Interspeech 2022
18-22 September 2022, Incheon, Korea

Incremental Layer-Wise Self-Supervised Learning for Efficient Unsupervised Speech Domain Adaptation On Device

Zhouyuan Huo, Dongseong Hwang, Khe Chai Sim, Shefali Garg, Ananya Misra, Nikhil Siddhartha, Trevor Strohman, Françoise Beaufays

Google LLC, USA

Abstract

Streaming end-to-end speech recognition models have been widely applied to mobile devices and show significant improvement in efficiency. These models are typically trained on the server using transcribed speech data. However, the server data distribution can be very different from the data distribution on user devices, which could affect the model performance. There are two main challenges for on device training, limited reliable labels and limited training memory. While self-supervised learning algorithms can mitigate the mismatch between domains using unlabeled data, they are not applicable on mobile devices directly because of the memory constraint. In this paper, we propose an incremental layer-wise self-supervised learning algorithm for efficient unsupervised speech domain adaptation on mobile devices, in which only one layer is updated at a time. Extensive experimental results demonstrate that the proposed algorithm achieves a 24.2% relative Word Error Rate (WER) improvement on the target domain compared to a supervised baseline and costs 95.7% less training memory than the end-to-end self-supervised learning algorithm.

Index Terms: speech recognition, domain adaptation, layer-wise, self-supervised, low-resource

1. Introduction

Streaming end-to-end speech recognition models have been widely applied on mobile devices and show significant improvement in efficiency [1, 2]. The deployed models trained using the transcribed speech data in the server could perform badly if the speech data distribution on the user's device is very different from the training data. We can treat adapting the model from the server data distribution (source domain) to the on-device data distribution (target domain) as a domain adaptation problem, where the domain discrepancy may come from different application domains, noise conditions or utterance lengths.

The speech domain adaptation task has been widely researched, including transfer learning [3], fine-tuning factorized hidden layer [4]. However, these methods require transcribed speech data from the target domain and cannot be applied to the unsupervised speech domain adaptation task. Recent results showed that self-supervised pre-training on untranscribed speech data improve speech recognition accuracy without much supervised data [5], including Autoregressive Predictive Coding (APC) [6], Contrastive Predictive Coding (CPC) [7] and Wav2vec2.0 [8]. Federated learning [9, 10] enables us to update models on mobile devices without compromising data privacy by keeping the training data decentralized. It has been researched or successfully deployed in different applications, e.g. emoji prediction [11], next-word prediction [12], and speech recognition [13]. There are two main challenges for federated speech domain adaptation on mobile devices: limited reliable labels and limited training memory. The former can be mitigated using self-supervised learning algorithms, which have been shown to be effective for domain adaptation using unlabeled data [5]. To address the memory requirement for on-device training, we propose using an incremental layer-wise training approach.

Layer-wise unsupervised pre-training followed by supervised fine-tuning enabled the successful training of deep neural networks before 2010s [14]. Authors of [15] empirically show that the layer-wise unsupervised pre-training can serve as a regularizer and adds robustness and generalization to deep neural networks, which does not disappear with more data. Recent research on layer-wise learning focuses more on faster distributed training, including both supervised learning [16] and unsupervised learning [17]. However, utilizing the layer-wise technique to reduce training memory for efficient unsupervised speech domain adaptation task on mobile devices is still an under-explored area.

In this paper, we propose an incremental layer-wise self-supervised learning algorithm to reduce the training memory cost for efficient unsupervised speech domain adaptation on mobile devices. Specifically, we update only one or a few layers at each step and all the layers are pre-trained in an incremental schedule from bottom to top after given training steps. Extensive experimental results demonstrate that the proposed algorithm obtains a Word Error Rate (WER) on the target domain 24.2% better than supervised baseline and costs 89.7% less training memory than the end-to-end self-supervised learning algorithm.

The rest of the paper introduces three self-supervised losses in Section 2, describes our proposed algorithm in Section 3, specifies the experimental setup (model, data and losses) in Section 4 and presents the results in Section 5. We conclude in Section 6.

2. Self-Supervised Losses

We consider three popular self-supervised losses for speech, Contrastive Predictive Coding (CPC) [7], Wav2vec2.0 [8] and Autoregressive Predictive Coding (APC) [6].

CPC [7] minimizes a probabilistic contrastive loss to induce the latent space to capture information that is maximally useful to discriminate future samples from negative samples. In the paper, we let CPC predict the future 12 frames and discriminate each future frame from 8 negative frames sampled from the same sequence.

Wav2vec2.0 [8], on the other hand, learns a good representation to discriminate the latent vectors of masked samples from negative samples. Because we use stacked log-Mel spectrograms as the input, we do not add convolutional layers to extract features as in [8]. We also remove the quantization module

10.21437/Interspeech.2022-10904

and use the log-Mel feature as the latent vector since no differences are observed in our task.

Unlike the contrastive loss in CPC or Wav2vec2.0, APC [6] is trained to encode the given speech utterances and predict future frames, where ℓ_1 or ℓ_2 loss is applied. In the paper, we use ℓ_2 loss and add a total variation regularization [18] between consecutive frames, with regularization weight 0.1.

3. Incremental Layer-Wise Self-Supervised Learning

One major issue with federated learning of speech models is the limited memory and compute resources available on mobile devices. Self-supervised learning offers the flexibility of attaching the loss to an arbitrary layer of the encoder and truncating the input sequence since there is no need to worry about the alignment between the input and label sequences (which is needed for supervised training). This flexibility allows the training memory to be controlled accordingly.

Self-supervised learning can be performed in a layer-wise manner, where one or more selected encoder layers are updated at each step. By attaching the self-supervised loss on top of the selected encoder layers, the layers above the selected layers need not be included in the training graph, and there is no need to compute the gradients for the layers below the selected layers. To reduce the training memory consumption, we propose the incremental layer-wise self-supervised learning algorithm, which updates only one or a few selected layers at each step and all the layers are pre-trained in an incremental schedule from bottom to top where a block of layers is pre-trained for a predefined number of steps before moving on to the next block.

Figure 1 visualizes the proposed incremental layer-wise self-supervised learning algorithm. At step n, we fix the parameters of the first three layers, drop the layers after layer 4 and update layer 4 by attaching a self-supervised loss on top of it. Since layers 1 to 3 are not updated, we only need to compute the gradients for layer 4. After pre-training layer 4 for r steps, we fix the parameters of the first four layers and update layer 5 only by attaching a self-supervised loss on top of it. All the layers can be updated from the bottom layers to the top layers progressively until all the layers are pre-trained. The incremental layer-wise self-supervised learning algorithm offers us a memory- and compute-efficient way to perform unsupervised domain adaptation, which is particularly important for on device learning where human transcriptions are not available. In the next sections, we will demonstrate that our algorithm can achieve comparable results to the end-to-end self-supervised learning with much less memory.

The incremental layer-wise self-supervised learning algorithm is very memory-efficient since we only update one layer at each step without storing activations for all the fixed layers. In the best case, the training memory requirement for updating layer 1 is roughly $\frac{1}{L}$ the training memory required to update all the L layers in an end-to-end fashion. When updating the intermediate layer l, we can save more time and memory by performing the feed-forward of the first $l-1$ layers using quantized weights.

4. Experimental Setup

4.1. Model Architecture Details

Following [2, 19], we use a Conformer RNN-T model in the experiments. The encoder has 17 Conformer layers with model

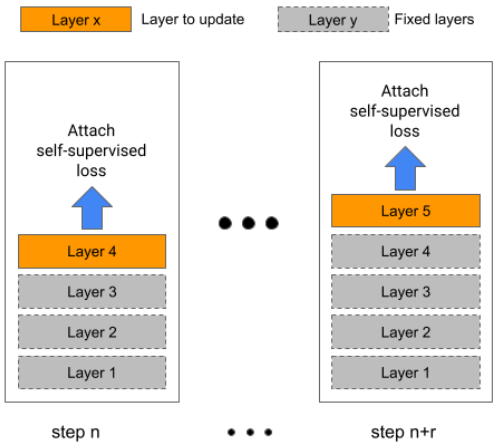

Figure 1: *Incremental layer-wise self-supervised learning.*

dimension 512. To restrict the model from using any future information as [2], each conformer layer is streaming and contains causal convolution, left-context attention layers and feed-forward dense layers. The convolution kernel size is 15 and the self-attention layer consists of 8 heads with 65 left context length. As the model is causal, the right context length is 0. The RNN-T decoder consists of a prediction network with 2 LSTM layers with 2048 units projected down to 640 output units, and a joint network with a single feed-forward layer with 640 units. The model is trained to predict $4,096$ word pieces. The model input is a vector of size 528, consisting of 4 contiguous frames of 128-dimension logMel features [20] sub-sampled by a factor of 3 and one-hot domain-id vector of size 16.

4.2. Data Sets

The unsupervised speech domain adaptation task in the paper utilizes the multi-domain utterances as described in [20]. These multi-domain (MD) utterances span domains of search, farfield, telephony and YouTube. All datasets are anonymized and hand-transcribed. As shown in Table 1, source domain data which has transcriptions is composed of Multi Domain (MD) data excluding Medium-form (MF) utterances. The target domain without transcriptions is the Medium-form utterances and we would like to improve performance on it. To make the Conformer RNN-T model general to both the source and target domains, we first pre-train the encoder layers of Conformer RNN-T model on MD data using self-supervised learning, and then fine-tune the encoder and decoder layers using the RNN-T loss on the source domain only (MD-MF). For performance evaluation, we calculate the Word Error Rate (WER) of Medium-form (MF) to measure the performance on target domain and the WER of Short-form (SF) for the performance on the source domain.

5. Experimental results

In this section, we conduct extensive experiments to evaluate the incremental layer-wise self-supervised learning algorithm on the unsupervised speech domain adaptation task.

5.1. Unsupervised Domain Adaptation Comparisons

In the paper, we assume that there are transcribed source domain data and untranscribed target domain data as shown in Table 1. Existing popular domain adaptation methods e.g. transfer

Table 1: *Overview of training data sets. MD-MF denotes the source domains utterances with transcriptions, MF denotes the target domain utterances without transcriptions. MD-MF and MF are subsets of MD. SF is a subset of MD-MF.*

Data set	Transcribed	Hours
Multi-domain (MD)	Yes & No	400k
Medium-form (MF)	No	26k
MD - MF	Yes	374k
Short-form (SF)	Yes	27k

Table 2: *Comparisons of Word Error Rate (WER) between supervised, semi-supervised baselines and three self-supervised learning algorithms. In parenthesis, the relative improvement with respect to the supervised baseline.*

Algorithms	Word Error Rate (%)	
	MF	SF
Supervised	6.2	6.3
Semi-Supervised	4.9	6.1
APC + Fine-tune	**4.5 (-27.4%)**	6.1
CPC + Fine-tune	**4.5 (-27.4%)**	6.1
Wav2vec2.0 + Fine-tune	4.6 (-25.8%)	6.3

learning [3], fine-tuning factorized hidden layer [4] cannot be used in this case, because they require transcribed data from the target domain. Semi-supervised learning and self-supervised learning methods are a great fit for this situation because neither of them require transcribed target data. Semi-supervised learning methods can learn from pseudo-transcriptions generated by the teacher network and self-supervised learning methods learn representations using speech only. In Table 2, we compare the performance of supervised, semi-supervised and three end-to-end self-supervised learning algorithms. The first row is the baseline model trained with supervised learning on the source domain (MD-MF) only. The second row shows the results of semi-supervised learning where the teacher is the supervised model trained on MD-MF. For self-supervised learning, we first perform self-supervised pre-training on MD data, and then fine-tune on the source domain (MD-MF) only. It is evident that self-supervised pre-training using APC, CPC, or Wav2vec2.0 algorithms can improve the WER performance on the target domain (MF) by a large margin compared to the supervised baseline using source domain data only. They also perform as well or better than the baseline on the source domain. We observe that the uni-directional self-supervised objective functions (APC, CPC) works better than the bi-directional objective function (Wav2vec2) when training a streaming model.

5.2. Comparison with Memory-Efficient Methods

We compare the proposed incremental layer-wise self-supervised learning with related memory-efficient methods for deep learning optimization. Gradient accumulation (GA) [21] splits a batch into multiple mini-batches and accumulates the gradients before updating model weights. Extremely, the training memory cost is lowest if the mini-batch (mb) size is equal to one. Instead of updating all variables, BitFit [22] reduces the training memory cost by updating the bias term only. Gradient checkpoint (GC) [23] drops intermediate activations at a cost of extra forward pass. Table 3 shows training memory cost comparison of the Conformer model with batch size 5 and input length 686. They are obtained by running an Android emulator which is only available if the updated layers per step is less than 5. The training memory 8508MB of the end-to-end algorithm which updates 17 conformer layers at a time is estimated by extrapolating the data points using linear regression, because it exceeds the memory available. Results in Table 3 demonstrate that the proposed incremental layer-wise algorithm achieves better memory reduction than compared methods. Although GA can reduce memory by using batch size 1, it still takes a lot of memory for immediate activations because the model is deep. BitFit still takes as much memory as the end-to-end training, because it only reduces the memory for the updated variables and most of the memory cost is from activations. Gradient checkpoint improves the training memory cost greatly, however, it is still too high for the on-device training. The memory cost for layer-wise method ranges from 748MB to 876MB when updating from layer 1 to 17 which involves different sizes of quantized models. Furthermore, combining layer-wise with GA and GC methods can save more training memory, taking only 157MB to 369MB. Regarding the WER performance, GA or GC behaves the same as end-to-end training since there is no batch normalization in the model. We did not test BitFit because it does not save memory. Results in Section 5.3 demonstrate that layer-wise algorithm can also obtain performance as good as end-to-end training.

Table 3: *Training memory cost comparisons of the memory-efficient methods with batch size* 5 *and input length* 686.

Memory-Efficient Methods	Memory Cost (MB)
-	8508
GA (mb=1)	2704
BitFit	8069
GC	1532
Layer-wise	**748** $\sim$ **876**
Layer-wise + GA (mb=1)	**222** $\sim$ **434**
Layer-wise + GA (mb=1) + GC	**157** $\sim$ **369**

5.3. Layer-Wise Self-Supervised Learning

The results in the last section show that end-to-end self-supervised learning algorithms can improve the performance on the target domain in the unsupervised speech domain adaptation task. In Table 4, we compare the proposed incremental layer-wise (ILW) with greedy layer-wise (GLW) [17] self-supervised learning algorithms. GLW attaches self-supervised losses on top of every encoder layer and updates all the layers at every step, which requires much more memory in the training. By contrast, IWL only updates one layer at each step and pre-train the bottom layers with more steps than top layers. For the encoder with 17 layers, ILW updates the model from layer 1 to layer 17 in an incremental way. All the results in Table 4 are obtained by running self-supervised pre-training for 174k steps on the multi-domain (MD) data and then fine-tuning on the source domain (MD-MF).

In Table 4, we can observe that the MF WER results of GLW and ILW algorithms are comparable and both achieve 4.6% WER on the target domain (MF) by using the CPC loss. APC performs the worst because applying reconstruction losses with the same targets on all the layers make them learn similar

Table 4: *Comparison between greedy layer-wise and incremental layer-wise scheme for three self-supervised learning algorithms. We pre-train on MD data and then fine-tune on source domain (MD-MF) only.*

Layer-wise scheme	Pre-training algorithm	Word Error Rate (%) MF	SF
Greedy	APC	5.0	6.3
	CPC	**4.6**	**6.1**
	Wave2vec2	4.8	6.2
Incremental	APC	5.1	6.2
	CPC	**4.6**	**6.1**
	Wave2vec2	5.1	6.2

representations. The disadvantage of Wav2vec2.0 loss is the model mismatch between bi-directional self-supervised learning and uni-directional fine-tuning. In terms of training memory consumption, ILW is much more efficient than GLW because ILW only updates one layer at each step and all the activations of the fixed layers are dropped after the forward pass. Since the CPC method obtains the best performance, we will use it to investigate the effect of hyper-parameters on incremental layer-wise CPC algorithm in Section 5.4.

5.4. The Effect of Hyper-Parameters

We investigate three important hyper-parameters for the incremental layer-wise CPC algorithm and reduce the training memory further by truncating input length without compromising performance.

5.4.1. Incremental Layer-wise Pre-Training Schedules

Firstly, we explore the effect of pre-training schedules and investigate how to distribute training steps on different encoder layers. As shown in Figure 2, we tried three different schedules: more steps at bottom layers, same steps for all layers and fewer steps on bottom layers. Figure 3 shows that pre-training more steps at bottom layers, can converge faster in the fine-tuning stage. This is expected since the top layers cannot learn anything useful without having the well-trained bottom layers.

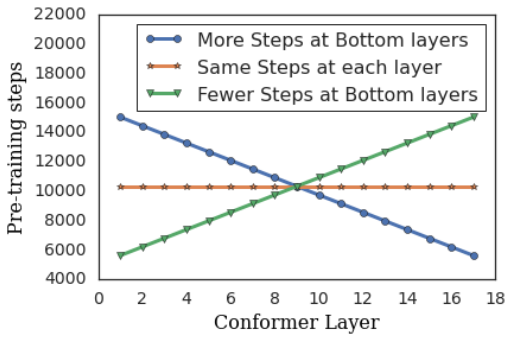

Figure 2: *Pre-training steps per layer for three schedules.*

5.4.2. Different Number of Updated Layers at Each Step

We update only 1 layer at each step in Section 5.3. Here, we vary the number of layers to be updated at each step. Table 5 shows that the proposed algorithm is very robust to the number of layers being updated at each step. Training memory is also reduced if we update fewer layers.

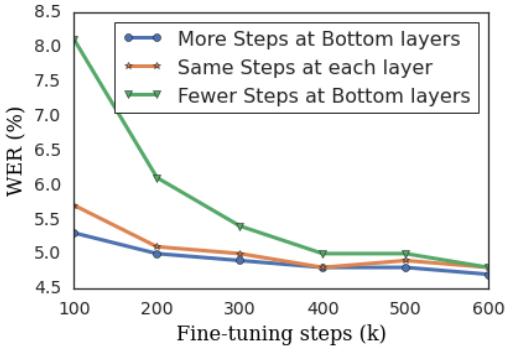

Figure 3: *Fine-tuning WERs for three schedules.*

Table 5: *WER of incremental layer-wise CPC algorithm with different number of bottom layers per step.*

# updated layers per step	WER (%) MF	SF	Memory Cost (MB)
End-to-end	4.5	6.1	8508
4 layers	4.5	6.1	2203
2 layers	4.6	6.1	1233
1 layer	4.6	6.1	748

5.4.3. Truncating Input Lengths

Furthermore, we reduce the training memory by truncating the input length of frames and fixing the updated layer per step to 1. We find in Table 6 that the training memory of the proposed algorithm with input length 100 (362MB) and GA, GC cost 95.7% smaller training memory than the end-to-end CPC algorithm (8508MB).

Table 6: *WER of Incremental layer-wise CPC algorithm with truncated input lengths.*

Input length / Truncated %	WER (%) MF	SF	Memory Cost (MB) Layer 1	Layer 17
686 / 0%	4.6	6.1	748	876
300 / 56.3%	4.6	6.1	380	565
200 / 70.8%	4.6	6.1	290	492
100 / 85.4%	4.7	6.1	189	405
ILW(100)+GA+GC	4.7	6.1	**129**	**362**

6. Conclusion

In this paper, we propose an incremental layer-wise self-supervised learning algorithm for efficient unsupervised speech domain adaptation on mobile devices. Extensive experimental results demonstrate that the proposed algorithm improves the word error rate on the target domain by 24.2% relative to a supervised baseline and uses 95.7% less training memory than the end-to-end self-supervised learning algorithm.

7. References

[1] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[2] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin *et al.*, "A better and faster end-to-end model for streaming asr," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5634–5638.

[3] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 17–36.

[4] K. C. Sim, A. Narayanan, A. Misra, A. Tripathi, G. Pundak, T. Sainath, P. Haghani, B. Li, and M. Bacchiani, "Domain adaptation using factorized hidden layer for robust automatic speech recognition," in *Proc. Interspeech 2018*, 2018, pp. 892–896. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-2246

[5] A. Misra, D. Hwang, Z. Huo, S. Garg, N. Siddhartha, A. Narayanan, and K. C. Sim, "A comparison of supervised and unsupervised pre-training of end-to-end models," in *Interspeech*, 2021.

[6] Y. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3497–3501.

[7] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[8] 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.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[10] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[11] S. Ramaswamy, R. Mathews, K. Rao, and F. Beaufays, "Federated learning for emoji prediction in a mobile keyboard," *arXiv preprint arXiv:1906.04329*, 2019.

[12] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[13] D. Guliani, F. Beaufays, and G. Motta, "Training speech recognition models with federated learning: A quality/cost framework," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3080–3084.

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[15] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 201–208.

[16] Z. Huo, B. Gu, H. Huang *et al.*, "Decoupled parallel backpropagation with convergence guarantee," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2098–2106.

[17] S. Löwe, P. O'Connor, and B. S. Veeling, "Putting an end to end-to-end: Gradient-isolated learning of representations," *arXiv preprint arXiv:1905.11786*, 2019.

[18] L. I. Rudin and S. Osher, "Total variation based image restoration with free local constraints," in *Proceedings of 1st International Conference on Image Processing*, vol. 1. IEEE, 1994, pp. 31–35.

[19] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[20] A. Narayanan, R. Prabhavalkar, C.-C. Chiu, D. Rybach, T. N. Sainath, and T. Strohman, "Recognizing long-form speech using streaming end-to-end models," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 920–927.

[21] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu *et al.*, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *Advances in neural information processing systems*, vol. 32, 2019.

[22] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.

[23] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *arXiv preprint arXiv:1604.06174*, 2016.