Kang *et al. EURASIP Journal on Audio, Speech, and Music Processing* (2018) 2018:6
https://doi.org/10.1186/s13636-018-0128-6

EURASIP Journal on Audio, Speech, and Music Processing

RESEARCH Open Access

Advanced recurrent network-based hybrid acoustic models for low resource speech recognition

Jian Kang[1], Wei-Qiang Zhang[1*], Wei-Wei Liu[2], Jia Liu[1] and Michael T. Johnson[3]

Abstract

Recurrent neural networks (RNNs) have shown an ability to model temporal dependencies. However, the problem of exploding or vanishing gradients has limited their application. In recent years, long short-term memory RNNs (LSTM RNNs) have been proposed to solve this problem and have achieved excellent results. Bidirectional LSTM (BLSTM), which uses both preceding and following context, has shown particularly good performance. However, the computational requirements of BLSTM approaches are quite heavy, even when implemented efficiently with GPU-based high performance computers. In addition, because the output of LSTM units is bounded, there is often still a vanishing gradient issue over multiple layers. The large size of LSTM networks makes them susceptible to overfitting problems. In this work, we combine local bidirectional architecture, a new recurrent unit, gated recurrent units (GRU), and residual architectures to address the above problems. Experiments are conducted on the benchmark datasets released under the IARPA Babel Program. The proposed models achieve 3 to 10% relative improvements over their corresponding DNN or LSTM baselines across seven language collections. In addition, the new models accelerate learning speed by a factor of more than 1.6 compared to conventional BLSTM models. By using these approaches, we achieve good results in the IARPA Babel Program.

Keywords: Gated recurrent units, Recurrent architectures, Low resource speech recognition

1 Introduction

Automatic speech recognition (ASR) has undergone rapid change in recent years. Deep neural networks (DNN) combined with hidden Markov models (HMM) have become the dominant approach for acoustic modeling [1, 2], replacing the traditional Gaussian mixture model-hidden Markov models (GMM-HMMs) approach. Utilizing increased availability of both computational power and training data, error rates have been reduced significantly across many speech recognition tasks [3, 4]. A wide variety of NN architectures have been introduced, each having associated advantages and disadvantages. Of all these architectures, recurrent neural networks have shown strong comparative performance.

Recurrent neural networks (RNNs) [5] are a neural network framework that include self-connections from the previous time step as inputs. Each unit contains a dynamic history of the sequence of input features sequence, instead of a fixed-size window. This approach exploits long-term feature dependencies across speech frames. As a result of this structure, RNNs are less affected by temporal distortion. Due to these properties, some researchers have investigated using RNNs to capture longer context [6–10], reporting improved performance compared to DNNs.

Although RNNs are well suited for sequence tasks, the time dependencies that can be learned are still limited to the vanishing and exploding gradient problem [11]. To solve this problem, the long short-term memory (LSTM) [12] has been proposed. LSTM units use gates to control information flow and effectively create shortcut paths across multiple temporal steps. This gate mechanism makes LSTM architectures well suited to sequence tasks and has improved robustness [13–15]. LSTM-based acoustic models have been successfully applied to several

*Correspondence: wqzhang@tsinghua.edu.cn
[1]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic, Engineering, Tsinghua University, Beijing 100084, China
Full list of author information is available at the end of the article

speech applications, such as voice search tasks [16, 17], with good performance.

Although LSTM models have achieved excellent results for large vocabulary continuous speech recognition, they still struggle when applied to certain tasks, such as training for low-resource languages.

Conventional LSTM and bidirectional LSTM (BLSTM) require complex training mechanisms which make them difficult to implement. Some of these mechanisms, such as clipping of cell activations and peephole connections, require careful tuning to a particular training set. The vanishing gradient problem across multiple layers is also a potential problem. We hope to solve these shortcomings using additional gating mechanisms that further constrain temporal dependencies and focus the training process.

In this paper, we aim at building advanced RNN structures with a hybrid acoustic model, to maximize use of prior knowledge in the speech signals. The solutions we propose include local window BLSTM, gated recurrent units, and residual architecture-based models. This work expands on our previous work [18–21]. Experiments are carried out on the Babel benchmark datasets, for low resource keyword search evaluations. By using these techniques, we achieve 3 to 10% relative improvements over the corresponding DNN or LSTM baselines. In addition, the new models improve training time by a factor of more than 1.6 compared to conventional BLSTM models.

The remainder of the paper is organized as follows. Section 2 briefly introduces the baseline LSTM model. Section 3 describes the proposed approaches. We report our experimental results in detail in Section 4, including experimental setup, hyperparameters evaluation, and comparisons between selected datasets. Finally, conclusions and future work are outlined in Section 5.

2 Baseline LSTM System

We first give a brief introduction of the DNN and LSTM network structures used as a baseline. A DNN contains a series of hidden layers, which for speech applications is most commonly fully connected with sigmoid activation functions.

RNNs are a neural network framework with self-connections from the previous time step used as inputs. This structure allows the network to capture a dynamic history of information about input feature sequences and is less affected by temporal distortion. Due to these properties, RNN have performed better than traditional DNNs in large vocabulary speech recognition tasks. Although conventional RNNs have feedback connections in the hidden layers to model temporal correlations, this structure captures short-term dependencies much better than long-term dependencies due to the vanishing and exploding gradients in the Stochastic Gradient Descent (SGD) training process [11].

The LSTM RNN topology is an advanced network structure designed to model long-term dependencies while limiting the rate of gradient decay through a gating mechanism.

LSTM units were first introduced in [12]. A popular LSTM structure is shown in Fig. 1. The forward pass from x_t to h_t follows the equations:

$$g_t = \phi\left(W_{gx}x_t + W_{gh}h_{t-1} + b_g\right) \tag{1}$$

$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i\right) \tag{2}$$

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_f\right) \tag{3}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t \tag{4}$$

$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + W_{oc}c_t + b_o\right) \tag{5}$$

$$h_t = o_t \otimes \phi(c_t) \tag{6}$$

Here, x_t is the input, and h_{t-1} is the previous output of the LSTM. W_{*x}, W_{*h}, W_{*c}, and b_* are forward matrices, recurrent matrices, diagonal peephole connections, and biases for all gates, respectively. σ is the sigmoid function, ϕ is the hyperbolic tangent function, and $\otimes$ denotes element-wise multiplication. For convenience, we denote the above calculations as $h_t = \texttt{LSTM}(x_t, h_{t-1})$.

The LSTM uses gates to control information flow and effectively creates shortcut paths across multiple temporal steps. The key ideas behind the LSTM unit are the addition of a memory cell block to maintain temporal information and the use of non-linear activation gates to control both the information flow into the memory cell and the output of the unit. Each LSTM unit consists of one cell unit and four control gates. These gates control the input and output as well as the temporal extent of the memory cell through a forget gate. The memory cell itself can also directly control the gates. The LSTM units implement this by peephole connections from the memory cell to the gates to learn precise timing information.

As a hybrid acoustic model, the network is trained to predict HMM states using a forced alignment. For both networks, a softmax layer is added at the top of the recurrent layers to generate posterior possibilities. The output of the softmax layer provides an estimate of the posterior probabilities $P(s|o)$ for states s, with given features o. The output in the softmax layer is computed by

$$P(s|o) = \text{softmax}(W_s h_{\text{out}} + b_s), \tag{7}$$

where (W_s, b_s) is the connection weight matrix and bias vector for the softmax layer, and h_{out} is the output of the top recurrent layer.

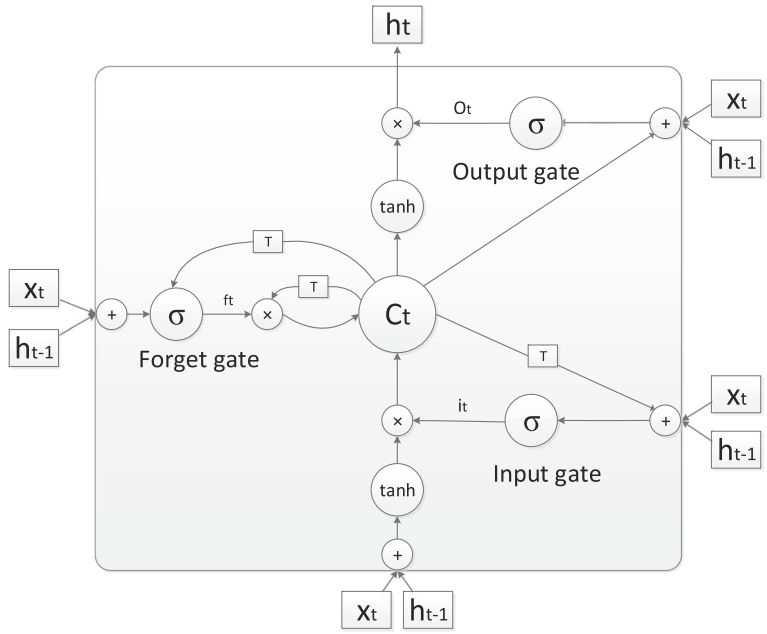

Fig. 1 Long short-term memory unit. "T" denotes a delay of one time step, σ denotes the sigmoid function, and "tanh" denotes hyperbolic tangent function

Inspired by DNNs, multiple LSTM layers can be stacked to build deep LSTM RNNs [9]. When input features propagate through the recurrent layer, the output features at each time step incorporate the history of temporal features from previous time steps. Compared to a shallow LSTM, the features generated by a deep LSTM are more generalizable and suitable for prediction. Thus, a deep LSTM RNN takes advantage of the merits of both DNNs and conventional LSTM.

Recently, linear recurrent projection layers have been proposed for reducing the number of parameters at no loss of accuracy [16]. Following this work, we use the term LSTM to denote such a deep LSTM-projected architecture and use this approach as our baseline.

3 Advanced recurrent architectures and training algorithms

Although LSTM models have achieved excellent performance in speech recognition tasks, they still have some shortcomings. For example, conventional unidirectional LSTMs only use preceding context information, which limits its ability to generate future context information. Furthermore, the architectures of traditional LSTM units are complex. This large size restricts their abilities to generalize and leads to a vanishing gradient problem across multiple layers. Because of the use of temporal context, training on entire sentence-length utterances requires extremely large training times, and the decoding real time factor is also poor. The proposed architectures and algorithms are designed to substantially reduce the above shortcomings.

In this section, we extend conventional LSTM and investigate in more depth the gated recurrent unit (GRU) element introduced in our previous work [18]. We also propose a new element called a residual GRU (rGRU) to alleviate the vanishing gradient problem across multiple layers.

3.1 Bidirectional LSTM

As discussed previously, the BLSTM [22, 23] is able to make use of both the preceding and following contexts within an utterance. The BLSTM does this by processing the data in both directions with two separate parameter sets, forward parameters and backward parameters.

The propagating output is calculated as follows:

$$\overrightarrow{h_t} = \overrightarrow{\texttt{LSTM}}(x_t, h_{t-1}), \tag{8}$$

$$\overleftarrow{h_t} = \overleftarrow{\texttt{LSTM}}(x_t, h_{t+1}), \tag{9}$$

where $h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right], t = 1 : L$, and L is the length of training sentences. (8) uses the forward parameters, while (9) uses the backward parameters.

After propagating in both forward and backward directions, the individual directional outputs are concatenated and fed forward to the next hidden layer. This bidirectional approach enables the system to more fully take advantage of the time dependencies of the input features.

3.2 Local window BLSTM

Although BLSTM networks often achieve better performance in speech recognition than LSTM, the latency during training and decoding is significant because it is necessary to wait until seeing a whole sentence. This makes BLSTM inappropriate for real-time online speech recognition.

However, in traditional backpropagation through time algorithm (BPTT) [24], the error signals are truncated based on the BPTT batch size parameter. This results in temporal isolation between blocks of the signal. The impact of the gradient becomes attenuated through these time steps, a process which is impacted by the batch size as well. For frames which are too far away from the current time, the impact of their gradient will be lower.

As we know, when considering the mechanism of pronunciation for human, the history phoneme information, including the shape changes of vocal cords, mouths, and noses, occupy the most impacts of sequential information. The future information play an auxiliary role, such as continuous changes of vocal organs. In addition, one of the successful key points behind the RNN is that they model the temporal relationships over phonemes. In general, the relationships between phonemes are constrained within word boundaries. The future information within word boundaries may be most useful to help to learn the current phoneme information.

Inspired by above thoughts, we consider the time dependencies within a local window. Based on this concept, in this subsection, we introduce the local window BLSTM (LW-BLSTM) approach.

Figure 2 shows the illustration of LW-BLSTM. All time dependencies are considered within a fixed local window N. The length of local window N is $Len(N)$. These local windows are non-overlapping chunks. Training LW-BLSTM is same as for the BLSTM. When using a new chunk during training, the initial $\overrightarrow{h_0}$ directly uses the final $\overrightarrow{h}_{Len(N)}$ from the previous chunk of the same utterance. In contrast to the traditional BLSTM, we do not need to wait until seeing a whole sentence, so both training and decoding speed are accelerated significantly. Moreover, the computational resources used for training are reduced sharply. In particular, the GPU memory requirements are reduced by a factor of about 10. This enables us to train a larger number of chunks in parallel, which accelerates the training speed further.

Some prior work has investigated methods to reduce latency and speed up the training process of BLSTM. This includes context-sensitive-chunk BLSTM (CSC-BLSTM) [25] and latency-controlled BLSTM (LC-BLSTM) [26]. Figure 2 shows the differences among these approaches. Comparing to the CSC-BLSTM approach, the LW-BLSTM approach introduced here incorporates the entire past history by using the final hidden states as the initial condition for the next block. This may lead to a more accurate approximation and be one of the key points behind the LW-BLSTM. Compared to LC-BLSTM, we do not distinguish truncated future context and preceding context, relative to a fixed local window. This may lead to fewer backward frames but avoids the potential issue of having the appended frames generate no output.

3.3 Gated recurrent units and LW-BGRU

Although LSTM RNNs have achieved excellent results, this architecture has some weaknesses. The architecture has a large number of parameters and can overfit relatively easily, especially for low resource tasks. In addition, training requires several complex mechanisms such as nonlinear clipping operations on cell activations and peephole connections [16] which may make it difficult to tune the parameters. To address these problems, we will adopt GRUs, another type of recurrent unit.

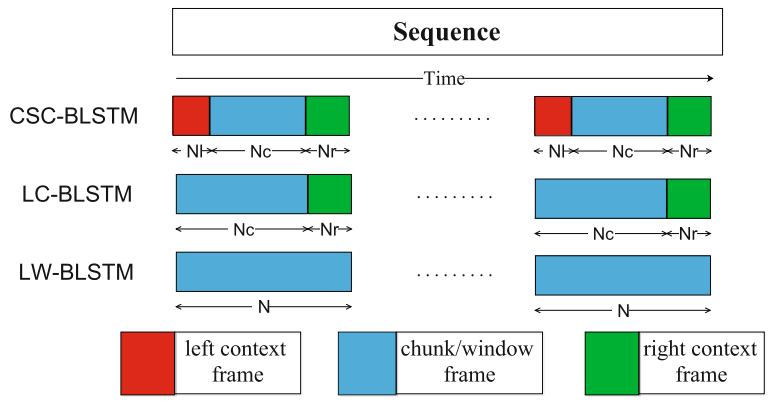

Fig. 2 Illustration of local window BLSTM (LW-BLSTM)

The GRU was recently proposed by Cho et al. [27]. Like LSTM, it was designed to adaptively reset or update memory content. As is shown in Fig. 3, each GRU has a reset gate and an update gate, which control the memory flow. The GRU fully exposes its memory content at each time step and balances output between the previous memory state and the new candidate memory state.

The GRU reset gate r_t is computed by

$$r_t = \sigma(W_{rx}x_t + U_{rh}h_{t-1} + b_r), \tag{10}$$

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

Similarly, the update gate z_t is computed by

$$z_t = \sigma(W_{zx}x_t + U_{zh}h_{t-1} + b_z), \tag{11}$$

where the parameters are as above.

Next, the candidate memory state m_t is calculated by

$$m_t = \phi(Wx_t + U(r_t * h_{t-1}) + b), \tag{12}$$

where ϕ is the hyperbolic tangent function and * denotes element-wise multiplication.

Lastly, the output of the GRU is calculated by

$$h_t = z_t * h_{t-1} + (1 - z_t) * m_t. \tag{13}$$

From the above propagating procedures, we can see that both GRU and LSTM use gates to control information flow and effectively create shortcut paths across multiple temporal steps. These gates and shortcuts help to detect and obtain the existence of an important feature in the input sequence. In addition, they allow the error to be backpropagated easily, thus reducing the difficulty due to vanishing or exploding gradients with respect to time [11].

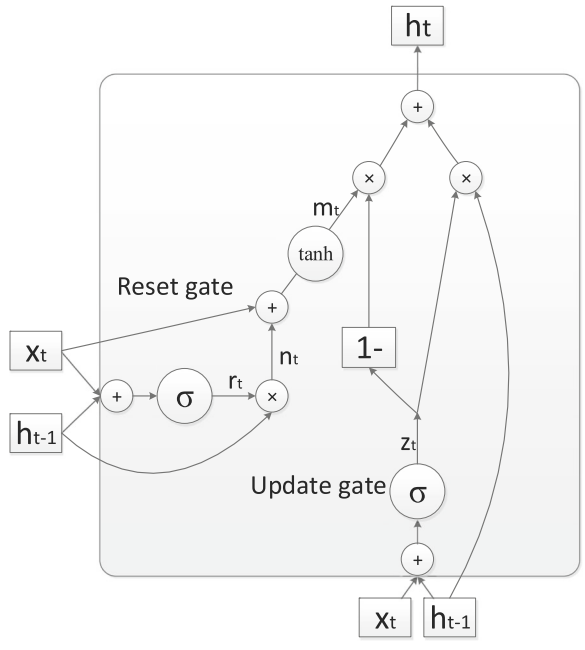

Fig. 3 Gated recurrent unit. σ denotes the sigmoid function, "tanh" denotes hyperbolic tangent function

The update gate helps the GRU to capture long term dependencies and plays a role like that of the forget gate in LSTM. The reset gate helps the GRU to reset whenever the detected feature is not necessary anymore. So when the GRU tries to learn temporally changed features, these gates activate differently.

The main difference between LSTM units and GRUs is that there is no output activation function or output gate to control the output in a GRU. Intuitively, because the output may be unbounded, this could hurt performance significantly. However, experimental results show that this is not true for GRUs, perhaps because coupling the reset gate and update gate avoids this problem and makes the use of an output gate or activation function less valuable [28–30]. Further, because an output gate is not used in GRUs, the total size of GRU layers is smaller than that of LSTM layers, which helps the GRU network avoid overfitting.

3.4 Residual BLSTM and residual BGRU

In LSTM or GRU architectures, a sigmoid function or hyperbolic tangent function is chosen as the nonlinear activation. These bounded functions may accentuate the vanishing gradient problem because it is easy for the gradients to become small when the error signal passes through multiple layers. This issue has attracted attentions of researchers in the machine learning community.

In recent years, some novel architectures, like residual net [31] and highway networks [32] have introduced an additional spatial shortcut path from lower layers for efficient training of deep networks with multiple layers.

The residual network approach [31] was successfully applied to train more than 100 convolutional layers for image classification and detection. The key insight in the residual network is the inclusion of a shortcut path between layers that can be used for an additional gradient path. The highway network [32] is an another way of implementing a shortcut path in a feed-forward neural network. Highway LSTM [33] is a recurrent version of highway network. This approach reuses shortcut gradient paths in the temporal direction for a highway shortcut in the spatial domain. Highway connections are used between internal memory cells instead of output layers. A new gate network was also introduced to control highway paths from the prior layer memory cells.

Inspired by these approaches, we propose the bidirectional residual LSTM and GRU (BrLSTM and BrGRU). These new architectures combine the merits of both bidirectional RNN and residual networks. In detail, (1) ∼ (5)

and (10) ∼ (12) do not change, while (6) is updated as follows:

$$h_t = o_t \otimes \phi(c_t) + W_{hx}x_t \tag{14}$$

Similarly, (13) is updated as follows:

$$h_t = z_t * h_{t-1} + (1 - z_t) * m_t + + W_{hx}x_t \tag{15}$$

There has been some similar recent work in this area, notably residual LSTM [34] and highway LSTM [33]. In contrast to residual LSTM, we add the residual directly to the output of the LSTM units, while [34] add the residual part to the hidden states before the projection functions. In addition, we apply the residual part to the new local window BLSTM and BGRU. In contrast to the highway LSTM, our methods use an output layer for the spatial shortcut connection instead of an internal memory cell, which reduces interference with a temporal gradient flow.

4 Experimental results and discussion

4.1 Data corpus

In order to evaluate our model, we implement experiments on a series of low resource speech recognition task, OpenKWS.

Since 2013, the National Institute of Standards and Technology (NIST) has conducted a series of keyword search evaluations called OpenKWS [35]. It is a part of the IARPA Babel program. These evaluations try to build a high-performance automatic speech recognition system for keyword search tasks. The data for the IARPA Babel program consists of conversational telephone speech from 25 languages. During the evaluations every year, an unknown and resource-limited surprise language is released, and participating teams are given only a short period of time to finish the task.

For each surprise language, the amount of training data is about 40 h. In addition to the training set, there are also tuning, development, and evaluation sets for each surprise language. The duration of the tuning set is about 5 h, while the development set contains about 10 h of transcribed data. The development set is for evaluation by the participants themselves. The evaluation set contains about 90 h of conversational speech. There is no pronunciation lexicon released. However, a language-specific peculiarities (LSP) document is available, which can help participating teams build the grapheme-to-phoneme lexicon.

We included seven different languages in our experimental works. These languages include Cantonese, Pashto, Vietnamese, Tamil, Swahili, Kazakh, and Georgian. Figure 4 shows all Babel languages and highlights our target languages, represented by white points.

4.2 Baseline setup

For these target languages, the pronunciation lexicons are processed based on LSP document information. The language model used for each language is a trigram, trained using just the transcript of the training data for each language and modified by Kneser-Ney smoothing.

We select two kinds of input features to train the individual models. The first is a vector of 40-dimensional Mel filterbank features concatenated with first and second order derivatives. For this inputs, a GMM-HMM is trained to generate targets using the Kaldi toolkit [36]. The

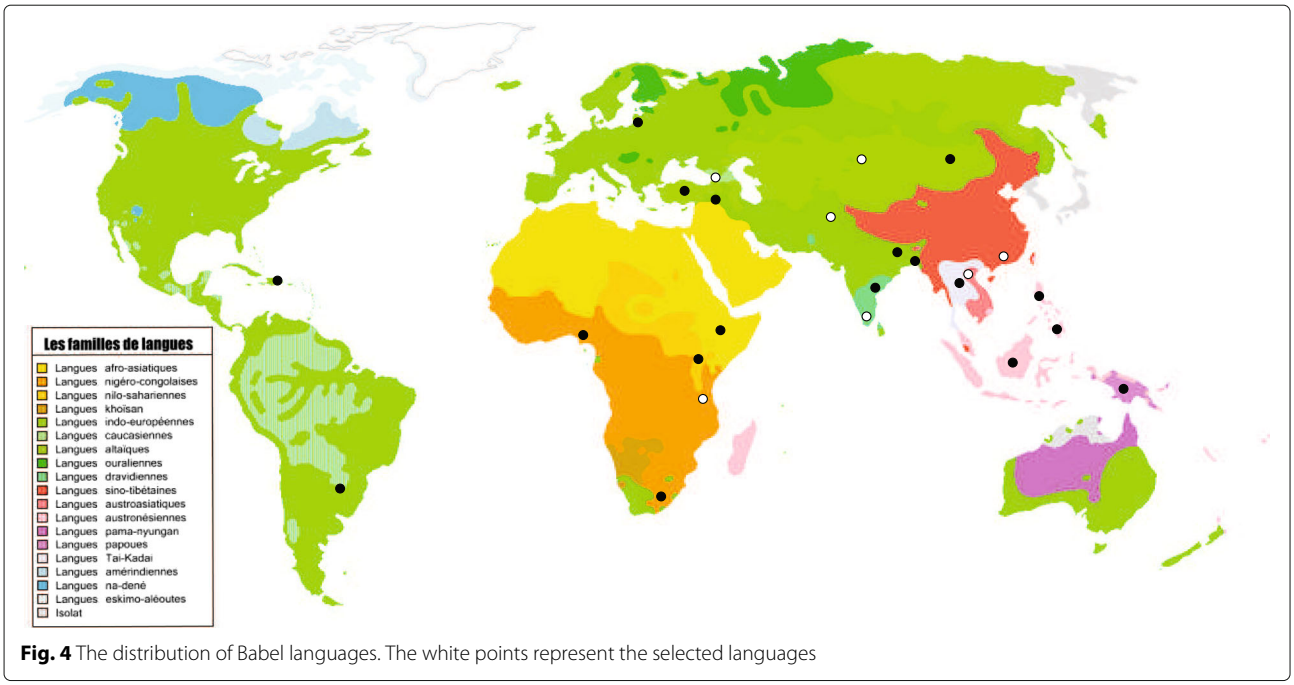

Fig. 4 The distribution of Babel languages. The white points represent the selected languages

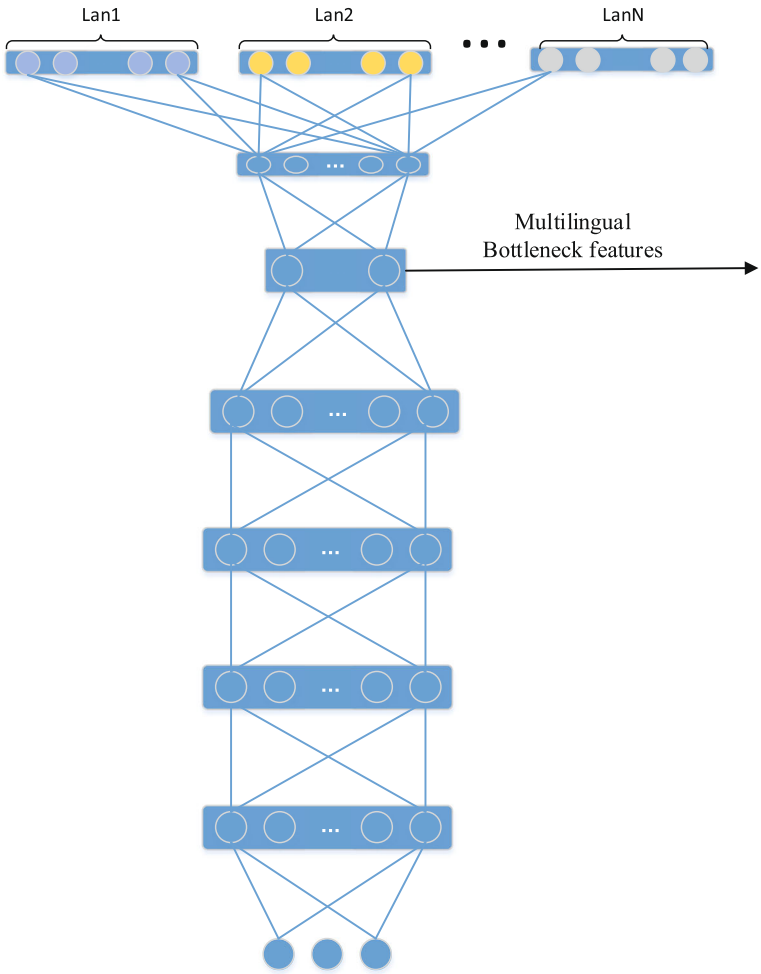

Fig. 5 The configuration of multilingual bottleneck extractor

GMM-HMM is first trained using 13-dimensional PLP features concatenated with 3-dimensional pitch features with zero means and unit variances. After that, LDA is implemented to reduce the feature dimensionality to 40. Next, the GMM-HMMs are trained by speaker adaptive training (SAT) and further enhanced by discriminative training using the boosted maximum mutual information (BMMI) criterion.

The second feature type is a vector of 128-dim multilingual bottleneck features [37]. Figure 5 shows the configuration of the bottleneck feature extractor. We use a six-layer TDNN [38] as the feature extractor. The splicing

Table 1 The corpus size and number of senones of the Babel languages

Language	Training hours	# senone	Language	Training hours	# senone	Language	Training hours	# senone
Cantonese(101)	140.7	4687	Assamese(102)	60.3	4707	Bengali(103)	61.1	4929
Pashto(104)	77.3	4823	Turkish(105)	76.4	4791	Tagalog(106)	83.8	4814
Vietnamese(107)	87.1	4692	Haitian(201)	66.5	4875	Swahili(202)	44.0	4638
Lao(203)	65.2	4667	Tamil(204)	64.1	4560	Kurdish (205)	41.7	4412
Zulu(206)	61.3	4464	Tok Pisin(207)	39.0	4565	Cebuano(301)	41.0	4603
Kazakh(302)	39.6	4714	Telugu(303)	41.8	4515	Lithuanian(304)	42.1	4755
Guarani(305)	42.6	4504	Igbo(306)	43.7	4659	Amharic(307)	43.2	4685
Mongolian(401)	45.7	4537	Javanese(402)	45.1	4763	Dholuo(403)	41.3	4571

The number in the () indicates the language id per the Babel program

Table 2 The baseline configuration of all languages, including vocabulary size, number of phonemes, and language model perplexity

Language	# words	# phonemes	LM perplexity
Cantonese	18512	216	124.87
Pashto	17646	126	217.55
Vietnamese	6210	375	176.99
Swahili	21890	75	435.74
Tamil	52369	34	656.51
Kazakh	19587	133	476.04
Georgian	34946	35	471.03

indexes used are $\{-2,-1,0,1,2\}$ $\{-1,2\}$ $\{-3,3\}$ $\{-7,2\}$ $\{0\}(*)$ $\{0\}$. The splicing indexes of $\{-2,-1,0,1,2\}$ indicate that the first layer sees five consecutive frames of input, and the $\{-1,2\}$ indicate that the second hidden layers see two frames of the previous layer, separated by three frames. All layers except the bottleneck layer contain 1024 neurons. The bottleneck layer is located at the fifth layer, denoted as $(*)$. The dimension of the bottleneck layer is 128. The original input to the TDNN feature extractor is the 40-dimensional Mel-filter bank features concatenated with 3-dimensional pitch features. To fully take advantage of available low resource languages, we include 24 languages from the IARPA Babel program dataset and use multitask learning methods [39] to train the bottleneck feature extractor. For each language in the 24 language set, a separate GMM-HMM is trained to generate the frame-level senone alignments as the targets of the feature extractor using the above methods. The details regarding corpus size and the number of senones for all 24 languages are listed in Table 1. The total duration of training data used to extract multilingual features is about 1400 h.

For this data, a separate GMM-HMM system is trained using the above multilingual bottleneck features of each target language.

A summary of baseline configurations, including the vocabulary size, the number of phonemes, and the language model perplexity, is given in Table 2.

We then build DNN and LSTM as the baseline acoustic models. The DNN consists of seven hidden layers with 1024 units per layer. The input feature is the multilingual bottleneck features or fbank features concatenated within a long context window of 11. We choose a sigmoid function for nonlinear activation. The parameters are randomly initialized with a normalized uniform distribution, and the DNN is pre-trained using DBN-based layer-wise pre-training. The DBN pre-training is performed with three epochs for each layer. The parameters are optimized using the CE criterion SGD algorithm. The batch size is 128 with an initial learning rate of 0.008. The initial learning rates are empirically tuned on the development set. At the end of every epoch, the learning rate is reduced by a factor of 2 if the frame accuracy on the development set drops.

The LSTM model contains three LSTM layers with 800 memory cells per layer, and each LSTM layer is followed by a low-rank linear recurrent projection layer of 512 units. The input features are the same as for the DNN. The target label is delayed for five frames. Training is performed using the truncated BPTT algorithm. Twenty frames are used in the BPTT training, and 20 sentences are trained in parallel. The initial learning rate is 0.0001, and the momentum is 0.9. The learning rate is controlled the same as DNN.

Discriminative learning methods [17, 40, 41] can help to improve the recognition performance. In order to achieve better performance, we select models using multilingual bottleneck features as inputs, then train them using the SMBR sequence training algorithm [17]. For fair comparison, we only show the cross-entropy WER for fbank input models, as in [21].

We use the Kaldi decoder with the beam set to 11.0, the lattice beam set to 8.0, and the acoustic weight set to 0.083333 in all of our evaluation experiments. For Cantonese, the evaluation metric is character error rate (CER), while for other target languages, the metric is word error rate (WER). The results of the baseline models are given in Table 3. These results are comparable with some previous published researches [42–47].

4.3 BLSTM and LW-BLSTM

The results of the BLSTM and LW-BLSTM acoustic models are presented in this subsection. We first evaluate the impact of hyperparameters in BLSTM and LW-BLSTM

Table 3 WER (%) of the baseline models for all languages

Model	WER						
	101 Cantonese	104 Pashto	107 Vietnamese	202 Swahili	204 Tamil	302 Kazakh	404 Georgian
DNN-fbank	44.8	51.2	53.1	46.2	66.7	54.1	50.5
LSTM-fbank	40.7	50.5	47.8	42.5	65	52.9	48.9
DNN-MBN	36.1	44.2	44.7	38.9	61.3	48.8	45
LSTM-MBN	35.7	44.9	45	39.6	61.3	49	45.2

Table 4 BLSTM accuracy as a function of the number of layers

Model	# layers		
	2	3	4
BLSTM-fbank	48.9	48.3	48.5
BLSTM-MBN	43.9	43.8	44.8

for Pashto. Following this, we give the results using the optimal strategy for each language.

The input features for these individual acoustic models are as described above, namely 128-dimensional multilingual bottleneck features and 40-dimensional Mel filter banks with delta and delta-delta derivatives. The target labels are also as mentioned above. All parameters are randomly initialized. The training algorithm is the same as that for baseline LSTM. We use an initial learning rate of 0.00005 and the momentum of 0.9.

For BLSTM, we focus on two parameters. The first is the hidden layer size. The second is the number of hidden layer. For LW-BLSTM, we evaluate the impact of the length of local window *N*.

4.3.1 The hyperparameters in BLSTM

For testing the hidden layer size and the number of hidden layer, we fix the number of parallel sentences as 20. The results are shown in Tables 4 and 5. For testing the first parameter, we fix the hidden layer size as 500, i.e., 500 units per direction. For testing the second parameter, we fix the number of hidden layers as 3.

From the results, it can be seen that when the number of BLSTM layer is larger than 3 and the number of hidden units per direction is larger than 600, the results become worse. This may be because when BLSTMs are applied to low resource tasks, a large model size has negative impact on generative ability.

In addition, results show that the BLSTM with three hidden layers and 500 units per direction has the best performance, achieving a WER of 43.8 and 48.3%, which are better than the LSTMs. The strong baseline results show that the WER of the LSTMs are 44.9 and 50.5% for different input features. This represents a relative decrease in WER of about 3%.

4.3.2 LW-BLSTM hyperparameters

For testing the length of the local window *N*, we use the best network structure, i.e., containing three bidirectional recurrent hidden layers and 500 units per direction. The results are shown in Tables 6 and 7.

Table 5 BLSTM accuracy as a function of the hidden layer size

Model	# hidden layer size				
	400	500	600	700	800
BLSTM-fbank	48.6	48.3	48.7	49.2	49.5
BLSTM-MBN	44	43.8	43.9	44.5	44.8

Table 6 WER results vs. the length of the local window

Model	the length of local window				
	10	20	30	40	50
LW-BLSTM-fbank	48.8	48.3	48.3	48.3	48.3
LW-BLSTM-MBN	44.5	43.9	43.9	43.9	43.8

From the results, we can conclude that when we apply all time dependencies calculations within a fixed local window whose length is larger than 20–30, the WER does not change significantly. This may be because the effective time dependencies last more than 20–30 frames but do not affect frames too far away. The farther the distance between two frames, the less they influence each other. So the time dependencies LW-BLSTM learn are not much less than that of the traditional BLSTM.

Apart from the WER, another matter to take into consideration is the training time. Compared to BLSTM, one potential benefit of LW-BLSTM is that it may be able to reduce the training and decoding time. To compare the training time, all experiments have been implemented on a single NVIDIA Tesla K80 using CUDA [48].

From the training time data presented in Table 7, we can see that the LW-BLSTM approach improves the training time by a factor of more than 1.6 compared to the baseline BLSTMs.

Another interesting note is when the length of local window is larger than 20–30, the WER results become stable. In order to explore the reasons, we use above word boundary thoughts, analysing the average duration of phonemes, the average number of phonemes in words for all seven languages. For agglutinating languages, we also analyse the average number of phonemes in morphemes.

By using the linguistic information in Table 8, for a local window containing 20–30 frames, considering context impact, it contains about three to four phonemes, which are almost equal to the the average number of phonemes in words or morphemes. We think a local window containing 20–30 frames has a complete word boundary, so it can learn the temporal dependencies within a whole word. It may be the reason why when the length of the local

Table 7 Training time per epoch (hours) vs. the length of local window

Model	The length of local window				
	10	20	30	40	50
LW-BLSTM-fbank	3.92	4.71	5.03	5.34	5.68
BLSTM-fbank			8.40		
LW-BLSTM-MBN	3.92	4.71	5.01	5.34	5.70
BLSTM-MBN			8.40		

Table 8 The linguistic information of all languages, including the average duration of phonemes and the average number of phonemes in words

Language	The average duration of phonemes	The average number of phonemes in words
Cantonese	14.01	3.92
Pashto	9.23	6.39(3.08)
Vietnamese	13.27	2.83
Swahili	12.77	7.54(3.44)
Tamil	11.02	9.30(3.87)
Kazakh	9.65	7.12(3.63)
Georgian	12.41	8.47(6.39)

The number in the () indicates the average number of phonemes in morphemes for agglutinating languages

window is 20–30, the LW-BLSTM can achieve nearly the same performance as the traditional BLSTM.

4.3.3 Results for all languages

Although the BLSTM and LW-BLSTM models give good results for Pashto, it is important to see how these models perform for other languages as well. We conduct experiments for all seven languages using the best network structure as described previously.

The results in Tables 9 and 10 show the BLSTM and LW-BLSTM model results across a variety of different languages. Compared with the baseline LSTM models, the relative improvements of the BLSTMs are about 3.0% for both fbank input features and bottleneck input features. The results also show that the BLSTMs with a local window have nearly the same performance as BLSTM, while reducing the training time significantly, confirming the discoveries in the previous experiments.

4.4 LW-BGRU and LW-BrGRU

In these experiments, we briefly evaluate the impact of hyperparameters in LW-BGRU and apply these two architectures to all seven languages.

4.4.1 LW-BGRU and LW-BrGRU hyperparameters

We now investigate the optimal network structure, number of hidden layers, and hidden layer size, fixing the local window length as 20. The results are shown in Table 11.

The results show that the average performance of LW-BGRU models containing two GRU layers are better than those containing three GRU layers. The results become worse when the layer size per direction is larger than 700. These phenomena are similar to the LW-BLSTM, i.e., the large models may hurt the generation properties for low-resource languages. In addition, we find that the average performance is better when LW-BrGRU models contain more layers. This may be because the residual architectures learn more information from the original signals and help to avoid the gradient vanishing problems across multiple layers.

In addition, the best result of LW-BrLSTM is 43.5 on Pashto and is achieved when model contains three hidden layers and 500 units per direction. As the phenomena are coincident with LW-BrGRU, we do not compare the parameters duplicately.

4.4.2 Results for all languages

The results are shown in Table 12. From the results, we can confirm that the GRU-based systems outperform the LSTM-based systems. The LW-BGRU models decrease the WER by about 7% relative to the baseline LSTM model, 5% relative to the baseline LSTM model, and about 3% relative to the LW-BLSTM. For the LW-BrGRU, these are 8, 6, and 4%, respectively. The reasons why GRU-based models are better than LSTM-based models for low resource tasks may be because the GRU-based systems can more easily avoid overfitting, and the more compact architectures are easier to tune.

4.5 Experimental results for different models

From the above results, all bidirectional recurrent systems achieve better performance than traditional feed-forward systems and unidirectional recurrent systems, because they yield more diversity of time dependencies and more fully take advantage of the sequential features. In addition, the GRU architecture and residual architecture learn more precise information. Due to these merits, our new models achieve excellent performance in a series of low resource tasks, OpenKWS. There has been some previous research focusing on low-resource speech recognition tasks. The architectures in this work make use of the prior knowledge of different aspects of speech signals. These improvements over the traditional CNN and RNN can help to learn more precise time domain or frequency domain information. For example, in [21],

Table 9 WER (%) of the BLSTMs and LW-BLSTMs for all languages

Model	WER						
	101 Cantonese	104 Pashto	107 Vietnamese	202 Swahili	204 Tamil	302 Kazakh	404 Georgian
BLSTM-fbank	39.5	48.3	45.8	41	63.7	50.3	46.6
LW-BLSTM-fbank	39.6	48.3	45.9	41.1	63.7	50.2	46.7
BLSTM-MBN	34.7	43.8	43.6	38	60.6	47.8	44
LW-BLSTM-MBN	34.7	43.9	43.7	38	60.6	47.7	44

Table 10 Training time per epoch (hours) of BLSTMs and LW-BLSTMs for all languages

Model	Training time per epoch (h)						
	101 Cantonese	104 Pashto	107 Vietnamese	202 Swahili	204 Tamil	302 Kazakh	404 Georgian
BLSTM-fbank	12.79	8.40	7.69	4.92	7.00	4.85	4.88
LW-BLSTM-fbank	8.01	4.71	4.83	3.06	4.37	3.03	3.04
BLSTM-MBN	12.78	8.40	7.67	4.93	7.01	4.85	4.87
LW-BLSTM-MBN	8.01	4.71	4.83	3.06	4.37	3.03	3.03

the convolutional maxout neural networks (CMNN) and recurrent maxout neural networks (RMNN) use local spectral properties within frames and long-term dependencies among frames. In our experiments, we compare the results achieved by our models to these already strong baselines of these other models.

All WER results of all acoustic models for all languages are listed in Table 13. Except for the conclusions mentioned above, we find that our new models achieve better performance than CNN, CMNN, and RMNN approaches. The WER of the LW-BLSTM are 5.2 to 12.5% lower relative to the CNN and 2 to 8% relatively lower than the CMNN. For LW-BGRU, the WER decrease is about 10.6, 5.9, and 1.3% relative to CNN, CMNN, and RMNN, respectively. For LW-BrGRU, these three values are 11.7, 7.0, and 2.1%. This indicates that our advanced recurrent network-based models achieve excellent performances compared to the traditional DNN, RNN, and CNN as well as other advanced models like CMNN and RMNN. Compared to RMNN, our advanced recurrent network-based models are benefit from local window architectures, advanced recurrent units, while RMNN are benefit from maxout no-bounded nonlinear activation, so we think our advanced recurrent networks are better than RMNN can be due to the impact of advanced recurrent units. Compared to CNN and CMNN, our networks are based on recurrent architectures, while CNN and CMNN are based on convolution architectures. Although in recent years, CNN-based models achieve excellent performances [49–51], the input to these models must be spectral features, such as filerbank features and spectrogram features. While for low resource speech recognition tasks, one of the most effective methods is using multilingual bottleneck features. Obviously, multilingual bottleneck features are not suitable for CNN models. Moreover, compared models that all use fbank features, our advanced recurrent network-based models are better than CNN and CMNN models. We think it is because RNN-based models are more suitable for low-resource speech recognition tasks.

Table 11 LW-BGRU and LW-BrGRU accuracy as a function of the number of layer and hidden layer size

# layers	Model	# hidden layer size				
		300	500	700	800	900
2	A	48.4	47.6	47.4	47.4	48
	B	48.1	47.3	47	47.1	47.8
	C	43.9	43.3	43.1	43.2	44.1
	D	43.7	43.1	42.7	42.9	43.5
3	A	48.2	47.7	47.9	48.5	48.9
	B	47.5	46.9	46.9	47.1	47.6
	C	43.8	43.5	43.7	44.3	44.9
	D	42.9	42.6	42.6	43.1	43.5

Here, "A", "B", "C," and "D" represent "W-BGRU-fbank," "LW-BrGRU-fbank," "LW-BGRU-MBN," and "LW-BrGRU-MBN," respectively

In addition, another interesting note is that the improvements of systems using MBN features are smaller than those of systems using fbank features. For example, the average WER of the LW-BGRU are 10.5% lower relative to the DNN when using fbank features, while only 3.8% lower when using MBN features. We believe that the reason behind this is because the MBN features use linguistic information of all 24 Babel languages and thus the features contain more information. This characteristic may not be noticeable when there are sufficient data resources; however, it has significant impact in low resource conditions. As we can see from the results, when MBN features are used, the performance of the DNN is better than that of the LSTM, which is contrary to what has been found for traditional speech recognition tasks [16]. When using MBN features, the merits of advanced acoustic models become smaller. Despite this, our new models still achieve more than 4% relative reduction in WER, which confirms that our new methods are well suited to low resource speech recognition tasks.

A further note is that the improvements among languages vary. This is likely due to different properties of the languages. For example, the vocabulary size of Tamil is much larger than that of other languages, which makes the speech recognition tasks for Tamil harder. Also, the amount of training data for Cantonese and Vietnamese is larger, so the improvements between models become smaller. Furthermore, the discriminative information between Georgian words is large, so it is easier to capture the ability of models.

Table 12 WER (%) of the LW-BrLSTMs, LW-BGRUs and LW-BrGRUs for all languages

Model	WER						
	101 Cantonese	104 Pashto	107 Vietnamese	202 Swahili	204 Tamil	302 Kazakh	404 Georgian
LW-BrLSTM-fbank	39.2	47.9	45.3	40.6	62.9	49.5	45.7
LW-BGRU-fbank	38.7	47.4	44.8	40	62.8	49	45.4
LW-BrGRU-fbank	38.5	47	44.3	39.5	62.2	48.5	44.1
LW-BrLSTM-MBN	34.4	43.5	43.1	37.4	60.1	47.2	43.3
LW-BGRU-MBN	34.1	43.1	42.7	37	59.7	46.7	42.9
LW-BrGRU-MBN	34.1	42.7	42.7	36.8	59.2	46.2	41.7

Other metrics we are interested in include the duration time of training and the decoding real-time factor (RTF). To measure these, we use the same configuration described above. These results are shown in Table 14. The training time per epoch for the LSTMs is about two or three times slower than the DNNs. This is because the non-linearity computation and recurrent calculation procedures in the recurrent structures are less efficient for GPUs. In addition, the training time for BLSTM is much slower than LSTM. This is because each sentence needs to be fully loaded during BLSTM training, which occupies more GPU cache and leads to much time delay. The high GPU cache occupation leads to fewer utterances being able to be trained in parallel, which further increases the training time. However, LW-BLSTM and LW-BGRU alleviate this problem. These approaches accelerate the learning speed by a factor of more than 1.6 compared to the conventional BLSTMs. The result is that the training time per epoch for the LW-BLSTMs or LW-BGRUs is about 1.5 times lower than the LSTMs and nearly equal to that of CNNs.

Finally, we combine the results of different systems. Since different models have different architectures and corresponding merits, they are adept in modeling different aspects of speech signals. When we combine these strong individual models, we may benefit from their complementary information.

Table 13 WER (%) across all languages

Model	WER						
	101 Cantonese	104 Pashto	107 Vietnamese	202 Swahili	204 Tamil	302 Kazakh	404 Georgian
DNN-MBN	36.1	44.2	44.7	38.9	61.3	48.8	45
LSTM-MBN	35.7	44.9	45	39.6	61.3	49	45.2
BLSTM-MBN	34.7	43.8	43.6	38	60.6	47.8	44
LW-BLSTM-MBN	34.7	43.9	43.7	38	60.6	47.7	44
LW-BrLSTM-MBN	34.4	43.5	43.1	37.4	60.1	47.2	43.3
LW-BGRU-MBN	34.1	43.1	42.7	37	59.7	46.7	42.9
LW-BrGRU-MBN	34.1	42.7	42.7	36.8	59.2	46.2	41.7
DNN-fbank	44.8	51.2	53.1	46.2	66.7	54.1	50.5
LSTM-fbank	40.7	50.5	47.8	42.5	65	52.9	48.9
BLSTM-fbank	39.5	48.3	45.8	41	63.7	50.3	46.6
LW-BLSTM-fbank	39.6	48.3	45.9	41.1	63.7	50.2	46.7
LW-BrLSTM-fbank	39.2	47.9	45.3	40.6	62.9	49.5	45.7
LW-BGRU-fbank	38.7	47.4	44.8	40	62.8	49	45.4
LW-BrGRU-fbank	38.5	47	44.3	39.5	62.2	48.5	44.1
CNN-fbank [21]	43.6	51.5	52.5	–	67.2	–	–
CMNN-fbank [21]	41.7	49.3	49.9	–	64.2	–	–
RMNN-fbank [21]	39	48.1	45.7	–	63.4	–	–
DNN + LW-BLSTM LW-BGRU + LW-BrGRU	33	41.5	41.3	35.7	58.2	44.6	40.6
DNN + LW-BLSTM LW-BrLSTM + LW-BGRU + LW-BrGRU	32.8	41.2	41	35.5	57.9	44.3	40.2

Table 14 Training time per epoch (hours) and decoding real time factor (RTF) of all models for all languages

Model	Training time per epoch (hours)							Decoding RTF
	101 Cantonese	104 Pashto	107 Vietnamese	202 Swahili	204 Tamil	302 Kazakh	404 Georgian	
DNN-fbank	2.35	1.32	1.22	0.73	1.04	0.69	0.70	1.65
LSTM-fbank	5.33	3.17	3.22	2.03	2.93	1.99	1.99	2.02
BLSTM-fbank	12.79	8.40	7.69	4.92	7.00	4.85	4.88	2.77
LW-BLSTM-fbank	8.01	4.71	4.83	3.06	4.37	3.03	3.04	2.31
LW-BrLSTM-fbank	9.08	5.26	5.47	3.40	4.99	3.43	3.48	2.41
LW-BGRU-fbank	7.10	4.11	4.21	2.75	3.83	2.69	2.72	2.27
LW-BrGRU-fbank	8.03	4.68	4.92	3.13	4.34	3.06	3.10	2.36
CNN-fbank [21]	7.07	4.61	4.22	–	3.70	–	–	–
CMNN-fbank [21]	7.09	4.62	4.30	–	3.13	–	–	–
RMNN-fbank [21]	4.78	2.93	3.00	–	2.19	–	–	–

We use the lattice-based system combination method proposed in [52], implemented in the Kaldi toolkit. We combine the top best individual models. The final WER results become stable when the number of individual models is above five. From the final results, the relative improvements of the combined results are relatively 1.7 to 3.5% better relative to the best LW-BrGRU models. This system achieved third place in OpenSAT 2017 Pilot Evaluation based on Pashto [53].

4.6 Visualization for different models

The final experiment, inspired by [54], investigates the evolution of different models when performing recognition. Following this approach, we draw the output of the last hidden layer using the t-SNE tool [55] when decoding an utterance. For these temporal traces, if they are smoother, the discriminative distribution of corresponding high dimension output data is stronger and the output confidence is higher. In addition, the amount of information contained in the output data depends on the number of traces. The results are shown in Fig. 6. The types of acoustic models are presented at the upper left corner of each subpicture.

Some interesting observations can be found from the figure. The temporal traces of LW-BGRU and LW-BLSTM are more concise and smooth than those of DNN and LSTM. This indicates that the LW-BGRU tends to remember more than DNN and LSTM. The novelty within a long span window leads to a smooth temporal trace. The output decision has higher confidence, so the temporal trace is more concise. In addition, the temporal traces of LW-BrGRU are more complex. We believe this indicates that LW-BrGRU retains more original feature information from lower layers.

In order to evaluate completely, we show the decoding results and reference of this utterance. From Table 15, we can find that the decoding result of LW-BrGRU is the most accurate. For low-resource speech recognition tasks,

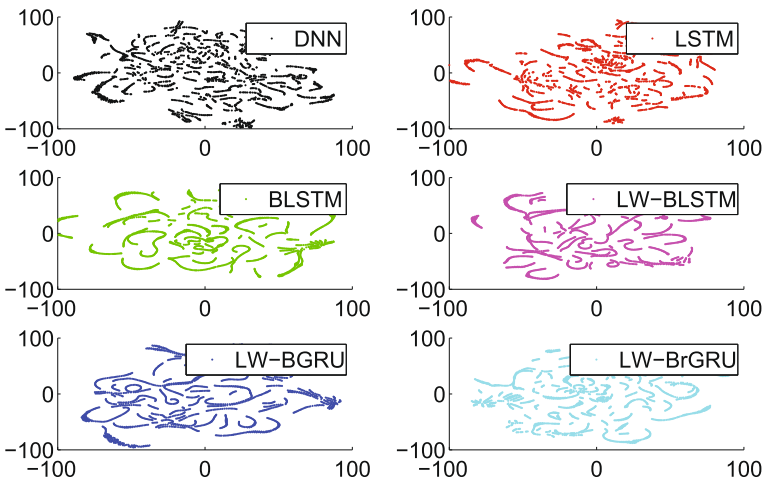

Fig. 6 The temporal trace of different acoustic models

Table 15 The decoding results and word error rates of different models

Model	Text (ins/del/sub percentage)
Reference	aa ilap'arak'e ravi biC'o sosom rom mankana daamt'vria da gamopKizlda aKla ra Kasiatzea
DNN	aa ilap'arak'e Kasiatzea (0/77/0)
LSTM	aa ilap'arak'e mankana aKla ra Kasiatzea (0/54/0)
LW-BLSTM	aa ilap'arak'e davT'ert masesKeo mankana daadgeba gamopKizlda aKla ra Kasiatzea (0/23/23)
LW-BGRU	aa ilap'arak'e mankana daamt'vria da gamopKizlda aKla ra Kasiatzea (0/31/0)
LW-BrGRU	aa ilap'arak'e ravi mankana daamt'vria da gamopKizlda aKla ra Kasiatzea (0/23/0)

The id of the utterance is 81424-B-20141123-000421-030633 for Georgian

the main part of errors is deleting error. It means that the real words are deleted because of the low acoustic and language scores. Deleting errors will become less if the output contains more helpful sequential information. So the LW-BrGRU and LW-BGRU learn more discriminative knowledge and achieve better performances.

5 Conclusions

In this work, we have proposed a local bidirectional RNN architecture, a new recurrent unit, gated recurrent units (GRU) as well as residual architectures, and combine these into a final system. Experiments are conducted on the benchmark datasets released under the IARPA Babel Program. Results show that the local window bidirectional RNN models decrease the WER about 3 to 8% relative to the baseline LSTM models and about 4 to 10% relative to the baseline DNN models across the selected seven target languages. In addition, the local window bidirectional RNN models decrease the learning time by a factor of more than 1.6 compared to conventional BLSTMs. In the future, we will extend these architectures with other useful components, such as convolutional architectures.

Abbreviations

ASR: Automatic speech recognition; BLSTM: Bidirectional long short-term memory; BNF: Bottleneck feature; GRU: Gated recurrent unit; LSTM: Long short-term memory; LW-BLSTM: Local-window bidirectional long short-term memory; LW-BGRU: Local-window bidirectional gated recurrent unit; RNN: Recurrent neural network; WER: Word error rate

Acknowledgements

This work is supported by National Natural Science Foundation of China under Grant No. 61370034 and No. 61403224.

Availability of data and materials

The datasets used or analysed during this paper are available from Babel program.

Authors' contributions

JK carried out these approaches, implemented the experiments, and finished this article. WQZ conceived of the study and participated in its design and coordination. WWL participated in implementation of the experiments and performed the statistical analysis. JL participated in the design of the study. MJ conceived of the study and helped to draft the manuscript. All authors read and approved the final manuscript.

Authors' information

Jian Kang received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013. He is currently working toward the Ph.D. degree in Tsinghua University, Beijing, China, and working on low-resource speech recognition. His current research interests include low-resource speech recognition and robust speech recognition.
Wei-Qiang Zhang was born in Hebei, China, in 1979. He received the B.S. degree in applied physics from University of Petroleum, Shandong, in 2002, the M.S. degree in communication and information systems from Beijing Institute of Technology, Beijing, in 2005, and the Ph.D degree in information and communication engineering from Tsinghua University, Beijing, in 2009. From 2016 to 2017, he was a visiting scholar at the Center for Computer Research in Music and Acoustics (CCRMA), Stanford University. He is an Associate Professor at the Department of Electronic Engineering, Tsinghua University, Beijing. His research interests are in the area of audio and speech signal processing, statistical pattern recognition machine learning.
Wei-Wei Liu received the B.S. degree in communication engineering from Xidian University, Xi'an, China, in 2003, the M.S. degree in communication engineering from National University of Defense Technology, Changsha, China, in 2006, and the Ph.D. degree in the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2015. Her research focuses upon speech processing and machine learning.
Jia Liu received his B.S., M.S., and Ph.D. degrees in communication and electronic systems from Tsinghua University, Beijing, China, in 1983, 1986, and 1990, respectively. He worked at the Remote Sensing Satellite Ground Station, Chinese Academy of Sciences, after his Ph.D., and worked as a Royal Society visiting scientist at Cambridge University Engineering Department during 1992–1994. He is now a professor in the Department of Electronic Engineering, Tsinghua University. His research fields include speech recognition, speaker recognition, language recognition, expressive speech synthesis, speech coding, and spoken language understanding.
Michael T. Johnson is currently a Professor and Chair of Electrical and Computer Engineering at the University of Kentucky in Lexington, KY. He received a Ph.D. from Purdue University in 2000, as well as an M.S. degree in Electrical Engineering from the University of San Antonio, TX in 1994 and B.S. degrees in Computer Science Engineering and Engineering with Electrical Concentration from LeTourneau University, Longview, TX in 1989 and 1990. He was a visiting professor at Tsinghua University in Beijing, China, in 2008–09 and 2014–15. His primary research area is speech and signal processing, with interests in articulatory kinematics, bioacoustics, and machine learning.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic, Engineering, Tsinghua University, Beijing 100084, China. [2]62315 Unit, Chinese People's Liberation Army, Beijing 100842, China. [3]Electrical and Computer Engineering, College of Engineering, University of Kentucky, Lexington, KY, USA.

Received: 25 October 2017 Accepted: 19 June 2018
Published online: 17 July 2018

References

1. G Hinton, L Deng, D Yu, GE Dahl, A-r Mohamed, N Jaitly, A Senior, V Vanhoucke, P Nguyen, TN Sainath, et al, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Proc. Mag. **29**(6), 82–97 (2012)

2. F Seide, G Li, D Yu, in *Proc. Interspeech*. Conversational speech transcription using context-dependent deep neural networks, (Florence, 2011), pp. 437–440
3. F Seide, L Gang, C Xie, et al, in *Proc ASRU*. Feature engineering in context-dependent deep neural networks for conversational speech transcription (IEEE, Waikoloa, 2011), pp. 24–29
4. GE Dahl, D Yu, L Deng, A Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio, Speech, Lang. Process. **20**(1), 30–42 (2012)
5. T Robinson, M Hochberg, S Renals, in *Automatic speech and speaker recognition*. The use of recurrent neural networks in continuous speech recognition (Springer, Boston, 1996), pp. 233–258
6. T Mikolov, et al, in *Proc. Interspeech*. Recurrent neural network based language model (ISCA, Makuhari, 2010)
7. W Chao, Y Dong, S Watanabe, et al, in *Proc ICASSP*. Recurrent deep neural networks for robust speech recognition (IEEE, Florence, 2014), pp. 5569–5573
8. N Boulanger-Lewandowski, J Droppo, M Seltzer, et al, in *Proc ICASSP*. Phone sequence modeling with recurrent neural networks (IEEE, Florence, 2014), pp. 5417–5421
9. A Graves, A Mohamed, GE Hinton, in *Proc ICASSP*. Speech recognition with deep recurrent neural networks (IEEE, Vancouver, 2013), pp. 6645–6649
10. AL Maas, QV Le, et al, *Recurrent neural networks for noise reduction in robust ASR*. (Proc INTERSPEECH: ISCA, 2012), pp. 22–25
11. Y Bengio, P Simard, P Frasconi, Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
12. S Hochreiter, J Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
13. TN Sainath, et al, in *Proc ICASSP*. Convolutional, long short-term memory, fully connected deep neural networks (IEEE, South Brisbane, 2015), pp. 4580–4584
14. JT Geiger, et al, in *Proc INTERSPEECH*. Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling (ISCA, Singapore, 2014), pp. 631–635
15. A Senior, H Sak, I Shafran, in *Proc ICASSP*. Context dependent phone models for LSTM RNN acoustic modelling (IEEE, South Brisbane, 2015), pp. 4585–4589
16. H Sak, AW Senior, F Beaufays, in *INTERSPEECH*. Long short-term memory recurrent neural network architectures for large scale acoustic modeling, (2014), pp. 338–342
17. H Sak, O Vinyals, G Heigold, A Senior, E McDermott, R Monga, M Mao, Sequence discriminative distributed training of long short-term memory recurrent neural networks. Entropy. **15**(16), 17–18 (2014)
18. J Kang, WQ Zhang, J Liu, in *Proc ISCSLP*. Gated recurrent units based hybrid acoustic models for robust speech recognition (IEEE, Tianjin, 2016), pp. 1–5
19. J Kang, C Lu, M Cai, WQ Zhang, J Liu, in *Proc ICASSP*. Neuron sparseness versus connection sparseness in deep neural network for large vocabulary speech recognition (IEEE, Brisbane, 2015), pp. 4954–4958
20. C Meng, Lv Zhiqiang, L Cheng, et al, in *Proc ASRU*. High-performance Swahili keyword search with very limited language pack: the THUEE system for the OpenKWS15 evaluation (IEEE, Scottsdale, 2015), pp. 215–222
21. M Cai, J Liu, Maxout neurons for deep convolutional and LSTM neural networks in speech recognition. Speech Comm. **77**, 53–64 (2016)
22. M Schuster, KK Paliwal, Bidirectional recurrent neural networks. IEEE Trans. Sig. Process. **45**(11), 2673–2681 (1997)
23. A Graves, N Jaitly, Mohamed A-r, in *Proc ASRU*. Hybrid speech recognition with deep bidirectional LSTM (IEEE, Olomouc, 2013), pp. 273–278
24. A Graves, J Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5-6), 602–610 (2005)
25. K Chen, Q Huo, Training deep bidirectional LSTMm acoustic model for lvcsr by a context-sensitive-chunk bptt approach. IEEE/ACM Trans. Audio, Speech Lang. Process. (TASLP). **24**(7), 1185–1193 (2016)
26. Y Zhang, et al, in *Proc ICASSP*. Highway long short-term memory rnns for distant speech recognition (IEEE, Shanghai, 2016), pp. 5755–5759
27. K Cho, et al, On the properties of neural machine translation: encoder-decoder approaches. arXiv preprint arXiv: 1409.1259 (2014)
28. J Chung, C Gulcehre, K Cho, Y Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv: 1412.3555 (2014)
29. R Jozefowicz, W Zaremba, I Sutskever, in *Proc ICML*. An empirical exploration of recurrent network architectures (International Machine Learning Society, Lille, 2015), pp. 2342–2350
30. K Greff, RK Srivastava, J Koutník, BR Steunebrink, J Schmidhuber, LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017)
31. H Kaiming, et al, Deep residual learning for image recognition. Proceedings of the IEEE, 770–778 (2016)
32. Srivastava, R Kumar, K Greff, et al, Highway networks. arXiv preprint arXiv: 1505.00387 (2015)
33. JG Zilly, et al, Recurrent highway networks. arXiv preprint arXiv: 1607.03474 (2016)
34. K Jaeyoung, M El-Khamy, J Lee, Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. arXiv preprint arXiv: 1701.03360 (2017)
35. NIST, Kws16 keyword search evaluation plan. https://www.nist.gov/itl/iad/mig/openkws16-evaluation
36. D Povey, A Ghoshal, G Boulianne, L Burget, O Glembek, Goel N, M Hannemann, P Motlicek, Y Qian, P Schwarz, et al, in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. The Kaldi speech recognition toolkit (IEEE Signal Processing Society, 2011). number EPFL-CONF-.192584
37. J Cui, et al, in *Proc ASRU*. Multilingual representations for low resource speech recognition and keyword search (IEEE, Scottsdale, 2015), pp. 259–266
38. V Peddinti, D Povey, S Khudanpur, in *Proc INTERSPEECH*. A time delay neural network architecture for efficient modeling of long temporal contexts (ISCA, Dresden, 2015), pp. 2440–2444
39. JT Huang, L Jinyu, Y Dong, et al, in *Proc ICASSP*. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers (IEEE, Vancouver, 2013), pp. 7304–7308
40. D Povey, PC Woodland, in *Proc ICASSP*. Minimum phone error and I-smoothing for improved discriminative training (IEEE, Orlando, 2002), pp. 105–108
41. B Kingsbury, in *Proc ICASSP*. Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling (IEEE, Taipei, 2009), pp. 3761–3764
42. M Karafiat, F Grezl, M Hannemann, et al, in *Proc INTERSPEECH*. BUT BABEL system for spontaneous Cantonese (ISCA, Lyon, 2013), pp. 2589–2593
43. K Martin, et al, *BUT 2014 Babel system: Analysis of adaptation in NN based systems*. (ISCA, Singapore, 2014)
44. NF Chen, S Sivadas, BP Lim, et al, in *Proc ICASSP*. Strategies for Vietnamese keyword search (IEEE, Florence, 2014), pp. 4149–4153
45. A Tanel, et al, in *Proc ICASSP*. The 2016 BBN Georgian telephone speech keyword spotting system (New Orleans, IEEE, 2017), pp. 5755–5759
46. C Ni, et al, in *Proc ICASSP*. Efficient methods to train multilingual bottleneck feature extractors for low resource keyword search (IEEE, New Orleans, 2017), pp. 5650–5654
47. W Hartmann, et al, in *Proc ICASSP*. Analysis of keyword spotting performance across IARPA babel languages (IEEE, New Orleans, 2017), pp. 5765–5769
48. CUDA Nvidia, Programming guide (2010)
49. O Abdel-Hamid, A-r Mohamed, H Jiang, L Deng, G Penn, D Yu, Convolutional neural networks for speech recognition. IEEE/ACM Trans. Audio, Speech, Lang. Process. **22**(10), 1533–1545 (2014)
50. T Sercu, C Puhrsch, B Kingsbury, Y LeCun, in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. Very deep multilingual convolutional neural networks for LVCSR (IEEE, 2016), pp. 4955–4959
51. C Szegedy, W Liu, Y Jia, et al, in *Proc CVPR*. Going deeper with convolutions, (2015), pp. 1–9
52. H Xu, D Povey, L Mangu, J Zhu, Minimum Bayes Risk decoding and system combination based on a recursion for edit distance. Comput. Speech Lang. **25**(4), 802–828 (2011)
53. opensat. https://www.nist.gov/itl/iad/mig/nist-2017-pilot-speech-analytic-technologies-evaluation
54. Z Tang, Y Shi, D Wang, Y Feng, S Zhang, in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. Memory visualization for gated recurrent neural networks in speech recognition (IEEE, 2017), pp. 2736–2740
55. LVD Maaten, GE Hinton, Visualizing highdimensional data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)