INCREMENTAL TRAINING AND CONSTRUCTING THE VERY DEEP CONVOLUTIONAL RESIDUAL NETWORK ACOUSTIC MODELS

Sheng Li[1]*, Xugang Lu*[1]*, Peng Shen*[1]*, Ryoichi Takashima*[1]*, Tatsuya Kawahara*[2] *and Hisashi Kawai*[1]

[1] National Institute of Information and Communications Technology, Kyoto, Japan
[2] School of Informatics, Kyoto University, Sakyo-ku, Kyoto 606-8501, Japan

sheng.li@nict.go.jp

ABSTRACT

Inspired by the successful applications in image recognition, the very deep convolutional residual network (ResNet) based model has been applied in automatic speech recognition (ASR). However, the computational load is heavy for training the ResNet with a large quantity of data. In this paper, we propose an incremental model training framework to accelerate the training process of the ResNet. The incremental model training framework is based on the unequal importance of each layer and connection in the ResNet. The modules with important layers and connections are regarded as a skeleton model, while those left are regarded as an auxiliary model. The total depth of the skeleton model is quite shallow compared to the very deep full network. In our incremental training, the skeleton model is first trained with the full training data set. Other layers and connections belonging to the auxiliary model are gradually attached to the skeleton model and tuned. Our experiments showed that the proposed incremental training obtained comparable performances and faster training speed compared with the model training as a whole without consideration of the different importance of each layer.

Index Terms— Speech recognition, Acoustic model, DNN, very deep convolutional residual network (ResNet)

1. INTRODUCTION

Since very deep convolutional networks (AlexNet [1], VGG [2], etc.) achieved impressive results in the visual and image process area, deeper and deeper structures have been investigated. These models revealed the importance of network depth. However, training a further deeper convolutional network model will face the challenge of the gradient-vanishing and overfitting problems. Instead of learning a new representation at each layer, deep convolutional residual network (also known as ResNet [3] [1]) use residual connections (shortcut/skip connections [4]) to learn residuals. The network had excellent convergence speed and showed compelling accuracy.

In speech recognition field, the deep residual networks also get magnificent performance [5, 6, 7, 8, 9, 10, 11]. However, these deep stacking models (always hundreds of layers) were initially designed for image recognition tasks, and extremely computationally demanding for speech recognition tasks, limiting the training and deployability. For most of the researchers, aggressively using massive GPUs to accelerate training speed is unrealistic. The motivation of this paper is focused on solving this problem: Given limited computational resources, how to efficiently accelerate training a very/ultra deep ResNet with little loss of precision.

Current research shows reducing the computational cost of the convolutional operation can speed up deep convolutional network training and compress the model size. For example, 2D filters are divided into independent-assumed 1D filters [12, 13]. Effective network structure pruning methods have been proposed, i.e. stochastic depth/dropping layers [14, 15] or learning structured sparsity [16].

Instead of pruning the model parameters, we try to keep all of the parameters for more precise decision boundaries. Recent research [17] also showed the current prevailing ResNet architecture with short-cut paths and some form of the ensemble of neural networks (a collection of many paths). Inspired by this research, we proposed an incremental training method. We first train a shallow skeleton model with the full training data set. The skeleton model includes the paths connecting the feature input and softmax output and substantial for both training and testing. Then the other layers are gradually attached to the skeleton model and trained. In this way, the whole very deep network can be constructed incrementally.

The rest of this paper is organized as follows. Section 2 briefly reviews the background knowledge of ResNet and its application in speech recognition. Section 3 describes our proposed method to simplify and speed up ResNet acoustic training. Section 4 presents evaluations of the proposed method. Conclusions and future works are given in Section 5.

[1] In this paper, "ResNet" is short for "very deep convolutional residual network" following the paper [3].

2. VERY DEEP CONVOLUTIONAL RESIDUAL NEURAL NETWORKS FOR SPEECH RECOGNITION

In this section, we take the benchmark 110-layer ResNet [3] as presented on CIFAR-10 [18] as an example to introduce the original structures of ResNet [2]. If we consider each frame of the speech signal as a 2D (temporal and frequency domain) image, the general concept of ResNet can also be successfully applied to speech recognition tasks.

The basic unit of a ResNet model is the residual block, and each block consists of the following two components:

1. Conv(3x3)-BN-ReLU-Conv(3x3)-BN [3] layer-sequence: Two convolutional layers with small 3x3 filters connected with the ReLU activation. Each convolution is followed by batch normalization [21].

2. Residual connection (short-cut/skip): Signals can bypass the transformation of the whole block from these short-cut/skip connections. We use 1x1 conv layers as the shortcuts (Fig. 1 right) when the dimensions of feature maps do not match. Otherwise, we use an identity shortcut (Fig. 1 left).

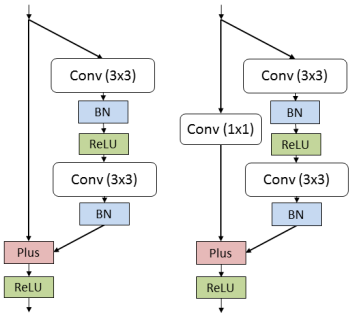

Fig. 1. Typical Structure of residual block.

There are three groups of residual blocks with small 3x3 filters that differ in the number of filters (16, 32 and 64) and we call them ResBlock-16, ResBlock-32 and ResBlock-64. And each group is a stack of 18 residual blocks in the 110-layer ResNet model. The convolutional layers mostly have 3x3 filters: for the same output feature map size, the layers have the same number of filters, and if the feature map size is halved, the number of filters is doubled. Zero padding is used to all the convolutional operations. Downsampling is done by convolutional layers (blocks are shown in Fig. 2 right) using a stride of 2x2. At last, the network ends up with an averaged 2D pooling (2x2 filter and 1x1 stride), a fully connected feed-forward layer (without batch normalization) and softmax of 8522-dim output.

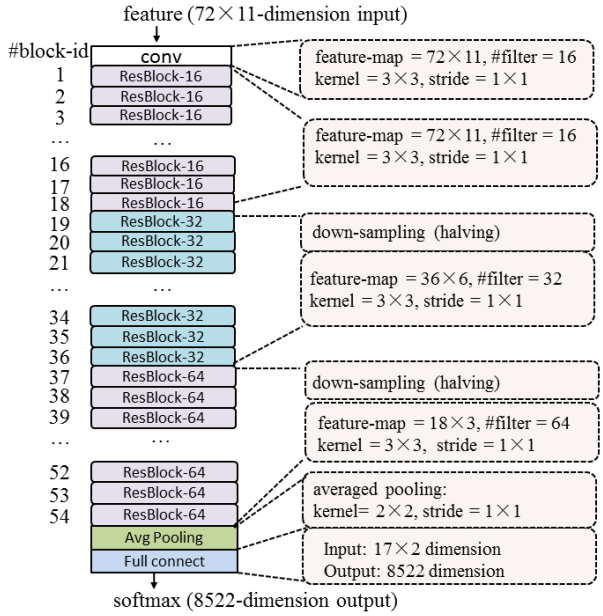

Fig. 2. ResNet Structure for Speech Recognition (54 blocks). (ResBlock-16/32/64 means the residual block with 16/32/64 filters.)

The work by Wang et. al [5] and the system we constructed (see Section 4) confirm that the ResNet acoustic model has better performance than conventional DNN models.

3. INCREMENTAL TRAINING AND CONSTRUCTING

3.1. Skeleton Network and Auxiliary Networks

A. Veit et al. [17] examined the current prevailing ResNet architecture with short-cut paths and some form of the ensemble of neural networks (a collection of many paths). They observed: during training, the updates were not happening uniformly across all layers, as they always would for traditional deep neural networks. Every training point would adjust the weights along a specific set of paths. And when testing, the input has its unique set of paths to the output, without having to go through every single layer.

We trained three seed ResNet acoustic models using the training data of 27.6 hours (CSJ-Train$_{\text{small}}$, see Section 4.1). These seed ResNet models are constructed using the similar network components and structures we introduced in Section 2 with 110 layers (18 ResBlock-16 + 18 ResBlock-32 + 18

[2]Current prevalent structures of ResNet such as wide residual network (WRN) [19] or Inception [12] are not followed in this paper.

[3]There are also residual blocks with other structure, e.g. Conv(1x1)-BN-ReLU-Conv(3x3)-BN-ReLU-Conv(1x1)-BN [3] used in ImageNet benchmarks [20] and the pre-activation structure proposed in [4]. In this paper, we focus on a simple structure.

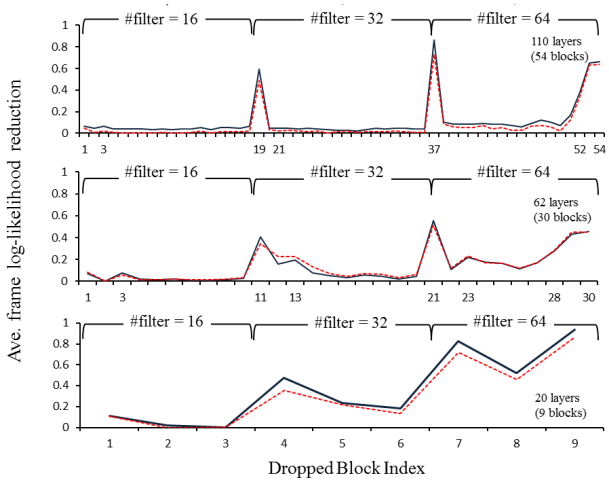

Fig. 3. Averaged frame log-likelihood change (Loglikelihood after dropping - Loglikelihood before dropping) of three ResNet models with different number of ResBlocks (The blue solid line is based on decoding 10 utterances selected from a male speaker in CSJ-Eval01 and the red dotted line is based on decoding 10 utterances selected from a female speaker in CSJ-Eval02 for verification).

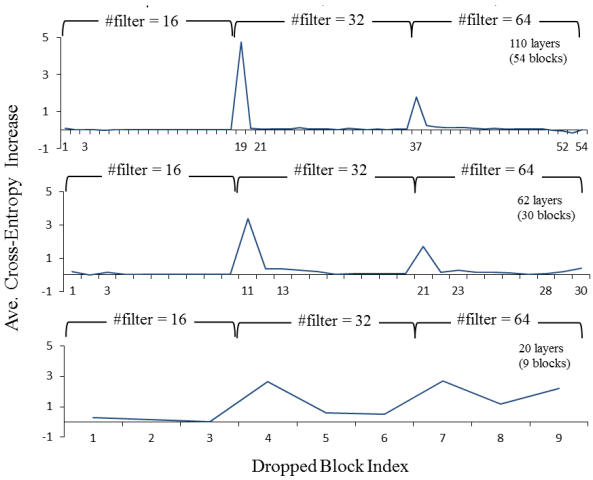

Fig. 4. Cross-Entropy change (CE after dropping - CE before dropping) of three ResNet models with different number of ResBlocks while training on the first 7000 samples of training data.

ResBlock-64, see Fig. 3 top), 62 layers (10 ResBlock-16 + 10 ResBlock-32 + 10 ResBlock-64, see Fig. 3 middle) and 20 layers (3 ResBlock-16 + 3 ResBlock-32 + 3 ResBlock-64, see Fig. 3 bottom).

Every time we drop a single block [4] from one of these seed models, we make a decoding using the modified model. We calculate the averaged frame log-likelihood change from the seed model results and show them in Fig. 3. We decode two data sets: 10 utterances selected from a male speaker in CSJ-Eval01 (the blue solid line in Fig. 3) and 10 utterances selected from a female speaker in CSJ-Eval02 (the red dotted line in Fig. 3). As validation, we repeat the block dropping processing on similar models when training with small data. The cross-entropy changes are shown in Fig. 4. The results suggest the followings:

The contribution from every single block to training and testing is different. Some blocks are playing significant roles in the whole network. Removing them from the network would cause a sharp change to the log-likelihood and the cross-entropy. The positions of these most important blocks are decided by the residual network topology and are independent of the testing data. This observation matches the fact reported in [17]. And these locations are summarized as follows to help us identify the skeleton from a full ResNet.

1. After the feature input (e.g. block 1-3 in Fig. 3 top, block 1-3 in Fig. 3 middle and block 1 in Fig. 3 bottom).
2. Before the network output (e.g. block 51-52-53-54 in Fig. 3 top, block 29-30 in Fig. 3 middle and block 9 in Fig. 3 bottom).
3. Between two group of blocks with different feature-map size (e.g. block 19-37 in Fig. 3 top, block 11-21 in Fig. 3 middle and block 4-7 in Fig. 3 bottom).
4. We also observed some individual blocks having relatively larger influence (although not so obvious compared to the blocks mentioned above) to the training and testing (e.g. block 21,24,31 in Fig. 3 top, block 13,17 in Fig. 3 middle and block 5,6 in Fig. 3 bottom).

We name these blocks playing significant roles in the whole network as "skeleton network", because it includes the paths connecting to the feature input and softmax output and is thus substantial for both training and testing. Similarly, the remaining layers can be regarded as "auxiliary networks".

Table 1. Skeleton Networks and Auxiliary Networks of 110-layer ResNet (not including the first convolutional layer and the last fully-connected layer)

	Num. of Blocks		
	#filter=16	#filter=32	#filter=64
Skeleton network (20% full depth)	2	4	5
Auxiliary network (80% full depth)	all other blocks excluding blocks belonging to the skeleton network		

[4] Dropping block means removing the Conv(3x3)-BN-ReLU-Conv(3x3)-BN sequence from a residual block and remaining the residual connection.

3.2. Proposed Training Method

Inspired by research in [17] and our observation, we propose an incremental training method as follows:

1. Train a seed model of 110-layer ResNet with a small-sized data set.
2. Extract a small number of layers (2 ResBlock-16 + 4 ResBlock-32 + 5 ResBlock-64, 24 layers in total) from the 110-layer seed model as the skeleton network (see Table 1). Then train the skeleton model with the full training data set.
3. Then the auxiliary networks can be copied from the seed model and attached to the trained skeleton network from the last step. The auxiliary network can be trained with less computational resources and the whole very deep network can be constructed efficiently.

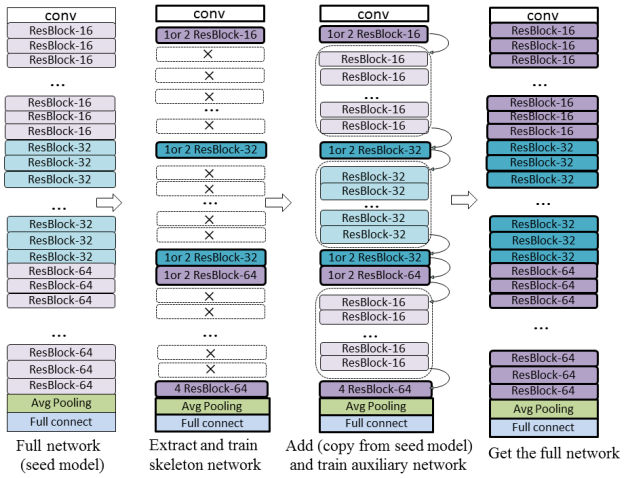

Fig. 5. Incremental training and constructing the very deep network. (We extract the skeleton model from 110-layer seed ResNet.)

The flowchart of this method is shown in Fig. 5, and this method is a little similar to the discriminative pretraining [22] and greedy layer-wise training [23], but is proposed to save computational resources for fine-tuning. We also tried to use a 20-layer ResNet (3 ResBlock-16 + 3 ResBlock-32 + 3 ResBlock-64) directly as the skeleton model and to self-copy the blocks to 110 layers during training. However, the training is very slow and difficult to converge.

4. EXPERIMENTS

4.1. Task and Data Descriptions

In this paper, we focusing applying the ResNet model to Japanese lecture transcription tasks. We tested our proposed method on the "Corpus of Spontaneous Japanese (CSJ)" [24]. The corpus consists of over 600-hour lecture recordings. It has three official evaluation sets (**CSJ-Eval01**, **CSJ-Eval02**, and **CSJ-Eval03**), each containing 10 lecture recordings [25]. We also pick up 10 lecture recordings for development and validation (**CSJ-Dev**). Finally, we used the 240-hour lecture recordings as the training set (**CSJ-Train**) according to [26, 27, 8, 25]. We also select 27.6-hour training data in CSJ (**CSJ-Train$_{\text{small}}$**) to train a seed model without parallelization for warm-start initialization [28]. And this seed ResNet model is also used to select the skeleton networks as described in Subsection 3.1.

Table 2. Data Sets of CSJ

		#Lectures	Hours
Training set	**CSJ-Train$_{\text{small}}$**	155	27.6
	CSJ-Train	957	240
Development set (**CSJ-Dev**)		10	2.0
Testing set	**CSJ-Eval01**	10	2.0
	CSJ-Eval02	10	2.1
	CSJ-Eval03	10	1.4

We train the baseline model using CSJ-Train. We first trained a GMM-HMM model using the MFCC feature with linear discriminant analysis (LDA), a maximum likelihood linear transform (MLLT) and fMLLR based speaker adaptive training (SAT). Then, we train a DNN model with five hidden layers each comprising 2048 hidden nodes. The output layer had about 8522 nodes that corresponded to the tied-triphone states of the GMM-HMM model. We used 24-dim filter-bank features together with its 1st and 2nd order derivatives to train DNN model. All these features are mean and variance normalized per speaker. The filter-bank features of both the previous and subsequent five frames (11 frames of features in total) are added when inputting them into the DNNs. The DNN model is initialized using unsupervised pre-training and supervised fine-tuning using standard stochastic gradient descent (SGD) based on the cross-entropy loss criterion. All these were implemented using the Kaldi toolkit [29].

We trained a 4-gram word language model (WLM) from the transcription of 591-hour CSJ training data. The vocabulary size of the WLM was 98K. These settings are the same as [27]. The DNN-HMM baseline acoustic model is shown in Section 4.3.

4.2. Experimental Settings for ResNet

We use the 110-layer ResNet structure introduced in Section 2. Similar to settings in [7], the features are concatenated into one channel ($static + \Delta + \Delta\Delta$). The label is the same to the state label of the DNN-HMM baseline.

We use the 27.6-hour training data (CSJ-Train$_{\text{small}}$) to train a 110-layer seed model without parallelization for

warm-start initialization. Then the 240-hour training data (CSJ-Train) is used for parallel training.

In the incremental training, the skeleton model is used for warm-up initialization. Then the 240-hour training data (CSJ-Train) is used for parallel training.

Instead of adding the blocks at once, we add the auxiliary blocks incrementally (50% at first and then the other 50%). Each time we add the auxiliary networks to the skeleton network, we slightly tune the whole network with part of the small data ($\text{CSJ-Train}_{\text{small}}$) several epochs and small learning rate before real retraining with 240-hour data set. The long span residual connections in the skeleton model are preserved.

In this paper, model training is implemented with CNTK toolkit [30] and python scripts. To be consistent with our former systems, we use Kaldi IO for both of the feature and label. We use stochastic gradient descent (SGD) with Nesterov momentum [31] and cross-entropy loss. To speed up the experiments, we use the block-wise model update filtering (BMUF) distributed training algorithm, originally proposed in [32]. The full networks and the skeleton networks are all trained on 4 Tesla K40 GPUs in parallel. The initial learning rate for each mini-batch is 0.1 and momentum is 0.9. The batch size is set to 4000. Maximum epoch number is set to 25. Decoding of ResNet model is done by feeding the scaled log-likelihood output of network to the Kaldi decoder [29].

4.3. Experimental Results

In this Subsection (see Table 3), we will report the performance of the 110-layer ResNet acoustic models (**ResNet110**) on the three CSJ evaluation sets. In the conventional training method, we train the **ResNet110** full network acoustic model (**Res110-full**) from a **ResNet110** seed model described in Section 4.2. We also get the seed models of the skeleton network (**Skeleton**) and the auxiliary networks (**Auxiliary**) from the same **ResNet110** seed model to perform incremental training. The WER% on CSJ-Eval01 of the **Skeleton** seed model (16.89%) has close performance compared to the **ResNet110** seed model (14.89%).

For comparison, we also train the conventional DNN-HMM hybrid acoustic model (**DNN-HMM**), the Bidirectional-LSTM-CTC (**BLSTM-CTC**) [27], and recent residual memory network (**RMN** [10], 15 layers, single direction) acoustic models.

As shown in Table 3, we use the word error rate (WER%) to be consistent with our former work [27]. The **BLSTM-CTC** model is decoded with EESEN [33], while other models are applied to Kaldi decoder.

Experiment results confirm that both ResNet110 full network (**Res110-full**) and the skeleton network (**Skeleton**) outperform the **DNN-HMM**, **BLSTM-CTC**, and **RMN**. We also notice that the performance gap between the skeleton networks and full network is small (around absolute 1% WER).

We estimate the training speed by calculating the frames per second on every GPU and the number of epochs. The speed to train the full network (**Res110-full**) with the 240-hour training set is around 290 frames per second on every GPU using 4 GPU in parallel. The whole training time is almost three weeks. On the other hand, the speed of training a skeleton network (**Skeleton**) is 960 frames per second on every GPU (3.3x faster than conventional method) with the same data set and GPUs. When the seed model of auxiliary network (**Auxiliary**) is attached to the fully trained skeleton model (**Skeleton**), we use the 240-hour training data (CSJ-Train) to tune the parameters of the whole network with 8 epochs. And the WER%s can be improved to 12.59% (CSJ-Eval01), 9.88% (CSJ-Eval02) and 14.35% (CSJ-Eval03) which are comparable to the conventional method. In our experiment, the proposed method can approximately save up to 30% of the training time compared to the conventional method. When the full network goes deeper, this method can save more training time.

Table 3. ASR performance (WER%) of acoustic models with different neural network structures (The improvements compared to the best results of conventional method with NO statistically significant difference are shown in bold fonts.)

Network		WER%		
		Eval01	Eval02	Eval03
DNN-HMM		14.40	11.84	15.62
BLSTM-CTC (EESEN prior) [27]		14.19	11.33	16.49
RMN (15 layers)		13.92	11.21	15.18
ResNet110 (conventional)	**Res110-full**	**12.26**	**9.83**	**13.73**
ResNet110 (incremental)	**Skeleton**	13.13	10.41	15.31
	+Auxiliary	**12.59**	**9.88**	14.35

5. CONCLUSIONS AND FUTURE WORK

In this paper, we focus on the efficient method to accelerate training of a very deep convolutional residual network (ResNet) with little loss of precision. We first train the skeleton model with the full training data set. The skeleton models has only a small number of layers. And it includes the paths connecting feature input and softmax output and thus is substantial for both training and testing. Other layers are gradually attached to the skeleton model and trained with relatively less time. Experiments show the model trained with our proposed method obtains comparable performances and much faster training speed than its counterpart trained with the conventional method. Rethinking our proposed method, when training a complex model, we can first train most important parameters (skeleton model) to get the basic decision boundaries, and then carving the more precise decision boundaries with larger parameter sizes. For the future work, we will investigate how to effectively train deeper ResNets with larger data set.

6. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. European Conference on Computer Vision (ECCV)*, 2016.

[5] Y. Wang, X. Deng, S. Pu, and Z. Huang, "Residual convolutional CTC networks for automatic speech recognition," in *arXiv preprint arxiv:1702.07793*, 2016.

[6] M. Bi, Y. Qian, and K. Yu, "Very deep convolutional neural networks for LVCSR," in *Proc. INTERSPEECH*, 2015.

[7] Y. Qian et al., "Very deep convolutional neural networks for noise robust speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 12, pp. 2263–2276, 2016.

[8] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-Attention based End-to-End speech recognition with a deep CNN Encoder and RNN-LM," in *Proc. INTERSPEECH*, 2017.

[9] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proc. IEEE-ICASSP*, 2017.

[10] M. Baskar et al., "Residual memory networks: Feed-forward approach to learn long-term temporal dependencies," in *Proc. IEEE-ICASSP*, 2017.

[11] M. Fujimoto, "Factored deep convolutional neural networks for noise robust speech recognition," in *Proc. INTERSPEECH*, 2017.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *arXiv preprint arXiv:1512.0056*, 2015.

[13] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *arXiv preprint arXiv:1610.02357*, 2016.

[14] G. Huang et al., "Deep networks with stochastic depth," in *Proc. European Conference on Computer Vision (ECCV)*. 2016, Springer International.

[15] D. Chen et al., "Deep networks with stochastic depth for acoustic modelling," in *Proc. Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016.

[16] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 2074–2082.

[17] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 550–558.

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," in *Tech Report*, 2009.

[19] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *arXiv preprint arXiv:1605.07146*, 2016.

[20] O. Russakovsky et al., "Imagenet large scale visual recognition challenge," in *arXiv preprint arXiv:1409.0575*, 2014.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015.

[22] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. IEEE-ASRU*, 2011, pp. 24–29.

[23] Y. Bengio et al., "Greedy layer-wise training of deep networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017.

[24] K. Maekawa, "Corpus of spontaneous japanese: Its design and evaluation," in *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[25] T. Kawahara, H. Nanjo, T. Shinozaki, and S. Furui, "Benchmark test for speech recognition using the corpus of spontaneous japanese," in *Proc. ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[26] T. Moriya, T. Shinozaki, and S. Watanabe, "Kaldi recipe for Japanese spontaneous speech recognition and its evaluation," in *Autumn Meeting of ASJ*, 2015, number 3-Q-7.

[27] N. Kanda, X. Lu, and H. Kawai, "Maximum a posteriori based decoding for CTC acoustic models," in *Proc. INTERSPEECH*, 2016, pp. 1868–1872.

[28] J. Dean et al., "Large scale distributed deep networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2012.

[29] D. Povey et al., "The Kaldi speech recognition toolkit," in *Proc. IEEE-ASRU*, 2011.

[30] A. Agarwal et al., "An introduction to computational networks and the computational network toolkit," in *Microsoft Technical Report MSR-TR-2014-112*, 2014.

[31] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence O($1/k^2$)," *Doklady ANSSSR (translated as Soviet.Math.Docl.)*, vol. 269, no. 3, pp. 543–547, 1983.

[32] K. Chen and Q. Huo, "Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering," in *Proc. IEEE-ICASSP*, 2016.

[33] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-End speech recognition using deep RNN models and WFST-based decoding," in *Proc. IEEE-ASRU*, 2015, pp. 167–174.