Interspeech 2022
18-22 September 2022, Incheon, Korea

Federated Pruning: Improving Neural Network Efficiency with Federated Learning

Rongmei Lin[1], *Yonghui Xiao*[2], *Tien-Ju Yang*[2], *Ding Zhao*[2], *Li Xiong*[1], *Giovanni Motta*[2], *Françoise Beaufays*[2]

[1]Emory University, [2]Google LLC

{rlin32,lxiong}@emory.edu, {yohu,tjy,dingzhao,giovannimotta,fsb}@google.com

Abstract

Automatic Speech Recognition models require large amount of speech data for training, and the collection of such data often leads to privacy concerns. Federated learning has been widely used and is considered to be an effective decentralized technique by collaboratively learning a shared prediction model while keeping the data local on different clients devices. However, the limited computation and communication resources on clients devices present practical difficulties for large models. To overcome such challenges, we propose Federated Pruning to train a reduced model under the federated setting, while maintaining similar performance compared to the full model. Moreover, the vast amount of clients data can also be leveraged to improve the pruning results compared to centralized training. We explore different pruning schemes and provide empirical evidence of the effectiveness of our methods.

Index Terms: federated learning, federated pruning, speech recognition, neural network, deep learning

1. Introduction

Neural network models have wide application in a variety of tasks, such as speech recognition, machine translation, and image recognition [1, 2, 3]. The performance of trained models largely depends on the quality and the amount of training data. Federated learning (FL) [4] provides a framework for leveraging the abundant data on edge devices with privacy preserved. However, FL faces several limitations in practice. One limitation is that the available memory on edge devices is highly limited. However, recent models are typically large, which makes on-device training challenging. For example, the successful model architecture for Automatic Speech Recognition (ASR), Conformer [5], has 130M parameters and requires 520MB memory solely for storing the parameters during training. Another limitation is that FL typically only updates the model parameters and leaves the model architecture unchanged. As a result, only model accuracy is improved but not model efficiency.

In this paper, we propose *Federated Pruning (FP)* to address the limitations mentioned above. Because models are usually over-parameterized to facilitate training, there are many redundancies. Several methods have been explored to exploit such redundancies to improve model efficiency. Among them, pruning is one of the most successful methods and has been widely studied under centralized training settings. At a high level, it identifies and removes redundant parameters from an over-parameterized model. The proposed FP applies the same idea to improve efficiency of federated learning and also allows leveraging on-device data to potentially achieve better efficiency than centralized pruning.

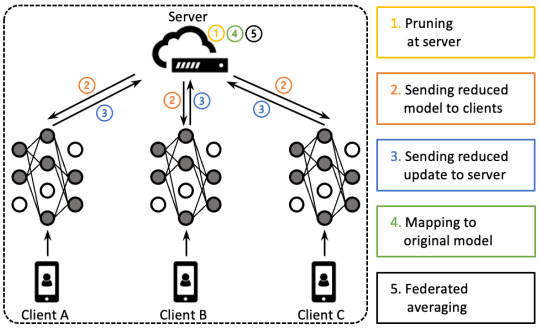


Figure 1: *A federated round of the proposed Federated Pruning. The white circles denote removed parameters.*

The pruning method has been extensively studied in centralized fashion [6, 7]. [8] shows that a well-initialized sub-network can match the accuracy of the full network and such sub-network was studied in centralized training [6, 7]. The main issue of this approach is that the parameters deemed unimportant and pruned at an early iteration may turn out to be important at a later iteration. To address this problem, [9, 10] use different pruning method time-wise and model-wise. Unlike these works focusing on centralized training, our work targets at federated learning and analyzes the impact of different pruning design decisions under this setting. A related work of model compression under the FL setting is Federated dropout [11]. Unlike our proposed method, federated dropout randomly generates reduced model and performs training on full model. Another preliminary work, PruneFL [12], also applies pruning to federated learning. It adopts sparse pruning instead of structural pruning as used in this work, so the resultant model will be less efficient when running on devices in practice. Moreover, we evaluate the proposed FP with production-grade models and datasets, which better reflects the real condition of deployment.

In summary, this work has the following contributions:

- **Improving the efficiency of federated learning:** We propose Federated Pruning (FP) to leverage on-device data to effectively prune redundant parameters from models. The resultant smaller models require less on-device memory to train and lower bandwidth for transporting models.
- **Exploring different pruning design decisions:** We explore and perform extensive ablation studies on two design decisions of pruning under federated learning: pruning patterns and pruning methods.

This work was done while Rongmei Lin was an intern at Google.

10.21437/Interspeech.2022-10787

Algorithm 1 Federated pruning. Initialize the server model with w^0 and the binary pruning mask M with *ones* like (w^0). The K clients are selected and indexed by k, federated pruning rounds are indexed by r, and n is the number of examples. *Shrink*(w, M) reduces the model size according to pruning mask M. *Expand*(w, M) maps the reduced model to original size. Related pruning methods include following functions: *GetImportanceScore* (w^r), *GenerateMask* (w^r, r, S).

Input: Pre-trained dense ASR model: w^0
  Binary pruning mask: M of *ones* like (w^0)
  Target sparsity level: S
  FL rounds: ΔR, $R^{fine-tune}$, R^{end}
Output: Sparse ASR model: $w^{R^{end}}$
function *FederatedPruning*
  initial sparsity level $s \leftarrow 0$
  for each round $r = 0, 1, 2, ..., R^{fine-tune} - 1$ **do**
    if $r \ mod \ \Delta R == 0$ **then** ▷ Every ΔR rounds
      GetImportanceScore (w^r)
      M = *GenerateMask* (w^r, r, s)
      if $s < S$ **then** ▷ Reaches refining phase if $s == S$
        increase s
      end if
    end if
    w^{r+1} = *FPTrain*(w^r, M)
  end for
  for each round $r = R^{fine-tune}, ..., R^{end}$ **do** ▷ Fine-turning
    Reduce the server model with mask M
    Training the reduced model with standard FL.
  end for
  return $w^{R^{end}}$
end function
function *FPTrain*(w^r, M)
  $W^r \leftarrow$ *Shrink*(w^r, M) ▷ Generate reduced model
  Randomly select K clients
  Server sends the reduced model W^r to K clients
  for each client k **in parallel do**
    $\hat{W}_k^r \leftarrow$ *ClientLocalUpdate*(k, W^r)
    $\Delta W_k^r = W^r - \hat{W}_k^r$
    Clients send ΔW_k^r to server
  end for
  $\Delta w_k^r \leftarrow$ *Expand*$(\Delta W_k^r, M)$ ▷ Map reduced updates
  $\bar{w}^r = \sum_{k=1}^{K} \frac{n_k}{n} \Delta w_k^r$ ▷ Federated Averaging [13]
  $w^{r+1} = w^r - \eta \bar{w}^r$
  return w^{r+1}
end function

- **Proposing a novel approach for adaptive sparsity:** We propose a novel adaptive per-layer sparsity approach that dynamically allocates the target global sparsity level to each layer. Therefore, there is no need to manually select the per-layer sparsity levels.
- **Experimenting with production-grade environments:** We evaluate the proposed Federated Pruning with production-grade models and datasets, which better reflects the real condition of deployment.

2. Federated Pruning

Figure 1 describes a federated round with the proposed Federated Pruning. The round starts from the generation of a set of variable masks on the server. The masks contain binary values signifying whether the parameters at the corresponding locations should be pruned (value 0) or not (value 1). The number of pruned values is determined by a given sparsity level, which is the ratio of pruned parameters. The server model is pruned based on the masks and sent to clients. Each client then trains this reduced model on its local data and returns the trained model. The server model finally aggregates the trained models from all the clients and moves on to the next federated round. Compared with standard federated learning, federated pruning prunes the model at the beginning of each round, and the pruned models are the ones trained on clients and transported.

Across federated rounds, we define three phases, (1) pruning, (2) refining and (3) fine-tuning, as shown in Figure 2. In the first phase (pruning), the sparsity level ramps up from 0 to the target sparsity level S. In the second phase (refining), the sparsity level is fixed at S, and the focus is refining the masks. Please note that there is a chance that a value in a mask flips from the 0 to 1, which means a pruned parameter revives. We show in Section 3.4 that this phase is important for improving model quality. In the last phase (fine-tuning), the sparsity and the masks are fixed, and the focus is fine-tuning the remaining parameters. This phase ends when the pruned model converges.

We summarize the details of our method in Algorithm 1. We consider one cloud server and K edge devices (referred to as clients) with their local speech data. Let w be the parameters of the full ASR model on the server side and $\{w_1, w_2, ..., w_k\}$ be the K reduced models to be trained on clients. FP starts with pre-trained ASR model w^0 on the server side and the target sparsity level S. At the beginning of each ΔR rounds, the server computes the importance scores of all variables using *GetImportanceScore()*. Then, a set of masks are generated by *GenerateMask()* based on the importance scores and the current sparsity level s. In the generated masks, all variables with small importance scores will be removed by *Shrink*(w^r, M). The reduced model will be sent to clients for training, and mapped back to the full model when the server receives clients' updates by *Expand*(w, M). When mapping the reduced model back, the masked regions (which are not sent to clients) will have 0 aggregated updates. Then the standard federated averaging process will be used to aggregate all clients updates. When the sparsity level s reaches the target level S, it enters the second phase of mask refinement. When the round r is equal to $R^{fine-tune}$, FP moves on to the last phase. We reduce the server model with the mask M, so that the server model is the same as client models. Finally, the reduced model will be trained until convergence.

2.1. Pruning patterns

We discuss the structure of the pruned variables, i.e. removed variables, as pruning patterns. Pruning patterns can be broadly categorized as: unstructured pruning [14] that prunes the less salient connections over any nodes, and structured pruning [15] that prunes on larger structures such as channel or layer. Since our objective is to reduce the actual model size, the training memory on clients' devices and the communication between server and clients, we use structured pruning to remove the *slices* of the variables, and thus physically reduce the model size. We implemented following pruning patterns:

- Whole row / column: prune the entire row or column of the two-dimensional weight matrices W.
- Half row / column: evenly partition the two-dimensional variables W into $[W_1, W_2]$ and prune each half of the row or column.

For higher (larger than 2) dimensional matrices, we first reshape them to two dimensional space, apply the above patterns, and transform back to the original dimensional space.

2.2. Pruning methods

We discuss how to decide the salience, i.e. the importance score, of variables. We use three methods including weight magnitude

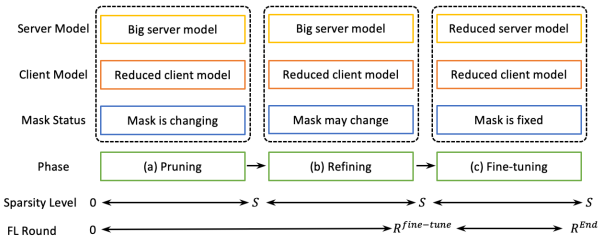


Figure 2: *Three phases of Federated Pruning across FL rounds.*

of variables, momentum of gradient magnitude of the variables and the multiplication of weight and gradient magnitude to measure the salience. At every ΔR round, we compute the $l1$ and $l2$ norm of the three methods as importance scores. Then given the sparsity level s, a threshold of the importance score can be derived by sorting the scores. The regions with less importance scores (i.e. smaller than the threshold) will be removed from the clients' model to form the reduced model. On server side, the regions with less importance scores are still kept in the pruning and refining phases, and will be removed in the final fine-tuning phase.

To achieve the target sparsity level S, we use pruning schedule to denote such process in the pruning phase. In this work, we explore the following two schedules:

- Constant sparsity level: instantly prune to the target sparsity level S at the beginning;
- Step-based sparsity level: gradually increase the sparsity level w.r.t current round r.

2.3. Mask refinement

In the refining phase, the masks are still re-generated. Because the masked regions in the model get 0 updates, their importance scores remain the same. For unmasked regions, their variables as well as the importance scores are updated. If the importance scores of some unmasked regions get smaller, then they might be replaced by the masked variables according to the rank of importance scores. Therefore, the masks are refined and retrained in this phase. In section 4, we also explore the variants of with and without mask refinement to show its utility. Note that the gradient based importance scores will yield relatively stale pruning mask, the masked regions will not have gradients and lose the ability of regrowth.

2.4. Adaptive per-layer sparsity

The layers of a deep neural network have different impacts on model accuracy. Based on the observations in [16], layers can be categorized as either "ambient" or "critical". Take the Conformer model [5] as example, "ambient" layers have little impact whereas pruning the "critical" layers will lead to severe quality degradation. Hence the sparsity level should get customized based on the importance of each layer. The sparsity level in the above federated pruning is unified among layers. Adaptive per-layer sparsity is proposed to allow dynamic sparsity level reallocation among layers.

We utilize a heuristic agent to determine the layer-wise sparsity level in two steps. First, we use predefined rules to measure the importance of each layer, such as the mean momentum of model deltas per layer or averaged weight magnitude per layer. Second, we assign the weighted sparsity level to

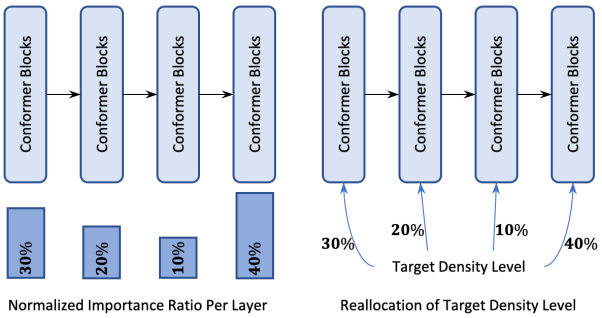


Figure 3: *Reallocation process*

each layer using estimated importance score. The less important layers get larger sparsity level. We use the averaged weight magnitude $||w_i||$ per layer as the importance score in this paper. As shown in Figure 3, the layers with larger magnitude are more important than those with smaller magnitude, thus the layer-wise density levels (1 - sparsity level) are assigned using the following equation for the L layers in the model.

$$LayerDensity = \frac{(1 - TargetSparsity) * ||w_i||}{\sum_{i=1}^{L} ||w_i||} \quad (1)$$

3. Experiment Results

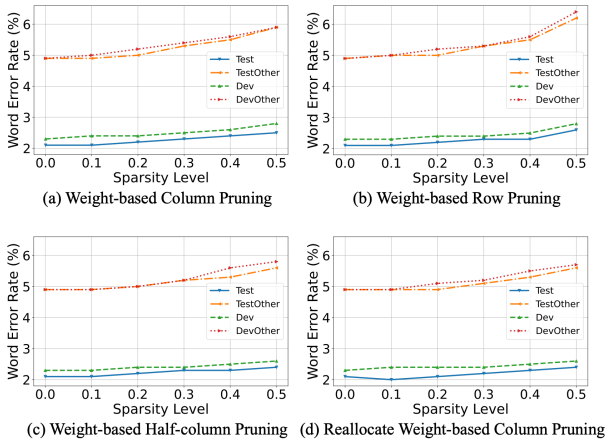
3.1. Experiment settings

Model architecture. We implemented the Federated Pruning in a distributed learning simulator. We use the state-of-the-art ASR model Conformer-transducer [17] as our base model. The model consists of 17 512-dimensional conformer encoder layers, a 640-dimensional embedding prediction network and a 2048-dimensional fully-connected joint network. In order to fit in our federated setting [18, 19], as suggested in [11], we change the original batch normalization to group normalization [20]. All the experiments share a same baseline model as initialization w^0. We follow the same settings of server/clients optimizer, SpecAugment [21] and number of clients as in previous work [22]. The baseline model is trained from scratch for 30k federated rounds. Our metric Word Error Rate (WER) on the baseline model is shown in Table 3.

Dataset. We train and evaluate the proposed model on the public LibriSpeech [23] corpus, which consists of 970 hours of labeled speech. We also explore the performance on industry-scale data collected from different domains as described in [24]. These multi-domain utterances contain 400k hours of speech and span domains of search, farfield, telephony and YouTube. Our work abides by Google AI Principles [25], all datasets are anonymized and hand-transcribed.

3.2. Federated pruning results

We prune all variables in each layer except the 1-D vector variable and the convolution layer which has specific utility and few parameters. The target sparsity level is the unified pruning percentage assigned for each variable. If we adopt the whole block pruning that prunes the entire column on W, it will further zero-out the corresponding row in next variable W' within the feed forward module in the Conformer blocks. The actual zero-parameter ratios of to-be-pruned variables are slightly



(a) Weight-based Column Pruning (b) Weight-based Row Pruning

(c) Weight-based Half-column Pruning (d) Reallocate Weight-based Column Pruning

Figure 4: *Experiment results of different settings of Federated Pruning on the Librispeech datasets.*

higher than the target sparsity level. The detailed numbers are listed in Table 1.

Figure 4 shows a rise in WERs on 4 Librispeech data subsets with the increase of sparsity levels. We also observed obvious quality degradation when the sparsity level $> 30\%$ in all pruning schemes. Concretely, 0.2% absolute increase on WER was observed on both Dev and Test evaluation set in our default setting Weight-based column pruning when the sparsity level reached 0.3. As we mentioned in Section 2.2, different *pruning methods* can be used to estimate the importance of variables. We conduct ablation experiments of different measurements on the Librispeech dataset. Our empirical finding in Table 2 suggests the weight-based score achieves similar WER as other metrics, while it is also the most communication efficient and stable metric. Thus, we rely on the weight-based score as the importance metric. In terms of the *pruning patterns*, we compare the WERs of several FP variants at sparsity level 50% as illustrated in Figure 4. One can observe that column pruning and especially the half-column pruning consistently outperforms the row-based pruning. Thus, all experiments below adopt weight magnitude as pruning method and whole column as pruning pattern. With the subsequent fine-tuning phase, the pruning schedule has very limited effect on the WER, so we further fix the constant sparsity level as the default setting.

Table 1: *Zero-Parameter Ratio w.r.t Target Sparsity Level*

Sparsity Level	0.10	0.20	0.30	0.40	0.50
Zero-Param Ratio	0.136	0.264	0.384	0.496	0.60

Table 2: *WERs of different pruning methods*

Pruning Methods	**WER on Test with sparsity level**				
	0.10	*0.20*	*0.30*	*0.40*	*0.50*
weight based	2.1	2.2	2.3	2.4	2.5
gradient based	2.2	2.3	2.3	2.3	2.4
weight×gradient based	2.1	2.2	2.2	2.3	2.4

3.3. Adaptive per-layer sparsity results

We propose the adaptive per-layer sparsity to make our method more flexible and dynamic, which allows reallocation of target pruning budgets among layers. Table 3 demonstrates that, compared to federated pruning with unified sparsity, the adaptive sparsity achieves lower WERs on all evaluation sets with 50% sparsity level.

Table 3: *WERs of unified / adaptive sparsity at Sparsity 50%.*

Exp.	**WER**			
	Test	*TestOther*	*Dev*	*DevOther*
Baseline	2.1	4.9	2.3	4.9
Unified Sparsity	2.5	5.9	2.8	5.9
Adaptive Sparsity	2.4	5.6	2.6	5.7

3.4. With and without mask refinement

To actually reduce the model size, the masked regions on server model are eventually zero-out by FP. On the other hand, the *Mask Refinement* phase introduced in Section 2.3 maintains the original values for masked regions. The pruned variables are allowed to grow back, leading to higher flexibility and thus, lower WERs as shown in Table 4.

Table 4: *WERs of w/o and w/ Mask Refinement at Sparsity 40%.*

Exp.	**WER**			
	Test	*TestOther*	*Dev*	*DevOther*
w/o Mask Refine	2.3	5.5	2.6	5.7
w/ Mask Refine	2.3	5.3	2.5	5.3

3.5. Results on short-form multi-domain dataset

Finally, we show the results on the large-scale short-form multi-domain dataset. The reduced model is trained on our multi-domain utterances and evaluated on the short-form dataset. Table 5 demonstrates the WERs on different sparsity levels. We conclude that our model can still achieve comparable performance to the baseline model (with sparsity level 0.0) on challenging dataset in the low sparsity level setting.

Table 5: *WERs of federated pruning on the voice search dataset.*

Sparsity Level	0.0	0.10	0.20	0.30	0.40	0.50
WER	6.4	6.7	7.0	7.4	7.9	8.9

4. Conclusion

We proposed the Federated Pruning method to find the efficient reduced model in FL settings. There are two advantages of FP. First, the training cost of clients, including the on-device training memory and communication, are alleviated due to the pruned and reduced model. Second, the model pruning quality can also be improved with the vast amount of clients data. We also proposed a new pruning method of the layer-wise sparsity level reallocation to improve the pruning quality. We showed in experiments that the FP trained model can achieve comparable quality of the full model.

5. References

[1] J. Li, Y. Wu, Y. Gaur, C. Wang, R. Zhao, and S. Liu, "On the comparison of popular end-to-end models for large scale speech recognition," *arXiv preprint arXiv:2005.14327*, 2020.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[3] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," *Advances in neural information processing systems*, vol. 30, 2017.

[4] P. Kairouz, H. Brendan McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[5] 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.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[7] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[9] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature communications*, vol. 9, no. 1, pp. 1–12, 2018.

[10] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4646–4655.

[11] D. Guliani, L. Zhou, C. Ryu, T.-J. Yang, H. Zhang, Y. Xiao, F. Beaufays, and G. Motta, "Enabling on-device training of speech recognition models with federated dropout," *arXiv preprint arXiv:2110.03634*, 2021.

[12] Y. Jiang, S. Wang, B. J. Ko, W. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *arXiv preprint arXiv:1909.12326*, 2019.

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data (2016)," *arXiv preprint arXiv:1602.05629*, 2016.

[14] Z. Wu, D. Zhao, Q. Liang, J. Yu, A. Gulati, and R. Pang, "Dynamic sparsity neural networks for automatic speech recognition," 2020. [Online]. Available: https://arxiv.org/abs/2005.10627

[15] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1812.08928

[16] L. Zhou, D. Guliani, A. Kabel, G. Motta, and F. Beaufays, "Exploring heterogeneous characteristics of layers in asr models for more efficient training," *arXiv preprint arXiv:2110.04267*, 2021.

[17] 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.

[18] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4387–4398.

[19] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.

[20] Y. Wu and K. He, "Group normalization," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[21] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[22] 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.

[23] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[24] 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.

[25] Google, "Artificial intelligence at google: Our principles." [Online]. Available: https://ai.google/principles/