ScaleFL: Resource-Adaptive Federated Learning with Heterogeneous Clients

Fatih Ilhan
Georgia Institute of Technology
Atlanta, GA
filhan@gatech.edu

Gong Su
IBM Research
Yorktown Heights, NY
gongsu@us.ibm.com

Ling Liu
Georgia Institute of Technology
Atlanta, GA
ling.liu@cc.gatech.edu

Abstract

Federated learning (FL) is an attractive distributed learning paradigm supporting real-time continuous learning and client privacy by default. In most FL approaches, all edge clients are assumed to have sufficient computation capabilities to participate in the learning of a deep neural network (DNN) model. However, in real-life applications, some clients may have severely limited resources and can only train a much smaller local model. This paper presents ScaleFL, a novel FL approach with two distinctive mechanisms to handle resource heterogeneity and provide an equitable FL framework for all clients. First, ScaleFL adaptively scales down the DNN model along width and depth dimensions by leveraging early exits to find the best-fit models for resource-aware local training on distributed clients. In this way, ScaleFL provides an efficient balance of preserving basic and complex features in local model splits with various sizes for joint training while enabling fast inference for model deployment. Second, ScaleFL utilizes self-distillation among exit predictions during training to improve aggregation through knowledge transfer among subnetworks. We conduct extensive experiments on benchmark CV (CIFAR-10/100, ImageNet) and NLP datasets (SST-2, AgNews). We demonstrate that ScaleFL outperforms existing representative heterogeneous FL approaches in terms of global/local model performance and provides inference efficiency, with up to 2x latency and 4x model size reduction with negligible performance drop below 2%.

1. Introduction

Mobile and Internet-of-Things (IoT) devices are the primary computing sources for most daily life tasks and they are becoming increasingly essential for billions of users worldwide (12; 18). These devices generate an unprecedented amount of data, which can be used to optimize services and improve user experience. Since the data is huge and mostly private, communicating, storing and organizing it in a central server poses serious privacy risks and brings logistic concerns (12). Federated learning (FL) emerged as a machine learning paradigm for this scenario, where storing the data and training the model in a central server is not feasible. In FL, instead of centralizing the data, the model is distributed to clients for local training and the central server aggregates the local updates received from clients (22).

Existing FL algorithms such as FedAVG (22), SCAFFOLD (13) and FedOpt (26) rely on the assumption that every participating client has similar resources and can locally execute the same model. However, in most real-life applications, the computation resources tend to differ significantly across clients (5; 19). This heterogeneity prevents clients with insufficient resources to participate in certain FL tasks that require large models. Although existing gradient compression (8) or model pruning techniques (7) may be applied to reduce the cost at the expense of small accuracy loss, these methods are not flexible enough to meet diverse constraint scenarios (computational, storage, network etc.) based on the heterogeneous resources of edge clients. We argue that FL should promote equitable AI practice by supporting a resource-adaptive learning framework that can scale to heterogeneous clients with limited capacity.

To this end, we present ScaleFL, a scalable and equitable FL framework. By design, ScaleFL has two novel features. First, ScaleFL can adaptively scale down the global model along the width and depth dimensions based on the computational resources of participating clients. The downscaling procedure in ScaleFL is inspired by EfficientNet (28), which demonstrates the importance of balancing the size of different dimensions while scaling a neural network. Since a deeper model is more capable of extracting higher-order, complex features while a wider model has access to a larger variety of lower-order, basic features, performing model size reduction across one dimension causes unbalance in terms of the learning capabilities of the resulting model. This motivates the design of ScaleFL that uniformly scales down the global model on both dimensions to provide the balance of preserving access to both complex and basic features as efficiently as possible. To perform splitting along the depth dimension, ScaleFL injects early exit

classifiers (29) to the global model at certain layers based on the model architecture and computational constraints at each complexity level. As a result, with ScaleFL, not only the global model achieves better performance compared to baseline FL approaches (22) and existing algorithms (5; 19) but also the local models at different complexity levels perform significantly better in case the clients are resource-constrained at inference time.

The second novelty of ScaleFL is providing effective aggregation mechanisms for combining local model updates from heterogeneous participating clients, and augmenting self-distillation for effective model update aggregation. During the training of local models, we perform self-distillation among the exit predictions to improve the knowledge transfer among subnetworks. Knowledge distillation enables transferring knowledge from a large (teacher) network to a smaller (student) network by training the student network on teacher predictions as soft labels (10). Self-distillation is a form of knowledge distillation, where the same network is used as both teacher and the student to improve performance during training, especially for multi-exit models (16; 24; 25; 33). In particular, we optimize these models with the additional objective of minimizing the KL divergence among the early exit (student) and final (teacher) predictions, which provides effective aggregation through increasing knowledge flow among local models. This procedure is an integrated component of the local optimization procedure and does not introduce any additional computational overhead as in standard knowledge distillation.

In summary, our main contributions are as follows: (1) We introduce a novel FL approach ScaleFL, which performs resource-adaptive 2-D model downscaling using early exits to handle system heterogeneity. Our method preserves the balance of basic and complex features at different complexity levels and enables efficient local models for resource-constrained clients. (2) We further enhance ScaleFL for effective integration of local training model updates by utilizing self-distillation among the exit predictions during local training to increase knowledge flow among subnetworks by minimizing the KL divergence among the early exit (student) and final (teacher) predictions. (3) We validate the advantages of ScaleFL in both model production quality for FL and model inference speedup for model deployment at edge clients. With extensive experiments on three vision benchmarks and two NLP benchmarks, we first demonstrate the significant improvements of ScaleFL in terms of global model performance on image/text classification tasks and various data heterogeneity settings, compared to recent approaches for FL with system heterogeneity. We then analyze the inference performance of local models and show that local models can provide up to 2x inference latency reduction and 4x model size reduction, with negligible performance drop under 2%.

2. Related Work

FedAVG is the baseline FL algorithm, where each round, clients download the updated global model, perform local training in parallel using gradient descent and send the updated local weights to the central server for aggregation by the average of the local weights from all participating clients in the given round (22). Three broad categories of efforts have been engaged to improve the FedAVG baseline. First, extensive studies have been dedicated to optimizing the communication efficiency of FL through gradient compression and quantization (1; 20). However, these studies only consider the scenario of homogeneous clients in which all participating clients are assumed to have similar computation capacity and can operate on the same model architecture with the same reduced model complexity. Second, there is significant research on robust federated learning to prevent client training data leakage due to model inversion attacks (30) or trojan detection against model poisoning attacks (2). The third category is the most recent efforts on addressing client heterogeneity in terms of data distribution (6; 13) and computation resources (5; 19). Our work is most relevant to the last category, where clients have heterogeneous resources. We below identify the most representative approaches in this category.

FedProx (17) allows each client to perform variable amounts of training iterations based on their computing power. However, this solution still assumes that all clients can operate on the same model. HeteroFL (5) proposes splitting the global model along width but, it keeps the full depth of the DNN architecture at each client and only adjusts the width split ratio for heterogeneous clients. This tends to result in very slim and deep subnetworks, which can lead to significant loss of basic features (28), resulting in a drastic drop in model quality. In addition, clients with limited resources will suffer from significant accuracy loss at model deployment time because they can only host and operate on those very slim and deep subnetworks. FedDF (19) proposes applying ensemble distillation to fuse models with different architectures. However, FedDF requires an additional dataset for the distillation operations and brings significant overhead between training rounds, whereas we apply self-distillation as an integrated component of the local training procedure without any additional overhead. Another recent study FLANC (23) shares a neural basis among all clients to efficiently construct models at various complexities. However, FLANC assumes that all clients can support at least the neural basis, and thus its adaptivity is bounded by the size of this pre-defined unified neural basis. In addition, the knowledge transfer among clients at different levels is only through the common neural basis.

Inserting multiple exits during DNN training enables early exiting to support adaptive inference (15). BranchyNet (29) introduced the idea of multi-exit classi-

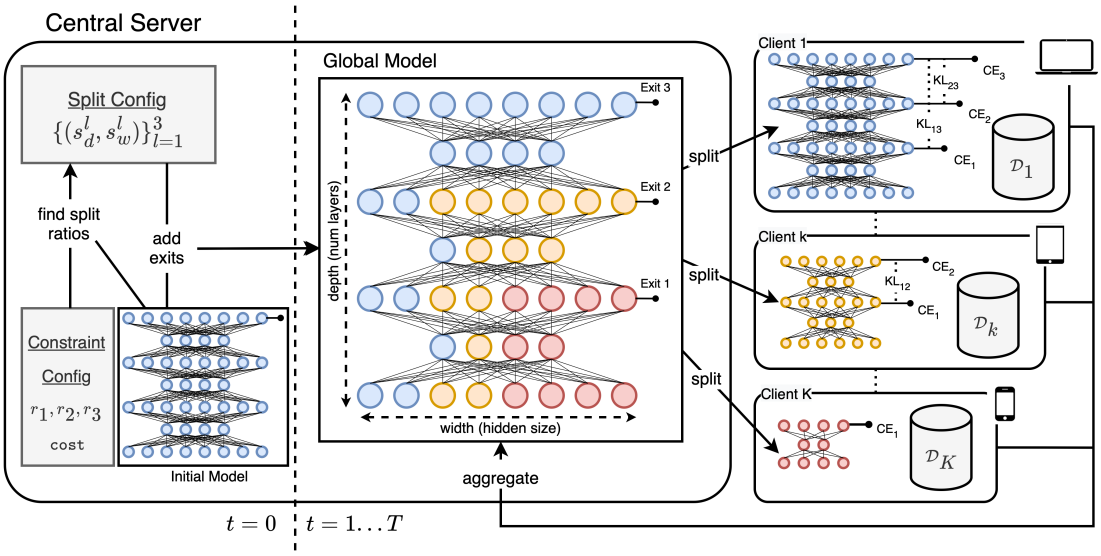

Figure 1. System architecture of ScaleFL with three levels. Given the constraint configuration, we compute the split ratios (Section 3.1.1) and based on computed width split ratios, we inject early exit classifiers to the given model. The global model is split along two dimensions (Section 3.1.2) and local models are trained using a combination of cross-entropy and KL-divergence losses as given in Eq. (5). Updates are aggregated back in the central server for the next round (Section 3.2).

fiers and early termination of inference for easier test samples. MSDNet (11) proposed a multi-scale dense-connected CNN architecture to improve performance in earlier exits. To improve the training efficiency, knowledge distillation (10) is employed among exits (16; 25). Early exiting has also shown effectiveness for NLP tasks (21; 35).

The design of ScaleFL is inspired by some of the above existing work with two original contributions. First, ScaleFL finds an optimal two-dimensional split plan for each client with limited resources and scales down a complex DNN model along both width and depth dimensions to provide the optimal balance of preserving basic and complex features in different model partitions. Second, ScaleFL further improves the aggregation and optimizes the performance of small subnetworks, produced by our two-dimensional downscaling, by applying self-distillation among exit predictions.

3. Methodology

Consider a federated learning system of K clients, each has its private local training dataset of size N_k, denoted by $\mathcal{D}_k = \{(\mathbf{X}_i, y_i)\}_{i=1}^{N_k}$, where $(\mathbf{X}_i, y_i)$ denotes the ith training data sample and its ground truth label, and $k \in \{1, \ldots, K\}$. We are also given the configuration of per-client resource constraints of each client k, such that the client can locally operate on a partition of the global model with the cost under the budget B_k and a level of tolerance ($\epsilon_k \geq 0$) for the budget constraint. A smaller value of ϵ_k indicates that the client has a tighter bound for the budget constraint B_k and a larger value implies the client has more elasticity regarding its resource budget.

In order to train a global model M collaboratively with K heterogeneous clients, ScaleFL first determines the model downscaling configuration for each of the K clients based on its resource budget B_k and budget tolerance ϵ_k (Section 3.1). Once the central server has the downscaling configurations for all K clients, it starts the federated learning process. At each round, only a p-fraction of the K clients participate in the joint training (*e.g.*, $p = 10\%$, $K = 100$) and operate on downscaled models based on their resources. Upon completion of each round, the central server follows the ScaleFL aggregation protocol to integrate the local model updates from heterogeneous clients (Section 3.2). We further improve the aggregation procedure by utilizing self-distillation (Section 3.3). Figure 1 gives a sketch of the ScaleFL system architecture.

3.1. Two-Dimensional Model Splitting

3.1.1 Generating 2-D Split Configuration

ScaleFL performs the split configuration as a part of the initialization process for a given federated learning (FL) task before starting the iterative FL rounds. The split configuration module makes two decisions: (1) Determining the number of model complexity levels L and mapping of each client k to the corresponding complexity level $l_k \in \{1, \ldots, L\}$, based on the resource constraints (B_k, ϵ_k), (2) given a DNN model M, determining the most uniform bi-dimensional split ratio along depth $s_d^{(l)}$ and width $s_w^{(l)}$ di-

mensions for each complexity level l.

Task 1: In order to downscale the global model M for resource-constrained clients, we first determine the total number of complexity levels, denoted by L, a hyperparameter in ScaleFL and the target cost reduction ratios r_l for each level l. We set L heuristically by performing clustering analysis over the set $\{B_k | 1 \leq k \leq K\}$ and assign each client with a model complexity level $l \in \{1, \dots L\}$ assuming at least one client can operate on the full model M. We consider $r_l = \frac{\min\{B_k | l_k = l\}}{\texttt{cost}(M)}$ for $l < L$, and $r_L = 1$ being the 100% complexity representing the full-size model M. Please note that this configuration can also be manually set by the user, e.g., it is possible to use the heuristic rule of halving the cost from the highest complexity level stepwise downward to the first level of complexity, so the local model M_l at level l will have the target cost reduction ratio of $r_l = 0.5 r_{l+1}$. Hence, we have r_1, r_2 and r_3 set to 12.5%, 25% and 50% respectively for $L = 4$. The smaller the level l_k is, the less complex the downscaled model will be for client k. Assigned level l_k for each client k satisfies the following: $r_{l-1}\texttt{cost}(M) \leq B_k < r_l \texttt{cost}(M)$ for $l_k \in \{1, \dots L-1\}$, and $\texttt{cost}(M) \leq B_k$ for $l_k = L$. Here, $\texttt{cost}(M)$ denotes the computational cost of the given model M, which can be measured in terms of $\#PARAMs(M)$ (number of weight parameters) and $\#FLOPs(M)$ (number of floating point operations).

Task 2: Now for each early exit complexity level l, we need to determine the most balanced depth and width split ratio pair $(s_d^{(l)}, s_w^{(l)})$, while satisfying that the cost of the submodel after the 2-D split will not exceed the resource constraints of those clients mapped to the exit complexity level l. Here, the depth split ratio $s_d^{(l)}$ determines where to place the lth early exit in the N layers of the global model M. The width split ratio $s_w^{(l)}$ indicates the fraction of the weight parameters to keep in the layers of the local models generated using the 2-D split ratio pair $(s_d^{(l)}, s_w^{(l)})$ for level l. For any $l \in \{1, \dots, L\}$, the following formulation will be followed to determine the most balanced split ratios $(s_d^{(l)}, s_w^{(l)})$ for those clients mapped to the exit complexity level l:

$$(s_d^{(l)}, s_w^{(l)}) = \underset{s'_d \in (0,1], s'_w \in (0,1]}{\arg\min} |s'_d - s'_w| \quad \text{such that}$$
$$\left| \frac{\texttt{cost}(\texttt{split}(M; s'_d, s'_w))}{r_l \texttt{cost}(M)} - 1 \right| \leq \epsilon_l. \qquad (1)$$

There may be multiple split ratio pairs that produce local models satisfying the target cost reduction ratio r_l. We find the most uniform bi-directional split ratio pair $(s_d^{(l)}, s_w^{(l)})$ for each early exit level l through grid search within the bound defined by the tolerance level ϵ_l. We set $\epsilon_l = \min_{j \in \mathcal{S}^l} \epsilon_j$, where $\mathcal{S}^l$ denotes the set of clients mapped to the exit level l based on their B_k, and ϵ_l is the minimum of tolerance levels of those clients in $\mathcal{S}^l$. A smaller value of ϵ_l enforces a tighter bound while satisfying the target cost reduction. A larger ϵ_l value provides more flexibility during the finding of the most balanced 2-D split configuration. For each level $l < L$, we place an early exit classifier to the $\lfloor s_d^{(l)} N \rfloor$th layer. Algorithm 1 lines 2-4 include these steps. Table 1 reports the computed depth and width split ratios for ResNet110 as the global model. The first column refers to the four complexity levels, the second and third columns reports the copmuted 2-D split ratios (s_d, s_w). The next two columns show the cost for the full-size model at $l = 4$ and the reduced model at each complexity level $l \in \{3, 2, 1\}$. Our depth and width-balanced split method results in uniformly shaped local models, as opposed to very slim and deep subnetworks due to splitting only along width in HeteroFL.

ResNet110	**Split Ratios**		**Cost**		
Level	s_d	s_w	#PARAMS	#FLOPS	r_l
4	1.00	1.00	1.73 M	253.1 M	1.000
3	0.88	0.75	0.86 M	138.5 M	0.500
2	0.77	0.70	0.46 M	99.7 M	0.250
1	0.66	0.70	0.21 M	83.4 M	0.125

Table 1. Split ratios and resulting local model statistics (#PARAMS, #FLOPS) for ResNet110 at each level.

3.1.2 Generating Local Models by Split Operation

Once the split configurations are produced, ScaleFL executes the 2-D split operation to generate the local model for each client k based on the split ratio pairs (s_w, s_d) to meet the budget constraint (B_k, ϵ_k) specified by the client. Concretely, at each communication round t, only p-fraction of the K clients, denoted by $\mathcal{S}_t$, are sampled and the global model M with weights θ_t is scaled down by executing the 2-D $\texttt{split}$ operation, which generates the local models with architecture M_{l_k} and weights $\theta_t^{(k)}$ for each client $k \in \mathcal{S}_t$ based on its complexity level l_k. In this operation, we first perform model splitting along the depth dimension by removing the layers after the $\lfloor s_d N \rfloor$-th layer where the lth early exit is placed.

Next, we perform splitting along the width dimension by selecting s_w-partition of the per-layer model weights along the hidden channel. To facilitate the execution of model reduction by width split ratio, we define an $\texttt{index}$ function that takes the weight matrix $\mathbf{W}$ and split ratio value s_w as inputs. Here $\mathbf{W} \in \mathbb{R}^{D_{in} \times D_{out}}$ is a hidden weight matrix where D_{in} and D_{out} are the number of input and output hidden channels in the corresponding layer. The $\texttt{index}$ function returns the Boolean index matrix $\mathbf{Z}$ of size $D_{in} \times D_{out}$, which indicates which weights to keep after the operation. We define the split index as follows: $\mathbf{Z}[i, j] = 1$ if

$i < \lfloor s_w D_{in} \rfloor$ and $j < \lfloor s_w D_{out} \rfloor$, and zero otherwise. This indexing results in keeping the upper-left submatrix by default, i.e., $\mathbf{W}' \triangleq \mathbf{W}[\mathbf{Z}] = \mathbf{W}[: \lfloor s_w D_{in} \rfloor, : \lfloor s_w D_{out} \rfloor]$. Our framework also allows using random selection of weight subsets (e.g. upper-right, lower-left, lower-right and middle etc.) for each client, so that different clients at the same complexity level can use different parts of the model weights and generate slightly different yet complimentary subnetworks. Algorithm 2 gives the pseudocode of this operation (see Appendix 6.1).

3.2. Aggregation

After the 2-D split operation, the ScaleFL aggregation module combines the updated local model weights received from heterogeneous clients at each round t. Concretely, every participating client $k \in \mathcal{S}_t$ may send a different subset of the global model to the FL server. For the default upper-left width split option to generate local models, the following aggregation procedure will be carried out to aggregate the overlapping weights received from the contributing clients at each round t, i.e., for every $\mathbf{W} \in \theta$, and $l \in \{1, \dots L\}$:

$$\mathbf{W}[\mathbf{Z}_l - \mathbf{Z}_{l-1}] \leftarrow \frac{1}{|\mathcal{S}_t^l|} \sum_{k \in \mathcal{S}_t^l} \overline{\mathbf{W}}_k[\mathbf{Z}_l - \mathbf{Z}_{l-1}]. \quad (2)$$

Here, $\mathcal{S}_t^l = \{k | k \in \mathcal{S}_t, l_k \geq l\}$ is the set of participating clients at round t with the mapped exit level at l or higher. $\mathbf{Z}_l = \texttt{index}(\mathbf{W}, s_w^{(l)})$ for $l \in \{l', \dots L\}$ and $\mathbf{0}$ for $l < l'$, where l' is the minimum level that $\mathbf{W}$ exists. For instance, the indexes defined by $\mathbf{Z}_4 - \mathbf{Z}_3$ will aggregate the updates from level 4 clients only. Similarly, the index defined by $\mathbf{Z}_3 - \mathbf{Z}_2$ will aggregate the updates from both level 3 and level 4 clients, and so forth. Lastly, $\overline{\mathbf{W}}_k$ is the zero-padded local weight $\mathbf{W}_k$ such that $\overline{\mathbf{W}}_k[\mathbf{Z}_l] = \mathbf{W}_k$ and $\overline{\mathbf{W}}_k[\mathbf{1} - \mathbf{Z}_l] = 0$. Hence, if all clients have the same complexity level (i.e. $L = 1$), Eq. (2) simplifies into $\mathbf{W} \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{k \in \mathcal{S}_t} \mathbf{W}_k$. Algorithm 3 gives the pseudo-code for the aggregation algorithm (see Appendix 6.1). This aggregation module is executed at FL server and it determines the overlapping regions using index matrices, and scales the sum of the weight updates received from contributing clients. One caveat for this baseline approach is the fact that low complexity level clients may not contribute to learning other parts of the global model weights belonging to higher complexity levels. This motivates us to further optimize our naïve baseline aggregation method with self-distillation.

3.3. Optimization with Self-Distillation

During the optimization of local models, we perform self-distillation, which is a form of knowledge distillation (10) where the technique is applied within the same network. Knowledge distillation enables a small (student) network to learn from another larger (teacher) network by treating the predictions of the teacher as soft targets. In ScaleFL, local models with $l > 1$ have multiple exits and can output multiple predictions. Therefore, to enhance the knowledge transfer among the subnetworks of the global model, we use self-distillation by using the final exit as the teacher and earlier exits as students.

For the local model M_j at level j, f_i is the ith core subnetwork with weights $\omega_{i,j}^{(f)}$. Likewise, g_i is the ith exit classifier subnetwork with weights $\omega_{i,j}^{(g)}$. $\hat{\boldsymbol{y}}_{i,j}$ is the output at the ith exit of M_j. We formulate the forward pass of local models as follows:

$$\mathbf{H}_{i,j} = f_i(\mathbf{H}_{i-1,j}; \omega_{i,j}^{(f)}), \quad (3)$$

$$\hat{\boldsymbol{y}}_{i,j} = g_i(\mathbf{H}_{i,j}; \omega_{i,j}^{(g)}), \quad (4)$$

for $1 \leq i \leq j \leq L$ where j is the level of the local model and $\mathbf{H}_{0,j} = \mathbf{X}$ is the input. In this study, we present our work on classification tasks so after obtaining the prediction logits at each exit for M_l at level l, the loss with self-

---

Algorithm 1: `ScaleFL`

Inputs: Dataset $\mathcal{D}_k = \{(\mathbf{X}_i, y_i)\}_{i=1}^{N_k}$ for each client k, number of complexity levels L, complexity level of each client $\{l_k\}_{k=1}^K$, target cost reduction ratios for each level $\{r_l\}_{l=1}^L$, client availability rate p, initial model architecture M_0
Parameters: number of communication rounds T, number of local training epochs E, batch size b, learning rate η
Outputs: Trained global model M and weights θ

Initialize global model M with architecture M_0 and weights θ_0
for level $l = 1, \dots L-1$ **do**
  Compute split ratio pair $(s_d^{(l)}, s_w^{(l)})$ using Eq. (1)
  Add early exit classifier to M at $\lfloor s_d^{(l)} N \rfloor$-th layer
end for
for round $t = 0, \dots T-1$ **do**
  $\mathcal{S}_t \leftarrow$ random subset of $\max(1, pK)$ clients
  for client $k \in \mathcal{S}_t$ in parallel **do**
    Split: $M_{l_k} \leftarrow \texttt{split}(M; s_d^{(l_k)}, s_w^{(l_k)})$
    for epoch $e = 1, \dots E$ in client k **do**
      for batch $\mathcal{B} \subset \mathcal{D}_k$ **do**
        $L = \frac{1}{b} \sum_{i \in \mathcal{B}} \mathcal{L}(M_{l_k}(\mathbf{X}_i; \theta^{(k)}), y_i)$ with Eq. (5)
        $\theta^{(k)} \leftarrow \theta^{(k)} - \eta \frac{\partial L}{\partial \theta^{(k)}}$
      end for
    end for
  end for
  Aggregate:
  $\theta \leftarrow \texttt{aggregate}(\{\theta^{(k)}\}_{k \in \mathcal{S}_t}, \{(s_d^{(l)}, s_w^{(l)})\}_{l=1}^L)$
end for
return M with θ

distillation is calculated as follows:

$$\mathcal{L} = \frac{1}{l(l+1)} \sum_{i=1}^{l} i(\beta \mathcal{L}_{KL}(\hat{\boldsymbol{y}}_{i,l}, \hat{\boldsymbol{y}}_{l,l}) + \mathcal{L}_{CE}(\hat{\boldsymbol{y}}_{i,l}, y)) \quad (5)$$

where $\mathcal{L}_{KL}(\hat{\boldsymbol{y}}_s, \hat{\boldsymbol{y}}_t) = \texttt{sum}(\sigma(\hat{\boldsymbol{y}}_t/\tau) \log \frac{\sigma(\hat{\boldsymbol{y}}_t/\tau)}{\sigma(\hat{\boldsymbol{y}}_s/\tau)})\tau^2$ is KL divergence with temperature $\tau > 0$, $\mathcal{L}_{CE}(\hat{\boldsymbol{y}}, y) = -\log \sigma(\hat{\boldsymbol{y}})[y]$ is cross-entropy loss for target class y, σ is softmax function and $\beta \in [0, 1)$ controls the self-distillation effect. τ and β are optimization hyperparameters of the system. Here, $\tau = 1$ corresponds to using the standard softmax function, and as τ grows, the output of the softmax becomes softer and more information on the score distribution of the teacher is provided to the student.

4. Experiments

In this section, we report the results of extensive experiments on five benchmarks: three image classification benchmarks (CIFAR-10/100, ImageNet) and two NLP benchmarks (SST-2, AgNews). We show that ScaleFL outperforms existing representative heterogeneous FL approaches in terms of global model performance under the same number of rounds and provides inference speedup with up to 2x latency reduction and 4x model size reduction while keeping the performance drop below 2%.

4.1. Datasets and Preprocessing

In image classification experiments, we consider common benchmark datasets on various scales: CIFAR-10, CIFAR-100 (14) and ImageNET (ILSVRC2012) (3). We hold out randomly selected 5000 images from CIFAR-10/100 train set and 25000 images from ImageNET train set for validation. We follow the data augmentation techniques applied in (9) with zero padding, center cropping and random horizontal flip with 0.5 probability. For normalization, we use fixed values, 0.5 as mean and 0.25 as standard deviation and disable statistics tracking for batch normalization operations due to the federated setting. In text classification experiments, we consider SST-2 (27) dataset from GLUE benchmark and AGNews (34) dataset. We hold out 872 sentences from SST-2 validation set and randomly selected 5000 sentences from AgNews train set for validation. For tokenization, we use the pre-trained tokenizer for BERT-base-uncased model provided by the open-source HuggingFace library (31). The details of the datasets are in Table 2.

Dataset	Train Size	Test Size	# Classes	Resolution
CIFAR-10	50K	10K	10	32x32
CIFAR-100	50K	10K	100	32x32
ImageNet	1.2M	150K	1000	224x224
SST-2	67K	1821	2	-
AgNews	120K	7.6K	4	-

Table 2. Statistics of the datasets used in image and text classification experiments.

4.2. Heterogeneous System Topology and Non-IID Client Data Distribution

In CIFAR-10/100 experiments, we set the number of clients $K = 100$ and the ratio of available clients in each training round $p = 0.1$. For ImageNET, SST-2 and AGNews experiments, $K = 50$ and $p = 0.2$. To simulate the system heterogeneity, we consider four complexity levels with the target cost reduction ratios $r_1 = 0.125$, $r_2 = 0.25$, $r_3 = 0.5$, $r_4 = 1$ and set the tolerance level $\epsilon_k = 0.1$. We set the number of clients assigned to each level to be equal. Different number of complexity levels or client distributions are also applicable and compatible with ScaleFL. We use Dirichlet distribution with concentration parameter α to create non-IID data splits for the local datasets in the federated setting as in (19; 32). We report the results in two levels of non-IID distribution with $\alpha = 1$ (more skewed) and $\alpha = 100$ (less skewed). The resulting class distribution of data at each client is illustrated in Figure 4 (see Appendix).

4.3. Model and Implementation Details

On CIFAR-10 and CIFAR-100 datasets, we perform experiments with two different models: ResNet (9) and MSDNet (11). ResNet is a common baseline for benchmarking and MSDNet is a multi-exit CNN architecture with dense connections. We use the default ResNet settings for 110-layer architecture from (9). For MSDNet, we use the recommended configuration with 24 layers, 3 scales and 16 initial hidden dimensions with a growth rate of 6 (11). On ImageNET, we consider EfficientNet (28) with B4 architecture (medium-size) and 224x224 input resolution (28). We set the number of FL rounds to 400 and 90, and batch size to 16 and 64 for CIFAR/ImageNET respectively. We optimize local models using gradient descent with the learning rate $\eta = 0.1$ decayed by 0.1 at the 100th/200th (for CIFAR) and 30th/60th (for ImageNET) rounds. We use 3-layer CNNs with ReLU activations as early exit classifiers. On SST-2 and AgNews datasets, we use the pre-trained BERT (4) provided by the open-source HuggingFace library (31). We optimize the local models for 100 rounds using gradient descent with the learning rate $\eta = 3e-5$ and batch size 16. We use one-layer FC layers as early exit classifiers. In all experiments, the number of local training epochs $e = 5$ and KL divergence temperature $\tau = 3$. For image/text classification experiments, loss weighting parameters $\beta = 0.1$ and 0.05, respectively.

We consider two baseline approaches: FedAVG (22) and Decoupled. In FedAVG, we train the Level-1 model with

Model	Algorithm	CIFAR-10				CIFAR-100			
		$\alpha = 100$		$\alpha = 1$		$\alpha = 100$		$\alpha = 1$	
		local	global	local	global	local	global	local	global
ResNet110	**FedAVG**	81.46	81.46	77.72	77.72	44.26	44.26	42.75	42.75
	Decoupled	77.16	-	74.83	-	36.60	-	35.78	-
	HeteroFL	82.93	84.35	77.60	79.91	44.66	47.12	42.97	42.95
	FedDF	83.35	84.44	77.08	78.57	43.50	46.99	42.29	44.50
	ScaleFL (Ours)	**84.49**	**85.53**	**79.61**	**80.83**	**46.63**	**49.94**	**43.52**	**44.95**
MSDNet24	**FedAVG**	82.69	82.69	75.28	75.28	46.44	46.44	42.75	42.75
	HeteroFL	81.54	83.02	75.77	76.74	44.65	47.77	42.32	43.00
	ScaleFL (Ours)	**84.61**	**84.77**	**77.81**	**78.69**	**49.19**	**50.25**	**45.25**	**46.12**

Table 3. Local and global model accuracy values on CIFAR-10/100 datasets with ResNet110 and MSDNet24.

EfficientNetB4	ImageNet			
	$\alpha = 100$		$\alpha = 1$	
	local	global	local	global
FedAVG	45.00	45.00	42.33	42.33
HeteroFL	43.68	46.61	41.74	43.59
ScaleFL (Ours)	**46.63**	**48.95**	**44.86**	**46.78**

Table 4. Local and global accuracy values on ImageNET.

BERT	SST-2				AG News			
	$\alpha = 100$		$\alpha = 1$		$\alpha = 100$		$\alpha = 1$	
	local	global	local	global	local	global	local	global
FedAVG	79.94	79.94	70.01	70.01	85.14	85.14	81.10	81.10
HeteroFL	76.02	88.83	76.21	82.86	89.92	91.51	88.85	90.85
FedDF	77.67	**88.95**	77.41	82.95	89.79	90.93	88.93	90.05
ScaleFL (Ours)	**83.72**	88.58	**79.65**	**83.79**	**90.53**	**92.13**	**89.72**	**91.20**

Table 5. Local and global accuracy values on SST-2 and AgNews.

splitting along width to use all available data. In Decoupled, we train separate models for each level using the corresponding data of the clients at that level. We also compare our results with recent studies: HeteroFL (5) and FedDF (19), where we use splitting along width dimension to form the local models at each level. For FedDF, we use CIFAR-10/100 and SST-2/AgNews train sets as distillation datasets for each other. Our implementation is on Python 3.7 with PyTorch 1.12 library. Our code is available at: https://github.com/git-disl/scale-fl. Each latency measurement is carried out 1000 times on a machine with an 8-core 2.4GHz CPU and 64GB RAM.

4.4. Global Model Performance

In this setting, we evaluate the global model on the test dataset of each client and report the average of obtained accuracy values. In Table 3, we report the results for image classification on CIFAR-10 and CIFAR-100 with ResNet110 and MSDNet24 models to observe the effect of different architectures on the performance of ScaleFL. We provide comparisons with FedAVG and HeteroFL for all three image datasets. We make two observations from Table 3. First, ScaleFL achieves 0.45-2.95% and 1.7-3.12% higher global performance compared to the closest approach on ResNet110 and MSDNet24 respectively. Second, in all settings, FedAVG baseline performs poorly since the small level-1 model is used to utilize the data of every client. Decoupled baseline performs the worst since the models for each level are trained independently.

In Table 4, we provide the results of ImageNet experiments with EfficientNet-B4 with 224x224 input resolution. Since FedDF uses 32x32 resolution on ImageNet experiments, we could not report the results for FedDF on ImageNet. ScaleFL consistently achieves around 2.34-3.19% higher global model accuracy under different data heterogeneity settings, compared to the closest competitor in each experiment. In addition, Table 5 reports the effectiveness of ScaleFL using NLP benchmarks with BERT as the backbone and ScaleFL again displays significant improvement especially in local model performance as investigated further in the next subsection.

4.5. Local Model Performance Analysis

This section reports the local model performances at different levels. In Tables 3, 4 and 5, 'local' columns contain the average of accuracy values obtained after evaluating the respective local model at each client based on its level. In addition, we illustrate the local models (their scales in width and depth dimensions) and accuracies for MSDNet24 on CIFAR100 in Figure 2. At each level, even though both submodels satisfy the target cost reduction, ScaleFL results in consistently better performance in terms of local model accuracy. This demonstrates the effectiveness of 2-D downscaling since it balances between preserving low-order basic features and extracting high-order complex features. We also analyze the performance with respect to the num-

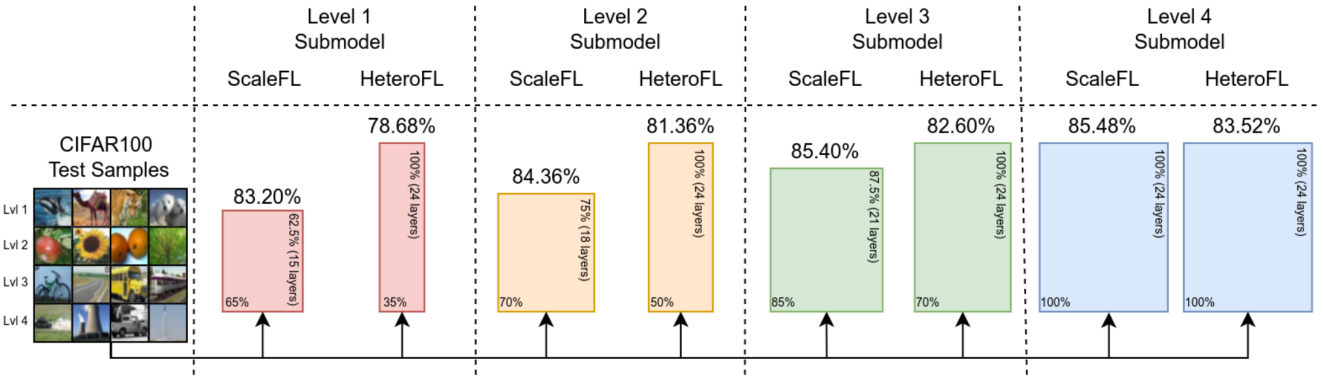

Figure 2. Local model accuracies of Level 1-4 models with ScaleFL and HeteroFL for MSDNet24 on their corresponding test sample sets from CIFAR100. Percentages on the bottom left and top right corners indicate the scaling ratios along width/depth dimensions. Percentages above the models report the test accuracy obtained by that model.

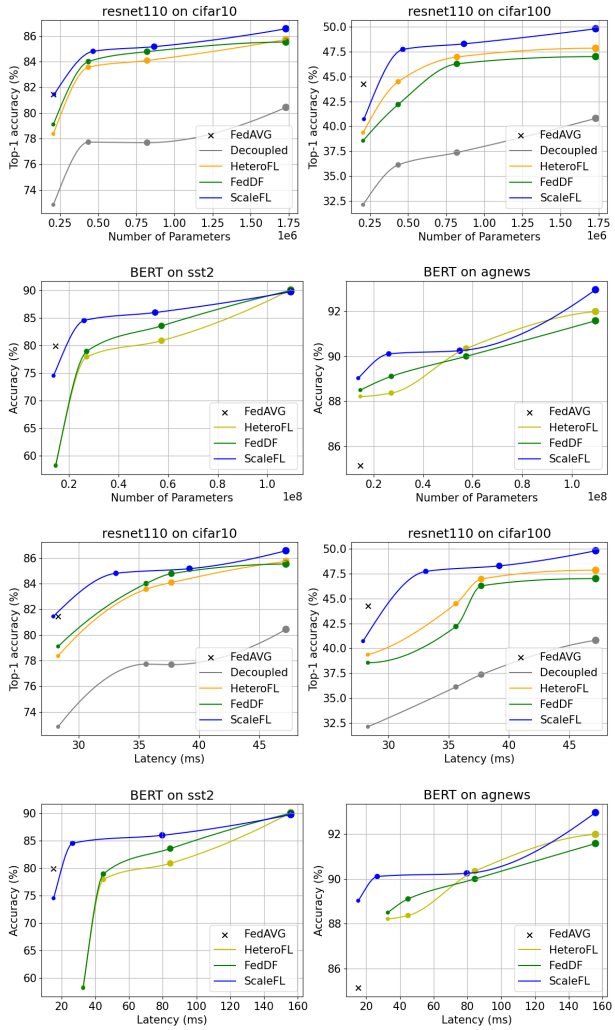

Figure 3. Local model performances with respect to #PARAMs and #FLOPs. Marker sizes represent complexity levels.

ber of parameters and inference time per sample on CPU (ms), as illustrated in Figure 3. In most settings, our approach achieves better test results at local models and the performance gap is more significant at lower levels, which demonstrates the efficiency of our two-dimensional splitting approach for model downscaling. For instance, the level-2 local model on AgNews with ScaleFL has 6x faster inference and 4x smaller size while having a lower 2.5% accuracy decrease (vs 3-4% with other methods) from the global model.

5. Conclusion

We have introduced ScaleFL, a novel resource-adaptive framework to address the system heterogeneity problem in federated learning. First, ScaleFL adaptively scales down the global model based on the computational constraints of participating clients. Our two-dimensional split methodology preserves the important balance of basic and complex features in local models. Second, we perform self-distillation among early exit and final predictions during local model training to improve the knowledge transfer among subnetworks and provide effective aggregation. We demonstrate the performance gains of ScaleFL compared to FedAVG as well as HeteroFL and FedDF, the two most representative federated learning approaches for system heterogeneity on benchmark image and text datasets.

Acknowledgements

This research is partially sponsored by the NSF CISE grants 2038029, 2026945, 1564097, and an IBM faculty award. The first author thanks for the summer 2022 internship at IBM Research with the group led by Dr. Donna Dillenberger.

References

[1] Mohammad Mohammadi Amiri, Deniz Gündüz, Sanjeev R. Kulkarni, and H. Vincent Poor. Federated learning with quantized global model updates. *ArXiv*, abs/2006.10672, 2020. 2

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020. 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 6

[5] Enmao Diao, Jie Ding, and Vahid Tarokh. Hetero{fl}: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021. 1, 2, 7

[6] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 2

[7] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 1

[8] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 1

[9] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 2, 3, 5

[11] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. 3, 6, 12

[12] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. 1

[13] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020. 1, 2

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 6

[15] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D. Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, EMDL'21, page 1–6, New York, NY, USA, 2021. Association for Computing Machinery. 2

[16] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1891–1900, 2019. 2, 3

[17] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020. 2

[18] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020. 1

[19] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 1, 2, 6, 7

[20] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018. 2

[21] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang

Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. In *ACL*, 2020. 3

[22] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017. 1, 2, 6

[23] Yiqun Mei, Pengfei Guo, Mo Zhou, and Vishal Patel. Resource-adaptive federated learning with all-in-one neural composition. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 2

[24] Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-distillation amplifies regularization in hilbert space. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 2

[25] Mary Phuong and Christoph Lampert. Distillation-based training for multi-exit architectures. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1355–1364, 2019. 2, 3

[26] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. 1

[27] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. 6

[28] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. 1, 2, 6

[29] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016. 2

[30] W. Wei, L. Liu, Y. Wut, G. Su, and A. Iyengar. Gradient-leakage resilient federated learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 797–807, Los Alamitos, CA, USA, jul 2021. IEEE Computer Society. 2

[31] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 6

[32] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 09–15 Jun 2019. 6

[33] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3712–3721, 2019. 2

[34] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 6

[35] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc., 2020. 3