STREAMING RESLSTM WITH CAUSAL MEAN AGGREGATION FOR DEVICE-DIRECTED UTTERANCE DETECTION

Xiaosu Tong, Che-Wei Huang, Sri Harish Mallidi, Shaun Joseph, Sonal Pareek, Chander Chandak, Ariya Rastrow, Roland Maas

Amazon, USA

ABSTRACT

In this paper, we propose a streaming model to distinguish voice queries intended for a smart-home device from background speech. The proposed model consists of multiple CNN layers with residual connections, followed by a stacked LSTM architecture. The streaming capability is achieved by using unidirectional LSTM layers and a causal mean aggregation layer to form the final utterance-level prediction up to the current frame. In order to avoid redundant computation during online streaming inference, we use a caching mechanism for every convolution operation. Experimental results on a device-directed vs. non device-directed task show that the proposed model yields an equal error rate reduction of 41% compared to our previous best model on this task. Furthermore, we show that the proposed model is able to accurately predict earlier in time compared to the attention-based models.

Index Terms— speech recognition, human-computer interaction, computational paralinguistics

1. INTRODUCTION

The smart-home devices such as Amazon Echo, Google Home, etc. are often used in challenging acoustic conditions, such as a living room with multiple talkers and background media speech. In these situations, it is crucial for the device to respond only to the intended (referred to as device-directed (DD)) and ignore unintended (referred to as non device-directed (ND)) speech. We refer to "device-directed speech detection" as the binary utterance-level classification task, which can be tackled by a binary classifier trained with different types of features. Historically, two main types of features, acoustic features and features from Automatic Speech Recognition (ASR) decoding, are used in the studies of device-directed speech detection [1, 2, 3, 4, 5]. First of all, acoustic features such as energy, pitch, speaking rate, duration and the corresponding statistical summaries are considered in [2]. Other acoustic features such as multi-scale Gabor wavelets are studied in [3]. Secondly, features coming from ASR decoder such as ASR confidence scores and N-grams are also proved to be valuable for the detection task in [2, 3]. Comparing to the acoustic features, however the ASR decoder features are computationally more expensive, and some of them may not even be available until the end of the utterance.

Our previous work [6, 7] investigated the device-directed speech detection task and proposed a classifier that integrates multiple feature sources, including the acoustic embedding from a pretrained LSTM, speech decoding hypothesis and decoder features from an ASR model, into one single device-directed model. In this paper, we focus in particular on the task of learning utterance-level acoustic embeddings to improve the device-directed speech detection accuracy. We consider two aspects: a) the model topology and b) the aggregation method to convert a frame-wise embedding into an utterance-level embedding.

As for aggregation methods, Norouzian et al. [8] showed the attention mechanism applied to the frame-wise output of the network can improve the equal error rate (EER) performance of the classifier. They used acoustic embedding features only and proposed a model topology consisting of a CNN and a bidirectional LSTM for the device-directed speech detection task. Kao et al. [9] compared different aggregation methods on top of the LSTM models for rare acoustic event classification, which is also an utterance classification task. The aggregation methods are applied to either the last hidden unit output h_t or the soft label prediction y_t.

Besides the aggregation mechanism, different model topologies for audio classification tasks are studied in [10, 11, 12, 13, 14, 15]. Cakır et al. [13] proposed a CRNN model structure for the sound event detection task, which is similar to the CLDNN model topology proposed in [16]. Since the results were evaluated at frame-level, no aggregation method was considered after the LSTM component. In [14], the authors explored a CLDNN model with bidirectional LSTM combined with the attention aggregation for an acoustic scene classification task. Ford et al. [10] experimented with different ResNet [17] structures, and concluded that a 50-layer ResNet shows the best performance on an audio event classification task. In [11], instead of stacking the LSTM on top of a CNN component, the authors proposed a parallel structure of LSTM and CNN components. Then, the outputs of LSTM

and CNN are concatenated and fed into the fully connected layers.

In this paper, we evaluate the performance of different model topologies on the device-directedness task using acoustic features only and find the ResLSTM to outperform the ResNet, LSTM, or CLDNN model structures. Secondly, we propose a new mechanism to incorporate historical information within an utterance using frame-level causal mean aggregation. Compared to the attention method used in [7, 8, 9], the causal mean aggregation

- is able to generate prediction at any frame, which could be easily applied for online streaming with much less computation.
- has same performance as attention aggregation when evaluated at the end of an utterance.
- outperforms the attention aggregation when evaluated at early time point of an utterance.

The rest of paper is organized as follows: The network architectures and different aggregation methods are discussed with details in Section 2. Section 3 and 4 presents the experiments setup and correspondingly results. We conclude with Section 5.

2. MODEL ARCHITECTURE

In this section, we will discuss our network architectures and the aggregation methods.

2.1. Model Topologies

Our ResLSTM model consists of one convolutional layer and one batch norm layer followed by six residual blocks and one average pooling layer, as shown in Figure 1. Each residual block has two convolution layers, two batch norm layers, and a residual connection. The second ReLU activation in the residual block is applied after the summation. There are 13 convolutional layers in total. The LSTM component has three unidirectional LSTM layers with 64 units. After the LSTM, there are two fully connected layers with hidden size 64.

2.2. Aggregation

After the frame-level embeddings h_t are generated from the network, an aggregation mechanism can be applied to the h_t. We categorize different aggregation methods into the following groups.

2.2.1. *Simple aggregation*

There are two types of simple aggregation considered in this paper. First, no aggregation is used. During training, we do frame-wise backpropagation on every frame with the frame-level labels which are obtained by repeating the utterance label. During inference, we use the embedding of the last frame as the embedding of the entire utterance. Second one is the global mean aggregation, which calculate the mean of the embedding of all frames as the utterance embedding. Then, we backpropagate once for each utterance with the utterance-level label.

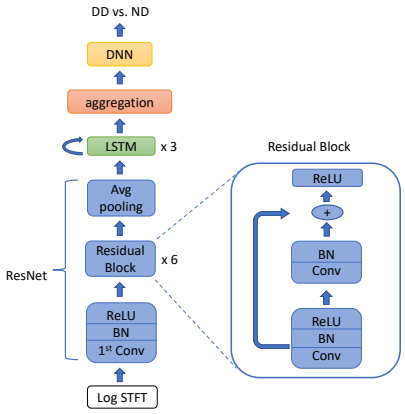

Fig. 1. The structure of the ResLSTM model

2.2.2. *Attention aggregation*

The attention aggregation calculates the utterance-level representation as a weighted average of all h_t. Similar to the global mean aggregation, it uses utterance-level label during training. Our previous work [7] showed that the attention method has better performance than utterance-level embedding with global mean aggregation and frame-wise embeddings without any aggregation.

2.2.3. *Causal mean aggregation*

The drawback of the attention methods used in the previous work [7, 8, 9, 10] is that the attention weights w_t for every frame are calculated once all frames are available, which is not feasible for online streaming tasks. Instead of generating the utterance-level representation at the end of the utterance, we generate frame-level representation by aggregating the past frames. Specifically, we average over all previous $h_i, i \leq t$ until current time point t as the representation of the tth frame. We call this the causal mean aggregation at frame-level:

$$s_t = \frac{1}{t}\sum_{i=1}^{t} h_i = \frac{t-1}{t} \cdot s_{t-1} + \frac{1}{t} h_t \qquad (1)$$

During online inference, we implement the causal mean aggregation with a counter and a mean operation to express

the logic as part of the neural network model definition in order to hide it from the inference engine. We find it is convenient to use two LSTM components, $LSTM_{counter}$ and $LSTM_{mean}$, to achieve this. The LSTM structure, with state values, naturally allows to "side-loading" the frame count to both h_t and s_{t-1}. The $LSTM_{counter}$ is for the frame counting, and the $LSTM_{mean}$ is used for summation and division, which is shown in the Figure 2. The two LSTM components have only one layer with fixed weights shown in Equation 2 and Equation 3, respectively. All the activation function σ in the $LSTM_{counter}$ and $LSTM_{mean}$ components are the LeakyReLU with $\alpha = 1$. The LSTM gates and weights are set as follows:

$$\begin{aligned} &W_f = 0, U_f = 0, b_f = 1 \Rightarrow f_t = 1 \\ &W_i = 0, U_i = 0, b_i = 1 \Rightarrow i_t = 1 \\ &W_o = 0, U_o = 0, b_o = 1 \Rightarrow o_t = 1 \\ &W_c = 0, U_c = 0, b_c = 1, c_0 = 0 \\ &c_t = f_t \cdot c_{t-1} + 1 \cdot (0 \cdot h_t + 0 \cdot h'_t + 1) = c_{t-1} + 1 \\ &h'_t = 1 \cdot c_t = h'_{t-1} + 1 = t \end{aligned} \tag{2}$$

where W, U, and b are weights and bias in the forget gate (f), input gate (i), output gate (o), cell input (c) in the $LSTM_{counter}$, respectively. h_t is the output of the original LSTM component, and h'_t is the output of the $LSTM_{counter}$ component. Then, we concatenate the reciprocal of h'_t with the original h_t as $[h_t, \frac{1}{h'_t}]$ and feed into the $LSTM_{mean}$. Let's assume the dimension of h_t is d, and the $LSTM_{mean}$ component has one LSTM layer with d hidden units.

$$\begin{aligned} &W'_f = [0]_{d\times(d+1)}, U'_f = [0]_{d\times d}, b'_f = [1]_{d\times 1} \Rightarrow f'_t = [1]_{d\times 1} \\ &W'_i = [0]_{d\times(d+1)}, U'_i = [0]_{d\times d}, b'_i = [1]_{d\times 1} \Rightarrow i'_t = [1]_{d\times 1} \\ &W'_o = \begin{pmatrix}[0]_{d\times d} & [1]_{d\times 1}\end{pmatrix}, U'_o = [0]_{d\times d}, b'_o = [0]_{d\times 1} \\ &W'_c = \begin{pmatrix}I_{d\times d} & [0]_{d\times 1}\end{pmatrix}, U'_c = [0]_{d\times d}, b'_c = [1]_{d\times 1} \\ &c'_t = c'_{t-1} + h_t, o'_t = [\frac{1}{t}]_{d\times 1} \\ &s_t = o'_t \circ c'_t = [\frac{t-1}{t}]_{d\times 1} \circ s_{t-1} + [\frac{1}{t}]_{d\times 1} \circ h_t \end{aligned} \tag{3}$$

where W', U', and b' are weights and bias in the forget gate (f), input gate (i), output gate (o), cell input (c) in the $LSTM_{mean}$, respectively. The $\circ$ is the element-wise product, $[k]_{d\times d}$ is a matrix with all elements equal to k and dimension $d \times d$. Similar to the idea showed in [9], we also considered enabling the aggregation after the DNN and apply it to the y_t instead of the h_t.

The LSTM is not the only choice to the frame counter in our implementation. A one-layer RNN with LeakyReLU $\alpha = 1$ can be used to replace the $LSTM_{counter}$:

$$h'_t = W''h_t + U''h'_{t-1} + b'' \tag{4}$$

where h is the output of the original LSTM component, $W'' = 0$, $U'' = 1$, $b'' = 1$, and h' is initialized with 0.

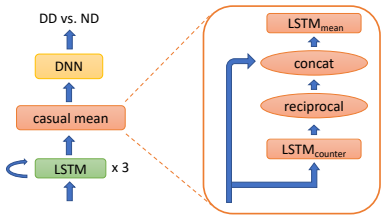

Fig. 2. The implementation of causal mean aggregation for online streaming

Same as $LSTM_{counter}$, the output of the RNN, h'_t is also the frame index t. However, the RNN cannot exactly converge to mimic the $LSTM_{mean}$ because the weight values ($W'' = \frac{1}{t}$ and $U'' = \frac{t-1}{t}$ in Equation 4) are time-dependent, which cannot be achieved by an RNN.

2.2.4. RNN aggregation

In the previous section, we showed how to calculate the embedding at each frame by causal mean aggregation, and potentially use LSTM or RNN as the frame counter in our implementation. Alternatively, one can use a trainable RNN layer as a different aggregation method besides the casual mean to get the aggregated embedding:

$$s_t = \sigma(W^T h_t + U^T s_{t-1}) \tag{5}$$

where W and U are the weights of the representation of the current frame and historical cumulation. The bias term b is set to be 0. Instead of having the weight as fixed or predefined hyper parameter related to t only, we use a one-layer RNN network to learn the weights for us. But the RNN layer potentially suffers from the gradient vanishing problem over time, which we will see in the result session.

2.3. Streaming CNN layer

In order to enable the model with convolutional operations for online streaming, we use a sliding window over the input of each convolutional layer, shifting in the time dimension during inference [18]. As the window is shifting to the right one frame at a time, we drop the oldest computation output from previous window, then cache and feed the rest of output into the next window of the same convolutional layer, which avoids wasting computes on redundant computations. We initialize the "previous" output at the first frame to zeros. As shown in Figure 3, we use one convolutional layer and one residual block with the first three frame inputs as an example for the online inference. For simplicity, we graph the the frequency dimension with size one.

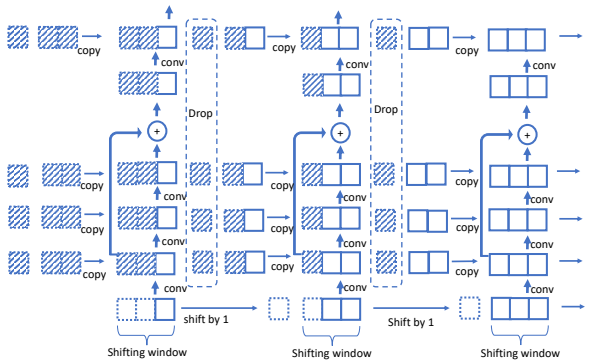

Fig. 3. Streaming convolutional operation. The shaded squares represent the initialized "previous" zero outputs at the first frame. The squares with a dash frame represent the frames of padded zeros, and the squares with a solid frame represent the frames of real input.

3. EXPERIMENTS

We use real recordings of natural human interactions with voice-controlled far-field devices for training and testing the models. The training data consists of $4,000$ hours of audio data comprised of 6M utterances. 4M of the utterances are device-directed examples and the rest of 2M are non device-directed examples. The testing data consists of 35,000 utterances. The binary label of each utterance is created based on human annotation.

The ResLSTM model has a kernel size as 3×3, and stride is 2×1 in which the time dimension stride is always 1. The output channel size of the first convolution layer is 8, and the following 6 residual blocks have $8, 8, 16, 16, 32, 32$ as the corresponding output channel sizes. After the last average pooling layer, we flat out the frequency dimension with the channel dimension, and feed the outputs to the LSTM. We compared our ResLSTM model with LSTM, ResNet, and CLDNN models individually where we fix the aggregation component to be attention. Two LSTM models are considered here. The LSTM-S has 3 LSTM layers of 64 units which is used in [7]. The LSTM-L has 5 layers of 128 units, which is comparable to the ResLSTM model in terms of the number of parameters. The ResNet only model is similar to the one in the [15]. We keep most of the ResNet50 [15] setup the same except the following changes based on our preliminary experiments. First, we set all kernel sizes and strides to be 3×3 and 2×1, respectively. We also remove the max pooling layer after the first convolutional layer. Second, we reduce the channel sizes to be $16, 32, 64, 128$ in each residual block. The CLDNN model we used is similar to the one used in [14]. It has 2 convolutional layers followed by one max pooling layer, and 5 LSTM layers with 128 units. All the convolutional layers are followed by a batch norm layer.

All our models are trained on the 256-dimensional log energy of short-time Fourier transform (log-STFT256) features. For global aggregation methods, such as attention and mean aggregation, the utterance label is used for loss calculation. We use Adam optimizer with the default setting [19] to minimize the cross-entropy loss. We use low frame rate input which has 30ms for each frame. We truncated the input audio at 300 frames (9 seconds) length.

During the training, we feed the entire utterance input to the network. In order to match the training with the online streaming inference, we pad $(kernel\_size-1)$ on the left side of time dimension of the input for every convolutional layer during training. Therefore, all the convolutional layers only see their corresponding inputs from the past but not future frames. We specify the stride in the time dimension of all convolutional layers to be 1.

Table 1. Performance of different model topology with attention aggregation

topology	AUC	EER	ACC	Para
LSTM-S	–	–	–	0.3M
LSTM-L	+7.6%	-22.6%	+5.8%	1.0M
CLDNN	+9.7%	-30.0%	+6.0%	1.1M
ResNet	+11.9%	-38.2%	+8.8%	1.5M
ResLSTM	+12.2%	-41.1%	+8.7%	0.9M

4. RESULTS

We first compare the performance across different model topologies. In the Table 1, we include AUC (area under curve), EER (equal error rate), and ACC (accuracy) as our performance metrics. We also show the number of parameters of each model in the Table 1 We use the results of LSTM-S model as the baseline. Increasing the width and depth of the LSTM-S to LSTM-L does reduce the EER by 22.6% and the number of parameters is increased from 0.3M to 1M. The CNN component in the CLDNN on top of the LSTM-L improves the EER by 30.0%. We also find simply adding more CNN layers in the CLDNN structure does not help to improve the performance on the test dataset. The ResNet only model has 50 convolutional layers, and it improves the EER by 38.2% relatively comparing to the baseline. But the number of parameters is about 1.5M, which is larger than other model topologies. Finally the ResLSTM model, which has 0.9M parameters, improves the EER the most by 41.1%.

Next, we fix the model topology to be the ResLSTM and compared different aggregation methods including frame-level training without any aggregation in the 2.2.1, utterance-level attention in the 2.2.2 and global mean, causal mean in the 2.2.3 and one layer RNN in the 2.2.4. We use the results of the ResLSTM without aggregation as the baseline. We applied the causal mean aggregation on either the LSTM output h_t or the prediction output from the DNN y_t. Results are

Table 2. ResLSTM model with different aggregation methods

aggregation method	AUC	EER	ACC
frame-wise	–	–	–
global mean	+1.3%	-7.8%	+1.0%
attention	+1.3%	-9.0%	+0.6%
causal mean on h_t	+1.3%	-8.5%	+1.1%
causal mean on y_t	+1.5%	-7.8%	+1.7%
RNN-ReLU	0%	+2.4%	-1.0%
RNN-tanh	+0.2%	-0.6%	0%

Table 3. EER at different relative time point of the utterance. L is the full length of an utterance

aggregation method	$0.5L$	$0.6L$	$0.7L$	$0.8L$	L
attention	–	–	–	–	–
causal mean on h_t	-16.0%	-13.3%	-7.6%	-3.6%	+0.6%

shown in the Table 2. As expected, the performance of global mean aggregation and attention method improves the EER by 7.8% and 9%, respectively, which matched the finding in our previous work [7]. The two models with frame-level causal mean aggregation show similar EER performance, which improves the EER by 8.5% and 7.8%. We conclude that the model with causal mean aggregation can achieve similar performance as model with attention when evaluate at the end of utterances. Since there is no significant performance difference between the two causal mean methods, we will use causal mean aggregation on the h_t for the rest of the paper. Using one layer RNN with $tanh$ activation function slightly improves the EER by 0.6% comparing to the baseline. RNN with ReLU activation function performed even worse, increases the EER by 2.4% comparing to the baseline. We believe this is due to the gradient vanish issue of the RNN layer over time.

Instead of evaluating the prediction results at the end of utterance, we also compared the causal mean aggregation to the attention by evaluating the prediction at early frames in an utterance. In Table 4, we evaluate the model performance in the first several seconds of each utterance. The causal mean always performance better than the attention method in terms of EER, especially when evaluating at the first two seconds. Moreover, in Table 3, we compare the two aggregation methods by evaluating the prediction results at different portions of each utterance. For example, $0.5L$ means the middle of an utterance. The causal mean aggregation method still consistently outperforms the attention method. Especially evaluating at middle of the utterance, it reduces the EER by 16% comparing to the attention method.

This robustness property of the causal mean aggregation method is critical for streaming ASR applications for two reasons: Firstly, in practice, the end of the utterance is determined by a separate end-of-utterance detector (aka, end-pointer) for the purpose of ASR and can therefore vary significantly from utterance to utterance. Secondly, depending on the application, an early DD/ND decision can be desirable in order to take action prior to reaching the end of the utterance.

Table 4. EER at different time in seconds since beginning of the utterances.

aggregation method	$1s$	$2s$	$3s$	$4s$	$5s$
attention	–	–	–	–	–
causal mean on h_t	-4.6%	-24.7%	-4.1%	-3.3%	-2.8%

We also tried a causal attention aggregation method, which mask out the future frames for the attention calculation at each frame. But the computational cost is prohibitive, since at every frame, the attention calculation has to be repeated which is much more computationally expensive than causal mean aggregation. We will consider this as a future work to continue seeking solution to reduce the training time.

5. CONCLUSIONS

In this paper, we proposed a ResLSTM model with causal mean aggregation for online streaming classification of device-directed speech detection. Experimental results showed that the ResLSTM model topology outperforms other topologies such as LSTM, ResNet, and CLDNN. We showed how to cache convolutional operations for online streaming inference with CNNs. We also proposed a causal mean aggregation method to obtain a more robust frame-level representation, and showed that causal mean aggregation method can achieve the same performance as the attention aggregation method on full utterances and significantly outperforms attention when used for early decision making, prior to reaching the end-of-utterance.

6. REFERENCES

[1] Daniel Reich, Felix Putze, Dominic Heger, Joris Ijsselmuiden, Rainer Stiefelhagen, and Tanja Schultz, "A real-time speech command detector for a smart control room," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[2] Elizabeth Shriberg, Andreas Stolcke, Dilek Hakkani-Tür, and Larry Heck, "Learning when to listen: Detecting system-addressed speech in human-human-computer dialog," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[3] Tomoyuki Yamagata, Tetsuya Takiguchi, and Yasuo Ariki, "System request detection in human conversation based on multi-resolution gabor wavelet features,"

in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[4] Heeyoung Lee, Andreas Stolcke, and Elizabeth Shriberg, "Using out-of-domain data for lexical addressee detection in human-human-computer dialog," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 221–229.

[5] Dong Wang, Dilek Hakkani-Tür, and Gokhan Tur, "Understanding computer-directed utterances in multi-user dialog systems," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8377–8381.

[6] Sri Harish Mallidi, Roland Maas, Kyle Goehner, Ariya Rastrow, Spyros Matsoukas, and Björn Hoffmeister, "Device-directed utterance detection," *arXiv preprint arXiv:1808.02504*, 2018.

[7] Che-Wei Haung, Roland Maas, Sri Harish Mallidi, and Björn Hoffmeister, "A study for improving device-directed speech detection toward frictionless human-machine interaction," in *Proc. Interspeech*, 2019.

[8] Atta Norouzian, Bogdan Mazoure, Dermot Connolly, and Daniel Willett, "Exploring attention mechanism for acoustic-based classification of speech utterances into system-directed and non-system-directed," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7310–7314.

[9] Chieh-Chi Kao, Ming Sun, Weiran Wang, and Chao Wang, "A comparison of pooling methods on lstm models for rare acoustic event classification," 2020.

[10] Logan Ford, Hao Tang, François Grondin, and James Glass, "A deep residual network for large-scale acoustic scene analysis," *Proc. Interspeech 2019*, pp. 2568–2572, 2019.

[11] Soo Hyun Bae, Inkyu Choi, and Nam Soo Kim, "Acoustic scene classification using parallel combination of lstm and cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 11–15.

[12] Hyungui Lim, Jeongsoo Park, and Yoonchang Han, "Rare sound event detection using 1d convolutional recurrent neural networks," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop*, 2017, pp. 80–84.

[13] Emre Cakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[14] Jinxi Guo, Ning Xu, Li-Jia Li, and Abeer Alwan, "Attention based cldnns for short-duration acoustic scene classification.," in *INTERSPEECH*, 2017, pp. 469–473.

[15] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al., "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

[16] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] "Fifo buffer op, https://github.com/apache/incubator-tvm/pull/4039," .

[19] "Torch optimizer, www.pytorch.org," .