COMPRESSING TRANSFORMER-BASED ASR MODEL BY TASK-DRIVEN LOSS AND ATTENTION-BASED MULTI-LEVEL FEATURE DISTILLATION

Yongjie Lv[1], *Longbiao Wang*[1*], *Meng Ge*[1], *Sheng Li*[2*], *Chenchen Ding*[2], *Lixin Pan*[4], *Yuguang Wang*[4], *Jianwu Dang*[1,3], *Kiyoshi Honda*[1]

[1]Tianjin Key Laboratory of Cognitive Computing and Application, College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]National Institute of Information and Communications Technology (NICT), Kyoto, Japan
[3]Japan Advanced Institute of Science and Technology, Ishikawa, Japan
[4]Huiyan Technology (TianJin) Co., Ltd., Tianjin, China
{yongjie_lv,longbiao_wang}@tju.edu.cn, sheng.li@nict.go.jp

ABSTRACT

The current popular knowledge distillation (KD) methods effectively compress the transformer-based end-to-end speech recognition model. However, existing methods fail to utilize complete information of the teacher model, and they distill only a limited number of blocks of the teacher model. In this study, we first integrate a task-driven loss function into the decoder's intermediate blocks to generate task-related feature representations. Then, we propose an attention-based multi-level feature distillation to automatically learn the feature representation summarized by all blocks of the teacher model. Under the 1.1M parameters model, the experimental results on the Wall Street Journal dataset reveal that our approach achieves a 12.1% WER reduction compared with the baseline system.

Index Terms— speech recognition, feature distillation, model compression, task-driven loss, transformer

1. INTRODUCTION

End-to-end speech recognition has attracted more and more researchers to study and explore because of its simple training process and excellent performance. In particular, the transformer-based models [1, 2, 3] can be easily trained in parallel using GPUs, and achieve remarkable performance compared with models based on DNN-HMM [4]. However, transformer-based models also have a severe drawback: they have a large number of parameters. This causes two main problems. First, the models need large disk space for storage, so they are difficult to deploy on resource-restricted devices, such as mobile phones. Second, large models always have a slow inference speed. Model compression is used to reduce the number of model parameters and control the loss of accuracy.

Many compression methods have been proposed to address the above issues. There are four main types of methods: low-rank matrix decomposition [5, 6, 7, 8], quantization [9, 10, 11], pruning [12], and KD [13, 14, 15, 16]. KD is relatively easy to implement by allowing the student (small) model to imitate the output of the teacher (big) model [13]. To better imitate the behavior of the teacher model, some researches [17, 18, 19, 20] have proposed techniques, such as patient knowledge distillation (PKD), that allow the student model to learn the knowledge of the teacher model's hidden blocks. The PKD-skip method is used to select the same number of blocks from the teacher model as the number of student blocks, then calculate the loss corresponding to these blocks. However, these methods cannot fully use the information of the teacher model's hidden blocks because they only manually select a limited number of intermediate blocks.

In this paper, we propose an approach that consists of two strategies to improve the student model's performance. First, we integrate an auxiliary task-driven loss function into the decoder's intermediate blocks to generate task-related feature representations, which can improve the student model's performance. Second, to take full advantage of the teacher model's information, we propose an attention-based multi-level feature distillation method, which can make use of the information of all blocks of the teacher model. Unlike the previous work for BERT[21], we propose two attention methods, which are used for the encoder and decoder distillation. To the best of our knowledge, this is the first work applying feature-based distillation in transformer-based speech recognition models.

*Corresponding author. This work was supported by the National Key RD Program of China (Grant NO. 2018YFB1305200), Alibaba Group through Alibaba Innovative Research Program and NICT international funding.

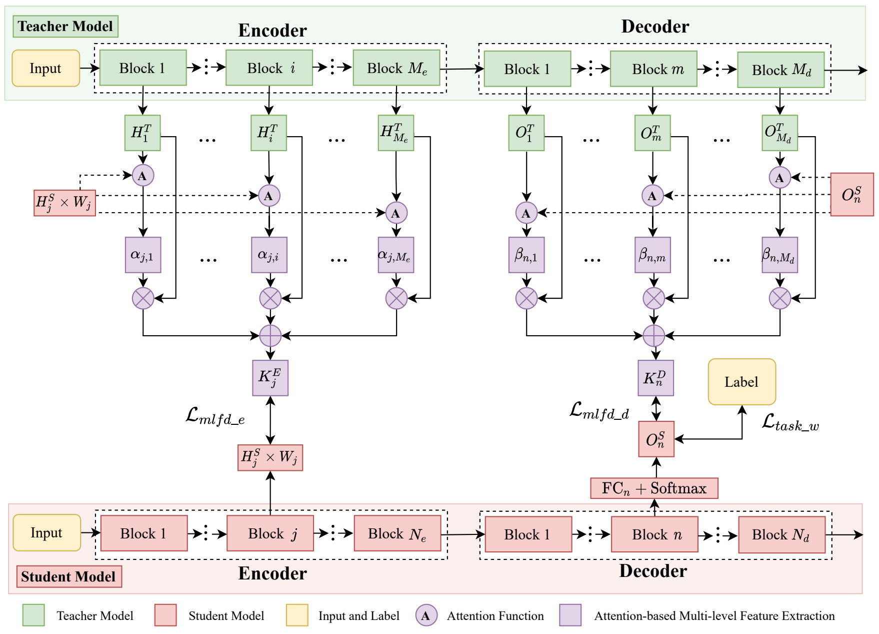

Fig. 1. The overall architecture of our proposed method, including task-driven loss and attention-based multi-level feature distillation. H and O represent the output of encoder block and the decoder block, respectively. For the encoder, the output of student (H_j^S) is projected to match the output dimension of teacher by a projection matrix $\mathbf{W_j}$. The M_e outputs from teacher model is used to calculated the attention weights and generate the knowledge K_j^E for knowledge distillation. For the decoder, the outputs of student and teacher have been matched because of the extra task-driven loss.

2. PROPOSED METHOD

We propose a hybrid method consisting of task-driven loss on decoder and attention-based multi-level feature distillation to improve the student model's performance.

2.1. Task-driven loss

We perform classification only on the output of the last block, and calculate a loss with the ground truth in the ordinary transformer model. The intermediate decoder blocks cannot obtain sufficient supervision information, and therefore the modeling ability of the middle block cannot be improved. This affects the decoding ability of the entire decoder. Considering this, we add an independent classifier to each block of the decoder and supervise it with ground truth y. The following equations describe this new structure:

$$\mathcal{L}_{task} = \frac{1}{N}\sum_{i=1}^{N} \text{CE}(\text{softmax}(\text{FC}_i(z_i)), y)) \tag{1}$$

$$z_i = \text{DecoderBlock}(h, z_{i-1}) \quad z_0 = \text{Embedding}(y)$$

where N is the number of decoder blocks, CE is the cross-entropy loss function, z_i is the output of the i-th decoder block, h is the output of the encoder, and FC_i is the projection layer of the i-th decoder block and projects the decoder block output to the vocabulary dimension.

The classifier of the decoder blocks closer to the top have better classification ability. Therefore, to improve the overall performance of the decoder, the N losses calculated by Eqn. (1) should not have the same weight. The high-level loss should be assigned a larger weight, and a low-level loss should be assigned a smaller weight. Therefore, Eqn. (1) is changed to the following:

$$\mathcal{L}_{task\_w} = \sum_{i=1}^{N} w_i \text{CE}(\text{softmax}(\text{FC}_i(z_i)), y), \sum_{i=1}^{N} w_i = 1 \tag{2}$$

where w_i is the weight of loss corresponding to the i-th block.

2.2. Attention-based multi-level feature distillation

Our attention-based multi-level feature distillation (MLFD) method can apply to both the encoder and the decoder. For the encoder distillation, we note the hidden output of each encoder block as $\{H_1^T, \ldots, H_{M_e}^T\}$ and $\{H_1^S, \ldots, H_{N_e}^S\}$ for the teacher and student, respectively, where M_e and N_e are the number of the encoder blocks of the teacher and the student. Additionally, $H_i^T \in \mathcal{R}^{L_e \times d^T}$, $H_j^S \in \mathcal{R}^{L_e \times d^S}$, where L_e is the length of encoder feature representation. d^T and d^S are the dimensions of the encoder block output. Since d^T and d^S are generally mismatch, we need to apply a learnable parameter matrix $\mathbf{W}$ to project the d^S to d^T. Our MLFD aims to calculate the similarity between the output of each encoder block of the student model and the outputs of all the encoder blocks of the teacher model. It then uses this similarity to perform a weighted summation of the teacher's encoder outputs, and uses the summation result K^E as the knowledge that this student encoder block needs to distill. This method can be described as the following equations and Figure 1.

$$\begin{aligned} K_j^E &= \sum_{i=1}^{M_e} \alpha_{j,i} \odot H_i^T \quad (j = 1, \ldots, N_e) \\ \alpha_{j,i} &= \frac{\exp(\text{Attention}(H_j^S, H_i^T))}{\sum_{i'=1}^{M_e} \exp(\text{Attention}(H_j^S, H_{i'}^T))} \\ \mathcal{L}_{mlfd\_e} &= -\sum_{j=1}^{N_e} \text{MSE}(H_j^S \mathbf{W}_j, K_j^E) \end{aligned} \tag{3}$$

where $\odot$ indicates the Hadamard product. MSE is the mean square error. There are two types attention implementations in Eqn. (3): add attention and dot product attention.

$$\text{Attention}(H_j^S, H_i^T) = \begin{cases} (H_j^S \mathbf{W}_j + H_i^T)\mathbf{P}, & \text{Add attention} \\ \text{Dot}(H_j^S \mathbf{W}_j, H_i^T), & \text{Dot attention} \end{cases} \tag{4}$$

where $\mathbf{P} \in \mathcal{R}^{d^T \times 1}$ and $\mathbf{W}_j \in \mathcal{R}^{d^S \times d^T}$ are learnable parameters, and Dot function means dot multiplication of vectors corresponding to time T in the two matrices.

For the decoder, we do not need to use an extra matrix $\mathbf{W}$ to match the decoder's hidden output dimension because the task-driven loss has projected the decoder's hidden output to the vocabulary size $|\mathcal{V}|$. We note the output of the decoder block as $\{O_1^T, \ldots, O_{M_d}^T\}$ and $\{O_1^S, \ldots, O_{N_d}^S\}$ for the teacher and the student, respectively. M_d and N_d are the number of decoder blocks for the teacher and student. $O_i \in \mathcal{R}^{L_d \times |\mathcal{V}|}$, where L_d is the length of the ground-truth label. We can calculate K_n^D similarly using the following equations.

$$\begin{aligned} K_n^D &= \sum_{m=1}^{M_d} \beta_{n,m} \odot O_m^T \quad (n = 1, \ldots, N_d) \\ \beta_{n,m} &= \frac{\exp(\text{Attention}(O_n^S, O_m^T))}{\sum_{m'=1}^{M_d} \exp(\text{Attention}(O_n^S, O_{m'}^T))} \\ \mathcal{L}_{mlfd\_d} &= -\sum_{n=1}^{N_d} \text{KLD}(O_n^S, K_n^D) \end{aligned} \tag{5}$$

where KLD is the Kullback-Leibler divergence loss function. And attention implementation is similar with Eqn. (4).

$$\text{Attention}(O_n^S, O_m^T) = \begin{cases} (O_n^S + O_m^T)\mathbf{Q} & \text{Add attention} \\ \text{Dot}(O_n^S, O_m^T) & \text{Dot attention} \end{cases} \tag{6}$$

where $\mathbf{Q} \in \mathcal{R}^{|\mathcal{V}| \times 1}$ is a learnable parameter.

So the loss function of our entire model is

$$\mathcal{L} = \lambda \mathcal{L}_{task\_w} + (1 - \lambda)\mathcal{L}_{mlfd\_d} + \gamma \mathcal{L}_{mlfd\_e} \tag{7}$$

where λ and γ can balance three losses.

According to the above equations, the attention weight α and β are optimized during training. Therefore, MLFD can dynamically adjust each teacher block's contribution to the student at each time step or label position. Using MLFD in both the encoder and decoder simultaneously, the student model can learn the information of all the teacher's hidden blocks without wasting teacher information and without the need for heuristic selection from the teacher blocks.

3. EXPERIMENTS

3.1. Experimental settings

We conduct a series of experiments on the Wall Street Journal dataset (LDC93S6B and LDC94S13B). We use train_si284, which contains about 81 hours of speech, as the training set. We use test_dev93 and test_eval92 as the validation set and test set, respectively. Data preparation and feature extraction are performed using the ESPnet toolkit [22]. The input feature is a 40-dimensional fbank with cepstral mean and variance normalization. The output vocabulary has 33 classes, consisting of the 26 English letters (uppercase), apostrophe, period, space, noise, unknown marker, padding marker, and a token marking the beginning or end of a sequence.

All experiments reported in this paper are conducted on the transformer model. For the front end, we use a two-layer CNN with a kernel size of 3 and a stride of 2 for downsampling. We perform the speech features using SpecAugment [23] before they were input to the model. In the training stage, we use the Adam optimizer [24], with $\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$, and a varying warmup learning rate, following [25], with warmup_steps set to 3000. Additionally, we set the attention and residual dropout rate to 0.1, following [1]. The gradient is clipped before the parameter update. The max norm of the gradients grad_clip is set to 5. Finally, we average the model parameters of the ten epochs as the final model.

At the inference stage, the beam search decoding algorithm, with a beam width of 12 and maximum decoding length of 300, is used for decoding test set. And length normalization is used with a length penalty of 1.0.

Both the encoder and the decoder of the teacher model have 6 blocks, and the attention and feedforward network (FFN) dimensions are 256 and 1024, respectively. The teacher model have 15.4M parameters. The student model have 4 encoder blocks and 2 decoder blocks, and both the attention and FFN dimensions are 128. The student model have 1.1M parameters. Additionally, both the teacher and the student model have 4 heads.

3.2. Effectiveness of task-driven loss

We use the original transformer structure as the baseline for both the teacher model and student model. Table 1 shows that the teacher model and student model perform better than

Table 1. Results of applying task-driven loss

Exp. ID	Model Settings (decoder)	Loss ($\mathcal{L}$)	WER%	
			Test	Dev
E0	Teacher, baseline	$\mathcal{L}_{ce}$	14.42	15.57
E1	+task-driven loss, equal weight	$\mathcal{L}_{task}$	14.30	14.66
E2	+task-driven loss, unequal weight	$\mathcal{L}_{task\_w}$	**13.46**	**14.16**
E3	Student, baseline	$\mathcal{L}_{ce}$	28.15	31.77
E4	+task-driven loss, equal weight	$\mathcal{L}_{task}$	27.05	32.00
E5	+task-driven loss, unequal weight	$\mathcal{L}_{task\_w}$	**26.75**	**29.62**

original models after introducing a task-driven loss. Then, we train the models using the Eqn. (2), and the weight w in Eqn. (2) can simply be set by Eqn. (8) in this work.

$$w = \mathrm{softmax}([0, \dots, M-1]) \tag{8}$$

where M is the number of decoder blocks.

After using unequal weights, we find that the performance of both the teacher and the student models have been further improved. To summarize, the WERs of the teacher and student models trained with task-driven loss and unequal weight achieve a 6.7% and 5.0% relative reduction compared with their corresponding baselines.

3.3. Effectiveness of multi-level feature distillation

Both the encoder and decoder are conducted following three experiments: using PKD to distill the selected blocks of the teacher model, using proposed MLFD implemented by two kinds of attention (add and dot product) to learn all the blocks of the teacher model, and all task-driven loss settings used in the decoder-related experiments is similar as E5.

For the experiments on the encoder (E6-E8), we use a 1:0.1 ratio for the ground truth and PKD ($\mathcal{L} = \mathcal{L}_{ce} + 0.1\mathcal{L}_{pkd}$) or MLFD ($\mathcal{L} = \mathcal{L}_{ce} + 0.1\mathcal{L}_{mlfd}$). Additionally, we select one block from every two teacher blocks for PKD training. In Table 2, it is obvious that our propsed MLFD with dot product attention can achieve a 2.6% WER relative reduction compared with PKD method.

For the experiments on the decoder (E9-E10), we use a 0.4:0.6 ratio for task-driven loss and PKD ($\mathcal{L} = 0.4\mathcal{L}_{task\_w} + 0.6\mathcal{L}_{pkd}$) or MLFD ($\mathcal{L} = 0.4\mathcal{L}_{task\_w} + 0.6\mathcal{L}_{mlfd}$). We select one block from every three teacher blocks for PKD training. Table 2 shows that our MLFD with add attention on decoder still has better performance than PKD and achieve 1.6% WER relative reduction. Table 2 also shows that MLFD with dot product attention in the encoder has better performance than add attention, while the MLFD with add attention is better for the decoder.

Table 2. Results of applying MLFD on either the encoder (E6-E8) or decoder (E9-E11) and both of them (E12).

Exp. ID	Model Settings E6-8 (encoder), E9-11 (decoder)	WER%	
		Test	Dev
E6	PKD	27.58	31.44
E7	MLFD (add attention)	27.05	31.45
E8	MLFD (dot-product attention)	26.86	30.82
E9	PKD+$\mathcal{L}_{task\_w}$	25.88	29.27
E10	MLFD (add attention)+$\mathcal{L}_{task\_w}$	25.46	28.90
E11	MLFD (dot-product attention)+$\mathcal{L}_{task\_w}$	26.09	29.80
E12	E8+E10	**24.75**	**28.72**

In the E12, we combine the E8 and E10 trained with a ratio of 0.4:0.6:0.1 for task-driven loss, MLFD in the decoder, and MLFD in the encoder ($\mathcal{L} = 0.4\mathcal{L}_{task\_w} + 0.6\mathcal{L}_{mlfd\_d} + 0.1\mathcal{L}_{mlfd\_e}$), and this achieve the best performance with WER relative 12.1% reduction compared with baseline.

3.4. Further discussion

To further verify the effectiveness of our method, we compare it with the latest ASR model compression result [14] using the sequence-level KD method. We reproduce the sequence-level method on the transformer model using the top-5 teacher decoding results as pseudo-labels. Table 3 shows that our proposed method has comparable accuracy with sequence-level KD. However, under the same computing resources, our method relatively reduces the training time by more than 50% compared with sequence-level KD. In addition, our method only introduces a small additional number of matrix calculations in the attention part compared to PKD, which only increases a slight training overhead.

Table 3. The comparison between our method and sequence-level KD

Exp. ID	Model Settings	WER%	
		Test	Dev
E12	E8 (encoder) + E10 (decoder)	24.75	**28.72**
E13	Sequence-level KD	**24.73**	28.77

4. CONCLUSION AND FUTURE WORK

In this study, we proposed a hybrid method consisting of task-driven loss on the decoder and MLFD on the entire model to make full use of the teacher model information. Our proposed method had better performance compared with PKD, and had comparable performance but lower training costs compared with sequence-level KD. In future work, we will explore novel methods to make better use of teacher information, and we intend to improve the performance of the student model.

5. REFERENCES

[1] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. IEEE-ICASSP*, 2018, pp. 5884–5888.

[2] Abhilash Jain, Aku Rouhe, Stig-Arne Grönroos, and Mikko Kurimo, "Finnish ASR with deep transformer models," in *Proc. INTERSPEECH*, 2020, pp. 3630–3634.

[3] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 5036–5041.

[4] Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "End-to-end speech recognition using lattice-free MMI," in *Proc. INTERSPEECH*, 2018, pp. 12–16.

[5] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE-ICASSP*, 2013, pp. 6655–6659.

[6] Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J Barezi, and Pascale Fung, "On the effectiveness of low-rank matrix factorization for LSTM model compression," in *Proc. PACLIC*, 2019, pp. 253–262.

[7] Rohit Prabhavalkar, Ouais Alsharif, Antoine Bruguier, and Lan McGraw, "On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition," in *Proc. IEEE-ICASSP*, 2016, pp. 5970–5974.

[8] Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung, "Lightweight and efficient end-to-end speech recognition using low-rank transformer," in *Proc. IEEE-ICASSP*, 2020, pp. 6144–6148.

[9] Alex Bie, Bharat Venkitesh, Joao Monteiro, Md Haidar, Mehdi Rezagholizadeh, et al., "A simplified fully quantized transformer for end-to-end speech recognition," *arXiv*, 2019.

[10] Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore, "Efficient 8-bit quantization of transformer neural machine language translation model," *arXiv*, 2019.

[11] Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh, "Fully quantized transformer for machine translation.," *arXiv*, 2019.

[12] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag, "What is the state of neural network pruning?," *arXiv*, 2020.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv*, 2015.

[14] Raden Mu'az Mun'im, Nakamasa Inoue, and Koichi Shinoda, "Sequence-level knowledge distillation for model compression of attention-based sequence-to-sequence speech recognition," in *Proc. IEEE-ICASSP*, 2019, pp. 6151–6155.

[15] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh, "Improved knowledge distillation via teacher assistant," *arXiv*, 2019.

[16] Yoon Kim and Alexander M Rush, "Sequence-level knowledge distillation," in *Proc. EMNLP*, 2016, pp. 1317–1327.

[17] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu, "Patient knowledge distillation for BERT model compression," in *Proc. EMNLP*, 2019, pp. 4314–4323.

[18] Ji Won Yoon, Hyeonseung Lee, Hyung Yong Kim, Won Ik Cho, and Nam Soo Kim, "Tutornet: Towards flexible knowledge distillation for end-to-end speech recognition," *arXiv*, 2020.

[19] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu, "TinyBERT: Distilling BERT for natural language understanding," in *Proc. EMNLP*, 2020, pp. 4163–4174.

[20] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou, "MobileBERT: a compact task-agnostic BERT for resource-limited devices," in *Proc. ACL*, 2020, pp. 2158–2170.

[21] Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu, "ALP-KD: Attention-based layer projection for knowledge distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 13657–13665.

[22] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai, "ESPNet: End-to-end speech processing toolkit," in *Proc. INTERSPEECH*, 2018, pp. 2207–2211.

[23] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[24] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.