Interspeech 2018
2-6 September 2018, Hyderabad

Filter sampling and combination CNN (FSC-CNN): a compact CNN model for small-footprint ASR acoustic modeling using raw waveforms

Jinxi Guo[1], *Ning Xu*[2], *Xin Chen*[2], *Yang Shi*[1], *Kaiyuan Xu*[1] *and Abeer Alwan*[1]

[1]Department of Electrical and Computer Engineering,
University of California, Los Angeles, CA, USA
[2]Snap Inc., Venice, CA, USA

lennyguo@g.ucla.edu, ning.xu@snap.com, xin.chen@snap.com, yangshi5@g.ucla.edu,
kyxu@g.ucla.edu, alwan@ee.ucla.edu

Abstract

Learning an ASR acoustic model directly from raw waveforms using CNNs has proved to be effective, where convolutional layers with learnable filters are able to automatically extract useful features. However, these filters, with independent parameters, can be highly redundant resulting in inefficient systems. In this paper, we propose a novel method to generate CNN filter parameters by first sampling from a low-dimensional parameter space and then using a trainable scalar vector to perform a linear combination. This filter sampling and combination method (denoted as FSC) not only naturally enforces parameter sharing in the low-dimensional sampling space, but also adds to the learning capacity of filters. The FSC-CNN model has a significantly smaller number of parameters and is more efficient compared to conventional CNN models, which makes it feasible for small-footprint ASR. Experimental results on the WSJ LVCSR task show that FSC-CNNs are able to achieve a WER of 3.67 with a standard decoder set-up with only 1.19M nonlinear-layer parameters (better than a strong baseline CNN model with 3.2x more parameters). It also outperforms a CNN model with a similar number of parameters by a relative improvement of 10.26%.

Index Terms: speech recognition, small-footprint, acoustic modeling, raw audio, filter sampling and combination, CNNs

1. Introduction

Recently, a great deal of attention has been paid to training ASR acoustic models directly from raw waveforms. Several types of neural-network architecture have been proposed. In [1, 2], feed-forward DNNs were adopted to learn features from raw waveforms. The authors in [3] proposed a network-in-network architecture that uses CNN filters to extract features from the signals and showed significant improvement over MFCCs. Similarly, shallow CNN models were used in [4, 5] and they are robust to noise. In [6], a combination of convolutional layers and long short-term memory (LSTM) layers was proposed to show the effectiveness of raw-waveform modeling. Moreover, a complex linear projection (CLP) layer with LSTM layers was proposed in [7], which achieves superior performance compared to filterbank features.

While the majority of those systems are focused on large-scale neural network models, few studies have focused on designing a compact model for small-footprint ASR applications, which can directly model the raw-waveform input.

Small-footprint ASR is very important in resource-constrained scenarios which require a smaller size of acoustic models while achieving high recognition accuracy. There have been several studies on small-footprint ASR. In [8, 9], the authors use matrix factorization methods on fully-connected layers to reduce parameters. In [10, 11], techniques based on teacher-student learning have been applied to distill knowledge from large models to small models. [12] investigated the use of low rank displacement of structured matrices for small-footprint networks. However, only a few studies have focused on raw-waveform modeling. The authors in [13] proposed a unified highway network (HW) with a time-delayed bottleneck layer in the middle to model the raw waveform after computing the discrete Fourier transform. The proposed thinner and deeper HW networks with complex DFT features show significant improvement over filterbank features and have smaller footprint. The effect of HW networks for small-footprint ASR is also studied in [14].

In this paper, we first propose a deep 1-D CNN model to extract features directly from raw waveforms. Since learned filters with independent parameters can be redundant, we then propose a novel method to generate CNN filters with a limited number parameters in two steps: first sample the filter parameters from a low-dimensional space; then use a set of trainable scalar vectors to perform a linear combination of the sampled filters. The proposed filter sampling and combination CNN model has a significantly smaller number of parameters compared with a standard CNN model and can still extract useful features from raw waveforms with no loss in accuracy.

2. CNN-based acoustic modeling using raw waveforms

In this section, we introduce our baseline system, which uses a CNN-based neural network architecture and raw waveforms as input for acoustic modeling.

The input features to the neural network are long duration segments (110ms or 1760 samples) of raw waveform signals. The raw waveforms are mean and variance normalized at the speaker level. For the neural network architecture, we propose a deep 1-D CNN model to extract features from raw waveforms as shown in Table 1. This CNN structure has 7 convolutional layers and 2 fully-connected layers. The first three layers have larger filter sizes in order to capture larger reception fields, such that more useful low-level features can be extracted from raw waveforms. The last four convolutional layers have a smaller filter size but a larger number of filters, which can efficiently extract higher-level features. The output of each convolutional layer is fed into a max-pooling layer with stride equal to two, in order to reduce the dimension of the feature maps and extract invariant features. In the end, two fully-connected layers are stacked, which can transform the extracted features into a space for discriminative classification. A fully-connected layer can be also treated as a 1-D CNN layer which has larger-size filters

10.21437/Interspeech.2018-1370

Table 1: *Proposed 1-D CNN structure (the last column shows the number of parameters for each layer).*

Layer	Filter size	#filters	#para
$conv1$	32	32	$1K$
$conv2$	32	64	$65K$
$conv3$	16	128	$131K$
$conv4$	8	128	$131K$
$conv5$	8	256	$262K$
$conv6$	8	512	$1M$
$conv7$	4	512	$1M$
$fc1$	2048	512	$1M$
$fc2$	512	512	$262K$

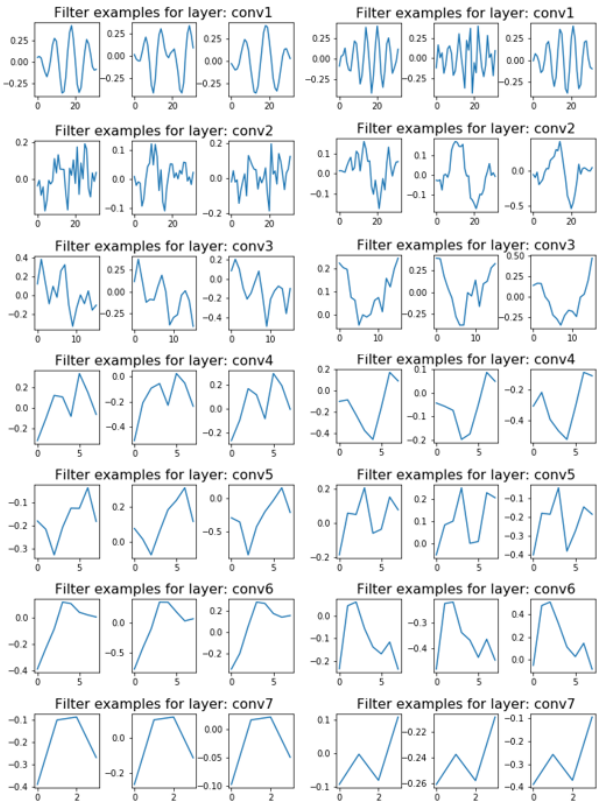

Figure 1: *Filters learned from each conv layer. Each row represents a layer and 6 different filters from that layer are shown.*

with depths equal to one.

In this paper, since we are aiming at designing compact and efficient neural network models for ASR, the parameters of the baseline CNN model are highly optimized and well designed. The total number of nonlinear-layer parameters of the baseline CNN is around 3.84M.

3. Filter sampling and combination CNN

In the following subsections, we will first show the learned filters from raw audio features and their redundancy using the proposed CNN model defined in the previous section, and then we will introduce our proposed filter sampling and combination method in detail.

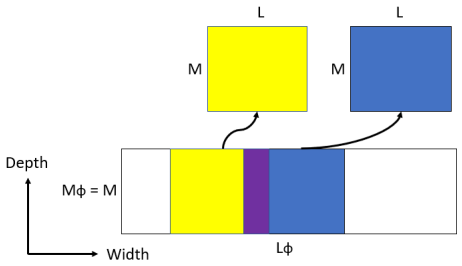

Figure 2: *Widthwise filter sampling in space* Φ.

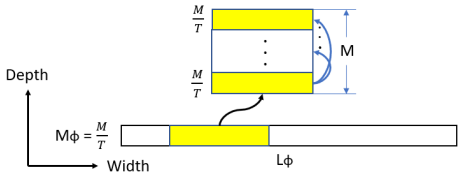

Figure 3: *Depthwise filter sampling in space* Φ.

3.1. Filters learned from raw audio

The proposed 1-D CNN model uses independent filters to learn representations from the input. In order to investigate the relationship between the filters, we plot the filters that are learned from convolutional layers in Figure 1. We select 6 one-dimensional filters from each layer. From the figure, we can observe that the learned filters in the same layer are redundant. Many of the filters have similar shapes but different ranges. This motivates us to use fewer parameters to represent the filters and still keep their learning ability. Therefore, in this paper, we propose a novel filter generation method, which first samples the filter parameters from a low dimensional space of parameters, and then uses a set of trainable scaler vectors to perform a linear combination. By doing filter sampling and combination, we are able to get various filters which share weights in a hidden low-dimensional space. The technique can also alleviate overfitting problems by introducing a smoother loss function, which is modeled with fewer parameters.

3.2. Filter sampling

We introduced the idea of weight sampling in [15], which efficiently reuses the weights among filters. In this paper, we apply the filter sampling step in a similar way.

For a given convolutional layer, the width of a filter is defined as the filter size of each 1-D filter, and the depth of a filter equals the number of feature maps outputted from the previous layer. Let L denote the filter width and M denote the filter depth.

For each convolutional layer, all filters are sampled from a low dimensional filter-sampling space Φ as illustrated in Figure 2. Φ is a two-dimensional space with width L_ϕ and depth M_ϕ.

To start, we set the depth of the sampling space M_ϕ equal to the depth of filter M and only do the sampling along the width. We then use a sliding window with the same size L and depth M as the filters to do the sampling in Φ, as shown in Figure 2. The stride (or the skipping step) of the sliding window is denoted as S. The smaller S is, the more parameter sharing among filters. Suppose that we sample N filters from Φ, then the equation between the width of Φ and the size and stride of

the window is:

$$L_\phi = NS + L - S. \quad (1)$$

The filter sampling method ensures that each generated filter shares the weights with adjacent filters, such that the number of independent parameters is reduced significantly. The compression ratio for widthwise sampling is $\frac{LN}{L_\phi} \approx \frac{L}{S}$, given that N is much larger than L.

Besides the widthwise sampling, we can also perform depthwise sampling by repeating the sampling several times, as illustrated in Figure 3. We set the depth of sampling space M_ϕ to be $\frac{M}{T}$, where T denotes the number of times sampling is repeated (i.e. reusing the same parameters) so that the compression ratio for depthwise sampling is $\frac{M}{M_\phi} = T$. Note that both widthwise and depthwise sampling can be applied simultaneously to generate filters.

3.3. Filter combination

Directly applying filter sampling can reduce the number of parameters significantly. However, simply tying the weights between adjacent filters will limit their learning ability. Therefore, in order to avoid tying parameters directly between filters, we introduce a set of trainable scaler vectors α_i as in Eq.2, where $i = 1, 2, ..., N$ and N is number of filters for a given layer. Each α_i consists of M scalers α_{ij}, where $j = 1, 2, ..., M$, and M is the depth of the filters. Let F_i denote the i_{th} generated filter from the filter sampling step as in Eq.2, where F_{ij} is a filter of size L in j_{th} depth of filter F_i . Each α_{ij} is multiplied to F_{ij} to generate a new set of filters $\hat{F}_i$ as shown in Eq.2.

$$\alpha_i = \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{iM} \end{bmatrix}, F_i = \begin{bmatrix} F_{i1} \\ F_{i2} \\ \vdots \\ F_{iM} \end{bmatrix}, \hat{F}_i = \begin{bmatrix} \alpha_{i1}F_{i1} \\ \alpha_{i2}F_{i2} \\ \vdots \\ \alpha_{i2}F_{iM} \end{bmatrix} \quad (2)$$

The new filter $\hat{F}_i$ can be interpreted as a linear combination of the original filter F_i, and the linear weights are all trainable and optimized based on the final loss of the neural network. The proposed filter combination method ensures that all the generated filters will now have unique parameters and they still naturally share the weights in the hidden sampling space Φ.

By adding a set of scaler vectors α_i, we introduce $M * N$ extra parameters for each layer. The motivation of adding α_i is to ensure that each generated filter will have different parameters to model them, and the filters generated from the filter-sampling step only share the weights with a limited number of filters. Therefore, it is not necessary to use $M * N$ independent parameters to represent α_i, and instead we can tie the weights of α_i, such that each generated filter has a unique combination of weights from Φ and scalers from α_i . Similar to the idea of weight sampling, we can tie the weights of either dimension M or N. If we do the widthwise filter sampling in the first step, we can tie the weights of α_i along dimension N by a ratio of R_N (i.e. weights are repeated every $\frac{N}{R_N}$); if depthwise filter sampling is conducted, the weights of α_i can be tied along dimension M by a ratio of R_M. By applying weight tying, we can significantly reduce the number of parameters needed for the linear combination step.

The above weight sampling and combination method can be conveniently generalized from convolutional layers to fully connected layers. As mentioned in Section 2, for a fully-connected layer, since its input has a single channel, its weights can be treated as filters with depth one. Those filters have large filter sizes which equal to the size (vector dimension) of the input to that layer. Since the depth of the filters already equals to one, we can only perform widthwise filter sampling for fully-connected layers.

4. Evaluation setup

We conduct our experiments on 80 hours of speech data using the Wall Street Journal (WSJ) continuous speech corpus [16]. We use the standard configuration: si284 dataset for training, dev93 for validation and eval92 for testing. The Word Error Rates (WER) on eval92 are reported. We compare with filterbank and raw waveforms based systems. For filterbank feature based systems, we use 40-dim Mel-filterbank features normalized on a per-speaker level, which are then spliced by a context window of 11 frames (i.e. $\pm$5). DNNs and CNNs with various sizes are compared. For raw waveforms based systems, in order to make fair comparison, we use 110ms raw waveforms as input, which covers the same context as Mel-filterbank features. The raw waveforms are also normalized on per-speaker level. For the raw waveforms baseline, we use the 1-D CNN structure as introduced in Section 2. The number of tied tri-phone states is 3362 and all the neural network systems are trained with the same alignment. No speaker adaptation is performed for any of the systems. All experiments in this paper are conducted using the Tensorflow neural network training toolkit [17] with the Kaldi decoder [18].

All the neural networks are trained using the Adam optimization strategy [19] with cross-entropy criterion. The networks are initialized with Gaussian random normal distributed weights with standard deviation equal to 0.01. The relu activation function is used for all layers. For each layer, before passing the tensors to the nonlinearity function, a batch normalization layer [20] is applied to normalize the tensors and speed up the convergence. The shuffling mechanism is applied on each epoch. All neural networks are trained from scratch. For decoding, we use Kaldi WSJ's default setup, which uses trigram language modeling and a large dictionary.

5. Experiments and results

5.1. Mel-filterbank features vs. Raw waveforms

In this section, we compare the performance of using Mel-filterbank features and raw waveforms. For Mel-filterbank features, both DNNs and CNNs with various numbers of parameters are investigated. We also do a VTLN-based data augmentation technique to increase the data by 5 times for filterbank features. For raw waveform features, the proposed 1-D CNN model is used. Table 2 compares results of different setups. From the first two rows, we can observe that for Mel-filterbank features, when decreasing the number of parameters from 17.6M to 3.86M, the performance degrades. Given a similar number of parameters as in rows 2, 3 and 5, 2-D CNNs perform better than DNNs for Mel-filterbank features. When using raw waveforms with the proposed 1-D CNN model, the WER decreases to 3.70, which is 22.6% relatively better than Mel-filterbank with DNNs and 21.4% relatively better than Mel-filterbank with CNNs. The proposed 1-D CNN model without any data augmentation also outperforms a 17.6M-parameter Mel-filterbank-based DNN model with data augmentation by relatively 8% as shown in the 4th row. The results indicate that the proposed 1-D CNN model with raw waveform as input is very efficient for acoustic modeling, and it is able to extract useful features automatically. Therefore, we will use the 1-D

CNN as our baseline system for the following subsections.

Table 2: *Baseline comparison: different features and nerual network structures.*

	Set-up	WER	#para
1	Mel-fbank+5-layer DNN 2048	4.55	$17.6M$
2	Mel-fbank+5-layer DNN 930	4.78	$3.86M$
3	Mel-fbank+2-D CNN	4.71	$3.86M$
4	Mel-fbank+5-layer DNN 2048 +data augmentation	4.02	$17.6M$
5	**raw waveform+1-D CNN**	**3.70**	**3.84M**

5.2. Filter sampling

In this section, we first show ASR performance using the proposed CNN models with decreasing number of parameters in Table 3. We can observe that when the number of parameters decreases from 3.84M to 1.27M (from row 1-4) by reducing the number of filters in the convolutional layers or the hidden nodes in the fully-connected layers, the WERs increase significantly. This result indicates that the number of filters and hidden nodes are very important for this CNN model.

Then, we apply the filter sampling method (denoted as FS) introduced in Section 3.2 to the baseline CNN model. For convolutional layers, we apply filter sampling along either width (denoted as cw in row 5) or depth (denoted as cd in row 6), respectively. For fully-connected layers, we can only do the sampling along width (denotes as fw). Both convolutional and fully-connected layers are compressed with a ratio of 4 (denoted as cw/4, cd/4 and fw/4), and therefore the total number of parameters is reduced by a factor of 4, which is 0.96M. From Table 3, we can see that the performances of filter sampling CNNs (FS-CNNs) degrade compared with the baseline CNNs due to much fewer parameters. However, given similar WERs as in rows 3 and 5, FS-CNNs have only two thirds of the parameters of a standard CNN. Filter sampling along width or depth has comparable performance (compare the last two rows in Table 3).

Table 3: *Filter sampling results for raw waveform CNNs. 'c*' indicates that the convolutional layers have half the number of filters in each layer compared with the baseline CNN. 'cw' and 'cd' represent compressing the parameters in the convolution layers using widthwise and depthwise filter sampling, respectively. 'fw' represents performing widthwise filter sampling in the fully connected layers. '/4' means reducing the number of parameters by a factor of 4.*

	Set-up	WER	#para
1	CNN: 7c+f512-512	3.70	3.84M
2	CNN2: 7c+f512-256	3.88	3.71M
3	CNN3: 7c^*+f512-512	4.00	1.41M
4	CNN4: 7c^*+f512-256	4.09	1.27M
5	CNN+FS (cw/4, fw/4)	4.00	0.96M
6	CNN+FS (cd/4, fw/4)	4.04	0.96M

5.3. Filter sampling and combination

From the previous section, we can notice that only using filter sampling will lead to performance degradation. One of the possible reasons is that tying weights between filters can limit their learning ability in extracting features. Therefore, in this section, we show the results when applying both filter sampling and combination in Table 4. Rows 4 and 8 show the effect of using both filter sampling and combination to generate CNN filters. Clearly, adding a linear combination step significantly improves the performance of the filter sampling CNNs. When performing widthwise sampling, FSC-CNNs improve the performance of FS-CNNs by 5.7%; when performing depthwise sampling, FSC-CNN outperforms FS-CNN by 4.5%. Hence, the improvement is more significant for widthwise sampling.

As mentioned in Section 3.3, we can significantly reduce the number of parameters of the linear combination step, by simply tying the weights of scaler vectors. In Table 4, results for compressing scaler-vector parameters by factors of 2 and 4 (denoted as /2 and /4) are shown in rows 5, 6, 9 and 10. We notice that by using less parameters for a linear combination, the FSC-CNN can achieve even better performance due to less over-fitting. When performing widthwise filter sampling and compressing the linear combination weights by a factor of 2, we achieve a WER of 3.67 with only 1.19M parameters, which is even better than the strong baseline CNN with x3.2 more parameters. When this best performing system is compared with the CNN4 model in 2nd row with a similar number of parameters, the WER decreases by 10.26%. When further reducing the number of parameters to 1.07M as in row 6, the WER increases to 3.81, which is only a small degradation.

Table 4: *Filter sampling and combination results for raw waveform CNNs. 'lin-MxN ' means doing linear combination using MxN different scalers.*

	Set-up	WER	#para
1	**CNN: 7c+f512-512**	3.70	3.84M
2	CNN4: 7c^*+f512-256	4.09	1.27M
3	CNN+FS (cw/4, fw/4)	4.00	0.96M
4	**CNN+FSC (cw/4, fw/4, lin-MxN)**	**3.77**	**1.41M**
5	**CNN+FSC (cw/4, fw/4, lin-MxN/2)**	**3.67**	**1.19M**
6	**CNN+FSC (cw/4, fw/4, lin-MxN/4)**	**3.81**	**1.07M**
7	CNN+FS (cd/4, fw/4)	4.04	0.96M
8	**CNN+FSC (cd/4, fw/4, lin-MxN)**	**3.86**	**1.41M**
9	**CNN+FSC (cd/4, fw/4, lin-M/2xN)**	**3.85**	**1.19M**
10	**CNN+FSC (cd/4, fw/4, lin-M/4xN)**	**3.86**	**1.07M**

6. Discussion

Theoretically, the proposed FSC-CNN can be easily generalized to 2D CNN with time-frequency features. However, we think FSC-CNN is more effective with 1-D CNN and raw-audio input, since the filter size for 1-D CNN is usually quite large in order to capture lager reception fields of the raw waveform. This may result in larger redundancy between the learned filters. Moreover, the proposed FSC-CNN can be combined with other model compression techniques, such as weight quantization, which can further reduce the number of parameters significantly with no loss in accuracy.

7. Conclusion

In this paper, we present a compact FSC-CNN model, which uses a filter sampling and combination method to efficiently generate filters with a relatively small number of parameters but with a strong learning capability. When applying FSC-CNN on WSJ LVCSR task with raw waveforms, it outperforms a strong baseline CNN with 3.2x fewer parameters.

8. References

[1] M. Bhargava, and R. Rose, "Architectures for deep neural network based acoustic models defined over windowed speech waveforms," *Interspeech*, 2015, pp. 6-10.

[2] Z. Tuske, P. Golik, R. Schluter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for LVCSR," *Interspeech*, 2014.

[3] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic modelling from the signal domain using CNNs," *Interspeech*, 2016.

[4] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," *ICASSP*, 2015, pp. 4624–4628.

[5] D. Palaz, R. Collobert, et al, "Analysis of CNN-based Speech Recognition System using Raw Speech as Input," *Interspeech*, 2015.

[6] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," *Interspeech*, 2015.

[7] E. Variani, T. N. Sainath, I. Shafran, and M. Bacchiani, "Complex linear projection (CLP): A discriminative approach to joint feature extraction and acoustic modeling," *Interspeech*, 2016.

[8] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," *Interspeech*, 2013, pp. 2365-2369.

[9] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," *ICASSP*, 2013, pp. 6655–6659.

[10] G. Hinton, O. Vinyals, J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[11] L. Lu, M. Guo, and S. Renals. "Knowledge distillation for small-footprint highway networks," *ICASSP*, 2017, pp. 4820–4824.

[12] V. Sindhwani, T. N. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," *NIPS*, 2015

[13] J. Guo, K. Kumatani, M. Sun, et al, "Time-delayed bottleneck highway networks using a DFT feature for keyword spotting," *ICASSP*, 2018.

[14] L. Lu, S. Renals, "Small-footprint deep neural networks with highway connections for speech recognition," *arXiv preprint arXiv:1512.04280*, 2015.

[15] X. Jin, Y. Yang. N. Xu, et al, "WSNet: Compact and Efficient Networks with Weight Sampling," *arXiv preprint arXiv:1711.10067*, 2017.

[16] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," *Proceedings of the workshop on Speech and Natural Language*, 1992, pp. 357–362

[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, et al, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467*, 2016

[18] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al, "The Kaldi speech recognition toolkit," *Proc. of ASRU*, 2011, pp. 1-4.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.