INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Knowledge Distillation for Streaming Transformer–Transducer

Atsushi Kojima

Advanced Media, Japan
a-kojima@advanced-media.co.jp

Abstract

We explore knowledge distillation methods from nonstreaming to streaming Transformer–Transducer (T–T) models. Streaming T–T truncates future context. It leads to recognition quality degradation compared with the original T–T. In this work, we explore knowledge distillation, which minimizes internal representations in all Transformer layers between nonstreaming and streaming T–T models. In the experiment, we compared two different methods: the minimization of the L2 distance of hidden vectors and the minimization of the L2 distance of heads. All experiments were conducted using the public LibriSpeech corpus. Results of the experiment showed that hidden vector similarity-based knowledge distillation is better than multi-head similarity-based knowledge distillation. We observed 3.5% and 2.1% relative reductions in word error rate compared with the original streaming T–T in test-clean set and test-other set, respectively.
Index Terms: end-to-end speech recognition, Transformer–Transducer, knowledge distillation, neural transducer, Transformer

1. Introduction

Recently, Transformer [1], which was originally proposed for neural machine translation, has widely been used for automatic speech recognition (ASR) [2, 3, 4]. Transformer can capture the temporal dependence more effectively than a typical architecture for sequence modeling such as the recurrent neural network (RNN). In many end-to-end ASR architectures [5, 6, 7], Transformer-based models overcome RNN-based models [3, 4, 8, 9, 10, 11]. Among the architectures, Transformer–Transducer (T–T) [8, 9] is promising in terms of efficiency and accuracy.

For many speech transcribing applications, which require rapid response, the manner of streaming is important. The original Transformer is poor at streaming decoding, because it needs the entire input sequence to calculate self-attention. To address this issue, streaming T–T simply truncates self-attention [8, 9]. There is a trade-off between recognition accuracy and latency.

In this work, we explore knowledge distillation from nonstreaming to streaming T–T models. The original knowledge distillation minimizes the Kullback–Leibler (KL) divergence loss between output probabilities from two models [12]. With a neural transducer, this approach is difficult because of differences in the trained alignment of acoustic features and output tokens between streaming and nonstreaming models [13]. Therefore, we explore knowledge distillation, which minimizes differences between internal representations in Transformer layers, inspired by the work of Aihara et al. and Aguilar et al. [14, 15]. In [14, 15], the authors minimize internal representation in long short-term memory (LSTM) and bidirectional encoder representations from Transformers [16] respectively. In this work, we compare two methods. The first method is to minimize the L2 loss of hidden vectors. The second method is to minimize the L2 loss of all heads. We will report the results of experiments using the public LibriSpeech corpus [1].

2. Transformer–Transducer

2.1. Neural transducer

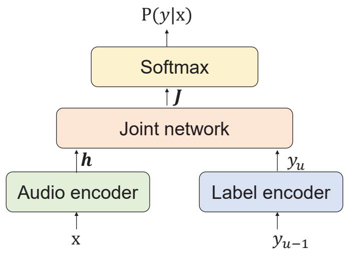

Figure 1: *Architecture of neural transducer.*

The neural transducer is an end-to-end ASR model. Figure 1 shows an overview of the neural transducer. The model consists of a label encoder, an audio encoder and a joint network. Given acoustic feature $\mathrm{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T)$ and previous tokens $\boldsymbol{y} = (y_1, y_2, \cdots, y_{U-1})$, the audio encoder converts acoustic feature $\boldsymbol{x}$ to hidden vector $\boldsymbol{h}$, and the label encoder predicts a new token 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.

The neural transducer is trained using RNN–Transducer (RNN–T) loss [6]. 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.

For inference, the neural transducer performs frame synchronous prediction. Therefore, the model can perform efficient streaming decoding when streaming architecture is chosen for both audio and label encoders.

2.2. Transformer–Transducer

T–T uses Transformer layers for encoders on a neural transducer. The Transformer layer consists of multi-head self-attention and position-wise feed-forward layers. In the

[1]Code available at `https://github.com/atsushiKojima`

multi-head self-attention layer, self-attention is calculated as

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{3}$$

where $Q \in \mathbb{R}^{T_q \times d_k}$, $K \in \mathbb{R}^{T_k \times d_k}$ and $V \in \mathbb{R}^{T_k \times d_v}$ are the query, key and value, respectively. d_k is a parameter. $\text{Softmax}(\frac{QK^T}{\sqrt{d_k}})$ is called self-attention. Again, the Transformer layer has multiple heads:

$$\begin{aligned} &\text{MultiHeadAttention}(Q, K, V) \\ &= \text{Concat}(\text{HEAD}_1, ..., \text{HEAD}_{\text{head}})W^O, \end{aligned} \tag{4}$$

$$\text{HEAD}_i = \text{Attention}(QW_i{}^Q, KW_i{}^k, QW_i{}^V). \tag{5}$$

where head is the number of heads and $W^O \in \mathbb{R}^{d_{in} \times d_{in}}$, $W_i^Q \in \mathbb{R}^{d_{in} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{in} \times d_k}$ and $W_i^V \in \mathbb{R}^{d_{in} \times d_v}$ are model parameters. The original T–T requires an entire input sequence to calculate self-attention. In contrast, streaming T–T truncates self-attention contexts. It leads to recognition quality degradation compared with the original T–T.

3. Knowledge distillation for streaming Transformer–Transducer

We explore internal representation similarity-based knowledge distillation from nonstreaming to streaming T–T models. The original knowledge distillation minimizes the KL divergence loss between output logits from two models. For the neural transducer, this approach is difficult because of differences in the trained alignment of acoustic features and output tokens between streaming and nonstreaming models [13]. Figure 2 shows alignments of streaming and nonstreaming T–T models for the same utterance. For streaming T–T, we set past and future context sizes for 10 and 0 respectively. We can see the difference in alignment. For instance, the length corresponding the first token in streaming T–T is larger than that in nonstreaming T–T. Later, we show that this difference in alignment is detrimental to the recognition accuracy in the experiment. Because of this issue, we explore internal representations similarity-based knowledge distillation. Note that minimizing the KL divergence of self-attention between two models directly is difficult, because streaming T–T truncates contexts in self-attention.

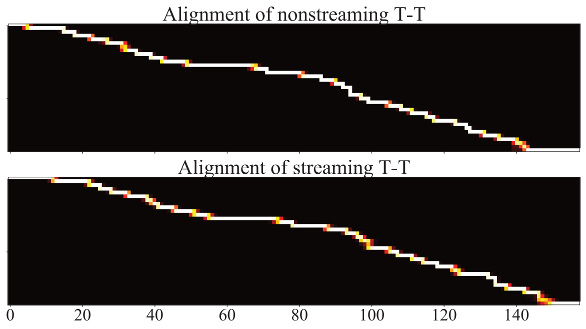

Figure 2: *Alignment difference between streaming and nonstreaming T–T models.*

As internal representations, we focus on two different features: hidden vectors (the output of the final layer) and multi-head. Figure 3 shows an overview of internal representation similarity-based knowledge distillation. In this work, we compare two different methods. The first method is to minimize L2 distance of hidden vectors ((a) in Figure 3). The loss function is written as

$$\lambda_{\text{hidden\_sim}} = \sum_{i=1}^{\text{N}} \sum_{t=1}^{\text{T}} \| \boldsymbol{h}_{i,t}^{\text{teacher}} - \boldsymbol{h}_{i,t}^{\text{student}} \|_2, \tag{6}$$

where $\boldsymbol{h}$ is the hidden vector and N and T are the number of Transformer layers and the length of the input sequence, respectively. The second method is to minimize L2 distance across all heads ((b) in Figure 3). The loss function is written as

$$\lambda_{\text{head\_sim}} = \sum_{i=1}^{\text{N}} \sum_{j=1}^{\text{head}} \sum_{t=1}^{\text{T}} \| \boldsymbol{h}_{i,j,t}^{'\text{teacher}} - \boldsymbol{h}_{i,j,t}^{'\text{student}} \|_2, \tag{7}$$

where $\boldsymbol{h}^{'}$ is the vector in multi-head after residual connection and head is the number of head.

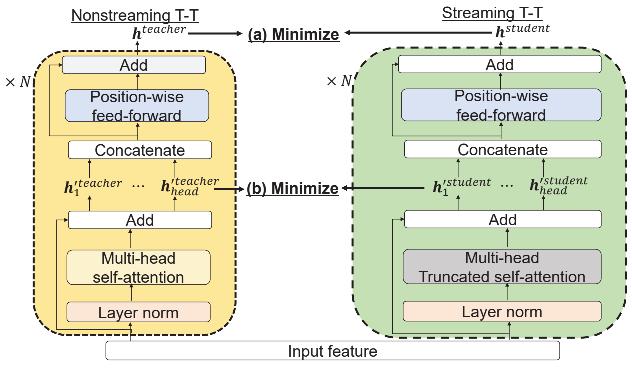
Nonstreaming T-T
$h^{teacher}$
(a) Minimize
Streaming T-T
$h^{student}$
× N
Add
Position-wise feed-forward
Concatenate
$h_1^{\prime teacher}$ ... $h_{head}^{\prime teacher}$
(b) Minimize
$h_1^{\prime student}$ ... $h_{head}^{\prime student}$
Add
Multi-head self-attention
Layer norm
× N
Add
Position-wise feed-forward
Concatenate
Add
Multi-head Truncated self-attention
Layer norm
Input feature

Figure 3: *Overview of internal representation similarity-based knowledge distillation.*

To train the distilled T–T model, we combine RNN–T loss and internal representation similarity loss. Therefore, we define two different loss functions as

$$\lambda_{\text{RNN-T+hidden\_sim}} = \lambda_{\text{RNN-T}} + \alpha\lambda_{\text{hidden\_sim}}, \tag{8}$$

$$\lambda_{\text{RNN-T+head\_sim}} = \lambda_{\text{RNN-T}} + \alpha\lambda_{\text{head\_sim}}, \tag{9}$$

where α is a hyperparameter.

4. Experiments and results

4.1. Corpus and setup

As the experimental dataset, we used the LibriSpeech corpus [17], which consists of 970 hours of transcribed speech. In all experiments, the ASR model output a subword token based on a sentence piece [18] and a blank token. The total number of tokens was 256. The model was trained using train-clean set and train-other set. As input acoustic features, we used a 40-dimensional log Mel filter bank, computed with a 25 ms window and shifted every 10 ms. Every three frames were stacked and these features were downsampled to a 30 ms frame rate [19]. In addition, gradient clipping was applied with a value of 5 to avoid an exploding gradient. Furthermore, we applied SpecAugment [20] to improve robustness. All networks were implemented using Pytorch [21]. For the evaluation metric, we used the word error rate (WER).

4.2. Experimental setup

We use Transformer only as the audio encoder, because Transformer performs well as the audio encoder but not as the label encoder [8, 9]. Table 1 shows the parameters of the Transformer encoder model. For both streaming and nonstreaming models, we use the same encoder architecture. For the decoder, we use a single unidirectional LSTM layer with 256 hidden nodes. The joint network obtains 256-dimensional vectors from audio and label encoders, and outputs 256-dimensional vector with Tanh activation. Finally, softmax outputs 256-dimensional logits. The total parameter size of the model is 13.3 (M).

In streaming T–T, we use two different context sizes: $L = 10, R = 0$ and $L = \infty, R = 0$. L and R are the past and future context sizes respectively. Both settings meet the streaming decoding condition. For training all models, we use the Transformer learning schedule [1]. We also use the Adam optimizer [22]. We set $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10$. For inference, we use beam search with a beam size of 4.

Table 1: *Transformer encoder architecture.*

Parameter	Value
Number of layers	10
Number of heads	4
Head dimension	64
Number of hidden nodes	256
Position-wise feed-forward dimension	2048
Context	$L = 10, R = 0$ / $L = \infty, R = 0$

For reference, we use streaming RNN–T with an architecture of four unidirectional LSTM layers with 650 hidden nodes and a single unidirectional LSTM layer with 256 hidden nodes for audio and label encoders, respectively. We set the parameters to be the same as those of T–T for joint network and softmax output layers. We apply SpecAugment, as in T–T training.

4.3. Results

The results ($L = 10, R = 0$) are summarized in Table 2. The meaning of the terms in the table are shown in Table 3.

Table 2: *Results of knowledge distillation methods ($L = 10, R = 0$). Results are given as relative word error rate reduction (WERR) [%]. The minus sign indicates improvement.*

ID	Description	L	R	α	test-clean	test-other
S1	Transducer	∞	∞	0	0.0	0.0
S2	Transducer	10	0	0	25.4 (0.0)	31.8 (0.0)
S3	KL	10	0	0.001	56.9 (25.1)	45.4 (10.3)
S4	L2_head	10	0	0.001	31.2 (4.6)	31.8 (0.0)
S5	L2_hidden	10	0	0.001	24.1 (**-1.1**)	31.8 (0.0)
S6	L2_hidden	10	0	0.1	21 (**-3.5**)	29 (**-2.1**)

S1 and S2 models are original nonstreaming and streaming T–T models respectively. S3-S6 models are distilled from S1. We can see that hidden vector similarity-based knowledge distillation is better than multi-head similarity-based knowledge distillation. Compared with the original streaming model, multi-head similarity-based knowledge distillation results in slightly lower recognition accuracy in test-clean set. We think that this result was caused by limited future context in self-attention. Figure 4 shows self-attention probabilities in 6th Transformer layer. We can see that self-attention of HEAD2-4 in nonstreaming T–T model has high probabilities in future context, which streaming T–T model can not use. By contrast, we can see a 1.1% relative reduction in WER with minimized hidden vector similarity-based knowledge distillation compared with the original streaming T–T model. KL divergence-based knowledge distillation (S3) also reduces the recognition accuracy. Finally, we can observe 3.5% and 2.1% relative reductions in WER relative to the original streaming T–T by setting α as 0.1 in test-clean set and test-other set, respectively.

Table 3: *Meaning of the terms.*

Term	Meaning
Transducer	The model is trained using the RNN–T loss function.
KL	The model is trained by the original knowledge distillation method, which minimizes the KL divergence of logits between two models.
L2_hidden	The model is trained using loss function Eq. (8).
L2_head	The model is trained using loss function Eq. (9).

Table 4: *Results of knowledge distillation methods ($L = \infty, R = 0$). Results are given as relative word error rate reduction (WERR) [%]. The minus sign indicates improvement.*

ID	Description	L	R	α	test-clean	test-other
S1	Transducer	∞	∞	0	0.0	0.0
S7	Transducer	∞	0	0	6.1 (0.0)	18.9 (0.0)
S8	L2_hidden	∞	0	0.1	4.1 (**-1.9**)	7.8 (**-9.3**)

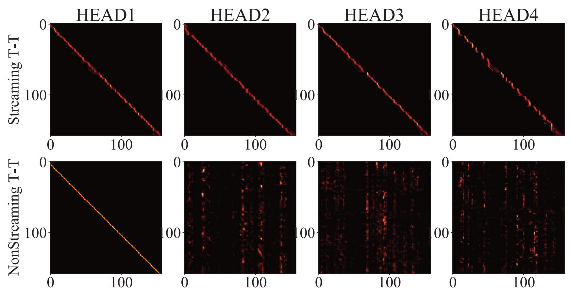

Figure 4: *Comparison of self-attention.*

The results ($L = \infty, R = 0$) are summarized in Table 4. S7 is the standard streaming T–T with the context size $L = \infty, R = 0$. There is a reduction in WER using hidden vector similarity-based knowledge distillation. We can observe 1.9% and 9.3% relative reductions in WER relative to the original streaming T–T in test-clean set and test-other set, respectively.

Table 5 shows results for RNN–T and T–T. When standard streaming T–T and RNN–T have the same context size, T–T is superior to RNN–T. By applying hidden vector similarity-based knowledge distillation, T–T could further reduce WER.

4.4. Analysis of Hidden Vectors in Distilled T–T Model

We analyze hidden vectors in distilled T–T. Table 6 shows mean and standard deviation of L2 distances of hidden vectors nonstreaming and streaming T-T models in test-other set. We

Table 5: *Results for other models. Results are given as relative word error rate reduction (WERR) [%]. The minus sign indicates improvement.*

ID	Model	L	R	Param size (M)	test-clean	test-other
S1	T–T	∞	∞	13.3	0.0	0.0
S7	T–T	∞	0	13.3	6.1	18.9
S8	Distilled T–T	∞	0	13.3	4.1 (0.0)	7.8 (0.0)
S9	RNN–T	∞	0	13.1	6.8 (2.6)	21.3 (12.5)

Table 6: *L2 distances of hidden vectors in test-other set.*

Layer	Distilled T–T (S8)	Original T–T (S7)
1	1.31 ± 0.19	8.13 ± 1.22
2	1.74 ± 0.35	9.33 ± 2.31
3	2.81 ± 0.41	13.37 ± 3.21
4	4.96 ± 0.64	15.80 ± 3.60
5	6.5 ± 0.41	18.93 ± 3.72
6	8.12 ± 0.41	21.95 ± 3.93
7	8.34 ± 0.39	27.56 ± 3.40
8	12.30 ± 0.71	29.92 ± 3.54
9	23.16 ± 2.43	36.90 ± 3.88
10	55.12 ± 8.76	54.37 ± 5.57

can see that L2 distance in the distilled streaming T–T model (S8) is smaller than that in the original streaming T–T model (S7) in most layers. Especially, we found that L2 distances of hidden vectors in lower layers become small.

Figure 5 shows hidden vectors from the 3th Transformer layer for an utterance using t-distributed stochastic neighbor embedding (t-SNE) [23]. t-SNE converts a 256-dimensional hidden vector to a two dimensional vector. We can see that hidden vectors in distilled T–T are distributed closer to hidden vectors in nonstreaming T–T than to hidden vectors in the original streaming T–T. These results suggest that distilled T–T learns hidden vectors similar to those in nonstreaming T–T.

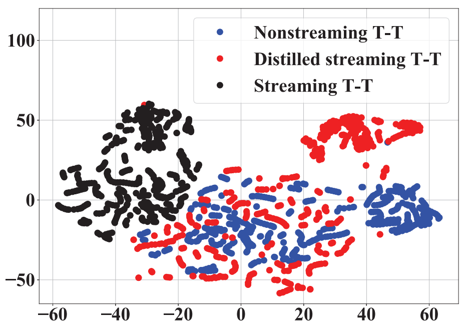

Figure 5: *Example of hidden vectors visualized by t-SNE.*

5. Conclusion

In this study, we explored knowledge distillation for nonstreaming to streaming T–T models. In particular, we examined knowledge distillation that minimizes internal representations in all Transformer layers. In the experiment, we compared two different methods: the minimization of the L2 distance of hidden vector and the minimization of the L2 distance of heads. As a result of the experiment using LibriSpeech, we found that hidden vector similarity-based knowledge distillation is better than multi-head similarity-based knowledge distillation. In the experiment, we observed 3.5% and 2.1% relative reductions in WER compared with the original streaming T–T in test-clean set and test-other set, respectively.

6. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017.

[2] N. Moritz, T. Hori and J. L. Roux, "Streaming Automatic Speech Recognition with the Transformer Model," in *Proc. ICASSP*, 2020.

[3] S. Karita, X. Wang, S. Watanabe, T. Yoshimura, W. Zhang, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. Yalta and R. Yamamoto, "A comparative study on transformer vs rnn in speech applications," in *Proc. ASRU*, 2019.

[4] S. Karita, N. Yalta, S. Watanabe, M. Delcroix, A. Ogawa and T. Nakatani, "Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *Proc. INTERSPEECH*, 2019.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] E. Tsunoo, Y. Kashiwagi, T. Kumakura and S. Watanabe, "Towards online end-to-end Transformer automatic speech recognition," *arXiv preprint arXiv:1910.11871*, 2019.

[11] H. Miao, G. Cheng, C. Gao, P. Zhang and Y. Yan, "Transformer-based online CTC/attention end-to-end speech recognition architecture," in *Proc. ICASSP*, 2020.

[12] G. Hinton, O. Vinyals and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learning Workshop*, 2014.

[13] G. Kurata and G. Saon, "Knowledge distillation from offline to streaming RNN transducer for end-to-end speech recognition," in *Proc. INTERSPEECH*, 2020.

[14] R. Aihara, T. Hanazawa, Y. Okato, G. Wichern and J. L. Roux, "Teacher-student deep clustering for low-delay single channel speech separation," in *Proc. ICASSP*, 2019.

[15] G. Aguilar, Y. Ling, Y. Zhang, B. Yao, X. Fan and C. Guo, "Knowledge distillation from internal representations," in *Proc. AAAI*, 2020.

[16] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional Transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[17] V. Panayotov, G. Chen, D. Povey and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*, 2015.

[18] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” *arXiv preprint arXiv:1808.06226*, 2018.

[19] H. Sak, A. Senior, K. Rao and F. Beaufays, “Fast and accurate recurrent neural network acoustic models for speech recognition,” *arXiv preprint arXiv:1507.06947*, 2015.

[20] 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.

[21] 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.

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

[23] L. Maaten and G. Hinton, “Visualizing data using t-SNE,” Journal of Machine Learning Research, vol. 9, no. Nov, pp. 2579-2605, 2008.