLLR and CNN-BLSTM features, and class discrimination is done by the topmost layer, i.e. softmax layer. The complete model is trained jointly.

3.2 Experimental Setup and Hardware Configuration

The various experiments on CNN-BLSTM architecture are carried out on a medium size corpus containing 200 K Hindi spoken utterances (around 150 h). All the models are fed with 40-dimensional log-mel FB features. These features are computed at an interval of 10 ms. Asynchronous stochastic gradient descent (ASGD)

[8] is a better choice for optimizing the network. It works with a constant learning rate $= 10^{-5}$ to pre-train all the neural networks with the CE criteria. The distributed ASGD [17] is also used to perform the HF sequence training experiments. The Glorot-Bengio strategy [11] is used to initialize the weights. All the layers in BLSTM are initialized to be Gaussian, with a variance of 1/(number of inputs). Moreover, the learning rate is explicitly selected for each network, and for stable training, the chosen value is the largest among others. Learning rates are exponentially decayed.

The proposed model is tested on a supercomputer named PARAM Shavak. It consists of two multicore CPUs having 18 cores each along with two accelerator cards (i.e. GPGPU). The computer has 64 GB RAM, 8 TB storage, Nvidia Pascal architecture based co-processing technologies, and deep learning GPU software environment. It works under Ubuntu v17.04 operating system. Kaldi toolkit is used for implementation of the proposed model with Python.

4 Analysis of Various Strategies

In this section, various experiments are performed on the proposed architecture to check its performance. The experiments given in Sections 4.1, 4.2, 4.3, and 4.4 are carried on CNN layers, and in Section 4.5, we check different combinations of BLSTM layers and FC layers. The experimental setup used in Sections 4.1, 4.2, 4.4, and 4.5 is described in Section 3.2, and in Section 4.3, max pooling strategy is changed with other pooling strategies.

4.1 Limited vs. Full Weight Sharing

Some advantages of weight sharing are the following: it reduces the model complexity and makes the network easier to train. In computer vision, the results are good for LWS as compared to FWS [23]. The properties of speech signal change on different frequency bands. Separate sets of weights for different frequency bands may be more fruitful because it allows the detection of distinct feature patterns in different filter bands along the frequency axis. Therefore, LWS scheme looks more appropriate for CNN acoustic models. In ASR task, most LWS work has been performed on a single layer using eight-band different LWS filter [1]. However, our architecture uses multiple convolutional layers because it has been proved that multiple convolutional layers enhance the performance of the system [29]. Therefore, multiple LWS convolutional layers with two LWS filters are used. Local activities of the LWS filter preserve the local information (i.e. locality), and the next LWS filter is fed by a lower LWS filter. Abdel-Hamid et al. [1] reported that an increase in the number of LWS filters and moving to a single layer would deteriorate the performance further. Therefore, the use of multiple FWS convolution layers is logical. The idea of FWS is implemented in the same way as in image recognition. Sainath et al. [30] also showed in their experiments that adding additional convolutional layers was beneficial and claimed that the differences between low- and high-frequency components are expertly captured by using FWS in the convolutional layers.

Table 1 shows the result of LWS and FWS for different setups. Two LWS filters are used in the experiment and compared with FWS in performance and number of parameters. The relative improvement in WER

Table 1: WER as a Function of Limited Weight vs. Full Weight Sharing.

Method	Number of hidden units in first/second convolutional layers	Parameters	WER (%) on training set	WER (%) testing set
FWS	256/256	6.9 M	18.4	20.2
LWS	256/256	7.9 M	18.3	20.1
FWS	384/384	8.9 M	18.2	20.1
LWS	384/384	10.1 M	18.1	20
FWS	512/512	11.3 M	17.9	19.8
LWS	512/512	12.7 M	17.7	19.7

(≈0.7%) seems negligible when comparing the computational cost. The performances offered by LWS with two filters and by FWS are almost the same. The implementation of FWS is easier, so we do not prefer filter locations for each limited weight ahead of time. FWS is adopted over LWS because FWS produces the best trade-off between WER and number of parameters with 6.9 M parameters (256/256 hidden units per convolutional layer). This setting is applied to subsequent experiments.

4.2 Number of Hidden Units

The local behavior of speech features varies differently in high-frequency regions when compared to the features in low-frequency regions [1]. This issue is addressed by applying weight sharing to frequency components. In simple words, different weights are used for different frequency components. Weight sharing can be applied across all the time and frequency components, although the differences between different frequency regions can also be captured with a large number of hidden units by performing FWS.

Table 2 shows the effect of different number of hidden units for two FWS convolutional layers on WER. The result confirms that as the number of hidden units increases, the WER steadily decreases. The reason of this gain is that increasing the hidden units with FWS excellently deals with the variations in frequency in the input signal.

4.3 Type of Pooling (Max, Stochastic, L_p)

Pooling is an important concept that transforms the joint feature representation into valuable information by keeping the useful information and eliminating irrelevant information. Small frequency shifts that are common in speech signal are efficiently handled using pooling. Pooling also helps in reducing the spectral variance present in the input speech. It maps the input from p adjacent units into the output by applying a special function. After the element-wise non-linearities, the features are passed through the pooling layer. This layer down-samples the feature maps coming from the previous layer and produces the new feature maps with a condensed resolution. This layer drastically reduces the spatial dimension of input. It serves the two main purposes. First, it reduces the amount of parameters or weights up to 65%, thus lessening the computational cost. Second, it controls the overfitting of the training data. Overfitting refers to when a model is so tuned to the training examples.

In this section, various pooling strategies that have been successfully applied in computer vision tasks are investigated for speech recognition tasks. A local region of the previous convolutional layer feeds the inputs to the pooling layer that down-samples the inputs to fetch a single output from that region. Max pooling is the most popular strategy for CNNs [1]. It selects the maximum value from the pooling region. The function for max pooling is given in Eq. (1).

$$s_j = \max_{i \in R_j} a_i \tag{1}$$

where R_j is a pooling region and $\{a_1, \ldots, a_{|R_j|}\}$ is a set of activations. Zeiler and Fergus [44] have shown in their experiments that overfitting of the training data is a major problem with max pooling. L_p pooling [34] and stochastic pooling [44] are alternate pooling strategies that address this problem. Bruna et al. [4] claim that the generalization offered by L_p pooling is better than max pooling.

Table 2: WER as a Function of Number of Hidden Units.

Number of hidden units in first/second convolutional layers	Parameters	WER (%) on training set	WER (%) on testing set
256/256	6.9 M	18.4	20.2
384/384	8.9 M	18.2	20.1
512/512	11.3 M	17.9	19.8

In L_p pooling, a weighted average of activations is taken in the pooling region. Equation (2) shows the operation for L_p pooling.

$$s_j = \left(\sum_{i \in R_j} {a_i}^p \right)^{1/p} \tag{2}$$

If $p = 1$ then it works as an average pooling, while $p = \infty$ leads to max pooling. The areas of high activation are downweighted by areas of low activation in average pooling because all elements in the pooling region are examined and their average is taken. It is the major problem with average pooling. The issues of max and average pooling are addressed using another pooling strategy known as stochastic pooling. Stochastic pooling first computes the probabilities p for each region R_j by normalizing the activations within the region, as given in Eq. (3).

$$\begin{aligned} p_i &= a_i \Big/ \sum_{k \in R_j} a_k \\ s_j &= a_l \quad \text{where} \ \ l \sim P(p_1, p_2, \ldots, p_{|R_j|}) \end{aligned} \tag{3}$$

These probabilities create a multinomial distribution that is used to select location l and corresponding pooled activation a_l. In simple words, the activations are selected randomly based on the probabilities calculated by multinomial distribution. Stochastic pooling prohibits overfitting because of its stochastic component. The advantages of max pooling are also available in stochastic pooling.

For L_p pooling, $p > 1$ is examined as a trade-off between average and max pooling. Both L_p and stochastic pooling strategies are compared to max pooling on the Hindi dataset. For both convolutional layers, the pooling size of 2 (i.e. $p = 2$) is used for all the three pooling strategies, and the stride value is also taken to be 2. Their results are shown in Table 3.

Max pooling shows improvements over stochastic and L_p pooling. However, these gains are not significant on speech tasks. The experiment concludes that the pooling strategies like L_p and stochastic pooling are not offering any improvements over max pooling for speech tasks.

4.4 Pooling in Both Frequency and Time

In speech recognition task, CNNs with pooling in frequency is generally investigated [1, 9, 29]. Although Toth [37] and Waibel et al. [41] explored CNNs for pooling in time only, in computer vision, pooling is generally applied in both frequency and time [22, 34]. We also analyze the pooling in both frequency and time for our architecture. Table 4 shows the result for pooling in both frequency and time for various pooling strategies. Pooling in both frequency and time slightly improves the stochastic pooling, but these gains are limited. In

Table 3: WER as a Function of Different Pooling Types.

Pooling method	WER (%) on training set	WER (%) on testing set
Max pooling	18.3	20.1
Stochastic pooling	18.4	20.3
L_p pooling	18.4	20.3

Table 4: WER as a Function of Pooling in Frequency and Time.

Pooling method	WER (%) on training set	WER (%) on testing set
Max pooling	18.2	20.0
Stochastic pooling	18.3	20.2
L_p pooling	18.4	20.3

Table 5: WER as a Function of the Number of BLSTM Layers and Fully Connected Layers.

Number of BLSTM vs. fully connected layers	WER (%) on training set	WER (%) on testing set
1 BLSTM, 4 fully connected layers	18.5	20.4
1 BLSTM, 3 fully connected layers	18.6	20.4
2 BLSTM, 3 fully connected layers	18.4	20.2
2 BLSTM, 2 fully connected layers	18.5	20.3
3 BLSTM, 3 fully connected layers	18.0	19.9
3 BLSTM, 2 fully connected layers	18.3	20.2
4 BLSTM, 2 fully connected layers	18.4	20.3

the analysis, it is clear that if pooling in time has the overlap between pooling windows, then it keeps the same performance. If pooling in time is applied without overlapped pooling windows, then it is seen as subsampling of the signal in time. Therefore, the overlapped pooling window is only a way to smooth out the signal in time. It is also observed that pooling in frequency and time is slightly better than pooling in frequency, but this gain will be diminished for large tasks.

4.5 Number of BLSTM and Fully Connected Layers

The proposed architecture is deep in both domains, i.e. time and spatial. Different structures may impact the performance of the proposed architecture. For example, the capability of the architecture to learn temporal relationship may be affected by BLSTM layers, and the capability of the architecture to learn feature transforms may be affected by the number of FC layers.

To get optimal performance of the architecture, seven different structures are designed by varying the number of BLSTM layers and FC layers. The non-linearity activation for the FC layer is maxout. The WER is a function of the number of BLSTM layers and FC layers, as shown in Table 5. The results show that increasing the number of BLSTM layers up to 3 improves the performance after that performance starts to deteriorate. At the same time, it is also observed that BLSTM layers offer improvements over FC layers for the same input feature set.

5 Analysis of Input Features

The features required for CNNs are locally correlated in time and frequency. Mel-frequency cepstral coefficients [7] features are generally the preferred choice in speech recognition, but they do not have locality in frequency and hence cannot be used with CNN [1]. This property, i.e. locality in frequency, is offered by mel FB features [26]. Various researchers [10, 25, 40] have proposed many speaker adaptation techniques that are adapted to improve ASR systems performance, and the results have shown marvelous improvements in recognition rate.

Combining extra time dynamic information of features is a common method to improve the performance. By combining time-derivative information of features called delta (Δ) and double-delta ($\Delta\Delta$), the performance of the system is also increased [43]. The experiments discussed in Section 4 were performed with log-mel + $\Delta + \Delta\Delta$ features. Another common method, vocal tract length normalization (VTLN), normalizes the speech differences like difference of speaker, accents, stress, etc., by averaging vocal tract length [40]. It keeps the locality in frequency by applying the warping to the log-mel FB of each speaker; hence, it looks more fruitful. In this section, the performance of complex features is analyzed.

The improvements in the recognition rate by including VTLN and delta information is presented in Table 6. Note that the same number of parameters is used for each input feature. The performance of different transformations of log-mel features is not tested for DNN because the performance of fMLLR has been observed to be better than VTLN-warped log-mel features [10]. Sainath et al. [29] applied fMLLR features to CNN by transforming the VTLN-warped log-mel features. This method has not shown any improvement in

Table 6: WER as a Function of Input Feature.

Feature	WER (%) on training set	WER (%) on testing set
Log-mel FB	19.2	21.1
fMLLR transformed Log-mel FB	19.1	21.0
VTLN-warped log-mel FB	19.0	21
fMLLR transformed VTLN-warped log-mel FB	18.9	20.8
fMLLR transformed Log-mel FB + Δ + $\Delta\Delta$	18.6	20.6
Log-mel FB + Δ + $\Delta\Delta$ + fMLLR(DNN)	18.4	20.3
VTLN-warped log-mel FB + fMLLR(DNN)	18.7	20.6
VTLN-warped mel FB + Δ + $\Delta\Delta$	18.3	20.1
VTLN-warped mel FB + Δ + $\Delta\Delta$ + fMLLR(DNN)	18.1	19.9

performance. Multi-scale neural networks [34] combine the input from different layers to improve the results. The same idea is applied to combine fMLLR features with CNN-BLSTM features in FC layers. The output of the fMLLR FC layer is fed into the next FC layer where it combines with CNN-BLSTM features. By combining fMLLR features and CNN-BLSTM features, 1.1% relative gain is achieved. In the analysis, it is also observed that combining fMLLR features through the FC layer is more effective than fMLLR transformed log-mel features.

6 Non-Linear Units and Dropout

In this section, the performance of different non-linear functions, i.e. sigmoid, maxout, rectified linear units (ReLU), and parameterized rectified linear unit (PReLU), is evaluated by varying the nature of fully connected layers with and without using dropout networks on the Hindi dataset.

6.1 Sigmoid Neurons

For acoustic modeling, the standard sigmoid is the preferred choice. Fixed function shapes and no adaptive parameters are its strengths. The sigmoid family's function has been widely explored for many ASR tasks.

$$f(\alpha) = \eta . \frac{1}{1 + e^{-\gamma\alpha+\theta}} \tag{4}$$

$f(\alpha)$ is the logistic function, and θ, γ, and η are known as the learnable parameters. Equation (4) denotes the p-sigmoid(η, γ, θ).

In the p-sigmoid function, the curve $f(\alpha)$ has different effects of η, γ, and θ. Among the three parameters, the curve $f(\alpha)$ is highly changed by η because it scales linearly. The value of $f(\alpha)$, i.e. $|f(\alpha)|$ is always less than or equal to $|\eta|$. η can be any real number. If $\eta < 0$, the hidden unit makes a negative contribution, if $\eta = 0$, the hidden unit is disabled, and if $\eta > 0$, the hidden unit makes a positive contribution that can be seen as a case of learning hidden unit contributions [32, 40] and outputs are constant values. If $\gamma \longrightarrow 0$, then $f(\alpha)$ is similar to input around 0. The horizontal difference to $f(\alpha)$ is managed using parameter θ. If $\gamma \neq 0$, then θ/γ is the x value of the mid-point.

6.2 Maxout Neurons

Maxout neurons came as a good substitute of the sigmoid neurons. The main issue with the use of conventional sigmoid neurons is vanishing gradient problem during SGD training. Maxout neurons effectively resolve this issue by producing a constant gradient. Cai et al. [5] achieved 1–5% relative gain using the maxout neurons instead of ReLU on the switchboard dataset. Each maxout neuron gets input from several pieces of alternative activations. The maximum value among its piece group is taken as the output of a maxout neuron as given in Eq. (5).

$$h_l{}^i = \max_{j\in 1,\ldots,k} z_l{}^{ij} \tag{5}$$

where ${h_l}^i$ represents *i*th maxout neuron output in the *l*th layer. *k* represents the number of activation inputs for the maxout neurons. ${z_l}^{ij}$ represents the *j*th input activation of the *i*th neuron in the *l*th layer as given in Eq. (6).

$$\boldsymbol{z}_l = {\boldsymbol{W}_l}^T \boldsymbol{h}_{l-1} + \boldsymbol{b}_l \tag{6}$$

where ${\boldsymbol{W}_l}^T$ represents the transpose of the weight matrix for layer *l*, $\boldsymbol{h}_{l-1}$ represents previous hidden layers output, $\boldsymbol{b}_l$ is a bias vector for layer *l*, and $\boldsymbol{z}_l$ is piece activations (or input activations).

The computation of $\boldsymbol{z}_l$ or $\boldsymbol{h}_l$ does not include any nonlinear transforms like sigmoid or tanh. The process of selection of maximum value which is the non-linearity of the maxout neuron is represented in Eq. (5). This process is also known as feature selection process.

During training for maxout neuron, the gradient is computed as given in Eq. (7).

$$\frac{\partial {h_l}^i}{\partial {z_l}^{ij}} = \begin{cases} 1 & \text{if } {z_l}^{ij} \geq {z_l}^{is} \, \forall_s \in 1, \ldots, k \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Equation (7) shows that the value of the gradient is 1 for the field with the maximum activation and 0 otherwise. The value of gradients is either 0 or 1 during training, so the vanishing gradient problem is resolved easily with maxout neurons. Therefore, the deep maxout neural networks are easily optimized as compared to conventional sigmoid neural networks.

6.3 Rectified Linear Units

Neural network training is generally performed by two types. First, a frame discriminative SGD CE criterion is used to train the DNN. Second, the sequence level objective function is used to readjust the CE-trained DNN weights [20]. Since speech is a non-stationary sequence level task, the second type is more relevant for speech recognition problem. Various researchers have demonstrated that sequence training improves the performance of ASR over a CE-trained DNN by 10–15% relatively [21, 29]. Second-order HF optimization is relevant in terms of performance gain with sequence training, though not as important for CE training [21]. Srivastava et al. [36] proposed a new way to regularize DNNs by the use of ReLU and dropouts. Dahl et al. [6] have shown in their research that a 5% relative reduction has been achieved in WER for CE trained DNNs using ReLU + dropout on a 50-h English Broadcast News large vocabulary continuous speech recognition. ReLU is a non-saturated linear activation function. Its output is 0 for negative values and input itself otherwise. ReLU function is defined as

$$h_l = \max(0, z_l) \tag{8}$$

However, subsequent HF sequence training without dropout rubbed out some of the gains and performance provided by a DNN trained with a sigmoid non-linearity without dropout. In this paper, dropout is efficiently applied with HF sequence training.

6.4 Parameterized Rectified Linear Units

ReLU units generate zero gradients whenever the units are not active. Therefore, gradient-based optimization will not update their weights. The result of constant zero gradients is slow down in the training process. To overcome this issue, He et al. [16] proposed PReLU which is an advanced version of ReLU and includes the negative part to fasten the learning. It successfully obviates the vanishing gradient problem. In this model, if the input is negative, then the output is produced by multiplying input with a slope of α; otherwise the output is the input itself. A PReLU function is defined as follows:

$$h_l = \begin{cases} \alpha z_l & \text{if } z_l < 0 \\ z_l & \text{otherwise} \end{cases} \tag{9}$$

Table 7: WER as a Function of Non-Linearity.

Non-Linearity	Model	WER (%) on training set	WER (%) on testing set
Sigmoid	No-drop	19.6	21.4
Sigmoid	Dropout	19.5	21.4
Maxout	No-drop	19.1	21.0
Maxout	Dropout	18.4	20.4
ReLU	No-Drop	19.3	21.1
ReLU	Dropout	18.6	20.7
PReLU	No-Drop	19.1	21.1
PReLU	Dropout	18.5	20.6

6.5 Dropout

Maxout neurons expertly manage the problem of under-fitting, so they have better optimization performance [38]. However, CNNs using maxout non-linearity are more prone to overfitting due to their high capacity. To address the overfitting problem, regularization methods like L_p-norm, weight decay, weight tying, etc., have been proposed. Srivastava et al. [36] proposed a promising regularization technique, dropout, to efficiently reduce the problem of overfitting. In this method, half of the activations within a layer are stochastically set to 0 for each training sample. By doing this, the hidden units cannot co-adapt to each other and learn better representation for the inputs. Goodfellow et al. [12] showed that dropout is an effective way to control the overfitting for maxout networks because of better model averaging. Various strategies are used for dropout regularization for the training and testing phase. Specifically, dropout ignores each hidden unit stochastically with probability p during the feed-forward operation in neural network training.

This anticipates complex co-adaptions between hidden units by making them independent from other units. Especially using dropout, y^l is given in Eq. (10).

$$\boldsymbol{y}^l = f\left(\frac{1}{1-p}.\boldsymbol{W}^l\left(\boldsymbol{r}^{l-1} * \boldsymbol{y}^{l-1}\right) + \boldsymbol{b}^l\right) \tag{10}$$

where $\boldsymbol{y}^l$ is the activation of layer l. $\boldsymbol{y}^{l-1}$ is the input into layer l. $\boldsymbol{W}^l$ represents the weight matrix for layer l. $\boldsymbol{b}^l$ represents a bias vector for layer l. $f(.)$ is the non-linear activation function such as sigmoid or maxout, and r is a binary mask where Bernoulli distribution with probability p of being 1 is used to draw each entry. During decoding, dropout is not used.

The hyper-parameter p is called dropout rate, and it is the ratio of the number of neurons to be omitted from training in order to improve generalization. A lower value of p means more information, and a higher value of p means more aggressive regularization. During testing, the factor $\frac{1}{(1-p)}$ is used to ensure that no units should be dropped out at test time and the total input will pass to each layer. During dropout training, the coefficient $(1-p)$ is used to scale down the neuron activations. This is an intelligent way to perform model averaging and to improve the model generalization ability.

The experimental results of different non-linear networks with or without dropout are shown in Table 7 and conclude that maxout networks converge faster than ReLU, PReLU, and sigmoidal networks. The WER is the same for sigmoid in both cases with or without dropout. During training, the maxout networks show better abilities to fit the training dataset. During testing, maxout and PReLU networks have shown almost the same WER, but sigmoidal and ReLU perform less. The sequence training is found more closely linked to objective function as compared to CE. The same dropout rate is applied for each layer. Dropout is varied with a step size of 0.05. The experiments confirmed that the dropout probability $p = 0.5$ is reasonable.

7 Result Analysis and Comparison

By the results of the experiments performed earlier, the maximum performance is achieved by CNN-BLSTM hybrid system with two CNN, three BLSTM, and three FC layers. The results are best for LWS with 512/512

Table 8: WER Values of Available Acoustic Models.

Method	WER (%)
GMM-HMM	23.5
DNN	19.8
Deep belief network	19.7
RNN	19.3
CNN	18.9
CNN-BLSTM	17.8

units, but its number of parameters is too high, i.e. 12.7 M. On the other hand, FWS with 256/256 units uses only 6.9 M parameters. FWS uses approximately half parameters as compared to LWS, and relative deterioration is only 2%. Therefore, FWS with 6.9 M parameters looks more appropriate to use. No doubt, the increase in hidden units also increases the recognition rate, but it increases the complexity of the network. Maxout neurons also performed well because of clean speech. In noisy environment, results may vary. Speaker adaption techniques improve the feature set, and as a result, the performance of the system rises. The comparison of WER of CNN-BLSTM hybrid systems with CNN, DNN, and RNN is shown in Table 8. The CNN-BLSTM hybrid trained by including speaker-adaption and maxout + dropout can achieve a 24.25%, 10%, and 5.8% relative improvement over the Gaussian-based hidden Markov model (HMM), DNN, and CNN, respectively. This helps to strengthen the hypothesis that CNN-BLSTM hybrid structure is better than other models for speech recognition task, and this advancement is attained due to its special structure.

8 Conclusion

In this paper, a more powerful hybrid acoustic model is proposed by including the advantages of CNN, BLSTM, and fully connected layers. The strength of CNN-BLSTM architecture is demonstrated for speech recognition task. Various weight sharing techniques and pooling strategies are explored that are normally used in computer vision. Unfortunately, none of the pooling techniques showed significant improvement in the ASR task. It is found that the structure having two CNN, three BLSTM, and three fully connected layers along with maxout neurons + dropout offers optimal result. In addition, VTLN-warped mel-FB $+ \Delta + \Delta\Delta$ are found to be the best locally correlated feature set for CNN. We also incorporated fMLLR features with CNN-BLSTM features by a new way. Overall, the proposed architecture achieved a relative improvement of 5.8% over best-performing CNN and 10% over a DNN system. The hybrid system achieved this high gain due to the maxout neuron, dropout, and speaker-adapted features. The experimental results proved that the hybrid of CNN-BLSTM is computationally efficient as well as competitive with the existing baseline system.

Conflict of Interest: We have no conflict of interest to declare.

Bibliography

[1] O. Abdel-Hamid, A. Mohamed, H. Jiang and G. Penn, Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition, in: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4277–4280, 2012.

[2] O. Abdel-Hamid, L. Deng and D. Yu, Exploring convolutional neural network structures and optimization techniques for speech recognition, *Interspeech* (2013), 3366–3370.

[3] Y. Bengio, P. Simard and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Netw.* **5** (1994), 157–166.

[4] J. Bruna, A. Szlam and Y. LeCun, Signal recovery from pooling representations, in: *31st International Conference on Machine Learning, ICML 2014*, Beijing, China, pp. 1585–1598, 2014.

[5] M. Cai, Y. Shi and J. Liu, Deep maxout neural networks for speech recognition, in: *Automatic Speech Recognition and Understanding (ASRU)*, 2013 IEEE Workshop on, pp. 291–296, 2013.

[6] G. E. Dahl, T. N. Sainath and G. E. Hinton, Improving deep neural networks for LVCSR using rectified linear units and dropout, in: *Acoustics, Speech and Signal Processing (ICASSP)*, 2013 IEEE International Conference on, pp. 8609–8613, 2013.

[7] S. B. Davis and P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, in: *Readings in Speech Recognition*, ed: Elsevier, pp. 65–74, 1990.

[8] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, A.Y. Ng, Large scale distributed deep networks, in: *Advances in Neural Information Processing Systems*, pp. 1223–1231, 2012.

[9] L. Deng, O. Abdel-Hamid and D. Yu, A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion, in: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6669–6673, 2013.

[10] M. J. Gales, Maximum likelihood linear transformations for HMM-based speech recognition, *Comput. Speech Lang.* **12** (1998), 75–98.

[11] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[12] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville and Y. Bengio, Maxout networks, in: *Proceedings of the 30th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, pp. 1319–1327, 2013.

[13] A. Graves and J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, *Neural Networks* **18** (2005), 602–610.

[14] A. Graves, N. Jaitly and A.-R. Mohamed, Hybrid speech recognition with deep bidirectional LSTM, in: *Automatic Speech Recognition and Understanding (ASRU)*, 2013 IEEE Workshop on, pp. 273–278, 2013.

[15] A. Graves, A.-R. Mohamed and G. Hinton, Speech recognition with deep recurrent neural networks, in: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, 2013.

[16] K. He, X. Zhang, S. Ren and J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[17] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior and M. Bacchiani, Asynchronous stochastic optimization for sequence training of deep neural networks, in: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5587–5591, 2014.

[18] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath and B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, *IEEE Signal Proc. Mag.* **29** (2012), 82–97.

[19] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato and Y. LeCun, What is the best multi-stage architecture for object recognition? in: *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, 2009.

[20] B. Kingsbury, Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling, in: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference* on, pp. 3761–3764, Taipei, Taiwan, 2009.

[21] B. Kingsbury, T. N. Sainath and H. Soltau, Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization, *Interspeech*, pp. 10–13, Portland, OR, USA, 2012.

[22] A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[23] N. Lambooij, *Applying image recognition to automatic speech recognition: determining suitability of spectrograms for training a deep neural network for speech recognition*, Bachelor thesis, Utrecht University, 2017.

[24] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, in: *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[25] L. Lee and R. C. Rose, Speaker normalization using efficient frequency warping procedures, in: *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings.*, 1996 IEEE International Conference on, pp. 353–356, Atlanta, GA, USA, 1996.

[26] A.-R. Mohamed, G. Hinton and G. Penn, Understanding how deep belief networks perform acoustic modelling, in: *Acoustics, Speech and Signal Processing (ICASSP)*, 2012 IEEE International Conference on, pp. 4273–4276, Olomouc, Czech Republic, 2012.

[27] T. Robinson, M. Hochberg and S. Renals, The use of recurrent neural networks in continuous speech recognition, in: C. H. Lee, F. K. Soong, and K. K. Paliwal, (Eds.), *Automatic Speech and Speaker Recognition. The Kluwer International Series in Engineering and Computer Science (VLSI, Computer Architecture and Digital Signal Processing),* vol. 355, pp. 233–258, Springer, Boston, MA, 1996.

[28] T. N. Sainath, B. Kingsbury, A.-R. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin and B. Ramabhadran, Improvements to deep convolutional neural networks for LVCSR, in: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 315–320, Olomouc, Czech Republic, 2013.

[29] T. N. Sainath, A.-R. Mohamed, B. Kingsbury and B. Ramabhadran, Deep convolutional neural networks for LVCSR, in: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8614–8618, Vancouver, Canada, May 26–31, 2013.

[30] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-R. Mohamed, G. Dahl, B. Ramabhadran, Deep convolutional neural networks for large-scale speech tasks, *Neural Networks* **64** (2015), 39–48.
[31] H. Sak, A. Senior and F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, *Interspeech*, pp. 338–342, Singapore, September 14–18, 2014.
[32] G. Saon, H. Soltau, A. Emami and M. Picheny, Unfolded recurrent neural networks for speech recognition, *Interspeech*, pp. 343–347, Singapore, September 14–18, 2014.
[33] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, *IEEE Trans. Signal Proc.* **45** (1997), 2673–2681.
[34] P. Sermanet, S. Chintala and Y. LeCun, Convolutional neural networks applied to house numbers digit classification, in: *Pattern Recognition (ICPR), 2012 21st International Conference* on, pp. 3288–3291, Stockholm, Sweden, 2012.
[35] H. Soltau, H.-K. Kuo, L. Mangu, G. Saon and T. Beran, Neural network acoustic models for the DARPA RATS program, *Interspeech*, pp. 3092–3096, Lyon, France, August 25–29, 2013.
[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* **15** (2014), 1929–1958.
[37] L. Toth, Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition, in: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 190–194, Florence, Italy, 2014.
[38] L. Toth, Convolutional deep maxout networks for phone recognition, *Interspeech*, pp. 1078–1082, Singapore, September 14–18, 2014.
[39] L. Tóth, Phone recognition with hierarchical convolutional deep maxout networks, *J. Audio Speech Music Proc.* **2015** (2015), 25. https://doi.org/10.1186/s13636-015-0068-3.
[40] E. Variani and T. Schaaf, VTLN in the MFCC domain: Band-limited versus local interpolation, in: *Twelfth Annual Conference of the International Speech Communication Association*, pp. 1273–1276, Florence, Italy, August 27–31, 2011.
[41] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. J. Lang, Phoneme recognition using time-delay neural networks, in: *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 328–339, Elsevier, Amsterdam, The Netherlands, pp. 393–404, 1989.
[42] R. G. Wijnhoven and P. de With, Fast training of object detection using stochastic gradient descent, in: *Pattern Recognition (ICPR), 2010 20th International Conference* on, pp. 424–427, Istanbul, Turkey, August 23–26, 2010.
[43] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, Odell, D. J. Ollason and D. Povey, *The HTK book*, Cambridge University Engineering Department, vol. 3, p. 175, Cambridge, UK, 2002.
[44] M. D. Zeiler and R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: *International Conference on Learning Representation*, Scottsdale, AZ, USA, 2013.