Ankit Kumar
Rajesh Kumar Aggarwal

HYBRID CNN-LIGRU ACOUSTIC MODELING USING SINCNET RAW WAVEFORM FOR HINDI ASR

Abstract *Deep neural networks (DNN) currently play a most vital role in automatic speech recognition (ASR). The convolution neural network (CNN) and recurrent neural network (RNN) are advanced versions of DNN. They are right to deal with the spatial and temporal properties of a speech signal, and both properties have a higher impact on accuracy. With its raw speech signal, CNN shows its superiority over precomputed acoustic features. Recently, a novel first convolution layer named SincNet was proposed to increase interpretability and system performance. In this work, we propose to combine SincNet-CNN with a light-gated recurrent unit (LiGRU) to help reduce the computational load and increase interpretability with a high accuracy. Different configurations of the hybrid model are extensively examined to achieve this goal. All of the experiments were conducted using the Kaldi and Pytorch-Kaldi toolkit with the Hindi speech dataset. The proposed model reports an 8.0% word error rate (WER).*

Keywords automatic speech recognition, CNN, CNN-LiGRU, DNN

Citation Computer Science 21(4) 2020: 397–417

1. Introduction

The ASR field has witnessed rapid improvement over the last few years. DNN has become the most prominent approach, almost replacing the Gaussian mixture model-hidden Markov model (GMM-HMM) approach [8, 17, 49]. Deep-learning technologies are responsible for making speech technologies commercially available at one's doorstep [15, 55], like Google Voice and Alexa. The deep-learning paradigm is continuously evolving and comes with new techniques and more-robust architectures every year. A wide variety of advanced architectures have been proposed, such as CNN [1], RNN [13], and the time-delay neural network (TDNN) [37]. Each one has its advantages and disadvantages. Despite these advancements, there is still a massive gap in performance.

CNN is an advanced version of DNN that holds several advantages over it. It can efficiently capture the local spatial correlation through local connectivity, which is hard to achieve via DNN [39]. CNNs are also able to deal with small shifts in frequency due to speaker variation or speaking style [2, 45]. Due to the sparse connectivity of the hidden neurons in each layer, CNN tremendously reduces the number of learnable parameters when compared to DNN. Recent research in speech recognition reports excellent performance with raw waveforms [20, 52, 57]. With standard features, CNN deals with fewer data points (which risk the loss of crucial information). Conversely, a raw waveform provides a richer and lossless input signal representation to a CNN. According to [40], standard CNN filters are not good enough to learn common acoustic features due to the lack of constraint on the neural parameter. These issues have been effectively dealt with by the recently proposed SincNet-CNN architecture [43], which uses prior knowledge in an efficient and interpretable manner. It uses sinc functions to explore the more meaningful filters. In SincNet-CNN, only the low and high cutoff frequencies of band-pass filters are directly learned from raw data, defining a very compact front-end filter bank with clear meanings [43]. According to [24], CNN does not perform well under noisy conditions. CNNs are also not able to handle variable-length inputs.

On the other side, RNNs are the most suitable choice for context-sensitive data (like speech), and the function of memory provided by the feedback loops offers higher accuracy. RNN can exploit long-term feedback dependencies across speech frames, which makes RNN less affected by temporal distortion [24]. This property makes them more suitable for speech recognition than other architectures, helping them to achieve high accuracy. Even though RNNs have a preferred position over different DNN architectures, they are still limited to the vanishing and exploding gradient problem [4]. To tackle this issue, the long-short term memory (LSTM) [19] model was proposed to control the flow of information. The LSTM model was widely applied in speech-recognition tasks with improved performance gain; however, it faces some issues when applied to low-resource languages [24]. In 2014, a gated recurrent unit (GRU) was proposed [7] with a simple architecture based on two gating units. These two gating mechanisms simplify the sophisticated LSTM cell design. Recently, LiGRU was proposed by Ravanelli and Bengio [41], which further disentangles the GRU design by removing the reset gate. This one-gated RNN model with the ReLu activation function showed high accuracy and low computational load, making it more popular among all RNN models [41].

In this paper, we aim to design a deep hybrid architecture by combining SincNet-CNN and LiGRU layers to take advantage of each model's merits. CNN is suitable for extracting position invariant features, and RNN is the best choice for sequence modeling [54]. CNN was shown to perform well on specific sequence processing problems at a considerably cheaper computational cost than RNN. The combined architecture of CNN-RNN is able to achieve both low computational cost and high accuracy [6]. The incorporation of a CNN stacked with LiGRU leads to a reduced WER for Hindi ASR. By using LiGRU rather than LSTM, we achieve a lower WER with reduced training time. In this work, we present a systematic comparison of the CNN, RNN, and Hybrid CNN-RNN acoustic models based on different acoustic features, hidden layers, activation functions, and hidden neurons. This paper also compares the computational efficiency of various acoustic models. The proposed architecture uses CNN capabilities to extract position-invariant features and the LiGRU model to model the long-term dependency. The performance of the proposed architecture was measured on the Hindi dataset. To the best of our knowledge, this work is the first attempt to explore this architectural variation in speech recognition.

The remaining part of the paper is organized as follows: Section 2 compares the previous work with the proposed work. Section 3 explains the system component used in this work. Section 4 discusses the proposed model. Section 5 gives the experimental setup and corpus details. Section 6 covers the experimental part of the paper, and Section 7 is the conclusion of the proposed system.

2. Prior work and contributions in Hindi ASR

The availability of a large amount of training data is the major obstacle for training the ASR system for most of the Indian languages, including Hindi. Hindi is the third-most-spoken language worldwide, with 637 million[1] speakers. However, Hindi is treated as limited-resource language due to the unavailability of a benchmarked large speech corpus and other resources. A limited vocabulary Hindi dataset of a duration of three hours is a well-known available resource for research. To our best knowledge, there has been no systematic comparison of Hindi ASR under the same training and testing conditions. For this purpose, we choose only those articles that were trained over the same Hindi dataset [5, 9–12, 26] (see Table 1). The majority of these have been done using GMM-HMM [5, 9–12] acoustic modeling and optimized integrated feature sets [9, 10, 12]. The impact of deep-learning techniques has not been properly investigated for the Hindi dataset. Recently, some works reported [35] that the CNN--LSTM, CNN-BLSTM model works well for the Hindi language. In [35,36], the author

[1]https://www.ethnologue.com/language/hin

trained Hindi ASR using 48 hours of a Hindi dataset from a different source. For better understanding the role of deep-learning techniques over Hindi ASR, we include only those works based on the Hindi dataset [46].

Table 1
Comparison with previous work (PER – Phoneme Error Rate; Acc. – Accuracy)

System	Feature Extraction	Acoustic Model	Language Model	Parameter measure	[%]
Biswas et al. (2014) [5]	WERBC	HMM	n-gram	PER	81.48
Dua et al. (2018) [10]	Q-PSO optimized MF-GFCC	GMM-HMM	n-gram	Acc.	80.40
Dua et al. (2019) [11]	MFCC	GMM-HMM	RNN LM	Acc.	84.40
Kumar et al. (2020) [26]	MFCC	TDNN	n-gram	Acc.	84.10
Dua et al. (2019) [12]	DE-optimized GFCC	GMM-HMM	n-gram	Acc.	86.90
Dua et al. (2018) [9]	PSO optimized MF-GFCC	GMM-HMM	n-gram	Acc.	87.96
Proposed work	Raw features	SincNet-CNN -LiGRU	n-gram	Acc.	92.0

3. System components

3.1. Convolutional Neural Network (CNN)

CNN [44] is another deep-learning architecture. CNN can efficiently capture the structural locality from the feature space [28]. CNN can also deal with a small shift in feature space due to the sparse connectivity of the hidden neurons in each layer. The CNN architecture is a combination of three sub-layers; namely, the convolution layer, the pooling layer, and the fully-connected (FC) layer. CNN-based acoustic modeling consists of alternating convolution and pooling layers, followed by FC layers. The speech data is organized in feature maps and feeds into the CNN architecture. The feature map term is borrowed from image processing, in which each pixel denotes the horizontal and vertical coordinates of a two-dimensional surface. In speech recognition, a feature map can be thought of as a spectrogram (two-dimensional) with a size of #times * #freqs (i.e., 11*40, where 11 is the context-window size, and 40 is the feature dimension) for static features. For dynamic features ($\Delta + \Delta\Delta$), the input feature map can be viewed as three different two-dimensional feature maps.

Convolution is the primary building block of the CNN architecture, in which a small window (filter or kernel) runs over the input feature map from the top-left corner to the bottom-right corner step by step. At each stage, the covered area by a small window (i.e., receptive field) performs a dot product with a filter and produces one output. All of the outputs generated by this process form a new feature map. Convolution output can be defined as follows:

$$fm^{(l)} = \sigma(k^{(l)} \cdot fm^{(l-1)} + b^{(l)}) \tag{1}$$

where $fm^{(l)}$ and $fm^{(l-1)}$ denote the feature maps, and $k^{(l)}$ denotes the kernel (filter) size. The convolution operation (*) is applied between the kernel and the feature map. The bias is denoted as $b^{(l)}$. Finally, the activation function (elu, sigmoid, ReLu, etc.) will be applied to generate a new feature map.

After the convolution operation, pooling operation is applied to the newly generated feature map to reduce the feature map dimensionality. This dimensionality reduction is achieved by the pooling function, which eliminates the irrelevant information and keeps the useful information. The pooling efficiently deals with a slight shift in the input pattern along with the frequency axis. Passricha and Aggarwal [36] investigate the performance of various pooling strategies like max-pooling, stochastic pooling, Lp pooling, mixed pooling, average pooling, etc. for the speech-recognition task. The output of the pooling layer feeds as an input to the second convolutional layer and so on. The last layers of the CNN are FC layers without any sparse connectivity. In the FC layers, the last one serves as the loss layer that computes the posterior probabilities of different classes.

The sparse connectivity of CNNs tremendously reduces the number of learnable parameters as compared to DNNs. The number of parameters in one convolutional layer can be computed as follows:

$$\#params^{(l)} = k^{(l)} \cdot fm_s^{(l)} \cdot fm_s^{(l-1)} \tag{2}$$

where $k^{(l)}$ denotes the filter size of layer l, and fm_s denotes the feature map size of layer l.

3.2. SincNet architecture

The numbers of time-domain convolutions are performed between the input raw speech signal and the finite impulse response (FIR) filters in the first convolution layer [30]. This process can be described as follows:

$$o[n] = i[n] \cdot k[n] = \sum_{x=0}^{X-1} i[x].k[n-x] \tag{3}$$

where $i[n]$ is the part of the speech signal, $k[n]$ is the kernel (filter) of length X, and $o[n]$ is the output of the convolution. In this operation, all of the elements of $k[n]$ are learnable parameters. In SincNet, $k[n]$ is replaced by function $g[n]$, which depends on only two learnable parameters.

$$o[n] = i[n] \cdot g[n, \theta] \tag{4}$$

In Sincnet, function g is implemented as a rectangular bandpass filter. In the frequency domain, the magnitude of rectangular bandpass filters can be defined as follows:

$$G[f, f_1, f_2] = rect\left(\frac{f}{2f_2}\right) - rect\left(\frac{f}{2f_1}\right) \tag{5}$$

where $rect_w(.)$ is the rectangular function with width w centered in 0.

The SincNet function used to get impulse response $g(n)$ in the time domain can be described as follows:

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n) \tag{6}$$

where f_1 and f_2 are the low- and high-frequency cutoffs of the rectangular bandpass filter. Here, f_1 and f_2 are the only two learnable parameters randomly initialized within a range of $[0, \frac{f_s}{2}]$, where f_s is the sampling frequency of the input signal. The Sinc function is described as $sinc(x) = \frac{sinc(x)}{x}$. Furthermore, this g is smoothed by the Hamming window [40].

In this way, SincNet helps us discover more meaningful and interpretable filters for CNNs. The SincNet based convolution operation is performed in the first layer, then multiple CNN or FC layers can be stacked before being fed into the softmax classifier. Unfortunately, SincNet was investigated by only a few works [29,30,34,43]. In this work, we proposed the hybrid architecture with SincNet-CNN and a recurrent neural network.

3.3. Recurrent Neural Network (RNN)

An RNN is a type of neural network that was proposed by Elman [13]. An RNN's loop-like structure allows it to remember previously processed information at future time instances. The hidden nodes in RNNs are self-connected and inter-connected to provide a full information exchange. Shuster and Paliwal [48] proposed a bidirectional RNN, which is an extension of a simple RNN (see Fig. 1).

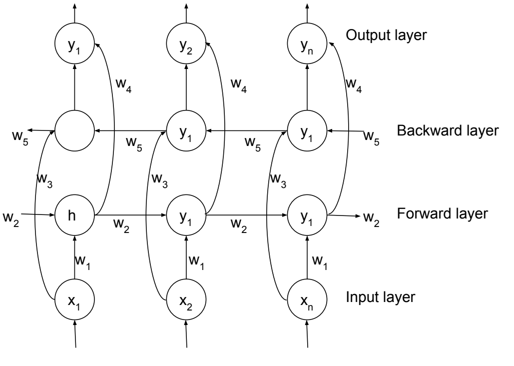

Figure 1. Bi-directional RNN

Bidirectional RNNs can process both past and future information. Speech is continuous data, and any model that has access to both past and future information can effectively predict the current information. An RNN's capability to process past and future information makes it a more suitable choice for speech recognition. RNN output can be calculated by the following equation:

$$h_t = tanh(W[h_{t-1}, x_t] + b_h) \tag{7}$$

The back-propagation through time (BPTT) training algorithm is used to train RNNs. These RNNs suffered from the vanishing/exploding gradient problem [47]. This issue was rectified by other RNN variants (like LSTM and GRU).

3.4. Gated Recurrent Neural Networks

3.4.1. Long Short-Term Memory (LSTM)

LSTM was proposed by [19] to overcome the limitations of the traditional RNN model. It is a distinct type of RNN model (see Fig. 2) with a memory cell and gating units. The memory cell helps remember the information over a long span of time, and the gating units help to control the information flow within the network [16]. The basic structure of the LSTM model contains three gates; namely, the input gates, output gate, and forget gate.

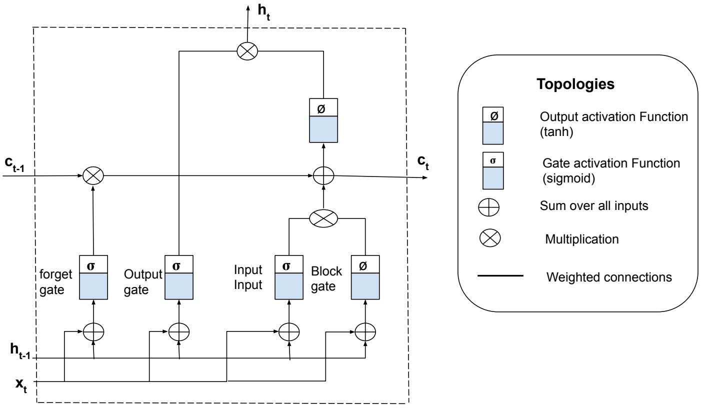

Figure 2. Basic building block of LSTM unit

By using these gates, the vanishing/exploding gradient problem is resolved, as they allow for an information flow in a more stable fashion over a lengthy span of time [56].

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{8}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{9}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{10}$$

$$\widetilde{c}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{11}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t \tag{12}$$

$$h_t = o_t \odot tanh(c_t) \tag{13}$$

3.4.2. Gated Recurrent Unit (GRU)

The GRU architecture simplifies the LSTM architecture by using only two gates (see Fig. 3). Due to the fewer gates, it has fewer parameters as compared to LSTM, which further helps it achieve great performance and fast convergence [53]. GRU was proposed by [7] in 2014. GRU removed the hidden cell state and combined the input gate and the forget gate into a single gate (update gate z) from the LSTM architecture. GRU is based on update gate z and reset gate r. The connection relationship can be understood by the following equations:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{14}$$

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{15}$$

$$\widetilde{h}_t = tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h) \tag{16}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{17}$$

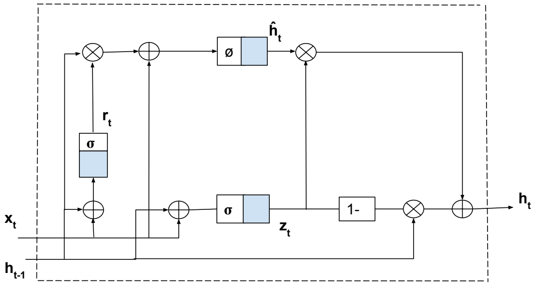

Figure 3. Basic building block of GRU unit

3.4.3. Light Gated Recurrent Unit (LiGRU)

Li-GRU is an extension of the GRU architecture that was proposed by Ravanelli [41] in 2018. In LiGRU, the architecture only works with update gate z, as it removed reset gate r from the GRU architecture. According to [41], the reset gate plays a vital role when dealing with the discontinuities that occur in the sequence. This situation rarely occurs in the case of speech, as history will always be helpful. The removal of reset gate rt leads to the modification of Equation 16:

$$\widetilde{h}_t = tanh(W_h[x_t, h_{t-1}] + b_h) \tag{18}$$

As proposed by [41], the standard tanh activation function is also replaced by the ReLu activation function for better results.

$$\widetilde{h}_t = ReLu(W_h[x_t, h_{t-1}] + b_h) \tag{19}$$

4. Proposed architecture

The recent work shifted to evolve a more interpretable neural network [32,34,43] by directly processing the raw speech waveform. Standard features like MFCC, MFSC, and FBANK consume fewer data points (e.g., 4,000 per second at a sampling frequency of 16 kHz) compared to the raw speech signal (16,000 per second at a sampling frequency of 16 kHz). Due to the richer and lossless input signal representation, raw speech input to the neural network model has become more popular [23, 33]. The standard features are based on human perception with no guarantee of producing optimal features [43]. In more recent work, CNNs were fed by raw speech waveforms [21, 23, 33, 52, 57]. The First CNN layer with a raw speech waveform has to deal with the high dimensionality of the data as well as the vanishing gradient issue (generally in a very deep architecture) [43]. Due to the lack of interpretability of the filterbank learned in the first convolution layer, Ravanelli and Bengio proposed a new convolution layer (SincNet) [40]. SincNet uses prior speech-processing knowledge to produce interpretable filters in the first convolution layer. Recently, a few other filters were also proposed for SincNet to enhance the system's accuracy [29, 32].

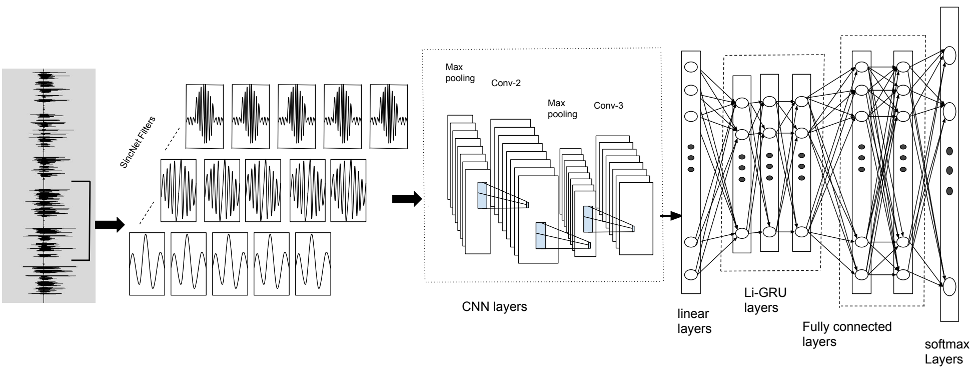

Figure 4. Proposed architecture

The proposed hybrid architecture incorporates the CNN, LiGRU, and DNN models. Figure 4 illustrates the hybrid architecture (and in Table 2 the configuration details are given). In the first convolution layer, we applied the raw speech signal as an input to obtain the output feature map. All of the convolution layers have 256 feature maps in this work. The dimensionality of the resultant feature map was further reduced by applying the max pooling layer. The pooling layer helps to keep the useful information and eliminate the insignificant details. The output of the pooling layer was fed as an input to the second convolution layer. After the third CNN layer, a linear layer was introduced to reduce the dimensionality without losing any information. LiGRU can encodes the sequence history in its internal state, which gives the potential to predict the next phoneme by using previously observed features. We used three LiGRU layers, with 550 units per layer. On top of the LiGRU layers, we put FC layers of a size of 5 with 1,024 units in each layer. We used FC layers to determine the posterior probabilities of different classes, as DNNs are the most suitable for class-based discrimination [17]. The batch-size was 8, with an initial learning rate of 0.0004. The learning rate was tuned on the development set for 24 epochs.

Table 2

Configurations for proposed architecture

Layers	#Layers	Feature Map Size	Filter Size	#Neurons
Convolutional	3	256,256,256	128 samples,5,5	–
Pooling	2	256,256	3,2	–
Li-GRU	3	–	–	550
Fully Connected	5	–	–	1,024

After each epoch, the learning rate was reduced by a factor of two If the frame accuracy on the development set went below a certain threshold. All other configuration details can be found in Table 2. We used a Kaldi decoder with the beam set to 13 and the lattice beam set to 8. Interestingly, this deep architecture has 11 layers (3 SincNet-CNN, 3 LiGRU, and 5 MLP) with fewer parameters ($\approx 10.6M$), which is approximately half in size of the five-layer LSTM model. This property makes the proposed architecture very computationally efficient when compared to the other RNN models.

5. Corpus details & experimental setup

5.1. Corpus details

We used a well-annotated and phonetically rich dataset [46] that was developed by TIFR, Mumbai (see Table 3).

Table 3

Hindi Speech Corpus Details

SN	Dataset	#Sentences	Duration	Duration after five-fold speed perturbation
1	Training	800	2.1 Hours	10.5 Hours
2	Development	100	10 Min.	50 Min.
3	Testing	100	10 Min.	–

The total duration of this dataset is 2.5 hours, which is treated as a low resource to train the DNN acoustic models. To cope with this issue, we apply a five-fold speed perturbation on the original train and development set. This dataset was recorded in a quiet room with two microphones at a 16 kHz sampling frequency. For the recording, 100 male/female speakers of various age groups were utilized. The pronunciation dictionary was made by using 68 phones. Each speaker was used to record ten sentences, out of which two sentences remained the same for all speakers. We used 80% of this dataset for training, 10% for development, and the remaining for testing purposes.

5.2. Experimental setup

The GMM-HMM baseline system was first built to produce an alignment for the proposed neural network training. The 39-dimension MFCC features were used to train the GMM-HMM system. These features were extracted by using a 25-ms Hamming window with a 10-ms frameshift. The baseline GMM-HMM system was built by using the Kaldi [38] toolkit with the TIMIT recipe. All of the neural network models in this work were trained using the Pytorch-Kaldi [42] toolkit. The feed-forward connections in the DNNs were initialized as described in [14]. The orthogonal matrices [27] were used to initialize the recurrent weights. To gain better generalization capabilities and performance improvement, we applied the dropout technique [18, 50] with the same mask across all time steps (as suggested by Moon et al. [31]). To address the internal covariate shift problem and speed-up the training time, we used the batch normalization [22] technique. Batch normalization also helps to improve the system's performance [18].

All of the training sentences were stored in ascending order according to their lengths. We used a mini-batch size of eight sentences starting from short utterances, and this minibatch size was progressively increased by the training algorithm. The Kaldi recipe was used for decoding and rescoring. The adaptive moment estimation [25] algorithm was used for optimization, which ran for 24 epochs. After each epoch, the system performance was measured on a development set. If the performance falls below a certain threshold (th = 0.001), the learning rate was updated as half the current value. An initial learning rate of 0.0004, a dropout factor of 0.2, and the ReLu activation function were chosen for all baseline experiments.

The texts from the various sources were collected to train the language model. These sources were the Emili corpus, magazines, newspapers, web text, and newsletters. For the Hindi language, 32,000 words were collected to train the language model. The SRI language modeling (SRILM) toolkit [51] was used to train the language model on the available text data of the Hindi language. The language model used in all experiments is the Kneser-Nay smoothed tri-gram language model.

6. Results & discussion

6.1. Experiments with different combinations of acoustic feature-set and activation functions

Table 4 shows the performance of Hindi ASR with various neural network-based acoustic modeling using a different combination of acoustic feature sets. TIMIT configurations of the PyTorch-Kaldi toolkit were used to train the acoustic models. Table 4 reports the WER achieved by the acoustic models. Except for CNN, all of the neural network architectures in this experiment contain 5 hidden layers with 550 hidden neurons in each layer with the ReLu activation function.

Table 4

Performance evaluation on different feature-set with different acoustic models

Acoustic Models	Features	Language Model	WER [%]
RNN	MFCC	3G	20.4
	FBANK		19.9
	MFCC+FBANK		17.4
	MFCC+FBANK+fMLLR		16.8
LSTM	MFCC	3G	19.2
	FBANK		18.6
	MFCC+FBANK		16.1
	MFCC+FBANK+fMLLR		15.9
GRU	MFCC	3G	17.5
	FBANK		16.9
	MFCC+FBANK		14.9
	MFCC+FBANK+fMLLR		14.2
Li-GRU	MFCC	3G	16.9
	FBANK		16.2
	MFCC+FBANK		13.6
	MFCC+FBANK+fMLLR		**12.7**
CNN	FBANK	3G	14.6
	MFSC		14.8
	Raw		14.4
	SincNet		13.9

For all of the RNN variants, we used the concatenated feature sets of MFCC, FBANK, and fMLLR. All of the features were extracted by Kaldi and concatenated by the Pytorch-Kaldi toolkit. These features with five frameshifts were used as input for the five-layer RNN models. In the training of the recurrent models, we truncated the long sentences with values of more than 1,000 to mitigate the out-of-memory issue. The RNN with ReLu activation function does not include any gating mechanism. LSTM performs slightly better ($\approx 1\%$) than the ReLu-RNN model. The GRU model shows a $\approx 1\%$ improvement over the LSTM model. The baseline CNN system uses 40-dimensional FBANK features and an 11-frame context window. Three convolutional layers with 256 feature maps and filter sizes of $9 \cdot 9$, $3 \cdot 3$, and $3 \cdot 3$ were chosen, and Max-Pooling was applied to reduce the dimension. We used MFSC, raw, and SincNet features as input for CNN as well. The best WER (12.7%) was achieved via the Li-GRU acoustic model. The Li-GRU architecture is based on a single gating mechanism. The combination of the MFCC, FBANK, and fMLLR feature sets gives the lowest WER in all of the neural network architectures.

In this experiment, we also investigated the role of activation functions like sigmoid, tanh, relu, leaky-relu, and elu on the accuracy of the Hindi ASR system. We found that the elu activation function gives the best WER (11.9%). All of the other configurations remained the same throughout all of the experiments. Overall, a 20% relative reduction in the WER was measured when using elu as compared to the sigmoid activation function for Hindi speech recognition (see Table 5). This work can be explored to investigate the other hyperparameters of the convolutional neural network.

Table 5

Performance evaluation of Hindi ASR using different activation functions

Models	Features	# Layers	sigmoid	tanh	relu	leaky-relu	elu
DNN	FBANK	7	25.8	24.6	23.9	23.3	22.8
LSTM	MFCC+FBANK+fMLLR	5	17.9	16.3	15.9	15.4	14.9
GRU	MFCC+FBANK+fMLLR	5	17.2	15.8	14.2	13.8	13.4
Li-GRU	MFCC+FBANK+fMLLR	5	14.9	13.5	12.7	12.3	**11.9**
CNN	FBANK	3	17.4	15.9	14.6	14.2	13.8
	MFSC	3	17.6	16.2	14.8	14.5	14.1
	RAW	3	17.2	15.7	14.4	14.1	13.6
	SincNet	3	16.8	15.4	13.9	13.7	**13.4**

In Table 6, the numbers of the parameters and training times are compared among the different acoustic models. The average training time per epoch was computed for comparison. For the same configuration, the LiGRU model's parameter size is just half of that in the LSTM model. For the Hindi dataset, an approx. 27% reduction was recorded by the LiGRU acoustic model as compared to the LSTM model. In the previous section, we already observed the impact of the LiGRU model on system accuracy. Due to its lower parameter size, less training time, and high accuracy, LiGRU becomes the most suitable choice for further investigation.

Table 6

Training time and parameter size comparison

Models	#Layers	#Neurons	#Parameters	Training times
DNN	7	1,024	6.7 M	380 s.
RNN	5	550	5.5 M	500 s.
LSTM	5	550	20.1 M	1,369 s.
GRU	5	550	15.2 M	1,207 s.
LiGRU	5	550	**10.4 M**	**997 s.**

6.2. SincNet-CNN with raw speech

In the case of raw speech as input to the first CNN layer, the input dimension is one, as the raw features are composed of a window of the temporal speech signal. Each speech sentence was split into chunks of 300 ms with 10 ms overlaps (i.e., the input size to the network is $0.300 \cdot 16{,}000 = 48{,}000$ at a sampling frequency of 16 kHz). This input is fed into the Sincnet filters. In this work, we use 256 SincNet filters (each with a length of 128) that are shifted by 10 samples. By this, each filter produces an output sequence of a size of $\left\lfloor \frac{(inputvectorsize - filterlength)}{stride} \right\rfloor + 1$ (i.e., $\left\lfloor \frac{(48{,}00-128)}{10} \right\rfloor + 1 = 468$).

Table 7

Network hyper-parameters for raw speech input

Parameters	Unit	Value
Input window size	ms	300
Kernel width of the first CNN layer	samples	128
Kernel width of other CNN layers	frames	4–5
Max-pooling kernel width	frames	2–3
No. of hidden layers in gated RNN	units	3
No. of hidden units in each RNN layer	units	550
No. of hidden layers in MLP	units	5
No. of hidden units in each MLP layer	units	1,024

The hyper-parameters of the CNN network with the raw speech waveform as input are given in Table 7. After the first convolution layer, the output dimension was reduced by applying max-pooling with a filter size of 3. The pooling layer reduces the output dimension by a factor of 3 (i.e., $468/3 = 156$). The second convolution layer takes 156 input samples from each of the 256 filters and produces $156 - 5 + 1 = 152$ samples. The overall output dimension of the second convolution layer will become $256 \cdot 152/2 = 19,456$. We used a non-overlapping stride of a size of 1 in all convolution layers except for the first CNN layer.

Table 8

Configurations used for SincNet-CNN architecture

#	Type	Input Stream	#Filters	Size	Stride	#Param.
1	SincNet-Conv.	1	256	128	10	512
	Max-Pooling	–	–	3	1	–
2	Conv.	256	256	5	1	1,638,400
	Max-Pooling	–	–	2	1	–
3	Conv.	256	256	5	1	1,638,400
	Max-Pooling	–	–	2	1	–
4	Conv.	256	256	5	1	1,638,400
	Max-Pooling	–	–	2	1	–
5	Conv.	256	256	5	1	1,638,400
	Max-Pooling	–	–	2	1	–
6	Conv.	256	256	4	1	1,048,576

The detailed configuration of SincNet-CNN can be found in Table 8. In the third convolution layer, the input samples of a size of 76 from each of the 256 filters were given as input, and the output sample size will become $76 - 5 + 1 = 72$ samples. Again, pooling with a filter size of 2 was applied to reduce the output dimension. In the fourth CNN layer, the sample size was reduced by $36 - 5 + 1 = 32$, which was further reduced to $32/2 = 16$ by adding the pooling layer. In the fifth CNN layer, this size will become $16 - 5 + 1 = 12$; the pooling layer reduces this to 6. Finally, in the sixth CNN layer, the sample size will become 3. Layer normalization [3] was applied in each convolution layer. The parameters of the first layer of SincNet-CNN were initialized by using the mel-scale lower and upper cut-off frequencies. All of the other CNN layers were initialized by the "Glorot" scheme [14]. We further stacked 3 LiGRU layers with 550 hidden units and 5 MLP layers with 1,024 hidden units.

Table 9

WER [%] comparison of proposed model having increased convolution layers and hidden units in FC layers

No. of hidden layers			WER [%]			
			Hidden Units			
Conv.	LiGRU	FC	550	800	1,024	2,048
2	2	5	9.8	9.6	9.5	9.4
		6	9.7	9.5	9.4	9.3
		7	9.6	9.5	9.3	9.2
		8	9.5	9.4	9.3	9.2
3	2	5	9.4	9.3	9.2	9.2
		6	9.3	9.2	9.1	9.0
		7	9.4	9.3	9.1	9.1
		8	9.2	9.1	9.0	8.9
4	2	5	9.1	9.0	8.9	8.8
		6	9.0	8.9	8.8	8.7
		7	8.9	8.8	8.6	8.5
		8	8.7	8.6	8.5	8.4
5	2	5	8.7	8.6	8.5	8.4
		6	8.6	8.5	8.4	8.3
		7	8.5	8.4	8.3	8.2
		8	8.4	8.3	**8.2**	**8.2**

We train four architectures based on Convolution Layers 2–6. The first architecture was composed of three convolution layers, three LiGRU layers, and five MLP layers (see Table 9). In the second through fourth architectures, only the convolution layer increases; all other configurations remain the same. The feature dimensions for all four architectures remained the same as per the table that was already explained. These architectures were extensively examined by increasing the number of MLP layers and hidden units in each MLP layer. Two observations were recorded here. First, as the number of CNN layers increases, the accuracy also increases. The best model was found to contain six convolution layers. Second, as the number of hidden layers and hidden units increases in the FC layers, the discrimination capabilities of DNN increases. This result leads to a corresponding WER reduction. We found 8 MLP layers that give the best results with 2,048 hidden units in each layer. There is a very slight difference in the WER between 1,024 and 2,048 hidden units. By considering computational efficiency, we found the 1,024-hidden-unit size most efficient for this work. The proposed very deep hybrid architecture (six CNN + three LiGRU + eight MLP layers) performs well with raw speech features.

Table 10
Varying length k of filters (WER [%])

Model	Filter length k in samples					
	64	80	128	256	512	1,024
CNN-SincNet	8.6	8.4	8.2	**8.0**	8.2	8.4
CNN-Raw	8.9	8.7	8.5	8.3	8.6	8.8

The length of the filter has a significant impact on system accuracy (see Table 10). The filter of a length of $k = 128$ has an impulse response that corresponds to $\frac{128}{16\,\text{kHz}} = 8\,\text{ms}$ at most or a bandwidth that is at least $\frac{16\,\text{kHz}}{128} = 125\,\text{Hz}$. Table 11 shows the WER of Hindi ASR when utilizing different window lengths. We found that 256 is the optimal window-length choice for Hindi ASR. Table 11 considers the 17-layer architecture (six SincNet-CNN + three LiGRU + eight MLP layers) for system accuracy.

Table 11
WER [%] for different acoustic features

SN	Models	Features	WER(%)
1	CNN-LSTM	FBANK	11.9
2	CNN-GRU		10.6
3	CNN-LiGRU		9.1
4	CNN-LSTM	MFSC	12.1
5	CNN-GRU		10.8
6	CNN-LiGRU		9.3
7	CNN-LSTM	Raw	11.2
8	CNN-GRU		9.8
9	CNN-LiGRU		8.3
10	CNN-LSTM	SincNet	10.8
11	CNN-GRU		9.4
12	CNN-LiGRU		**8.0**

6.3. Comparison of hybrid acoustic models with different acoustic features

In this experiment, we compared the WER [%] of different hybrid acoustic models with various acoustic features. We found that raw speech as input for the acoustic models to be the right choice rather than hand-crafted pre-computed features like MFSE and FBANK. All of the acoustic models are the combination of six CNN, three LiGRU, and eight MLP layers. The LiGRU layers use 550 hidden units, and the MLP layer uses 1,024 hidden units in each layer. We found that the proposed SincNet-CNN-LiGRU model performs better with a 1% WER reduction as compared to the FBANK features.

7. Conclusion

This paper describes the impact of our proposed hybrid neural network architecture on the Hindi ASR system. The proposed architecture has shown significant performance improvement when compared to the other acoustic models. Except for the performance improvement, the proposed architecture also considerably reduces the computational load and increases the convergence speed. We obtained the lowest WER (8.0%) when using the proposed model. The other findings of this work can be summarized as follows:

- Increasing the convolutional layers by the careful selection of the kernel and pooling sizes helps improve the system's accuracy by up to 1%.
- The proposed architecture has 17 layers (very deep architecture) and fewer parameters.
- Raw speech features were found to a more suitable choice as input for CNN as compared to handcrafted features like FBANK.

References

[1] Abdel-Hamid O., Mohamed A.r., Jiang H., Deng L., Penn G., Yu D.: Convolutional Neural Networks for Speech Recognition, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22(10), pp. 1533–1545, 2014.

[2] Abdel-Hamid O., Mohamed A.r., Jiang H., Penn G.: 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, IEEE, 2012.

[3] Ba J.L., Kiros J.R., Hinton G.E.: Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] Bengio Y., Simard P., Frasconi P.: Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks*, vol. 5(2), pp. 157–166, 1994.

[5] Biswas A., Sahu P.K., Chandra M.: Admissible wavelet packet features based on human inner ear frequency response for Hindi consonant recognition. *Computers & Electrical Engineering*, vol. 40(4), pp. 1111–1122, 2014.

[6] Chawla A., Lee B., Fallon S., Jacob P.: Host Based Intrusion Detection System with Combined CNN/RNN Model. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 149–158, Springer, 2018.

[7] Cho K., Merriënboer van B., Bahdanau D., Bengio Y.: On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[8] Dahl G.E., Yu D., Deng L., Acero A.: Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20(1), pp. 30–42, 2011.

[9] Dua M., Aggarwal R.K., Biswas M.: Discriminative Training Using Noise Robust Integrated Features and Refined HMM Modeling, *Journal of Intelligent Systems*, vol. 29(1), pp. 327–344, 2018.

[10] Dua M., Aggarwal R.K., Biswas M.: Optimizing Integrated Features for Hindi Automatic Speech Recognition System, *Journal of Intelligent Systems*, vol. 29(1), pp. 959–976, 2018.

[11] Dua M., Aggarwal R.K., Biswas M.: Discriminatively trained continuous Hindi speech recognition system using interpolated recurrent neural network language modeling, *Neural Computing and Applications*, vol. 31(10), pp. 6747–6755, 2019.

[12] Dua M., Aggarwal R.K., Biswas M.: GFCC based discriminatively trained noise robust continuous ASR system for Hindi language, *Journal of Ambient Intelligence and Humanized Computing*, vol. 10(6), pp. 2301–2314, 2019.

[13] Elman J.L.: Finding structure in time, *Cognitive Science*, vol. 14(2), pp. 179–211, 1990.

[14] Glorot X., Bengio Y.: 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.

[15] Goodfellow I., Bengio Y., Courville A.: *Deep learning*, MIT Press, 2016.

[16] Greff K., Srivastava R.K., Koutník J., Steunebrink B.R., Schmidhuber J.: LSTM: A Search Space Odyssey, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28(10), pp. 2222–2232, 2016.

[17] Hinton G., Deng L., Yu D., Dahl G.E., Mohamed A.r., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Sainath T.N., Kingsbury B.: Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, *IEEE Signal Processing Magazine*, vol. 29(6), pp. 82–97, 2012.

[18] Hinton G.E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R.R.: Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[19] Hochreiter S., Schmidhuber J.: Long Short-Term Memory, *Neural Computation*, vol. 9(8), pp. 1735–1780, 1997.

[20] Hoshen Y., Weiss R.J., Wilson K.W.: Speech acoustic modeling from raw multichannel waveforms. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4624–4628. IEEE, 2015.

[21] Hoshen Y., Weiss R.J., Wilson K.W.: Speech acoustic modeling from raw multichannel waveforms. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4624–4628. IEEE, 2015.

[22] Ioffe S., Szegedy C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *arXiv preprint arXiv:1502.03167*, 2015.

[23] Jung J.W., Heo H.S., Yang I.H., Shim H.J., Yu H.J.: Avoiding Speaker Overfitting in End-to-End DNNs Using Raw Waveform for Text-Independent Speaker Verification, *Extraction*, vol. 8(12), pp. 23–24, 2018.

[24] Kang J., Zhang W.Q., Liu W.W., Liu J., Johnson M.T.: Advanced recurrent network-based hybrid acoustic models for low resource speech recognition, *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018(1), p. 6, 2018.

[25] Kingma D.P., Ba J.: Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Kumar A., Aggarwal R.: A Time Delay Neural Network Acoustic Modeling for Hindi Speech Recognition. In: *Advances in Data and Information Sciences*, pp. 425–432, Springer, 2020.

[27] Le Q.V., Jaitly N., Hinton G.E.: A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv preprint arXiv:1504.00941*, 2015.

[28] LeCun Y., Bottou L., Bengio Y., Haffner P.: Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE*, vol. 86(11), pp. 2278–2324, 1998.

[29] Loweimi E., Bell P., Renals S.: On Learning Interpretable CNNs with Parametric Modulated Kernel-Based Filters. In: Interspeech, pp. 3480–3484, 2019.

[30] Mitra S.K., Kuo Y.: *Digital Signal Processing: A Computer-Based Approach*, vol. 2, McGraw-Hill, New York, 2006.

[31] Moon T., Choi H., Lee H., Song I.: RNNDROP: A novel dropout for RNNS in ASR. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 65–70. IEEE, 2015.

[32] Noé P.G., Parcollet T., Morchid M.: CGCNN: Complex Gabor Convolutional Neural Network on raw speech. In: *ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7724–7728, IEEE, 2020.

[33] Palaz D., Magimai-Doss M., Collobert R.: Analysis of CNN-Based Speech Recognition System Using Raw Speech as Input. Technical report, Idiap, 2015.

[34] Parcollet T., Morchid M., Linares G.: E2E-SINCNET: Toward Fully End-To-End Speech Recognition. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7714–7718. IEEE, 2020.

[35] Passricha V., Aggarwal R.K.: a Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition, *Journal of Intelligent Systems*, vol. 1 (ahead--of-print), 2019.

[36] Passricha V., Aggarwal R.K.: A comparative analysis of pooling strategies for convolutional neural network based Hindi ASR. *Journal of Ambient Intelligence and Humanized Computing*, vol. 11(2), pp. 675–691, 2020.

[37] Peddinti V., Povey D., Khudanpur S.: A time delay neural network architecture for efficient modeling of long temporal contexts. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[38] Povey D., Ghoshal A., Boulianne G., Burget L., et al.: The Kaldi speech recognition toolkit. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, 2011.

[39] Qian Y., Bi M., Tan T., Yu K.: Very Deep Convolutional Neural Networks for Noise Robust Speech Recognition, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24(12), pp. 2263–2276, 2016.

[40] Ravanelli M., Bengio Y.: Speech and Speaker Recognition From Raw Waveform with Sincnet. *arXiv preprint arXiv:1812.05920*, 2018.

[41] Ravanelli M., Brakel P., Omologo M., Bengio Y.: Light Gated Recurrent Units for Speech Recognition, *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2(2), pp. 92–102, 2018.

[42] Ravanelli M., Parcollet T., Bengio Y.: The Pytorch-Kaldi Speech Recognition Toolkit. In: *ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6465–6469. IEEE, 2019.

[43] Ravanelli M, Bengio Y.: Interpretable Convolutional Filters with SincNet. *arXiv preprint arXiv:1811.09725*, 2018.

[44] Rumelhart D.E., Hinton G.E., Williams R.J.: Learning Internal Representations by Error Propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[45] Sainath T.N., Kingsbury B., Saon G., Soltau H., Mohamed A.r., Dahl G., Ramabhadran B.: Deep Convolutional Neural Networks for Large-Scale Speech Tasks, *Neural Networks*, vol. 64, pp. 39–48, 2015.

[46] Samudravijaya K., Rao P.V.S., Agrawal S.S.: Hindi Speech Database. In: *Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, 2000.

[47] Schmidhuber J.: Deep Learning in Neural Networks: An Overview, *Neural Networks*, vol. 61, pp. 85–117, 2015.

[48] Schuster M., Paliwal K.K.: Bidirectional recurrent neural networks, *IEEE Transactions on Signal Processing*, vol. 45(11), pp. 2673–2681, 1997.

[49] Seide F., Li G., Yu D.: Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In: *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[50] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *The Journal of Machine Learning Research*, vol. 15(1), pp. 1929–1958, 2014.

[51] Stolcke A.: SRILM – an extensible language modeling toolkit. In: *Seventh International Conference on Spoken Language Processing*, 2002.

[52] Tüske Z., Golik P., Schlüter R., Ney H.: Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR. In: *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[53] Xu C., Shen J., Du X., Zhang F.: An Intrusion Detection System Using a Deep Neural Network with Gated Recurrent Units, *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[54] Yin W., Kann K., Yu M., Schütze H.: Comparative Study of CNN and RNN for Natural Language Processing. *arXiv preprint arXiv:1702.01923*, 2017.

[55] Yu D., Deng L.: *Automatic Speech Recognition*, Springer, 2016.

[56] Yu D., Li J.: Recent progresses in deep learning based acoustic models, *IEEE/CAA Journal of Automatica Sinica*, vol. 4(3), pp. 396–409, 2017.

[57] Zeghidour N., Usunier N., Synnaeve G., Collobert R., Dupoux E.: End-to-End Speech Recognition From the Raw Waveform. *arXiv preprint arXiv:1806.07098*, 2018.

Affiliations

Ankit Kumar
National Institute of Technology, Department of Computer Engineering, Kurukshetra, Haryana, India, anketvit@gmail.com

Rajesh Kumar Aggarwal
National Institute of Technology, Department of Computer Engineering, Kurukshetra, Haryana, India, rka15969@gmail.com

Received: 06.04.2020
Revised: 29.06.2020
Accepted: 07.07.2020