Expert Systems With Applications 139 (2020) 112840

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Adaptive windows multiple deep residual networks for speech recognition

Toktam Zoughi[a], Mohammad Mehdi Homayounpour[a,*], Mahmood Deypir[b]

[a] Laboratory for Intelligent Multimedia Processing Department of Computer Engineering& Information Technology Amirkabir University of Technology, Tehran, Iran

[b] Faculty of Computer Engineering, Shahid Sattari Aeronautical University of Science and Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 12 July 2018
Revised 18 July 2019
Accepted 24 July 2019
Available online 29 July 2019

Keywords:
Speech recognition
Deep neural network
Adaptive windows convolutional neural network (AWCNN)
Multiple Residual Networks (MRes)
Adaptive windows convolutional neural network and Multiple Residual networks (AMRes)

ABSTRACT

The hybrid convolutional neural network and hidden Markov model (CNN-HMM) has recently achieved considerable performance in speech recognition because deep neural networks, model complex correlations between features. Automatic speech recognition (ASR) as an input to many intelligent and expert systems has impacts in various fields such as evolving search engines (inclusion of speech recognition in search engines), healthcare industry (medical reporting by medical personnel, and disease diagnosis expert systems), service delivery, communication in service providers (to establish the callers demands and then direct them to the appropriate operator for assistance), etc. This paper introduces a method, which further reduces the recognition error rate. In this paper, we first propose adaptive windows convolutional neural network (AWCNN) to analyze joint temporal-spectral features variation. AWCNN makes the model more robust against both intra- and inter-speaker variations. We further propose a new residual learning, which leads to better utilization of information in deep layers and provides a better control on transferring input information. The proposed speech recognition system can be used as the vocal input for many artificial and expert systems. We evaluated the proposed method on TIMIT, FARSDAT, Switchboard, and CallHome datasets and one image database i.e. MNIST. The experimental results show that the proposed method reduces the absolute error rate by 7% compared with the state-of-the-art methods in some speech recognition tasks.

1. Introduction

Automatic Speech Recognition (ASR) systems convert human speech or audio signals to text. Automatic speech recognition is used in intelligent applications such as interactive voice response (IVR) systems, spoken document retrieval and dictation tools. We need a reliable speech recognition system for these applications. Notable improvements have emerged in some automatic speech recognition applications. For example, discriminative training approaches including minimum classification error training (Juang, Chou, & Lee, 1997; McDermott, Hazen, Le Roux, Nakamura, & Katagiri, 2007), maximum mutual information estimation (Kapadia, Valtchev, & Young, 1993), and minimum phone error training (McDermott et al., 2007; Povey & Woodland, 2002) have been developed. Acoustic modeling techniques such as conditional random fields (CRFs) (Hifny & Renals, 2009), segmental CRFs (Zweig & Nguyen, 2010), and the Gaussian Mixture Model-Hidden Markov Models (GMM-HMMs) (Sha & Saul, 2006) have also been developed. In spite of these improvements, research for ASR in real-world conditions requires continuation.

Among these ASR approaches, GMM-HMMs are a popular approach. Most of large vocabulary speech recognition (LVSR) algorithms (Dahl, Yu, Deng, & Acero, 2012; Hinton et al., 2012; Mohamed, Hinton, & Penn, 2012) use Hidden Markov models (HMM) in their approaches. Researchers have used HMM for temporal modeling. In addition, researchers estimate the probabilities of HMM states using GMMs. GMMs estimate parameters such as initial state probability, posterior probabilities of an observation, and transition between HMM states in a method called GMM-HMM (Dahl et al., 2012; Hinton et al., 2012; Mohamed et al., 2012). GMM-HMM is the basic method for speech recognition (Robinson et al., 2002). Among their disadvantages, GMMs constrain the abilities of GMM-HMM (Mohamed et al., 2012). Acoustic modeling techniques like conditional random fields (CRFs) (Hifny & Renals, 2009), segmental CRFs (Zweig & Nguyen, 2010), and Hidden Conditional Random Fields (HCRF) (Sung & Jurafsky, 2009) are pro-

* Corresponding author.
E-mail addresses: tzoughi@aut.ac.ir (T. Zoughi), homayoun@aut.ac.ir (M.M. Homayounpour), mdeypir@ssau.ac.ir (M. Deypir).

https://doi.org/10.1016/j.eswa.2019.112840
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

posed as ways to overcome GMMs shortcomings; however, the results show no considerable improvement compared to GMM-HMM (Sha & Saul, 2006).

GMM computes state transition probabilities using a constant formula, which leads to a weak performance of GMM-HMM in adverse conditions and speaking variations (Ramesh & Wilpon, 1992; Russell & Cook, 1987). In this paper, we propose a method to tackle GMM problems.

Some HMM-based methods like HSMM and ESHMM produce a sequence of observations at each state (Ramesh & Wilpon, 1992; Russell & Cook, 1987) to model various statement lengths. ESHMMs use the probability density function of a state to model variations in phone duration (Yu, 2010). ESHMM replaces an HMM state with another HMM. Other methods are based on Segmental Conditional Random Fields (SCRF) (Kao & Nguyen, 2011). SCRF tries to model various statement lengths of different speakers by extracting features of statement length.

Deep Neural Networks (DNNs) are also a popular approach in many applications such as automatic speech recognition (ASR), text and image classification, and video action recognition (Lee, Grosse, Ranganath, & Ng, 2009; rahman Mohamed, Dahl, & Hinton, 2012; Taylor, Fergus, LeCun, & Bregler, 2010). Deep networks are more powerful tools for speech recognition than other networks. Inspiring the brain functionality and data analysis and using this in deep layers causes this success. The hybrid of Deep Neural Networks and HMM are used for acoustic modeling called DNN-HMM (Dahl et al., 2012; Hinton et al., 2012; Mohamed et al., 2012). These systems use DNNs to compute HMM state probabilities. These deep models receive a sequence of speech frames as input and provide posterior probabilities of HMM states as the output of models. DNN models have performed well for LVSR (Bourlard & Morgan, 1993; Dahl et al., 2012; Hinton et al., 2012; Mohamed et al., 2012; Sainath, Mohamed, Kingsbury, & Ramabhadran, 2013a; Sainath et al., 2015), image, and text classification tasks (Salakhutdinov, 2009; Salakhutdinov & Hinton, 2012). DNN-HMMs outperform other methods such as GMM-HMMs (Hinton et al., 2012) by a large margin in speech recognition. HMM reduces the time and complexity of speech recognition training process in LVSR. GMMs are inefficient to model the data which lie near or on a non-linear manifold since GMMs use a large number of diagonal Gaussians. However, some appropriate approaches like DNNs, which use back-propagation training algorithm, model these data by a few parameters.

Deep Bidirectional Long Short-Term Memory Recurrent Neural Network, when combined with HMM (DBLSTM-HMM), has been successfully used for ASR. The DBLSTM-HMM system outperforms the GMM-HMM system in LVSR. The DBLSTM-HMM system presents better results than those achieved by DNN-HMM on ASR. Graves et al. believe that this hybrid approach with DBLSTM is suitable for ASR and acoustic modeling (Graves, Jaitly, & Mohamed, 2013).

A recurrent neural network coupled with connectionist temporal classification (RNN-CTC) maps audio signal input to phoneme outputs. RNN is used to model temporal information and CTC is used as its loss function. The RNN-CTC model is learned end-to-end and performs well in ASR (Graves, Fernandez, Gomez, & Schmidhuber, 2006). The combination of CTC and RNN have proven useful in handwriting recognition (Graves, 2008; Graves, Mohamed, & Hinton, 2013a). Graves et al. proposed an approach in which deep bidirectional Long Short-term Memory RNNs are combined with CTC training for phoneme recognition (Graves, Mohamed, & Hinton, 2013b).

DNN-HMM takes advantage of neural network for discriminative classification and HMM for context learning. However, NN-HMM structures are highly depended on the HMM model which are gradually losing their ground in speech recognition. Therefore, HMM-independent methods like LSTM-CTC methods are more effective than HMM-dependent methods. LSTM can handle noise, distributed representations, and continuous values. LSTM does not need a priori choice of a finite number of states in contrast to HMMs.

Convolutional neural networks (CNNs) are another deep method to compute HMM state probabilities. CNNs are deep graphical models with many hidden layers. CNNs could learn deep representations of data, the same as the other deep networks (Abdel-Hamid et al., 2014; Chen, Mak, Leung, & Sivadas, 2014; Lee et al., 2009; Sainath et al., 2013a; Sainath et al., 2015). Some researchers combine convolutional neural networks (CNNs) with HMM for speech recognition called CNN-HMM (Sainath et al., 2013a; Sainath et al., 2015). In the real world, CNN-HMMs are superior to DNN-HMMs and GMM-HMMs methods (Abdel-Hamid et al., 2014; Chen et al., 2014; Lee et al., 2009). We discuss convolutional neural networks in Section 2.1. Recent research has used deep residual networks (Res-Nets), which show high performance for text and image recognition (Dong, Loy, He, & Tang, 2016; He, Zhang, Ren, & Sun, 2016; Lee & Kwon, 2017). They use Res-Net output, the same as CNN output, to compute HMM state probabilities for speech recognition. Deep learning automatically extracts effective representations from data, which makes use the complete information compared to raw features like MFCCs. However, Convolutional neural network architecture reduces the computational overhead over DNN structure.

Convolutional deep neural networks (Abdel-Hamid et al., 2014; Chen et al., 2014; Lee et al., 2009; Sainath et al., 2013a; Sainath et al., 2015) and deep residual learning networks (Dong et al., 2016; He et al., 2016; Lee & Kwon, 2017) are new networks used for image and speech recognition. Each CNN hidden layer contains one convolution and one pooling sub-layer. These two sub-layers construct a hidden layer. CNN hidden layers capture local information.

As shown in Hinton et al. (2012) and Bengio (2009), deeper models represent data structures more expressively than shallower models. Previous works (Bengio, 2009; Hinton et al., 2012; Sainath et al., 2013a) indicate that by increasing the DNN depth, model accuracy improves. As the number of DNN hidden layers increases, the number of parameters also increases. In such conditions, training a DNN with a large number of parameters would be very difficult to converge toward the appropriate answers. It is possible to use DNNs with millions of parameters by using DNN pre-training, GPU access, residual learning, and fast computers. These properties make it possible to use deeper DNNs, which were previously impossible to use (Hinton et al., 2012; Mohamed et al., 2012; Sainath, Kingsbury, Soltau, & Ramabhadran, 2013). When the dataset is small or we have a complicated deep model, and if we randomly initialize DNNs millions of parameters, DNNs would not converge to the correct result or would take a long time to converge. Pre-training and residual learning are helpful methods to train deep networks and are most essential for convergence in some models with very complex structures (Celikyilmaz et al., 2016; Erhan & Manzagol, 2009; Glorot & Bengio, 2010; Martens, 2010).

Considerable developments for solving speech problems have arisen in the last several decades (Kim, Hori, & Watanabe, 2017; Li, Sainath, Weiss, Wilson, & Bacchiani, 2016; Miao & Metze, 2017; Qian, Bi, Tan, & Yu, 2016; Zhang, Chan, & Jaitly, 2017b; Zhang et al., 2018). Researchers denote the shortcomings of acoustic models (Qian et al., 2016) and criticize standard modeling approaches (Li et al., 2016; Qian et al., 2016; Zhang et al., 2018). Most of the practical approaches for speech recognition accept existing structures and slightly reduce the error rate without significant changes in the overall structure. These approaches include adaptive techniques such as maximum likelihood linear regression (Leggetter & Woodland, 1995), vocal tract normalization (Lee & Rose, 1996; Welling, Kanthak, & Ney, 1999), discriminative training

(McDermott et al., 2007; Povey, 2003), and improving language modeling (Chen & Goodman, 1996). Availability of very large data and capability of using the entire data is very helpful. Most speech recognition systems require powerful computers and advanced searching techniques to perform a fast recognition task. The final goal of a speech recognition system is to achieve the recognition power of humans. In this way, we analyzed CNN and Res-Net systems and considered their shortcomings in modeling speech signals.

Automatic speech recognition can be used as a part of an expert system. For example an expert system used to diagnose diseases may receive speech signal explaining the symptoms of a patient's illness at the input and in response declare his/her type of illness. Or, for example, in an intelligent interactive voice response system or in a voice search engine, the input to the intelligent system is the user's speech which should be first recognized and then followed up.

Automatic speech recognition is a challenging task due to intra- and inter-speaker variations in speech signals. Moreover, information extracted from input data cannot spread well in deep networks. Res-Nets are among the recent deep networks architectures (He et al., 2016; Huang, Xu, Sun, & Yang, 2017) and are popular in speech and image processing tasks.

We present a new structure to model speech in automatic speech recognition task. We also present a modification of Res-Net called Multiple Residual networks (MRes), which is able to be stacked unboundedly deep. Deep residual leaning makes it possible to learn a very deep network without pre-training. We have improved the structure of the deep residual learning to better direct the weights toward the correct answer in a very deep model. This model incorporates input information into the networks top layers, which leads to additional benefits such as application to large vocabulary speech recognition (LVSR). Moreover, we present an adaptive window convolutional neural network (AWCNN) in the first layers of the model to deal with the speaker varieties. The objective of this model is to try to cope with variations in the speech signal due to changes both intra- and inter-speaker. In addition, MRes adds general speech and speaker information to the model by using Subspace Gaussian Mixture Model (SGMM). We call the combination of AWCNN and MRes as AMRes network. These aspects distinguish this work from previous uses of residual learning (Tan et al., 2018; Vydana & Vuppala, 2017; Wu, Zhong, & Liu, 2018; Zhang, Zuo, Chen, Meng, & Zhang, 2017a).

We illustrate the application of AMRes to ASR in detail and show how different AMRes configurations affect final ASR performance. Evaluations on TIMIT, FARSDAT phone recognition, Switchboard, CallHome word recognition, and MNIST handwritten digit recognition confirms that the proposed method improves accuracy in both speech and handwritten character recognition tasks. The results demonstrate that the proposed method significantly outperform state-of-the-art methods such as DNN-HMM, CNN-HMM, Residual LSTM, and Res-HMM for image and phone recognition. Throughout this paper and for all methods, the first layer _ or input layer _ is located at the bottom of the model.

We summarize the contributions of this paper as follows:

1. We propose a new method that changes the structure of CNN and Res-Net.
2. The proposed adaptive window convolutional neural network approach is a very easy-to-implement framework for speech recognition in deep models.
3. Use of input information is controllable in the proposed method. Thus, input information enters all layers to direct the hyperplanes and parameters of the model.
4. This method achieves high recognition performance over the state-of-the-art training approaches on the TIMIT, MNIST, FARSDAT, Switchboard, and CallHome datasets.

We organize the rest of the paper as follows: Section 2 reviews related works. Section 3 presents the proposed algorithm AMRes and discusses its properties. Section 4 presents results and comparisons with recent techniques. Finally, Section 5 concludes the paper.

2. Related works

Many algorithms are proposed for phone recognition tasks (Dahl et al., 2012; Hinton et al., 2012; Mohamed et al., 2012; Sainath et al., 2013a; Sainath et al., 2015). We review the most important and the most relevant algorithms to our work. Among these methods, Convolution Neural Network (CNN) and residual learning are the most relevant. In this paper vectors were denoted by small letters. A boldface letter refers to a vector not a single number. In addition the matrices were represented by capital and boldface letters. Vector sizes were depicted by capital letters and not in boldface.

2.1. Convolutional neural networks

Convolutional neural networks (CNNs) have an input layer and some hidden layers (Sainath et al., 2013a). As presented in Fig. 1, each CNN hidden layer is composed of one convolution sub-layer and a pooling sub-layer.

Matrix weights $\mathbf{w}_{i,j}$ $(i = 1, \ldots, I;\ j = 1, \ldots, J)$ connect each input feature $\mathbf{o}_i (i = 1, \ldots, I)$ to multiple neurons (units) $\mathbf{q}_j (j = 1, \ldots, J)$ in convolution sub-layers, where I and J are the total number of input and feature maps, respectively. The mapping in a convolution sub-layer is as follows:

$$q_{j,m} = \sigma\left(\sum_{i=1}^{I}\sum_{n=1}^{F} o_{i,n+m-1} w_{i,j,n} + \mathbf{w}_{0,j}\right),\ (j = 1, \ldots, J) \quad (1)$$

where $o_{i,m}$ is the m^{th} unit of i^{th} input feature. F is the filter size, which defines the number of frequency bands in mapping input features to the convolutional layer. Using the convolution operator, Eq. (1) can be rewritten as Eq. (2):

$$\mathbf{q}_j = \sigma\left(\sum_{i=1}^{I} \mathbf{o}_i * \mathbf{w}_{i,j}\right),\ (j = 1, \ldots, J) \quad (2)$$

The number of feature-maps in the convolution layer determines the number of the local weight matrix in the convolution sub-layer. CNN considers a window in which this window length is equal to the input frame number, and its width is F. The convolutional operation decreases the output of the convolution sub-layer dimension by a factor of F.

CNNs differ from DNNs in two ways. First, the convolution sub-layer receives its input from a local region. Thus, each unit in the convolution sub-layer represents local area features. Second, the units in hidden layers split into multiple feature-maps. Each unit in the same feature-map shares the same weights, which receives its input from different locations in the lower layer.

As shown in Fig. 1, the pooling operation is applied to the convolution sub-layer. The pooling sub-layer has some feature-maps. The number of feature-maps in this sub-layer is the same as those in the convolution sub-layer, but the size of feature-maps in the pooling sub-layer is smaller than that of the convolution sub-layer. The goal of the pooling sub-layer is to reduce feature-map size. Thus, the units in this sub-layer have more general features than the lower layers. This generalization applies to different frequencies, which causes CNN to be robust to minor changes in frequency

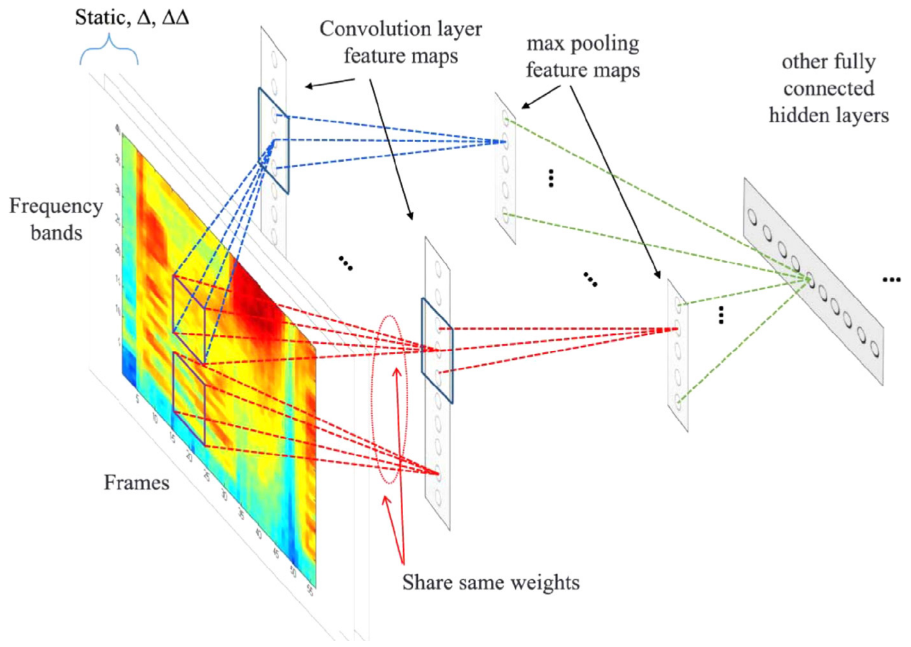

Fig. 1. A representation of Convolutional Neural Network (CNN), the convolution operator is applied along different frequency bands (Sainath et al., 2013a).

bands (Sainath et al., 2015). This reduction applies the pooling function to different local regions in convolution sub-layer. We can use the maximum or mean function for the pooling function. We apply the pooling function to each feature-map in convolution sub-layer. When the maximum function is used, the pooling sub-layer is defined as Eq. (3):

$$p_{i,m} = max \, {}^{G}_{n=1} q_{i,(m-1)} \times s + n \tag{3}$$

where G is the pooling size and s is the shift size (Sainath et al., 2013a).

2.2. Deep residual networks

Residual learning network (Res-Net) is a type of deep neural network. Deep residual networks are not fully connected networks. Res-Nets include shortcut connections. The overall architecture of Res-Nets is similar to convolutional neural networks (CNN). Both networks have one input layer and a number of hidden layers. Each hidden layer in both networks uses convolutional sub-layer. Each convolutional layer is composed of a convolution sub-layer and pooling sub-layer (Sainath et al., 2015). The main difference between CNNs and Res-Nets is due to their internal interconnections. Res-Nets have shortcut connections in addition to CNN connections (Tan et al., 2018; Vydana & Vuppala, 2017; Zhang et al., 2017a). Here, a shortcut is a skip connection to jump over some layers. In Res-Nets, some layers have a direct connection from their second previous layer. A simple solution for a shortcut connection is to use a linear layer (Vydana & Vuppala, 2017; Zhang et al., 2017a). Shortcut connections prevent vanishing or exploding gradients and thus improve classification performance. As Fig. 2 shows, each shortcut connection connects two layers but ignores one or more intermediate layers. Thus, in some layers the information is received directly from the previous layers. The proposed method extends this schema and adds some shortcut connections to receive generative information from the input layer.

2.2.1. Learning residual networks

Residual learning improves network learning (He et al., 2016). Residual learning is computed using Eq. (4):

$$\mathbf{y} = F(\mathbf{x}, \{\mathbf{w}_i\}) + \mathbf{x}, \qquad H(\mathbf{x}) = \mathbf{y} \qquad H(\mathbf{x}) - \mathbf{x} = F(\mathbf{x}) \tag{4}$$

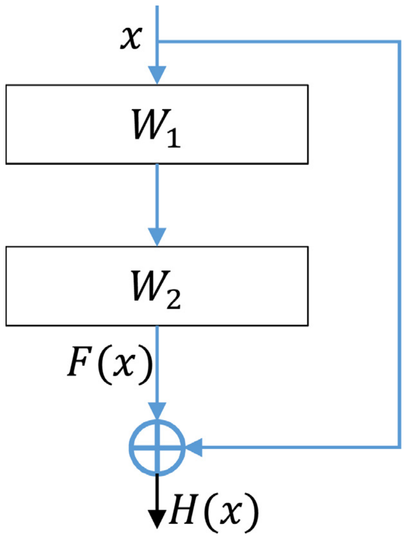

Fig. 2. Block diagram of a residual learning.

where, **x** and **y** are the input and output of the residual block diagram, respectively, as shown in Fig. 2. We consider $H(\mathbf{x})$ as a mapping of two stacked layers in which **x** is its input (Fig. 2). We expect the stacked layers to estimate the residual function $H(\mathbf{x}) - \mathbf{x}$ instead of approximating a function. The convolutional filter $\mathbf{w}_i$ maps input to output $(\mathbf{y} - \mathbf{x})$. He et al. (2016) prove that the optimization of $\mathbf{w}_i$ with residual mapping is easier than unreferenced mapping. The residual networks use ReLU as the activation function. The ReLU function causes non-linearity of the output layer.

2.3. Subspace Gaussian mixture model

Subspace Gaussian mixture model (SGMM) is used to provide a more efficient speech recognition system for low source languages (e.g. Persian and Arabic) by providing more speech and speaker information. SGMM improves multilingual speech recognition systems (Burget et al., 2010; Povey et al., 2010). The Subspace

Gaussian mixture model is defined as follows:

$$p(\mathbf{x}|j,s) = \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} N(\mathbf{x}; \mu_{jmi}^{(s)} \textstyle\sum_i) \tag{5}$$

$$\mu_{jmi}^{(s)} = \mathbf{M}_i \mathbf{v}_{jm} = N_i \mathbf{v}^{(s)} \tag{6}$$

$$w_{jmi} = \frac{exp(\mathbf{w}_i^T \mathbf{v}_{jm})}{\sum_{i'=1} exp(\mathbf{w}_{i'}^T \mathbf{v}_{jm})} \tag{7}$$

where j, I and w_{jmi} are phoneme, number of Gaussian in each state, and the weight of each Gaussian, respectively. Σ_i, $\mu_i^{(s)}$ are the covariance matrix and the Gaussian's mean, respectively. The vector $\mathbf{v}^{(s)}$ is defined for each speaker and c_{jm} is the weight of Gaussian mixture model in each state. The parameter N_i is a common parameter between speakers. If the parameters of SGMM and GMM are well tuned, the SGMM model will have fewer parameters than GMM model (Povey et al., 2010) thus this model is attractive. In SGMM an overall GMM named UBM is used for all HMM states in which general speech and speaker information is used to construct the model.

3. The overall AMRes architecture

Variations among different speakers, including difference in their vocal tract lengths and variation in environmental noises lead to considerable variations in speech signals. For better convergence, we require a network to model phone duration, precisely benefiting the input data and ferrying it all over in a very deep network.

In the proposed method, we use input data, speaker, and phone duration information to construct a comprehensive model. The proposed AWCNN considers better phone duration handling. By using SGMM we insert general speech and speaker information to the network. In addition, by using the proposed MRes network we insert the raw input layer to some layers with weighted connections. The main idea is quite simple but produces interesting results. Some ASR methods use information from two-dimensional time-frequency features obtained from successive frames of speech signal. Moreover, these features are fed to deep networks as input data, and the corresponding observation probabilities required from HMM states are acquired at the network output layer. The phone error rate is directly related to the information extracted by the network from the data. Since a network receives more effective information from input data, it produces better results.

In very deep neural networks, some input information is missed in deeper layers. Experimental results show that when we have a very deep network, the model will not converge to appropriate results. Thus, we should direct the model to the correct results. Here, we direct the layers with reference to the *previous information* and input information and propose a model called Multiple Residual neural network (MRes-Net). Fig. 3(a) and (b) show Res-Net and the proposed MRes-Net, respectively. As shown in Fig. 3(b), MRes-Net has connections all over the network. We control the combination of discriminative and generative information in this method. Thus, the lower layers receive more generative information while the higher layers receive more discriminative information.

3.1. MRes: Multiple Residual neural network

Fig. 3 (a) shows a Res-Net with 19 convolution layers. As shown in Fig. 3(a), in the standard form of Res-Net, each layer $\mathbf{h}^{lr}$ has a shortcut connection from layer $\mathbf{h}^{lr-2}$ because information from layer $\mathbf{h}^{lr-2}$ is useful for the current layer $\mathbf{h}^{lr}$, where lr and $\mathbf{h}$ are the layer number and hidden units, respectively. However, we believe that the first layer information, $\mathbf{h}^1$, also conveys valuable information. The effect of this information decreases as it moves toward deep layers within the networks. The input layer contains generative information, which is required in deeper layers. Using shortcut connections to flow the input information to the deeper layers is helpful. Fig. 3(b) shows an example of the proposed method, including Multiple Residual network (MRes-Net), which has 19 convolutional hidden layers. As shown in this figure, the first layer connects to the next layer. Moreover, some layers receive shortcut connections from the first layer output. Such connection is not a simple connection; it should be a weighted connection to control the flow of input information in deep layers. In other words, each shortcut connection from the first layer conveys information proportional to its own weight. Each layer $\mathbf{h}^{lr}$ in a one-in-between manner connects to $\mathbf{h}^1$ in addition to the $\mathbf{h}^{lr-2}$ layer. We call these two connections the *previous information*. Each of these two connections contain information to direct decision boundaries. $\mathbf{h}^1$ information directs the input information to the lower layers. These shortcut connections cause all one-in-between layers to receive *previous information* along with current information. They need this information to better analyze the input data in deeper layers. As Fig. 4 indicates, $F(\mathbf{x})$ uses two layers for the shortcut connection, which can use more layers.

Some phonemes are very similar to each other due to their category e.g., m and n as nasal or p and b as plosives. Therefore, without low-level information, the trained model cannot discriminate them in higher layers of the network. The proposed method uses low-level information in high-level neurons, which helps the network to recognize them better.

The third layer of MRes in Fig. 3(b) receives the same information twice. This situation only occurs for the third layer. We use two shortcut connections from the layers $\mathbf{h}^{(lr-2)}$ and $\mathbf{h}^1$ to the current layer $\mathbf{h}^{lr}$. However, these two layers ($\mathbf{h}^{(lr-2)}$ and $\mathbf{h}^1$) are equal for $lr = 3$ ($\mathbf{h}^{(3-2)}$ and $\mathbf{h}^1$ are equal). For the other layers, the $\mathbf{h}^{(lr-2)}$ and $\mathbf{h}^1$ would be different with each other, thus they do not contain the same information.

Another shortcoming of Res-Net is that each one-in-between layer, $\mathbf{h}^{lr}, \forall lr \in 2L+1$, sums its output with the second previous layer, $\mathbf{h}^{lr-2}$, but this summation is not a weighted summation (Dong et al., 2016; He et al., 2016; Lee & Kwon, 2017). In Res-Nets, one layer usually has a shortcut connection but the next layer does not. If the output of the current layer $\mathbf{h}^{lr}$ without shortcut connection is denoted by A, then the next layer with a shortcut connection from $\mathbf{h}^{lr-2}$ layer would produce a value near $2A$ as its output. As discussed in Section 4, this lack of balance causes a high error rate in the results. Therefore, as depicted in Fig. 4, the proposed method uses a weighted connection between the current layer $\mathbf{h}^{lr}$, second previous layer $\mathbf{h}^{lr-2}$, and $\mathbf{h}^1$. This weighted connection controls the model to produce the output values of each layer near A either in the layer that has a shortcut connection or does not have one. MRes defines its connections using Eq. (8):

$$H(\mathbf{x}) = \gamma(F(\mathbf{x})) + (1-\gamma)(\alpha^{lr}\mathbf{x}^1 + (1-\alpha^{lr})\mathbf{x}) \tag{8}$$

In Fig. 4, we obtain $F(\mathbf{x})$ using Eq. (9):

$$F(\mathbf{x}) = \mathbf{w}_2 \sigma(\mathbf{w}_1 \mathbf{x}) \tag{9}$$

In Eq. (8), $\mathbf{x}^1$ is the output of the first hidden layer (Fig. 4). Every two or three layers receive $\mathbf{x}^1$ as their input. $\mathbf{x}$ is the second previous layer output. $F(\mathbf{x})$ has two or three hidden layers but experiments show that using two layers is the best choice. The first hidden layer output $\mathbf{x}^1$, is multiplied by $(1-\gamma)\alpha^{lr}$ parameter to perform a weighed sum with the second previous layer, $\mathbf{x}$, which is multiplied by $(1-\gamma)(1-\alpha^{lr})$. Then, as indicated by Eq. (8), this *previous information* is aggregated with the current layer output, $\gamma F(\mathbf{x})$, to produce the next layer input using γ parameter.

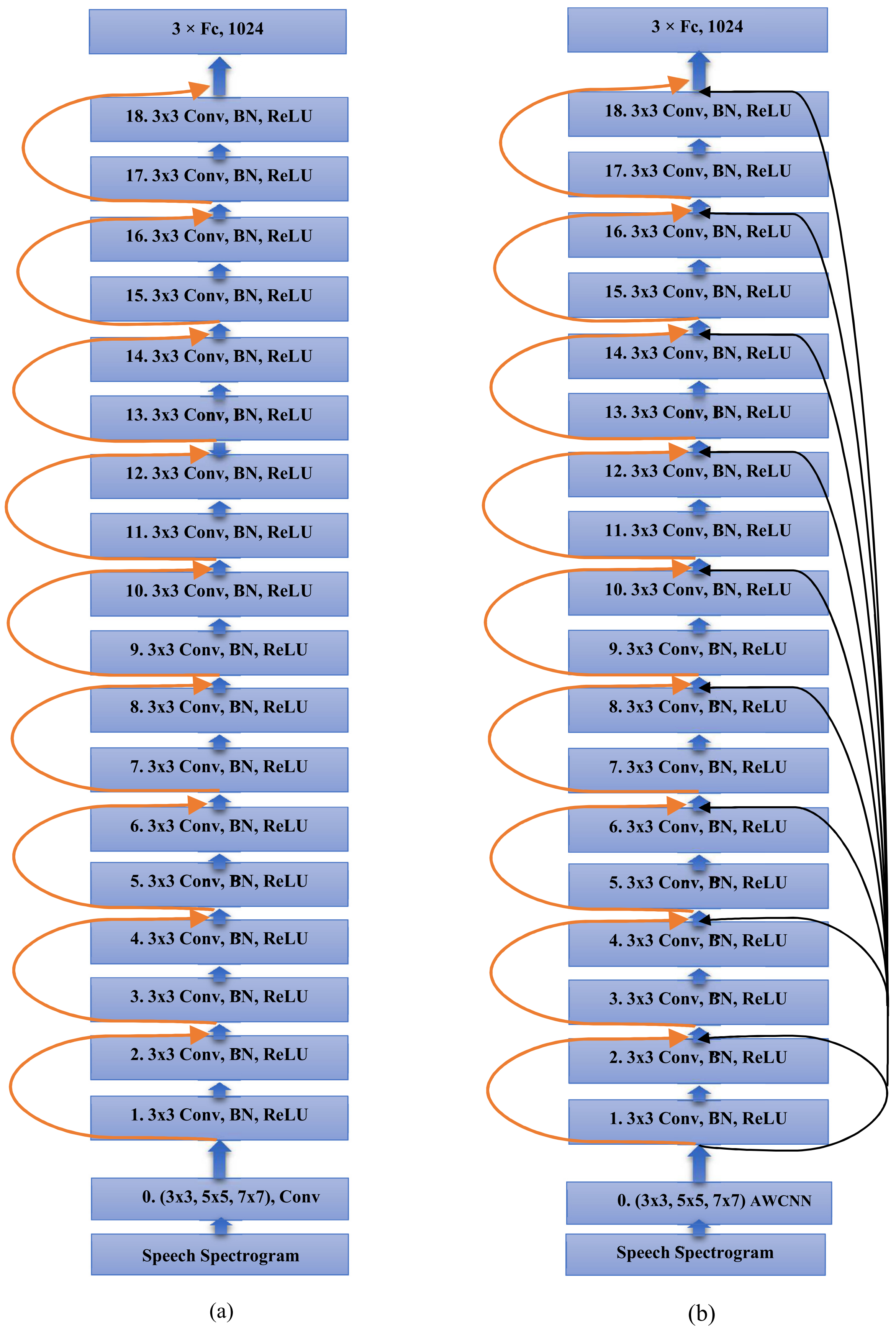

Fig. 3. a) The Residual network with 19 layers. b) The AMRes network with 19 layers. The first layer is an AWCNN, which has three convolution window size (3×3, 5×5, and 7×7). After this first layer, each box shows the layer number and the convolution window size. Batch normalization and ReLU are applied after convolutional function in each layer.

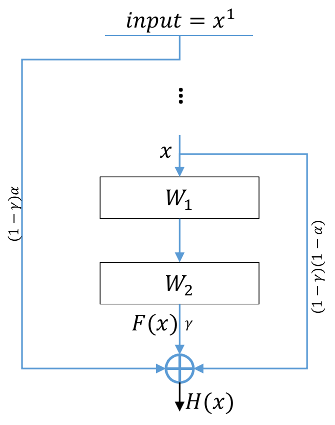

Fig. 4. Block diagram of a multiple residual learning.

As shown in Eq. (10), the $\mathbf{h}^{lr-2}$ layer information is added to the $\mathbf{h}^1$ layer information, as presented by $(\alpha^{lr}\mathbf{h}^1 + (1-\alpha^{lr})\mathbf{h}^{lr-2})$ _ or *previous information*. The previous layer $\mathbf{h}^{lr-1}$ is added to the *previous information* in Eq. (10) to estimate the next layer input. *Previous information* is multiplied by $(1-\gamma)$ and the current information is multiplied by γ.

$$\mathbf{h}^{lr} = \sigma(\gamma \mathbf{w}^{lr-1}\mathbf{h}^{lr-1} + (1-\gamma)(\alpha^{lr}\mathbf{h}^1 + (1-\alpha^{lr})\quad \mathbf{h}^{lr-2}))$$
$$\forall lr \in 2L+1, \quad \alpha^{(l+1)} = \alpha^{(l)} - Const \tag{10}$$

In Eq. (10), the α parameter for the next layer decreases by the *Const* parameter. The method uses a usual summation if the dimensions of the input and output are the same. However, if the input and output dimensions are not the same, two solutions can be used: 1) Adding zeros to the input to have the same size as the output, adds no extra parameter to the model; or 2) Using a convolution layer with a 1×1-window size and stride of two, performs down sampling. The stride size shows the number of filter shift. These solutions change the input dimension to the desired output dimension. In addition, these two cases use the stride of two.

In the proposed method, if the input and output dimension sizes are the same, we use the shortcut connection directly. If the input and output dimension sizes are different, we use a convolution layer to change the input dimension to the desired output dimension. We conclude that if the input and output feature-map sizes were the same, the number of filters would be the same. If the size of the output feature-maps were half of the input size, we would double the number of filters without any extra computational time.

However, MRes corrects the flow of input information all over the network. Nevertheless, MRes cannot model various phone durations among different speakers or a single speaker. Therefore, we modified the first layer of the MRes network to overcome this shortcoming.

3.2. Adaptive window convolutional neural network

Speakers might express an identical phoneme differently. Even one speaker may express a phoneme slow or fast. If we scale fast and slow expressions of a phoneme to a normal expression, we can treat them similarly. CNN networks consider one window size for each speech frame. In CNN networks, if we use different window sizes on the speech spectrogram, we can consider a phoneme with different durations. Thus, we can normalize a fast or slow expression of a phoneme to a normal expression length.

Inter- and intra-speaker phoneme duration variation is a significant problem in speech recognition since it decreases recognition results. That is, a phoneme with different lengths must be simply classified in the same class. CNNs use weight sharing, which makes it possible to use windows with varied sizes. The standard CNN uses one window at each time as the input of the network. Since phone duration and time-frequency variations are common in speech signals, using variable window sizes in the input layer of the convolution sub-layer instead of one window size may be more appropriate.

The first layer of the proposed network is an adaptive window convolutional neural network called AWCNN. As shown on the left side of Fig. 5, AWCNN contains one input layer and one hidden layer. The AWCNN hidden layer contains one convolution sub-layer and two pooling sub-layers, and a different hidden layer structure compared to CNN. AWCNN input includes several windows, which show the number of convolution sub-layers. The size of each window is directly related to phone duration. As shown on the left side of Fig. 5, three-convolution sub-layers are present for three window lengths. Each window considers a different phone duration length.

The goal of speech recognition is to convert speech to text. ResNets and CNNs use local features, weight sharing, and max pooling. These lead to high performance speech recognition. Res and CNN networks consider a time-frequency window of speech spectrogram as input. This small time-frequency window considers speech features locally. Weight sharing and using local features reduce the number of parameters during network learning to improve the robustness of the model against small frequency band variations and reduce over-fitting.

Thus, the speech spectrogram is used as the Res-Nets input in speech processing tasks (Abdel-Hamid et al., 2014; Sainath et al., 2013a; Sainath et al., 2015). We can use two other dimensional inputs instead of the spectrogram; however, the speech spectrogram is the input of the proposed method because it represents time and frequency information simultaneously.

Different phonemes are expressed with different durations. For each phoneme, we consider three windows sizes. By using max pooling in AWCNN, the best of these three windows is selected for the next layer based on phoneme duration. The network requires a short window size when expression of a phoneme appears in a brief time period. On the other hand, slow expressions require longer windows of a phoneme. Therefore, different window lengths model phone duration variations in the first layer of the proposed method.

The learning procedure of AWCNN method is summarized as follows. As shown in Fig. 5, AWCNN applies the convolutional function to the three different window sizes of the input layer. AWCNN applies mean function to the convolution sub-layer output. Finally, AWCNN applies the max function to the mean pooling sub-layers. Using various window sizes, the AWCNN model is more robust against speech signal variations for a speaker, variations between speakers, and phoneme length variations.

Eq. (11) shows the mapping of the input to the convolution sub-layer in AWCNN:

$$q_{j,m,nW} = \sigma\left(\sum_{i=1}^{I(nW)}\sum_{n=1}^{F} o_{i,n+m-1} \quad w_{i,j,n} + \mathbf{w}_{0,j}\right),$$
$$(j = 1, \ldots, J;\ nW = 1, \ldots, NW) \tag{11}$$

where nW is the window index e.g. $nW = 1, \ldots, 3$, as in Fig. 5. The nWth feature-map gets its input from the nWth input window. As seen in Fig. 5, the input features $\mathbf{o}_i (i = 1, \ldots, I(nW))$ have different dimensions. $I(nW)$ is the length of the nWth window. Each input feature, $\mathbf{o}_i$, connects to the units in the convolution sub-layer, $\mathbf{q}_j (j = 1, \ldots, J)$, by the weight matrices $\mathbf{w}_{i,j} (i = 1, \ldots, I;\ j = 1, \ldots, J)$. $o_{i,m}$ is the mth units of the ith input. F is the filter size, which defines the number of frequency bands from the input features to the convolution sub-layer. A convolution operator changes

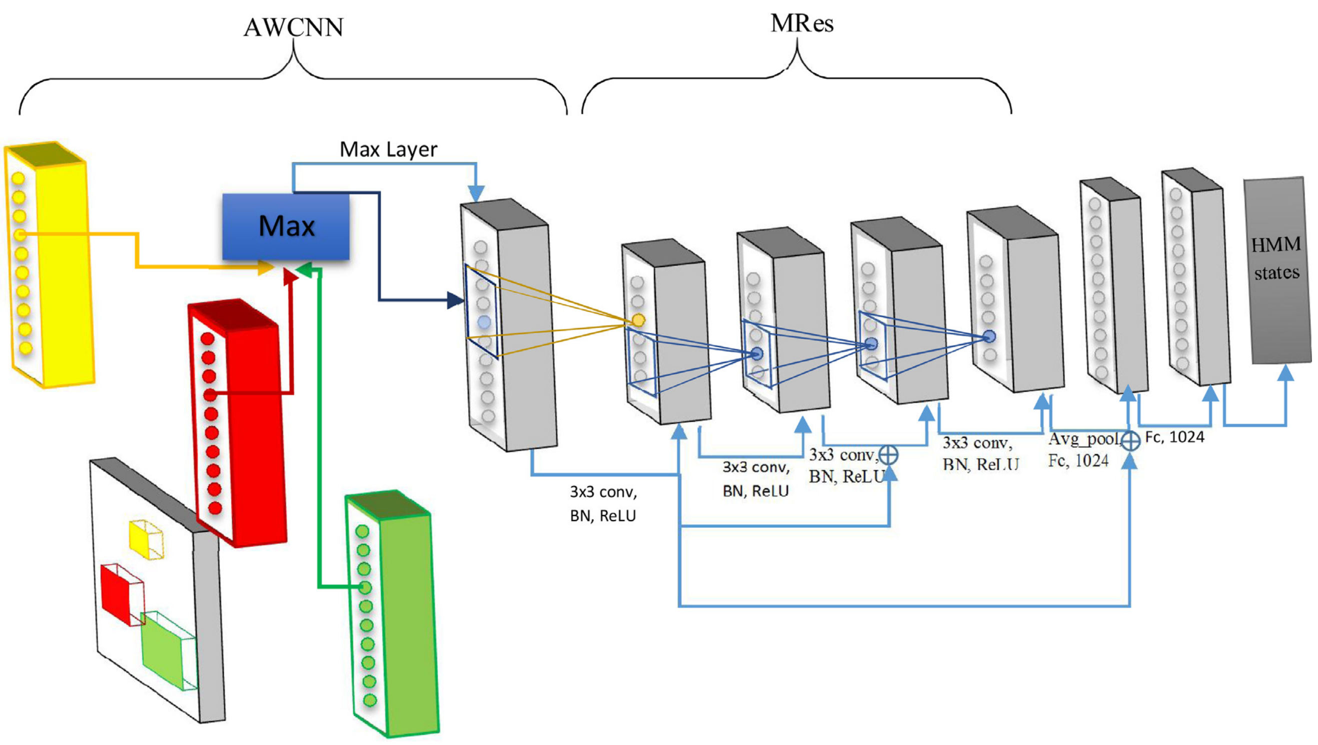

Fig. 5. A representation of the proposed method (AMRes). The first layer shows AWCNN in color mode then 4 MRes layer are represented following with 2 fully connected layers.

Eq. (11) to Eq. (12) as follows:

$$\mathbf{q}_{j,nW} = \sigma\left(\sum_{i=1}^{I(nW)} \mathbf{o}_i \ * \ \mathbf{w}_{i,j}\right), \quad (j = 1, \ldots, J; \quad nW = 1, \ldots, NW) \tag{12}$$

The number of feature-maps in the convolution sub-layer determines the number of local weight matrices in the convolutional maps. AWCNN considers three windows for its computations. Each window has a length I and width F.

As shown in the middle of Fig. 5, in order to connect the convolution sub-layer to the pooling sub-layer, we apply the mean-pooling function to each convolutional feature-map. Thus, the mean of these units is used as the output of the first pooling sub-layer. AWCNN computes each unit in the mean-pooling sub-layer as Eq. (13):

$$t_{j,m} = r \sum_{nW=1}^{NW} q_{j,m,nW} \tag{13}$$

In this equation, r is the normalization coefficient. The next pooling sub-layer uses a max function. This method applies max function to the output of the mean sub-layer. AWCNN computes each unit in the max-pooling sub-layer using Eq. (14):

$$p_{j,m} = max_{n=1}^{G} \quad t_{j,(m-1)} \times s + n \tag{14}$$

where G is the pooling window size, and s is the stride size. The stride size shows the overlapping of two windows.

The first layer of the proposed method uses AWCNN containing one convolution and two pooling sub-layers. Aside from the first layer, the other layers use MRes hidden layers, which use one convolution and no pooling sub-layer. These layers use convolutional function, batch normalization, and ReLU in a convolution sub-layer.

As shown in Fig. 6, we can summarize the AMRes-HMM as follows. The first layer of the network uses AWCNN. We then add MRes-Net on the top of the AWCNN layer, which allows the model to be deeper. MRes-Net learns from multiple inputs. That is, each layer $\mathbf{h}^{lr}$ receives information from $\mathbf{h}^1$ and the second previous layer, $\mathbf{h}^{lr-2}$ (Fig. 3(b)). The MRes applies batch normalization, and ReLU activation function after the convolutional function. MRes uses ReLU for all layers except the last layer. Subsequently, we apply fully connected layers on the top of MRes net to combine all frequency band features. We use a Softmax layer as the final layer of the networks. The Softmax function computes the probabilities of HMM states. Then, we use a back-propagation algorithm to train the model. Finally, the network uses a Viterbi decoder to decode the phoneme sequence. In addition, we add general speech and speaker information to the model by using the Subspace Gaussian Mixture Model (SGMM) to improve speech recognition results. SGMM is shown in the lower branch of Fig. 6. It is combined with the upper branch using a voting function.

4. Experiments

We conducted experiments on speech tasks and an image task to evaluate the AMRes. We report the results on TIMIT, MNIST, Switchboard, and CallHome datasets to evaluate the proposed method and compare it with existing approaches. In this section, we analyze the obtained results on these datasets after describing the experimental setting.

Here, we benefit from stochastic gradient descent. The mini-batch size is 256. The learning rate starts from 0.08 and scales by 0.5 when the error difference between two evaluation iterations is less than 0.02. We consider the momentum term of 0.9 in these experiments. We implemented AMRes using Python and Kaldi toolkits to evaluate the proposed method. In addition, the proposed method utilizes Theano and NumPy libraries. Theano is a library to define, optimize, and evaluate mathematical expressions. NumPy is a scientific package in Python. Theano uses NumPy and GPU, which speed up the derivatives computation. It is possible to develop multi-dimensional arrays like matrices by using the

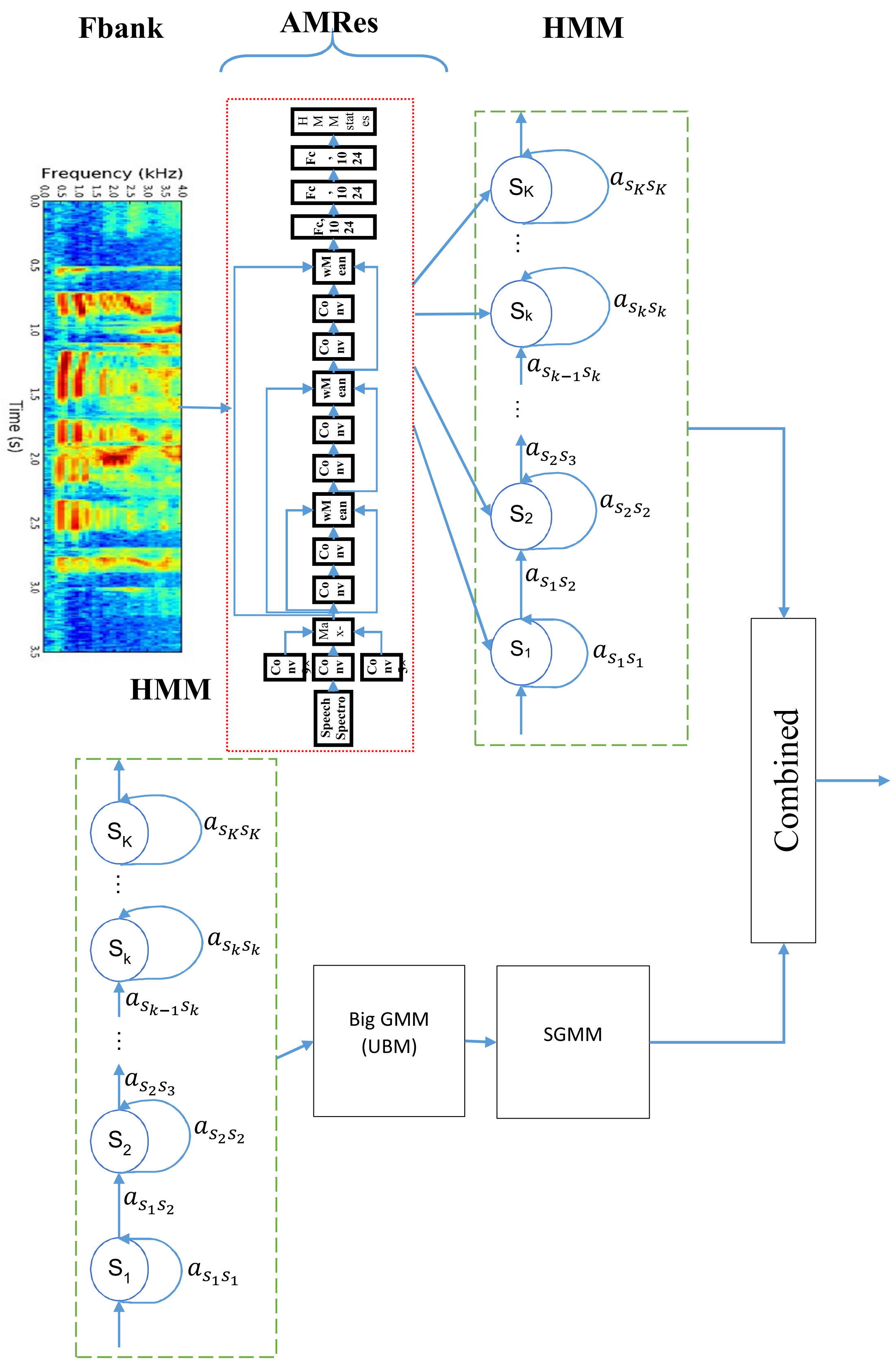

Fig. 6. The overall architecture of the proposed method (AMRes-HMM).

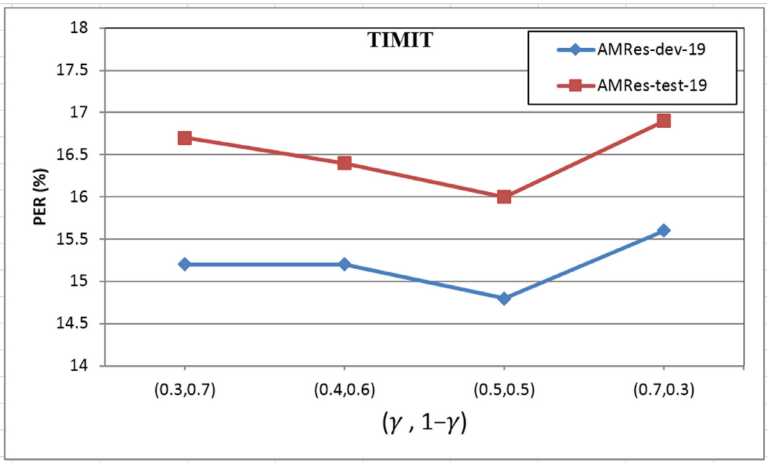

Fig. 7. PER versus different γ values on TIMIT dataset for the proposed AMRes method.

Table 1
TIMIT dataset characteristics.

Data set	# Speakers	# Sentence	# Hours
Train set	426	3696	3.14
Core test set	24	192	0.16
Complete test set	168	1344	0.81
Total	654	5232	4.11

Python library. We executed all our experiments on an eight-core computer, 2.88 GHz with 32G RAM which has 6144 MB GPU memory. The GPU has 2304 Cuda processing core.

4.1. Phone recognition results on TIMIT

Some researches on speech recognition use TIMIT as the evaluation dataset. Likewise, we used TIMIT to evaluate our method, specifically for phone recognition tasks. This dataset contains 6300 sentences from 630 speakers. Each speaker spoke 10 sentences from eight different dialect regions of the United States. All speakers spoke two of these sentences (SA1 and SA2 sentences). We report the results using a 24-speaker core test set. As Table 1 shows, the proposed method is trained on 3693 sentences and tested on 192 sentences core test set, which has no overlap with the development set. We compared the proposed method with all methods for speech recognition tasks and report their phone error rates (PER). We performed our experiments on the TIMIT core test set. Table 1 shows the features of this data set. The second and third columns show the number of speakers and sentences, respectively. The fourth column in Table 1 shows total test and train time.

TIMIT has 61 phones. We considered three states for each phone; thus, there were 183 class labels. We have 183 HMM states as the output of the proposed network. We used tree-state HMM, thus each tree-state is equivalent to a phoneme. Therefore, the output of the proposed network is 61 phone for TIMIT. After decoding, we mapped 61 phones to 39 phones as described in Lee and Hon (1989). We used the Bigram language model in our experiments.

4.1.1. Studying the effects of different γ values

In this experiment, we considered the effect of different γ parameters in Eq. (10). In the proposed model *Previous information* is multiplied by $(1-\gamma)$ and the current information is multiplied by γ. Fig. 7 shows the impact of γ parameter on PER. The horizontal axis shows $(\gamma, 1-\gamma)$, while the vertical axis shows the PER on TIMIT dataset. We have tested different values for $(\gamma, 1-\gamma)$, as shown in Fig. 7.

For this experiment, AMRes uses the speech signal spectrogram as its input. This method considers 11 consecutive speech frames at each iteration. We use a Fourier transform-based filter bank with 40-log energy coefficient distributed on Mel scale of these 11 frames as the network input. We normalize all speech data. AMRes uses AWCNN for its first layer. The network then uses MRes and the last layer is a Softmax layer. The method uses three fully connected layers and then HMM.

As depicted in Fig. 7, the model uses 50% of the prior information and 50% of the current layer information for the best result. Such result is achieved when $(\gamma, 1-\gamma)$ parameters are (0.5, 0.5). This experiment shows the importance of the *previous information* and remembers the input information in the middle layers. In addition, this figure shows the benefit of flowing input information in the model. Although input information can also flow from other shortcut connections, adding input information as a separate connection to the residual networks leads to more general models. These properties process input features in detail in deeper layers.

4.1.2. Studying the effects α hyper-parameter

In this section, we analyze the effect of varying the α parameter on AMRes. We have shown the results on the TIMIT core test set and development set. We consider the effect of α parameter on the ASR performance. As shown in Eq. (10), α parameter decreases in each algorithm iteration by *Const* parameter. Thus, if we change the *Const* parameter, the α parameter changes, and we consider the influence of *Const* parameter on the network performance in the phoneme recognition task. Here, we consider AWCNN as the first layer of our model. The window sizes of AWCNN are 5×5, 7×7, and 9×9. After this AWCNN layer, we consider 18 MRes convolution layers with 3×3-window size and three fully connected layers. The $(\gamma, 1-\gamma)$ parameters take (0.5, 0.5) as their values, which lead to the best results in the previous experiment. Fig. 8 shows the impact of the *Const* parameter (horizontal axis) on the PER (vertical axis) for model with 19 convolutional layer.

α parameter controls the transfer of the input information in the networks. The $\mathbf{h}^{lr-2}$ layer information is added to the $\mathbf{h}^1$ layer information by the α parameter. Fig. 8 shows that the best result emerges when $Const = 0.3$. $Const = 0.3$, shows the effect of $\mathbf{h}^1$ layer information, which decreases by the factor of 0.3 when it enters each deeper layer. For this *Const* value, the third layer in Fig. 3(b) receives 100% of its information from the $\mathbf{h}^1$ layer. The fifth layer receives 70% of its information from the $\mathbf{h}^1$ layer, and the remaining 30% of its information comes from the $\mathbf{h}^{lr-2}$ layer.

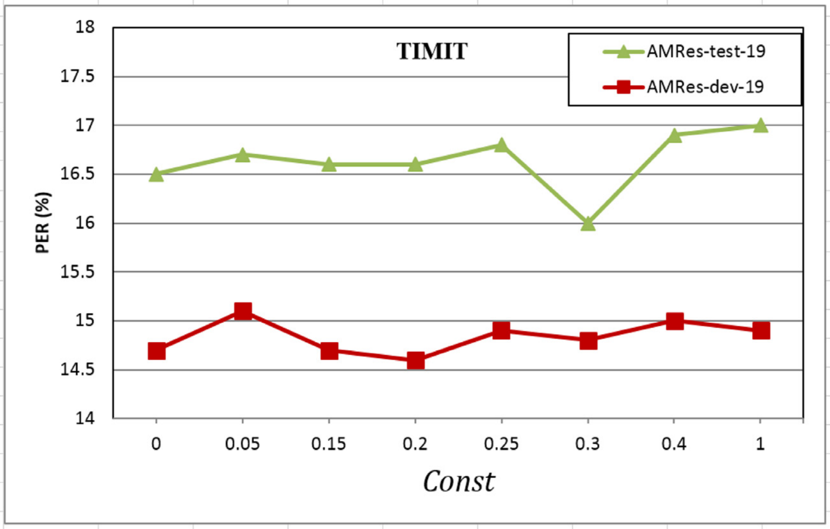

Fig. 8. PER versus different *Const* values for the proposed AMRes method on TIMIT.

The seventh layer receives 40% of its information from the $\mathbf{h}^1$ layer and the remaining 60% from the $\mathbf{h}^{lr-2}$ layer. The ninth layer receives 10% of its information from the $\mathbf{h}^1$ layer and the remaining 90% from the $\mathbf{h}^{lr-2}$ layer. In this example, the value of *Const* for the layers following the 11th layer is negative. Therefore, the algorithm considers zero value for them. Thus, the layers after the ninth layer act the same as Res-Net in this example (He et al., 2016). Therefore, these deep layers do not use $\mathbf{h}^1$ layer information.

For larger *Const* values, lower layers receive more information from the $\mathbf{h}^1$ layer, while deeper layers do not obtain any information from $\mathbf{h}^1$. For example, if we consider $Const = 0.4$, the higher layers following the seventh layer do not receive $\mathbf{h}^1$ layer information. The lower layers need more input information, but the amount of input information is not enough for the lower layers of the network in this example. Thus, the error rate increases when we consider $Const = 0.4$.

However, for $Const = 0.05$, up to the 51st layer receives the $\mathbf{h}^1$ layer information. The higher layers of the model do not require $\mathbf{h}^1$ layer information, which contains input information, because deeper layers mostly require discriminative information. Fig. 8 shows that a small value for α usually decreases performance and that by selecting a correct α value, AMRes can perform better due to its ability to learn generative features along with discriminative features. These properties cause the error rate to increase when *Const* value is 0.05.

As mentioned above we test the proposed method on TIMIT core test set to compare the proposed method with the other speech recognition systems. However, Figs. 7 and 8 are not statistically different, because TIMIT core test set has limited amount of data. Nevertheless, we want to tune the α and γ parameters for Table 2, thus we perform the experiment on core test set which have the similar result on TIMIT complete test set.

4.1.3. Overall evaluation

Here, we compare the proposed method, AMRes-HMM, with the newest and popular methods. We discuss the results of phone error rate on both core test set and development set of TIMIT. Most of these models are deep and all of them use HMM, so comparing them is meaningful. The baseline architecture is a Res-HMM model. As shown in Fig. 3(a), we implemented Res-HMM with 7, 15, and 19 convolution layers.

Table 2 rows 1 and 2 show GMM-HMM with three left-to-right states, estimated by maximum likelihood. HMM with 40 Gaussians in each state has a 0.3% better performance than 27 Gaussians in absolute PER. The proposed AWCNN-Res-HMM produces more than 25% relative reduction in PER compared with GMM-HMM. Rows 3–15 consider some new and popular methods for phoneme recognition using a TIMIT dataset. Row 17 shows the proposed method has better performance than these methods.

The term LWS in Table 2 (row 17) shows the use of local weight sharing (Sainath et al., 2013a; Sainath et al., 2015) for convolutional layers. *J* is the number of sub-layer feature-maps, *f* is the convolution window size, *p*1 is the first pooling window size, *p*2 is the second pooling window size, and *s* is the stride. The last parameter shows the number of fully connected layers. For example, 1024×3 shows a network with three fully connected hidden layers, which has 1024 units in each layer. In this experiment, we consider max function for both pooling sub-layers. AWCNN considers three different windows sizes including 3×3, 5×5, and 7×7 in the first layer. Table 2 shows only the smallest windows (shown as *f*: 3) for simplicity. We achieve two other window sizes by adding *f* with 2 and 4, respectively. The other convolution layers except the first layer use a window size of 3×3.

The 16th and 17th rows of Table 2 show that the proposed method reduces absolute PER by 6.9% compared with DNN-HMM.

4.1.4. Studying different structures of the proposed method

Here, we evaluate and compare different structures of the proposed method. The parameters we used in Table 3, is the same as the Table 2. We discuss the results of phone error rate on core test set of TIMIT. Table 3 rows 1 and 2 show that by using AWCNN-HMM, the error rate on the test sets decreases from 20.5% to 19.5% over CNN-HMM. Table 3, rows 3 and 4 show the effect of the proposed method for two convolutional and 10 fully connected layers. AWCNN-HMM with 10 fully connected layers obtains 0.3 absolute error reduction over CNN-HMM, but this is not better than the model with three fully connected layers. AWCNN cannot direct the information all over the deep network, thus the model accuracy decreases. The AMRes-HMM method presents better results and reduces PER by increasing the model depth.

The fifth and the sixth rows of Table 3 show the effect of the AWCNN method for seven convolutional layers. In these two rows, we combine Res-Net with CNN and AWCNN, which leads to 1% improvement. In addition, we tested the effect of AMRes on different structures. As shown in rows 6 and 10, the proposed AMRes-HMM model leads to a 3.4% PER reduction over the

Table 2
Comparison of different configuration of AMRes-HMM with state-of-the-art methods on TIMIT.

ID	Model	No. layer	Test PER%	Dev PER%
1	GMM-HMM (27 Gauss.)	–	26.4	25.4
2	GMM-HMM (40 Gauss.)	–	26.1	25.0
3	LR DHDPHMM (Torbati & Picone, 2016)	–	21.4	20.5
4	Data-driven HMM (Petrov, Pauls, & Klein, 2007)	–	21.4	–
5	Large Margin GMMs (Bengio, 2009)	–	21.1	–
6	GMMs/Full Cov. (Bengio, 2009)	–	26.0	–
7	HMM/MMI (20 Gauss.) (Rasmussen & Ghahramani, 2001)	–	24.6	23.2
8	HCRF/SGD (Rasmussen & Ghahramani, 2001)	–	21.7	20.3
9	CTC (Graves, Jaitly et al., 2013)	–	21.57	19.05
10	DBRNN (Graves, Jaitly et al., 2013)	–	21.92	19.91
11	DBLSTM (Graves, Jaitly et al., 2013)	–	19.34	17.44
12	LSTM (Huang et al., 2017)	4	20.8	–
13	Residual LSTM (Huang et al., 2017)	4	20.7	–
14	LSTM Res-1 (Huang et al., 2017)	4	19.3	–
15	Fast LSTM Res-3 (Huang et al., 2017)	4	19.3	–
16	DNN-HMM {pre-training + 4 × 1024}	4FC	20.52	19.68
17	AMRes-HMM {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	19Cov+3Fc	16.0	14.8

Table 3
Comparison of different configuration of AMRes-HMM with state-of-the-art methods using TIMIT dataset.

ID	Model	No. layer	Test PER%	Dev PER%
1	CNN-HMM {LWS(j:75 p:3 s:1 f:5) + 3 × 1024}	2Cov+3Fc	20.5	18.8
2	AWCNN-HMM {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	2Cov+3Fc	19.5	18.5
3	CNN-HMM {LWS(j:147 p:3 s:1 f:7) + 10 × 1024}	2Cov+10Fc	20.8	19
4	AWCNN-HMM {LWS(j:147 p1:3 p2:3 s:1 f:7) + 10 × 1024}	2Cov+10Fc	20.5	19
5	Res-HMM {LWS(j:75 p:1 s:1 f:3) + 3 × 1024}	7Cov+3Fc	20.4	18.5
6	AWCNN-Res-HMM {LWS(j:75 p1:1 p2:3 s:1 f:5) + 3 × 1024}	7Cov+3Fc	19.4	18.0
7	AMRes-HMM {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	7Cov+3Fc	16.6	15.5
8	AMRes-HMM {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	15Cov+3Fc	16.3	15.0
9	Res-HMM {LWS(j:75 p:1 s:1 f:3) + 3 × 1024}	19Cov+3Fc	20.4	19.1
10	AMRes-HMM {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	19Cov+3Fc	16.0	14.8
11	AMRes-HMM-(without SGMM) {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	19Cov+3Fc	18.2	16.5
12	AMRes-HMM-(without shortcut connection) {LWS(j:75 p1:3 p2:3 s:1 f:5) + 3 × 1024}	19Cov+3Fc	19.3	17.2

AWCNN-Res-HMM. Rows 5 and 9 show that if the model is trained without the proposed shortcut connections, the test error of network with 19 convolutional layers is equal to the error of the network with seven convolutional layers. Rows 9 and 10 show that the AMRes achieves 4.4% error reduction over the basic Residual network with the same structure. We examined the model with and without SGMM on rows 10 and 11. As shown in these two rows, by using SGMM in the model the result improves more than 2% which shows general speech and speaker information is necessary for the ASR. Rows 10 and 12 study the importance of using shortcut connection in the proposed method. As can be inferred by comparing these two rows, using shortcut connection leads to more than 3% reduction in PER.

We performed a statistical test (P-value) to show if the difference in performance between the AMRes-HMM and LSTM Res-1 (Huang et al., 2017) on TIMIT dataset is significant. We have obtained $P = 0.172$ for these two algorithms after performing required calculations which shows the difference in performance is statistically significant.

4.2. Phone recognition results on FARSDAT

We applied AMRes to a Farsi database named FARSDAT to confirm the appropriateness of the proposed method for Persian speech recognition. FARSDAT consists of 304 speakers including both men and women. Table 4 shows the characteristics of this database. We used 3994 sentences for training the models, 457 sentences as a development set, and 287 sentences as a test set similar to BabaAli (2016).

We compared AMRes-HMM with CNN-HMM and DNN-HMM on both the test set and development set of FARSDAT. Table 5 compares the proposed method, AWCNN-Res-HMM method, with the newest and popular methods that have used FARSDAT as their evaluation dataset for Persian speech recognition. Table 5 compares DNN-HMM, CNN-HMM, and Res-HMM with the proposed method. These methods use HMM similar to the proposed method. The last column evaluates the relative error rate of each method compared with the AMRes-HMM with 19 convolutional layers and three fully connected layers on test set of FARSDAT.

Table 4
Characteristics of FarsDat dataset used in the experiments (BabaAli, 2016).

Data set	# Speakers	# Sentence	# Hours
Train set	224	3994	2.9121
Development set	50	475	0.3946
Test set	30	287	0.2296
Total	304	4756	3.5364

The first and second rows in Table 5 show that by using AWCNN-HMM, the error rate on the test sets decreases from 21.5% to 20.6% over CNN-HMM. These are considerable results for speech recognition tasks on FARSDAT. The third and seventh rows show that the proposed method reduces the **absolute PER by 7.59%** compared with DNN-HMM.

We also evaluate the effect of MRes and AWCNN on CNN networks. As observed on the fourth and fifth rows of Table 5, the proposed AMRes-HMM model obtains 6.2% absolute PER reduction over the Res-HMM. In addition, rows 5 and 7 show that if the model is trained with the proposed shortcut connections, the error of 19 conv-layer improve compared to the seven conv-layers due to prevent vanishing gradients and remember the input data in deep layers. Rows 6 and 7 show that the AMRes achieves 5.59% error reduction over the Residual network with the same structure.

Table 5
Comparison of different configuration of AMRes-HMM with state-of-the-art methods on FARSDAT.

ID	Model	No. layer	Test PER%	Dev PER%	Relative Test PER% compared with AMRes
1	CNN-HMM {LWS(j:75 p:1 $\times$ 3 s:1 f:5) + 3 $\times$ 1024}	7Conv+3Fc	21.5	21.7	32.04
2	AWCNN-HMM {LWS(j:75 p1:1 $\times$ 1 p2:1 $\times$ 3 s:1 f:5)+3 $\times$ 1024}	7Conv+3Fc	20.6	20.9	29.07
3	DNN-HMM {pre-training + 10 $\times$ 1024}	10FC	22.2	23.2	34.19
4	Res-HMM {LWS(j:75 p:1 $\times$ 1 s:1 f:3)+3 $\times$ 1024}	7Cov+3Fc	21.3	21.4	31.40
5	AMRes-HMM {LWS(j:75 p1:1 $\times$ 3 p2:1 $\times$ 3 s:1 f:5) +3 $\times$ 1024}	7Cov+3Fc	15.10	15.36	3.24
6	Res-HMM {LWS(j:75 p1:1 $\times$ 1 p2:1 $\times$ 3 s:1 f:5)+3 $\times$ 1024}	19Cov+3Fc	20.2	20.6	27.67
7	AMRes-HMM {LWS(j:75 p1:1 $\times$ 3 p2:1 $\times$ 3 s:1 f:5)+3 $\times$ 1024}	19Cov+3Fc	14.61	15.09	0

Table 6
Comparison of different configuration of AMRes-HMM with state-of-the-art methods on MNIST.

ID	Model	No. layer	Test PER%
1	DNN {pre-training + 10 $\times$ 1024}	3FC	2.8
2	CNN {LWS(j:75 p:1 $\times$ 3 s:1 f:5) + 3 $\times$ 1024}	3Conv+3Fc	0.99
3	AWCNN {LWS(j:75 p1:1 $\times$ 1 p2:1 $\times$ 3 s:1 f:5) + 3 $\times$ 1024}	5Conv+3Fc	0.76
4	Res {LWS(j:75 p:1 $\times$ 1 s:1 f:3) + 3 $\times$ 1024}	3Conv+3Fc	1.19
5	AWCNN-Res {LWS(j:75 p1:1 $\times$ 1 p2:1 $\times$ 3 s:1 f:5) + 3 $\times$ 1024}	7Conv+3Fc	0.68
6	AMRes {LWS(j:75 p1:1 $\times$ 3 p2:1 $\times$ 3 s:1 f:5) + 3 $\times$ 1024}	7Conv+3Fc	0.55

4.3. Recognition results on MNIST

The MNIST dataset has 60,000 training data and 10,000 test data including 28 $\times$ 28 images. MNIST database contains 10 handwritten digits, 0–9. We train the network with all 60,000 train data and we evaluate the network with 10,000 test data.

As seen on the first, second, and sixth rows of Table 6, the proposed AMRes model with seven convolution layers obtains 2.25% and 0.44% absolute error reduction over the DNN and CNN, respectively.

Rows 2 and 3 indicate the proposed AWCNN with the weaker structure achieves 0.23% error reduction over the CNN.

The fourth and fifth rows in Table 6 show the effect of the proposed AWCNN method. By using AWCNN, the error rate on the test sets decreases from 1.19% to 0.68% over Res-Net.

Rows 4 and 6 compare the AMRes with Res network, indicating that the proposed method improves performance relative to Res by 0.64%.

We computed the *P*-value to show if the difference in performance between the AMRes and Res on MNIST dataset is significant. After performing required calculations we have obtained $P = 0.198$ for these two algorithms which shows the difference in performance is statistically significant.

4.4. Recognition results on LVSR

We apply AMRes to the Switchboard database to confirm the proposed method's appropriateness for automatic speech recognition using large databases. Switchboard contains conversations between people on a predefined topic. Switchboard uses 2000 h of training data, each call length is about five minutes. This database represents American English speech sample of location, gender, race, and channel. We also applied AMRes to the CallHome speech corporation data. CallHome contains conversations between friends and family with no preassigned topic and consists of 18 h of training data.

Table 7 compares different approaches on the two large vocabulary tasks. In this experiment, most of these models are deep architectures, so comparing them is meaningful. The proposed architecture is a AMRes model. We implemented AMRes with 19 convolution layers. In this experiment the WER is reported.

Table 7 shows the impact of the proposed method on WER. For this experiment, AMRes uses the speech signal spectrogram as its input. This method considers 11 consecutive speech frames at each iteration. We used a Fourier transform-based filter bank with 40-log energy coefficient to extract speech features. We normalize all speech data. AMRes uses AWCNN for its first layer. The network then uses 19 MRes layers and the last layer is a Softmax layer. The method uses three fully connected layers and then HMM. The $(\gamma, 1-\gamma)$ parameters are considered (0.5, 0.5), thus the model uses 50% of the prior information and 50% of the current layer information. AMRes uses $\alpha = 0.5$, and this parameter decreases by a factor of 0.5. The number of units in all AMRes hidden layers is 2048.

As shown in Table 7, AMRes reduces absolute error rate over CD-DNN and DNN-CNN by 7.1% and 4.4% on Switchboard (Hub500 SWB), respectively. AMRes also improves accuracy by 2.3% and 0.9% over CNN-RNN and RNN+LSTM on CallHome (Hub500 CH), respectively. Results show that AMRes is not better than LSTM+ResNet. As can be seen in these two datasets, AMRes outperforms CNN and RNN in most of the cases.

4.5. Comparison of the number of parameters for different networks

We compared the number of parameters for different networks and reported the results in Table 8. The second and third columns of this table show the network names and layer numbers, respectively. We fixed the total number of hidden layers to ten to have a fair comparison. We computed the number of learnable parameters for each network in the fourth column of Table 8. The input layer reads the input speech signal, so there are no parameters to be learned for the input layer. The number of parameters for convolutional layers is $l \times h \times f \times k$, where h and l are the number of hidden layer and hidden units, respectively. In addition, $f \times k$ is the filter size. There is a bias term for each feature map, which was ignored for all layers. In the pooling layers, we replaced some neighborhoods by their maximum value. Therefore, there is no parameter to be learned in a pooling layer.

For each fully connected layer, all input units have a separate weighted connection to a hidden unit. Thus, the number of parameters is $n \times h$, for the first layer with n inputs and h hidden units. For the next hidden layer, we have $h \times h = h^2$ parameters to be learned. This procedure is repeated for the other hidden layers. Therefore, we have $n \times h + l \times h^2$ parameters, for the fully connected network with l hidden layers.

We computed the number of parameters for each network in the fifth column of Table 8. In most experiments of this paper, we used a 7 $\times$ 7 filter size, thus the values of f and k are equal to 7 ($f = k = 7$). We calculated the sixth column by considering

Table 7
Comparison of Different Methods on two different large vocabulary tasks.

ID	Task	Hub500 Switchboard	Hub500 CallHome
1	CD-DNN (Saon, Sercu, Rennie, & Kuo, 2016)	13.2	23.3
2	DNN-CNN (Saon et al., 2016)	10.5	18.9
3	CNN-RNN (Sainath, Mohamed, Kingsbury, & Ramabhadran, 2013b)	8.0	14.1
4	RNN+LSTM (Sercu & Goel, 2016)	6.6	12.2
5	LSTM+ResNet (Saon et al., 2017)	5.5	10.3
6	AMRes	6.1	11.8

Table 8
Comparison of the number of parameters for different networks.

ID	Model	No. layer	No. parameters	No. parameters	No. parameters
1	DNN	10FC	$n \times h + 10h^2$	$420h + 10h^2$	10.92 M
2	CNN	7Conv+3Fc	$7h \times f \times k + 3h^2$	$343h + 3h^2$	3.49 M
3	AWCNN	7Conv+3Fc	$9h \times f \times k + 3h^2$	$441h + 3h^2$	3.59 M
4	Res	7Conv+3Fc	$7\ h \times f \times k + 3 + 3h^2$	$346h + 3h^2$	3.50 M
5	AWCNN-Res	7Conv+3Fc	$9h \times f \times k + 7 + 3h^2$	$448h + 3h^2$	3.60 M
6	AMRes	7Conv+3Fc	$9h \times f \times k + 3 + 3h^2$	$444h + 3h^2$	3.60 M

$h = 1024$, which is the number hidden units used in our implementations.

4.6. Discussion

Experiments show that the proposed method has a higher accuracy rate according to the state-of-the-art models because this model benefits the statistics of the input data and pass them all over in a very deep network. Moreover, weighted connections in the residual networks improve model accuracy. AMRes network uses the generative power of features to construct a network more robust against the phone duration, speaker variations, and time-frequency variations in speech signals.

We conclude from the above tables that using the AMRes method with more layers is better than the shallower AMRes, in contrast to the CNN method. In addition, a deep MRes network has a lower error rate on the test set and more generalization on the validation set. The number of parameters increases in the proposed method compared to CNN and Res. However, we can deal with this problem by using GPU and parallel programming. Thus, we conclude that the proposed method can be used and implemented easily in industrial and commercial ASR products.

5. Conclusion and future work

Deep neural networks are among the best tools for acoustic modeling. Deep neural networks when combined with HMM improve phone accuracy rate. We propose a new method that changes the structure of CNN and Res-Net to estimate the probabilities of HMM states. We proposed Adaptive windows convolutional networks along with Multiple Residual networks, named AMRes. This model can better model speech signals. The AMRes method applies to the speech spectrogram and models time-frequency varieties.

The proposed method considerably improves ASR performance relative to DNN, CNN, Res-Net, and other methods with similar layer numbers using TIMIT, FARSDAT, Switchboard, and CallHome datasets (about 8% absolute error reduction). Based on the results obtained from the conducted experiments, we conclude that the use of input information across general speech and speaker information is very beneficial for recognition accuracy. The other advantage of the proposed method is that with limited training data this network can be deep.

We believe that the proposed method is a way toward a powerful acoustic model. As future work, the proposed acoustic model can first be enhanced using recurrent connections in the Multiple Residual network. Then, we can insert attention-based method to this modified RNN network for more efficiency. We would examine the proposed method on other applications such as image, text to speech (TTS), EEG, and ECG signal processing to show its potential in feature extraction and better information flow to the deeper layers.

Acronyms

AWCNN	Adaptive Windows Convolutional Neural Network
AMRes-Net	Adaptive Windows Multiple Residual networks
ASR	Automatic Speech Recognition
CNN-HMM	Convolutional Deep Neural Network - Hidden Markov Network
CNN	Convolutional Neural Network
DBLSTM	Deep Bidirectional Long Short-Term Memory
DNN	Deep Neural Network
DNN-HMM	Deep Neural Network-Hidden Markov Model
GMM-HMM	Gaussian Mixture Model Hidden Markov Model
ESHMM	Expanded State HMM
HCRF	Hidden Conditional Random Fields
HMM	Hidden Markov Model
HSMM	Hidden Semi-Markov models
LVSR	Large Vocabulary Speech Recognition
LSTM	Long Short-Term Memory
MFCC	Mel-frequency Cepstral Coefficients
MRes-Net	Multiple Residual Network
RNN	Recurrent Neural Network
Res-Net	Residual Learning Network
SCRF	Segmental Conditional Random Fields
SGMM	Subspace Gaussian Mixture Models

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Toktam Zoughi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **Mohammad Mehdi Homayounpour:** Conceptualization, Validation, Formal analysis, Investigation, Resources, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Mahmood Deypir:** Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft.

References

Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22*(10), 1533–1545.

BabaAli, B. (2016). A state-of-the-art framework for Persian speech recognition. *Signal and Data Processing, 13*(3), 1–13.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning, 2*(1), 1–127.

Bourlard, H., & Morgan, N. (1993). Continuous speech recognition by connectionist statistical methods. *Neural Networks, 4*(6), 893–909.

Burget, L., Schwarz, P., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., et al. (2010). Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models. In *Ieee international conference on acoustics, speech and signal processing* (pp. 4334–4337).

Celikyilmaz, A., Sarikaya, R., Hakkani-Tur, D., Liu, X., Ramesh, N., & Tur, G. (2016). A new pre-training method for training deep learning models with application to spoken language understanding. In *Interspeech* (pp. 3255–3259).

Chen, D., Mak, B., Leung, C.-C., & Sivadas, S. (2014). Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition. In *ICASSP* (pp. 5592–5596). IEEE.

Chen, S. F., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on association for computational linguistics* (pp. 310–318). Morristown, NJ, USA: Association for Computational Linguistics.

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-Vocabulary speech recognition. *IEEE Transactions on Audio, Speech and Language Processing, 20*(1), 30–42.

Dong, C., Loy, C. C., He, K., & Tang, X. (2016). Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(2), 295–307.

Erhan, D., & Manzagol, P. A. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Aistats* (pp. 153–160).

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Aistats: vol. 9* (pp. 249–256).

Graves, A. (2008). Supervised sequence labelling with recurrent neural networks. In *Image rochester NY* (p. 124).

Graves, A., Fernandez, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on machine learning* (pp. 369–376).

Graves, A., Jaitly, N., & Mohamed, A. R. (2013). Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE workshop on automatic speech recognition and understanding, ASRU 2013 - proceedings* (pp. 273–278).

Graves, A., Mohamed, A.-R., & Hinton, G. (2013a). Speech recognition with deep recurrent neural networks. *Icassp*, (3), 6645–6649.

Graves, A., Mohamed, A.-R., & Hinton, G. (2013b). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP)* (pp. 6645–6649). arXiv:1303.5778.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778).

Hifny, Y., & Renals, S. (2009). Speech recognition using augmented conditional random fields. In *IEEE transactions on audio, speech and language processing: vol. 17* (pp. 354–365).

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A. R., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine, 29*(6), 82–97.

Huang, L., Xu, J., Sun, J., Yang, Y., et al. (2017). An improved residual LSTM architecture for acoustic modeling. In *International conference on computer and communication systems (ICCCS)* (pp. 101–105). arXiv:1708.05682.

Juang, B. H., Chou, W., & Lee, C. H. (1997). Minimum classification error rate methods for speech recognition. In *IEEE transactions on speech and audio processing: vol. 5* (pp. 257–265).

Kao, J. T., Zweig, G., & Nguyen, P. (2011). Discriminative duration modeling for speech recognition with segmental conditional random fields. In *ICASSP* (pp. 4476–4479).

Kapadia, S., Valtchev, V., & Young, S. J. (1993). MMI training for continuous phoneme recognition on the TIMIT database. *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2*, 491–494.

Kim, S., Hori, T., & Watanabe, S. (2017). Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *ICASSP, IEEE international conference on acoustics, speech and signal processing - proceedings* (pp. 4835–4839).

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International conference on machine learning: vol. 2008* (pp. 1–8).

Lee, H., & Kwon, H. (2017). Going deeper with contextual CNN for hyperspectral image classification. *IEEE Transactions on Image Processing, 26*(10), 4843–4855.

Lee, K. F., & Hon, H. W. (1989). Speaker-independent phone recognition using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 37*(11), 1641–1648.

Lee, L., & Rose, R. C. (1996). Speaker normalization using efficient frequency warping procedures. In *Acoustics, speech, and signal processing conference proceedings: vol. 1* (pp. 353–356).

Leggetter, C. J., & Woodland, P. C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language, 9*(2), 171–185.

Li, B., Sainath, T. N., Weiss, R. J., Wilson, K. W., & Bacchiani, M. (2016). Neural network adaptive beamforming for robust multichannel speech recognition. In *Interspeech* (pp. 1976–1980).

Martens, J. (2010). Deep learning via Hessian-free optimization. In *International conference on machine learning: vol. 951* (pp. 735–742).

McDermott, E., Hazen, T. J., Le Roux, J., Nakamura, A., & Katagiri, S. (2007). Discriminative training for large-vocabulary speech recognition using minimum classification error. In *Ieee transactions on audio, speech and language processing: vol. 15* (pp. 203–223).

Miao, Y., & Metze, F. (2017). End-to-end architectures for speech recognition. In *New era for robust speech recognition* (pp. 299–323).

Mohamed, A. R., Dahl, G., & Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing, 20*(1), 14–22.

Mohamed, A. R., Hinton, G., & Penn, G. (2012). Understanding how deep belief networks perform acoustic modelling. In *ICASSP* (pp. 4273–4276).

Petrov, S., Pauls, A., & Klein, D. (2007). Learning structured models for phone recognition. In *Proceedings of the empirical methods in natural language processing and computational natural language learning* (pp. 897–905).

Povey, D. (2003). *Discriminative Training for Large Vocabulary Speech Recognition* Ph.D. thesis.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., et al. (2010). Subspace Gaussian mixture models for speech recognition. In *IEEE international conference on acoustics, speech, and signal processing* (pp. 4330–4333).

Povey, D., & Woodland, P. C. (2002). Minimum phone error and I-smoothing for improved discriminative training. In *ICASSP* (pp. 105–108).

Qian, Y., Bi, M., Tan, T., & Yu, K. (2016). Very deep convolutional neural networks for noise robust speech recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing, 24*(12), 2263–2276.

Ramesh, P., & Wilpon, J. G. (1992). Modeling state durations in hidden Markov models for automatic speech recognition. In *Acoustics, speech, and signal processing: vol. 1* (pp. 381–384). IEEE.

Rasmussen, C. E., & Ghahramani, Z. (2001). Occam's razor. In *Advances in neural information processing systems: vol. 13* (pp. 294–300).

Robinson, A. J., Cook, G. D., Ellis, D. P. W., Fosler-Lussier, E., Renals, S. J., & Williams, D. A. G. (2002). Connectionist speech recognition of broadcast news. *Speech Communication, 37*(1–2), 27–45.

Russell, M., & Cook, A. (1987). Experimental evaluation of duration modelling techniques for automatic speech recognition. In *International conference on acoustics, speech, and signal processing: vol. 12* (pp. 2376–2379). IEEE.

Sainath, T., Mohamed, A. R., Kingsbury, B., & Ramabhadran, B. (2013a). Deep convolutional neural networks for LVCSR. In *International conference on acoustics, speech and signal processing (ICASSP)* (pp. 8614–8618).

Sainath, T. N., Mohamed, A. R., Kingsbury, B., & Ramabhadran, B. (2013b). Deep convolutional neural networks for LVCSR. *Icassp, ieee international conference on acoustics, speech and signal processing - proceedings*.

Sainath, T. N., Kingsbury, B., Saon, G., Soltau, H., rahman Mohamed, A., Dahl, G., et al. (2015). Deep convolutional neural networks for large-scale speech tasks. *Neural Networks, 64*, 39–48.

Sainath, T. N., Kingsbury, B., Soltau, H., & Ramabhadran, B. (2013). Optimization techniques to improve training speed of deep neural networks for large speech tasks. *Transactions on Audio, Speech and Language Processing, 21*(11), 2267–2276.

Salakhutdinov, R. (2009). *Learning Deep Generative Models* Ph.D. thesis.

Salakhutdinov, R., & Hinton, G. (2012). An efficient learning procedure for deep Boltzmann machines. *Neural Computation, 24*(8), 1967–2006.

Saon, G., Kurata, G., Sercu, T., Audhkhasi, K., Thomas, S., Dimitriadis, D., et al. (2017). English conversational telephone speech recognition by humans and machines. In *Proceedings of the annual conference of the international speech communication association, INTERSPEECH*.

Saon, G., Sercu, T., Rennie, S., & Kuo, H. K. J. (2016). The IBM 2016 English conversational telephone speech recognition system. In *Proceedings of the annual conference of the international speech communication association, INTERSPEECH*.

Sercu, T., & Goel, V. (2016). Advances in very deep convolutional neural networks for LVCSR. In *Proceedings of the annual conference of the international speech communication association, INTERSPEECH*.

Sha, F. S. F., & Saul, L. K. (2006). Large margin Gaussian mixture modeling for phonetic classification and recognition. In *ICASSP* (pp. 265–268).

Sung, Y. H., & Jurafsky, D. (2009). Hidden conditional random fields for phone recognition. In *Automatic speech recognition and understanding* (pp. 107–112).

Tan, T., Qian, Y., Hu, H., Zhou, Y., Ding, W., & Yu, K. (2018). Adaptive very deep convolutional residual network for noise robust speech recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing, 26*(8), 1393–1405.

Taylor, G. W., Fergus, R., LeCun, Y., & Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *Lecture notes in computer science* (pp. 140–153). vol. 6316 LNCS.

Torbati, A. H. H. N., & Picone, J. (2016). A doubly hierarchical dirichlet process Hidden Markov Model with a non-ergodic structure. *IEEE Transactions on Audio Speech and Language Processing, 24*(1), 174–184.

Vydana, H. K., & Vuppala, A. K. (2017). Residual neural networks for speech recognition. In *European signal processing conference* (pp. 543–547). vol. 2017-Janua.

Welling, L., Kanthak, S., & Ney, H. (1999). Improved methods for vocal tract normalization. In *Ieee international conference on acoustics, speech and signal processing (ICASSP)* (pp. 761–764).

Wu, S., Zhong, S., & Liu, Y. (2018). Deep residual learning for image steganalysis. *Multimedia Tools and Applications, 77*(9), 10437–10453.

Yu, S. Z. (2010). Hidden semi-Markov models. *Artificial Intelligence, 174*(2), 215–243.

Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017a). Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing, 26*(7), 3142–3155.

Zhang, Y., Chan, W., & Jaitly, N. (2017b). Very deep convolutional networks for end-to-end speech recognition. In *Ieee international conference on acoustics speech and signal processing (ICASSP)* (pp. 4845–4849). arXiv:1610.03022.

Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A. E.-D., Jin, W., & Schuller, B. (2018). Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST), 9*(5). arXiv:1705.10874.

Zweig, G., & Nguyen, P. (2010). A segmental conditional random field toolkit for speech recognition. In *Interspeech* (pp. 2858–2861).