Self-Supervised Learning for Multi-Channel Neural Transducer

Atsushi Kojima

Advanced Media, Inc.

a-kojima@advanced-media.co.jp

Abstract

Self-supervised learning such as wav2vec 2.0 improves accuracy of end-to-end automatic speech recognition (ASR) significantly. Typical self-supervised learning such as wav2vec 2.0 has been applied for single-channel end-to-end ASR model. In this work, we explorer self-supervised learning method for multi-channel end-to-end ASR model based on wav2vec 2.0 framework. As multi-channel end-to-end ASR model, we focus on multi-channel neural transducer. In the experiment, we compared three different methods for feature quantization to train multi-channel conformer audio encoder using contrastive loss: joint quantization, feature-wise quantization and channel-wise quantization. In fine-tuning, we train the multi-channel conformer–transducer. All experiments were conducted using the far-field in-house dataset and CHiME-4 dataset. Results of the experiment showed that feature-wise quantization is the most effective method. We observed 66% relative reductions in character error rate compared with the model without any pre-training in far-field in-house dataset.

Index Terms: Self-supervised learning, wav2vec 2.0, Multi-channel end-to-end speech recognition, Neural transducer, Conformer

1. Introduction

Self-supervised learning improves accuracy of end-to-end automatic speech recognition (ASR) model such as attention-based encoder-decoder [1], CTC [2] and neural transducer [3] significantly. As a self-supervised learning, the most popular framework is wav2vec 2.0 [4]. In wav2vec 2.0 pre-training, the model is trained similar to masked language modeling [5]. In the fine-tunning, the model is trained using ASR loss function. Typical self-supervised learning based on wav2vec 2.0 have been applied mainly for single-channel end-to-end ASR model [4, 6, 7]. In this work, we train multi-channel end-to-end ASR model based on wav2vec 2.0 framework.

Multi-channel end-to-end ASR models can improve robustness for far-field ASR in noisy environments, because the models can capture not spectral, but also spatial information of target and interference signals captured from different microphones [8, 9]. In many end-to-end multi-channel ASR architectures [10, 11, 12, 13, 14, 15], multi-channel neural transducer [14] is promising in terms of efficiency and accuracy.

Multi-channel neural transducer is based on neural transducer such as Transformer–Transducer [16, 17] and Conformer–Transducer [18]. Multi-channel neural transducer can learn the contextual relationship across channels using channel-wise and cross-channel self-attention layers without beamforming [19, 20, 21]. Multi-channel neural transducer overcomes typical multi-channel end-to-end ASR models, which cascaded with neural beamforming [14].

In this work, we train multi-channel neural transducer based on wav2vec 2.0 pre-training. For training, we explorer three quantization methods. First method is joint quantization. Second method is feature-wise quantization. Third method is channel-wise quantization. We will report the results of experiments using far-field in-house dataset and the public CHiME-4 dataset [9].

In the experiment, we show that feature-wise quantization method is the best among quantization methods. We observed 66% and 4.2% relative reductions in character error rate compared with model without any pre-training in far-field in-house dataset and CHiME-4 datasets, respectively.

2. Background

2.1. Multi-channel neural transducer

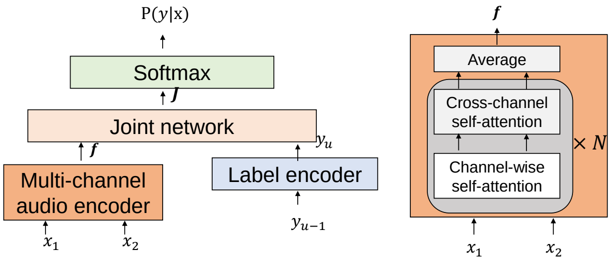


Figure 1: *Architecture of multi-channel neural transducer in the case of two channels.*

We describe architecture of multi-channel neural transducer. Figure 1 shows overview of multi-channel neural transducer in the case of two channels. Given acoustic feature $\boldsymbol{x}$ and previous tokens $\boldsymbol{y}_{u-1}$, the multi-channel audio encoder converts acoustic feature $\boldsymbol{x}$ to hidden vector $\boldsymbol{f}$, and the label encoder predicts a new token $\boldsymbol{y}_U$ based on past tokens except for blank token. The joint network outputs vector J using two hidden vectors from audio and label encoders and softmax outputs logits.

Multi-channel audio encoder consists of channel-wise self-attention layers and cross-channel self-attention layers. In the channel-wise self-attention layers converts inputs from each channels to hidden vectors independently via multi-head attention (MHA) [22]. In the cross-channel self-attention layers learn the contextual relationship across channels. We convert hidden vector h_i in $i-$th channel from channel-wise self-attention layer to query Q_i, and averaging vector of hidden vectors from channel-wise self-attention layers is converted to K_i and value V_i, respectively. MHA is calculated using Q_i, K_i and V_i. Finally, hidden vectors from the cross-channel attention layer are fused by a simple average. For input features, multi-channel audio encoder gets not only amplitude feaetures, but also phase features unlike single-channel neural transducer.

The multi-channel neural transducer is trained using

RNN–Transducer (RNN–T) loss [3]. Given acoustic features $\boldsymbol{x}$ and label sequence $\boldsymbol{y}$, the neural transducer outputs logits $T \times U$. The RNN–T loss is calculated as the sum of probabilities for all paths using a forward–backward algorithm. The RNN–T loss function is written as

$$\lambda_{\mathrm{RNN--T}} = -\sum_{i} \log P(\boldsymbol{y}|\mathrm{x}), \tag{1}$$

$$P(\boldsymbol{y}|\mathrm{x}) = \sum_{\boldsymbol{J} \in \mathcal{Z}(\boldsymbol{y},T)} P(J|\mathrm{x}), \tag{2}$$

where $\mathcal{Z}(\boldsymbol{y}, T)$ is the set of all alignments of length T for the token sequence.

2.2. Self-supervised learning based on wav2vec 2.0 framework

We describe self-supervised learning based on wav2vec 2.0 framework. In wav2vec 2.0 pre-training, the audio encoder is trained by minimizing contrastive loss. Given target quantized feature $\boldsymbol{q}_t$ and K distractors (nontarget quantized features), the model needs to identify true quantized feature in K+1 quantized features $\tilde{\boldsymbol{q}} \in \boldsymbol{Q}_t$. The contrastive loss is calculated as:

$$\lambda = -\log \frac{\exp(sim(\boldsymbol{f}_t, \boldsymbol{q}_t))}{\sum_{\tilde{\boldsymbol{q}} \sim \boldsymbol{Q}_t} \exp(sim(\boldsymbol{f}_t, \tilde{\boldsymbol{q}}_t))}, \tag{3}$$

where f is hidden vectors from masked audio encoder. sim is function for calculating cosine similarity between two vectors: $sim(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T \boldsymbol{b}/||a||||b||$.

Audio encoder consists of Transformer [22] or Conformer [18]. Quantizer consists of quantization layer [4] or linear layer [6].

3. Self-supervised learning for multi-channel neural transducer

3.1. Joint quantization

Figure 2 shows joint quantization. In this figure, $X^{\mathrm{amplitude}}$ and X^{phase} are amplitude and phase features, respectively. $\boldsymbol{f}$ and $\boldsymbol{q}$ are hidden vector from masked features and quantized features, respectively. This example shows 2-channels case. In this approach, quantizer converts concatenated vectors $[{X_1}^{\mathrm{amplitude}}; {X_2}^{\mathrm{amplitude}}; {X_2}^{\mathrm{amplitude}}; {X_2}^{\mathrm{phase}}]$ to quantized vector $\boldsymbol{q}$. Quantizer consists of a single linear layer.

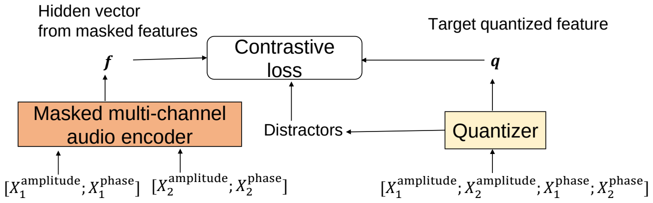

Hidden vector
from masked features
$\boldsymbol{f}$
Contrastive loss
Target quantized feature
$\boldsymbol{q}$
Masked multi-channel audio encoder
Distractors
Quantizer
$[X_1^{\mathrm{amplitude}}; X_1^{\mathrm{phase}}]$
$[X_2^{\mathrm{amplitude}}; X_2^{\mathrm{phase}}]$
$[X_1^{\mathrm{amplitude}}; X_2^{\mathrm{amplitude}}; X_1^{\mathrm{phase}}; X_2^{\mathrm{phase}}]$

Figure 2: *Joint quantization in the case of two channels.*

3.2. Split quantization

3.2.1. Feature-wise quantization

Figure 3 (a) shows feature-wise quantization. This example shows 2 channels case. In this method, quantization modules consists of amplitude quantizer, phase quantizer and joint quantizer. Amplitude quantizer converts amplitude features $[{X_1}^{\mathrm{amplitude}}; {X_2}^{\mathrm{amplitude}}]$ to quantized amplitude features $\boldsymbol{q}^{\mathrm{amplitude}}$. Phase quantizer converts phase features ${X_1}^{\mathrm{phase}}; {X_2}^{\mathrm{phase}}$ to quantized phase features $\boldsymbol{q}^{\mathrm{phase}}$. Joint quantizer gets the two vectors $[\boldsymbol{q}^{\mathrm{amplitude}}; \boldsymbol{q}^{\mathrm{phase}}]$from two quantizers, and converts the vectors to quantized feature $\boldsymbol{q}$. All quantizers consist of linear layers.

3.2.2. Channel-wise quantization

Figure 3 (b) shows channel-wise quantization. In this method, quantization modules consists of channel quantizers, attention and joint quantizer. channel quantizers converts amplitude and phase features ${X_c}^{\mathrm{amplitude}}; {X_c}^{\mathrm{phase}}$ from each input channels to channel-wise quantized features $\boldsymbol{q}_c$. To capture channel the contextual relationship across channels, attention was calculated. For instance, attention for first channel in the case of two channel is calculated as:

$$\boldsymbol{a}_1 = \mathrm{Softmax}(\boldsymbol{w}^T \mathrm{Tanh}(UX_1 + HX_2 + \boldsymbol{b})), \tag{4}$$

where $X_1 = [{X_1}^{\mathrm{amplitude}}; {X_1}^{\mathrm{phase}}]$ and $X_2 = [{X_2}^{\mathrm{amplitude}}; {X_2}^{\mathrm{phase}}]$. The attention is used for calculating weighted quantized vector $\boldsymbol{q_c}^{'}$. Joint quantizer converts weighted quantized features from each channel quantizers to quantized vecor $\boldsymbol{q}$.

4. Experiments

4.1. Data preparation

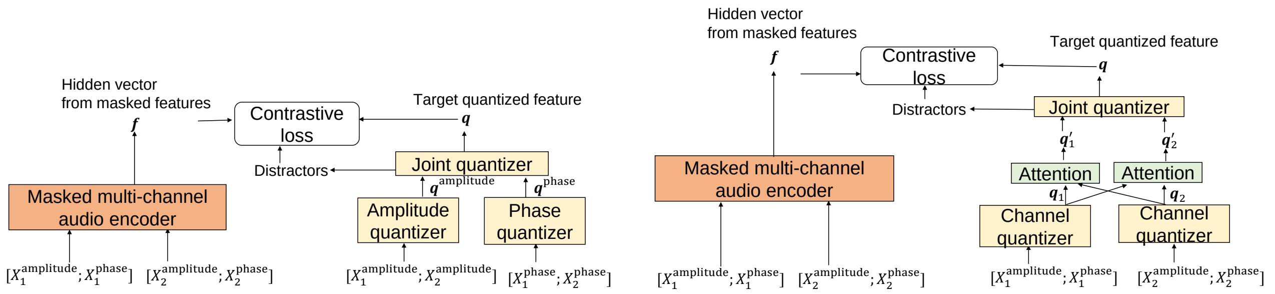
Far-field in-house dataset As the experimental dataset, we use far-field in-house dataset, which consists of 104.3 hours of transcribed speech in Japanese speech. The speech is recorded by a 2-channel linear microphone array, with inter-microphone spacing of 8 millimeters. Amount of train and test data are 102 and 2.3 hours, respectively. For self-supervised learning and fine-tuning, we used train set. For evaluating the model, we use test set. In this dataset, we report character error rate (CER).

CHiME-4 dataset In addition, we use also CHiME-4 datasets for evaluation our proposed method. The speech is recorded by 6 microphones in English speech. For efficient training, we pickup the first and sixth microphone channel as input signals. In this experiment, we use train and eval sets on real data. For self-supervised learning and fine-tuning, we used train set. For evaluating the model, we use eval set. In this dataset, we report word error rate (WER) and CER.

4.2. Model details

We describe architecture of multi-channel neural transducer. we use 8 conformer layers and a unidirectional LSTM layer for multi-channel audio encoder and label encoder, respectively. Table 1 shows the parameters of the Conformer [18] encoder model. The parameter size of the multi-channel Conformer encoder is 15.0 (M). The joint network obtains 512-dimensional vectors from audio and label encoders, and outputs 256-dimensional vector with Tanh activation. Finally, softmax outputs logits.

For amplitude feature, we use log-STFT square magnitude. For phase feature, we use cosine interchannel phase differences (cosIPD) and sinIPD [23].The features are extracted every 10 ms with a window size of 25 ms from audio samples. We set fft size as 512.


Hidden vector
from masked features
f
Contrastive loss
Target quantized feature
q
Distractors
Joint quantizer
$q^{\text{amplitude}}$
q^{phase}
Masked multi-channel audio encoder
Amplitude quantizer
Phase quantizer
$[X_1^{\text{amplitude}}; X_1^{\text{phase}}]$
$[X_2^{\text{amplitude}}; X_2^{\text{phase}}]$
$[X_1^{\text{amplitude}}; X_2^{\text{amplitude}}]$
$[X_1^{\text{phase}}; X_2^{\text{phase}}]$
Hidden vector
from masked features
f
Contrastive loss
Target quantized feature
q
Distractors
Joint quantizer
q'_1
q'_2
Masked multi-channel audio encoder
Attention
Attention
q_1
q_2
Channel quantizer
Channel quantizer
$[X_1^{\text{amplitude}}; X_1^{\text{phase}}]$
$[X_2^{\text{amplitude}}; X_2^{\text{phase}}]$
$[X_1^{\text{amplitude}}; X_1^{\text{phase}}]$
$[X_2^{\text{amplitude}}; X_2^{\text{phase}}]$

(a) Feature-wise quantization.

(b) Channel-wise quantization.

Figure 3: *Split quantization in the case of two channels.*

Table 1: *Multi-channel Conformer encoder architecture.*

Parameter	Value
Number of layers	8
Number of channels	2
Number of heads	8
Head dimension	32
Kernel size	7
Number of hidden nodes	256
Position-wise feed-forward dimension	512

In wav2vec 2.0 pre-training, we train multi-channel audio encoder by minimizing contrastive loss. We masked 50 % of time steps, and set number of distractor as 100. Distractors are uniformly sampled from other masked time steps of the same utterance.

For fine-tunning, we train multi-channel neural transducer by minimizing RNN–T loss. As baseline system, we use multi-channel neural transducer without any pre-traiing. In the far-field in-house dataset, the model outputs 715 characters and blank token. In CHiME-4 dataset, the model outputs 26 lowercase alphabet characters, 3 special tokens (apostrophe, period and whitespace) and blank token. In addition, gradient clipping was applied with a value of 5 to avoid an exploding gradient. We apply SpecAugment [24] for improving robustness. For training all models, we use the Transformer learning schedule [22]. We also use the Adam optimizer [25]. We set $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10$. All networks were implemented using Pytorch [26].

4.3. Results

4.3.1. Far-field in-house dataset

Table 2 shows the result of feature-wise quantization in far-field in-house dataset. In this experiment, we investigate the effect activation function for amplitude and phase quantizers. Comparing to the result of model without any pre-trainng (exp0), we can see improvement by any pre-training methods (exp1, exp2, exp3, exp4). In addition, we can see difference of improvement gain depending on activation function (exp1,2,4). We can observe 66% relative reduction in CER using the quantization method, which uses amplitude quantizer with Swish activation [27] and phase quantizer without any activation (exp4). The CER of the method was lower than the joint quantization (exp3).

Table 2: *The result of feature-wise quantization on far-field in-house dataset. Results are given as relative character error rate reduction (CERR) [%]. The plus sign indicates improvement.*

ID	pre-training	style	amplitude quantizer activation	phase quantizer activation	CERR (%)
exp0	-	-	-	-	0
exp1	✓	split	Swish	Swish	58.1
exp2	✓	split	Swish	ReLU	60.5
exp3	✓	joint	-	-	62.1
exp4	✓	split	Swish	None	**66**

Table 3: *The result of channel-wise quantization on far-field in-house dataset. Results are given as relative character error rate reduction (CERR) [%]. The plus sign indicates improvement.*

ID	pre-training	style	CERR (%)
exp0	-	-	0
exp3	✓	joint	62.1
exp5	✓	split	49.1

Table 3 shows the result of channel-wise quantization. Comparing to the result of model without any pre-trainng (exp0), the quantization method also reduced CER (exp5). Comparing the feature-wise quantization (exp1, exp2, exp4) and joint quantization (exp3), we observe improvement gain was small.

4.3.2. CHiME-4 dataset

Table 4 shows the result of feature-wise quantization in CHiME-4 dataset. Comparing to the result of model without any pre-trainng (expA), all method for pre-training (expB, expC, expD, expE) reduced CER same as far-field in-house dataset. For WER, we can observe 2.4% relative reduction in WER using the quantization method, which uses amplitude quantizer with Swish activation and phase quantizer with Swish activation (expB). For CER, we can observe 4.2% relative reduction in CER using the quantization method, which uses amplitude quantizer with Swish activation and phase quantizer without any activation (expE). Comparing the result of experiment using far-field in-house dataset, improvement gain was small. We concluded that the reason for this is that amount of training data in CHiME-4 dataset is small comparing amount of training data in far-field in-house dataset.

Table 4: *The result of feature-wise quantization on CHiME-4 dataset. Results are given as relative character error rate reduction (CERR) [%] and relative word error rate reduction (WERR) [%]. The plus sign indicates improvement.*

			amplitude	phase		
			quantizer	quantizer		
ID	pre-training	style	activation	activation	CERR (%)	WERR (%)
expA	-	-	-	-	0	0
expB	✓	split	Swish	Swish	3.6	**2.4**
expC	✓	split	Swish	ReLU	2.6	1.7
expD	✓	joint	-	-	3.2	1.4
expE	✓	split	Swish	None	**4.2**	0.1

4.4. Analysis of hidden vectors

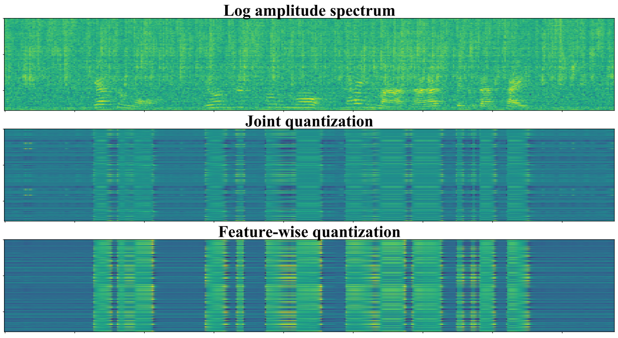


Figure 4: *Analysis of hidden vectors after self-supervised learning.*

We analyze hidden vectors from multi-channel audio encoder after pre-training. Figure 4 compares hidden vectors from multi-channel multi-channel audio encoders trained by different quantization methods using on far-field in-house dataset. Upper figure shows log-STFT square magnitude. Middle figure shows hidden vector from multi-channel audio encoder trained by joint quantization (exp3). Middle figure shows hidden vector from multi-channel audio encoder trained by feature-wise quantization (exp4). Comparing hidden vectors, we can observe more clear contrast between speech and noise section on feature-wise quantization. This multi-channel audio encoder trained by feature-wise quantization learns latent representation better than multi-channel audio encoder trained by joint quantization in terms noise robustness.

5. Conclusion

In this work, we train multi-channel neural transducer based on wav2vec 2.0 pre-training. For training, we explorer three quantization methods. First method is joint quantization. Second method is feature-wise quantization. Third method is channel-wise quantization. We will report the results of experiments using far-field in-house dataset and the public dataset [9]. In the experiment, we show that feature-wise quantization method is the best among quantization methods. We observed 66% and 4.2% relative reductions in character error rate compared with model without any pre-training in far-field in-house dataset and CHiME-4 datasets, respectively.

6. References

[1] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016.

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[3] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[4] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv, abs/1810.04805, 2018.

[6] Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang, Q. V. Le, and Y. Wu, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504*, 2020.

[7] S. Sadhu, D. He, C.-W. Huang, S. H. Mallidi, M. Wu, A. Rastrow, A. Stolcke, J. Droppo, R. Maas, "Wav2vec-C: A self-supervised model for speech representation learning," in *Proc. INTERSPEECH*, 2021.

[8] R. Haeb-Umbach, J. Heymann, L. Drude, S. Watanabe, M. Delcroix, and T. Nakatani, "Far-field automatic speech recognition," Proceedings of the IEEE, 2020.

[9] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," Computer Speech & Language, vol. 46, pp. 535-557, 2017.

[10] T. Ochiai, S. Watanabe, T. Hori, and J. R. Hershey, "Multichannel end-to-end speech recognition," *arXiv preprint arXiv:1703.04783*, 2017.

[11] X. Chang, W. Zhang, Y. Qian, J. Le Roux, S. Watanabe, "MIMO-SPEECH: End-to-end multi-channel multi-speaker speech recognition," in *Proc. ASRU*, 2019.

[12] A. S. Subramanian, X. Wang, S. Watanabe, T. Taniguchi, D. Tran, Y. Fujita, "An Investigation of end-to-end multichannel speech recognition for reverberant and mismatch conditions," *arXiv preprint arXiv:1904.09049*, 2019.

[13] F. Chang, M. Radfar, A. Mouchtaris, B. King, and S. Kunzmann, "End-to-end multi-channel Transformer for speech recognition," in *Proc. ICASSP*, 2021.

[14] F. Chang, M. Radfar, A. Mouchtaris, and M. Omologo, "Multichannel Transformer Transducer for speech recognition," in *Proc. INTERSPEECH*, 2021.

[15] B. Li, T. N. Sainath, R. J. Weiss, K. W. Wilson, and M. Bacchiani, "Neural network adaptive beamforming for robust multichannel speech recognition," in *Proc. INTERSPEECH*, 2016.

[16] C. F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen and M. L. Seltzer, "Transformer–Transducer: End-to-end speech recognition with self-attention," in *Proc. INTERSPEECH*, 2020.

[17] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, S. Kumar, "Transformer Transducer: A streamable speech recognition model with Transformer encoders and RNN–T loss," in *Proc. INTERSPEECH*, 2020.

[18] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, R. Pang, "Conformer: Convolution-augmented transformer for speech recognition,"in *Proc. INTERSPEECH*, 2020.

[19] J. Heymann, L. Drude, and R. Haeb-Umbach, “Neural network based spectral mask estimation for acoustic beamforming,” in *Proc. ICASSP*, 2016.

[20] H. Erdogan, J. Hershey, S. Watanabe, M. Mandel, and, J. Le Roux, “Improved MVDR beamforming using single-channel mask prediction networks,” in *Proc. INTERSPEECH*, 2016.

[21] T. Higuchi, K. Kinoshita, N. Ito, S. Karita and T. Nakatani, “Frame-by-frame closed-form update for mask-based adaptive MVDR beamforming,” in *Proc. ICASSP*, 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in neural information processing systems, 2017.

[23] Z. Wang, J. Le Roux and J. R. Hershey, “Multi-channel deep clustering: Discriminative spectral and spatial embeddings for speaker-independent speech separation,” in *Proc. ICASSP*, 2018.

[24] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in *Proc. INTERSPEECH*, 2019.

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. ICLR*, 2015.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An imperative style, high-performance deep learning library,” in *Proc. NeurIPS*, 2019.

[27] P. Ramachandran, B. Zoph, and Q. V Le, “Searching for activation functions,” *arXiv preprint arXiv:1710.05941*, 2017.