Recurrent Early Exits for Federated Learning with Heterogeneous Clients

Royson Lee [1 2] **Javier Fernandez-Marques** [3] **Shell Xu Hu** [1] **Da Li** [1] **Stefanos Laskaridis** [4] **Łukasz Dudziak** [1] **Timothy Hospedales** [1 5] **Ferenc Huszár** [2] **Nicholas D. Lane** [2 3]

Abstract

Federated learning (FL) has enabled distributed learning of a model across multiple clients in a privacy-preserving manner. One of the main challenges of FL is to accommodate clients with varying hardware capacities; clients have differing compute and memory requirements. To tackle this challenge, recent state-of-the-art approaches leverage the use of early exits. Nonetheless, these approaches fall short of mitigating the challenges of joint learning multiple exit classifiers, often relying on hand-picked heuristic solutions for knowledge distillation among classifiers and/or utilizing additional layers for weaker classifiers. In this work, instead of utilizing multiple classifiers, we propose a recurrent early exit approach named ReeFL that fuses features from different sub-models into a single shared classifier. Specifically, we use a transformer-based early-exit module shared among sub-models to *i)* better exploit multi-layer feature representations for task-specific prediction and *ii)* modulate the feature representation of the backbone model for subsequent predictions. We additionally present a per-client self-distillation approach where the best sub-model is automatically selected as the teacher of the other sub-models at each client. Our experiments on standard image and speech classification benchmarks across various emerging federated fine-tuning baselines demonstrate ReeFL's effectiveness over previous works.

1. Introduction

Federated Learning (FL) has become an indispensable tool to train machine learning models collaboratively without exchanging raw data from clients' edge devices. In many practical scenarios, especially in the cross-device setting, these devices often differ in computational resources and may not have sufficient compute and/or memory resources to participate in federated training. This may yield convergence and fairness issues, especially if the least capable devices are consistently excluded from training. Hence, a key challenge of FL is to divide the global model into heterogeneous sub-models to fit into a wide range of diverse devices while maintaining high performance of the global model. To this end, various existing works split the global model by pruning its channels, also known as *width-based scaling* (Diao et al., 2021; Horvath et al., 2021; Mei et al., 2022; Hong et al., 2022), and/or utilizing early exits, also known as *depth-based scaling* (Liu et al., 2022; Kim et al., 2023; Ilhan et al., 2023; Kang et al., 2023).

Recently, approaches that use depth-based scaling, some of which, in addition to width-based scaling, showed significantly better performance compared to approaches using width-based scaling solely. These depth-based works split the global model based on depth and deploy additional classifiers, allowing each sub-model to exit early. Having individual classifiers in a network, however, has been previously observed to degrade performance due to the competing optimization criteria across sub-models, leading to the accumulation of conflicting gradients from these classifiers (Huang et al., 2018; Song & Chai, 2018; Li et al., 2019; Laskaridis et al., 2020). Another limitation of depth-based scaling is that shallower sub-models are trained more often than their deeper counterparts as these deep sub-models are not trained with data from clients with lower resource budgets (Kim et al., 2023). In addition, training with more data does not necessarily translate to performance gains as these shallower sub-models might not have sufficient parameters to learn good representations (Liu et al., 2022).

To counteract these limitations, existing FL works use knowledge distillation among either classifiers or layers in the backbone model and/or add additional layers to improve the representation capacity of shallower sub-models at a cost of additional resources. These approaches often require manual selection of which layers to distill from/to, *e.g.* the largest sub-model acts as a teacher for the other sub-models. However, the optimal choice of which layers

[1]Samsung AI Center, Cambridge, UK [2]University of Cambridge, Cambridge, UK [3]Flower Labs, Cambridge, UK [4]Brave Software, London, UK [5]University of Edinburgh, Edinburgh, UK. Correspondence to: Royson Lee <dsrl2@cam.ac.uk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

to distill differs greatly for different FL scenarios; the best-performing sub-model is dependent on the client's dataset. For instance, the largest sub-model might not be the best performing as it is only trained with a subset of data and the smallest sub-model might be insufficiently large to learn good representations of the dataset.

This paper proposes a different depth-based scaling early exit approach to counteract existing limitations better. Unlike previous works, we use a lightweight transformer-based recurrent early exit module ReeFL which is shared across all sub-models. ReeFL is trained to *1)* exploit and fuse features from multiple sub-models for task-based prediction, allowing us to use a single shared classifier, and *2)* modulate the features for deeper sub-models to yield better representations. Learning to perform task-based prediction using a shared classifier on aggregated features of multiple sub-models allows deeper sub-models to leverage earlier sub-models' features. Additionally, as the classifier is trained on the full dataset, it overcomes previous limitations where deeper classifiers are trained on partial data. To pick the right teacher sub-model for knowledge distillation, we propose using the estimated best-performing sub-model per client to distil knowledge to the other sub-models. This adaptive best exit selection can also potentially induce computation savings during inference as each client picks its best-performing exit rather than its deepest exit. Our contributions are summarized as follows:

- We present a novel approach to tackle heterogeneous clients in FL where representations of different sub-models are implicitly leveraged for both early exiting and feature learning for deeper sub-models.
- We propose a dynamic way to select the best-performing sub-model as the teacher model for knowledge distillation for each client.
- Through our experiments, we show that our approach, ReeFL, is scalable by accommodating a diverse range of client resources while consistently outperforming state-of-the-art baselines in both full federated fine-tuning (Qu et al., 2022; Nguyen et al., 2023; Chen et al., 2023) and emerging federated parameter-efficient fine-tuning (PEFT) (Sun et al., 2022; Zhang et al., 2023; Zhao et al., 2024) scenarios on standard image and speech benchmarks. Lastly, our comprehensive ablation studies help to elucidate the contributions of knowledge distillation and different federated aggregation strategies.

2. Related Work

Federated Learning. FL research spans a wide range of problems and challenges such as privacy, fairness, communication, personalization, and many more. More details can be found in surveys (Kairouz et al., 2021; Zhang et al., 2021; Wen et al., 2023). In this paper, we focus on the learning of a global model, first proposed in FedSGD (Shokri & Shmatikov, 2015) and made popular when FedAvg (McMahan et al., 2017) was introduced. Subsequent works aim to optimize the global model performance by better handling the data heterogeneity among clients. For instance, existing works finetune the global model with IID data (Zhao et al., 2018), use regularizers to minimize the Euclidean distance (Li et al., 2020) or maximize the agreement (Li et al., 2021) between the global and local models, shift the local models to alleviate its divergence with the global model (Karimireddy et al., 2020), or perform exact minimization such that local models converge to a stationary point of the global loss (Acar et al., 2021).

Transformers. Besides algorithmic changes, researchers also found that self-attention-based architectures such as Transformers (Vaswani et al., 2017) are better than convolution-based models at handling data heterogeneity (Qu et al., 2022). Additionally, starting from pre-trained models as opposed to random model initialization plays a key role in stabilizing federated training and improving performance (Nguyen et al., 2023; Chen et al., 2023). More recently, recent FL works utilize Parameter-Efficient Fine-Tuning (PEFT) methods to substantially reduce the communication cost with little or no degradation in performance (Sun et al., 2022; Zhang et al., 2023; Zhao et al., 2024). We thus extend the conventional FL experiments to incorporate the PEFT training in our evaluation.

System Heterogeneity in FL. Towards handling client heterogeneity, where participants have different hardware resources, researchers have leveraged various techniques such as *quantization* (Yoon et al., 2022), *pruning* (Caldas et al., 2018b; Jiang et al., 2022), low-rank decomposition (Yao et al., 2021), *neural architecture search* (Dudziak et al., 2022), *client selection* (Lai et al., 2021), *asynchronous aggregation* (Huba et al., 2022), or simply varying the *number of local epochs* (Nguyen et al., 2023; Wang et al., 2020b; Lee et al., 2023). Orthogonal to these approaches, but more closely to our setting, many works proposed to *partition* the global model by *width* (Diao et al., 2021; Horvath et al., 2021; Hong et al., 2022; Mei et al., 2022) or *depth* (Kim et al., 2023; Liu et al., 2022) or both width and depth (Kang et al., 2023; Ilhan et al., 2023). These approaches underperform due to the joint training of several classifiers, which are often competing with one another, along with a partial view over the dataset and selection of a sub-optimal teacher sub-model for knowledge distillation (Hinton et al., 2015; Zhang et al., 2019; Phuong & Lampert, 2019). In contrast, our method, ReeFL, learns to leverage representations across sub-models for task-based prediction on a single shared classifier. The use of a shared classifier also allows us to overcome the limitation of previous depth-based scaling works where deeper classifiers are trained with insufficient data. Additionally, we dynamically define

the teacher-student combination, based on their per-client performance on the downstream task.

3. Proposed Method

3.1. Preliminaries

Transformers with Early Exits. We primarily focus on the classical Transformers (Vaswani et al., 2017; Dosovitskiy et al., 2020) due to their surge in popularity and also growing evidence of their capacity to better handle heterogeneous data in FL (Qu et al., 2022). A Transformer model typically consists of:

- A tokenizer that maps each element of the input sequence to a d-dimensional vector (aka token).
- A learnable positional embedding for each position i is added to the outcome of tokenization, which yields the i-th token denoted by z_i^0.
- A learnable class token: $z_{\text{cls}}^0 \in \mathbb{R}^d$, where the superscript 0 indicates it is located at the entrance.
- A stack of architecturally identical blocks that apply multi-head self-attention (MSA), layer normalization (LN) and feed-forward multilayer perceptron (MLP) transformations on the initial sequence of tokens $\mathbf{z}^0$.

The forward pass of each Transformer block, for $l = 1 \cdots L$, takes the form:

$$\begin{aligned} \mathbf{z}^l &= \bar{\mathbf{z}} + \text{MLP}^l(\text{LN}_2^l(\bar{\mathbf{z}})) \\ \bar{\mathbf{z}} &= \mathbf{z}^{l-1} + \text{MSA}^l(\text{LN}_1^l(\mathbf{z}^{l-1})) \end{aligned} \tag{1}$$

where $\mathbf{z}^l := [z_{\text{cls}}^l, z_1^l, \ldots, z_n^l]$ is the output from block l, yielding the l-th transformation of $\mathbf{z}^0$. Since none of the transformations change the shape of the intermediate representation $\mathbf{z}^l$, it can be fed into a classifier for early exiting.

Federated Learning with Heterogeneous Clients. We consider a centralized setting of FL where the central server holds a global transformer model θ_g with sub-models $\theta_g[:\, l]$, for $l = 1 \cdots L$, representing L early exits. Each client, i, has a device with a maximum resource budget of r_i and N_i private training examples $\{(x_j, y_j)\}_{j=1}^{N_i}$ with x_j the input sample and y_j the target label. For each round, c participating clients are randomly sampled from a pool of C clients and the corresponding sub-models $\theta_i := \theta_g[:\, r_i]$ are selected to fit into each client's budget. These sub-models are then sent to the respective clients for training (i.e., one or few sub-model updating steps) and the updated sub-models are sent back to the server for central aggregation. Existing aggregation strategies such as FedAvg (McMahan et al., 2017), FedAdam (Reddi et al., 2021), FedDyn (Acar et al., 2021) can be naturally adapted for the early-exit setting.

3.2. Recurrent Early Exits

Given the heterogeneous setting, there are two main challenges for learning early-exiting sub-models: i) shallow sub-models face a dilemma of choosing between modulating subsequent features for later predictions or exploiting current features for prediction at early exits; ii) deeper sub-models are likely overfitted because they are only affordable on high-capacity clients. Prior work attempts to mitigate these issues via knowledge distillation, typically hand-picking the largest sub-model as the teacher (Ilhan et al., 2023) or utilizing deep mutual learning (Zhang et al., 2018) where each sub-model learned from one another (Kim et al., 2023) to counteract these limitations.

Architecture of ReeFL. We propose a single Transformer-based block module, named *recurrent early exits (Ree)* with parameters ϕ, to facilitate fine-tuning of early-exiting sub-models based on a pre-trained backbone transformer. As suggested by its name, Ree is applied recurrently at each early exit to simultaneously achieve feature exploration and exploitation of the class token.

Specifically, as illustrated in Fig 1, if the client is required to early exit at block l, Ree appends the class token z_{cls}^l to the queue of the class tokens $\mathbf{q}_{\text{cls}}[:\, l] := [z_{\text{meta}}, z_{\text{cls}}^1, \ldots, z_{\text{cls}}^l]$, and then transform it to obtain the "exploitation" and "exploration" versions of class token z_{cls}^l:

$$\mathbf{m}^l = \text{Ree}_\phi(\mathbf{q}_{\text{cls}}[:\, l] + \mathbf{p}[:\, l]). \tag{2}$$

The outputs $\mathbf{m}^l := [m_0^l, \ldots, m_l^l]$ are considered as modulated class tokens, where z_{meta} and $\mathbf{p}$ are learnable parameters analogous to the class token and the positional embedding of the backbone transformer.

There are numerous ways to parse the output $\mathbf{m}^l$. Empirically, we find that it is important to make the most of the foundation model and use the modulated meta-class token, m_0^l, as an additive modification to enhance the discriminative power of z_{cls}^l. Thus, we build the shared classifier (with parameters ψ) among all early-exit blocks as follows:

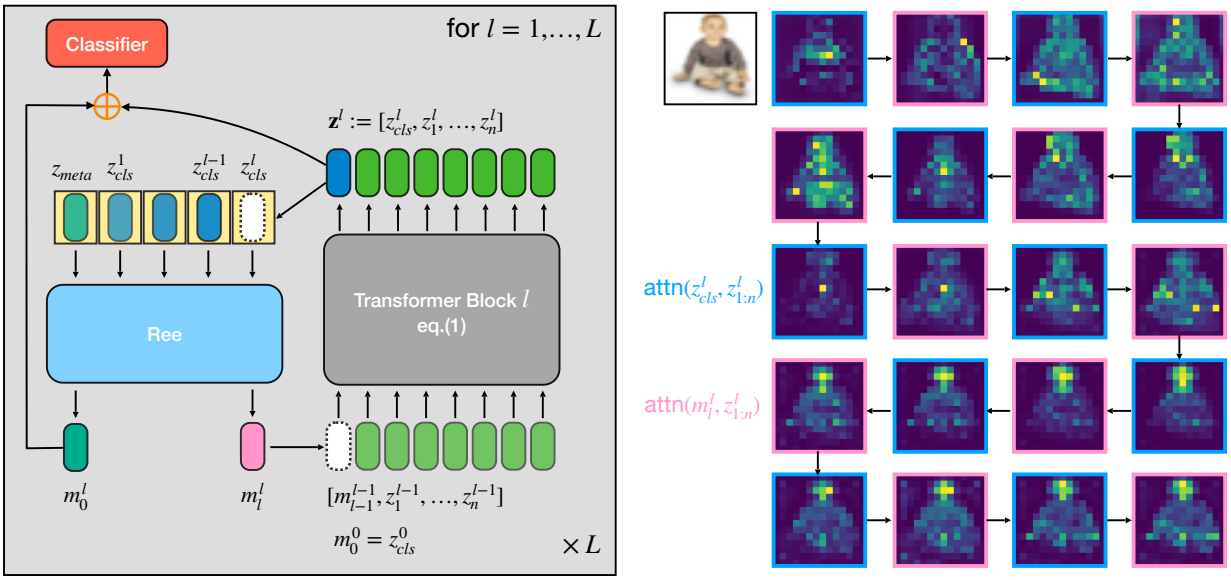
$$\hat{y}^l = \text{Classifier}_\psi(m_0^l + z_{\text{cls}}^l). \tag{3}$$

To improve the feature representation of deeper layers for subsequent prediction, Ree modulates the features of the backbone transformer by simply replacing

$$z_{\text{cls}}^l \leftarrow m_l^l. \tag{4}$$

As such, the input to the next transformer block of the backbone is slightly changed: $[z_{\text{cls}}^l, z_1^l, \ldots, z_n^l] \rightarrow [m_l^l, z_1^l, \ldots, z_n^l]$. We show in Fig 1(b) an example of the modulated feature development process, which alternates between the original backbone transformer blocks and Ree.

Federated Training of ReeFL. In each federated round, for each client, we train θ_i which consists of both ϕ and ψ on the local train dataset using SGD. Note that in the case where the backbone model is frozen, then the learnable



(a) ReeFL Architecture

(b) Feature Development Process

Figure 1. **Overview of ReeFL.** (a) Early exiting of block l: Ree takes as input the meta class token z_{meta}, the history of class tokens, $[z^1_{\text{cls}} \ldots z^{l-1}_{\text{cls}}]$, and the most recent class token, z^l_{cls}, and produces two tokens: *1)* the modulated meta-class token m^l_0 which participates in early-exit classification and *2)* the modulated latest class token m^l_l which is used to replace z^l_{cls} as a part of input to block $l+1$. Assuming the case where there is an early exit after every block, the forward pass involves running the shown architecture L times with shared Ree module. We assume $m^0_0 \equiv z^0_{\text{cls}}$ to be the starting point. (b) We visualize Ree's feature modulation during the forward pass of a CIFAR-100 image by showing a sequence of attention maps. Starting from block $l = 1$, we show the attention map between z^l_{cls} vs. $z^l_{1:n}$ (in blue) and the attention map between m^l_l and $z^l_{1:n}$ (in pink) alternatively. In this particular example of an image of a baby, the distinctive feature is the face, as learnt by later layers. In the earlier layers, particularly the 2nd, 3rd, and 4th layers, the modulated class token shown in pink aids the backbone model to focus more on the distinguishing parts of the image as compared to the use of the unmodulated class token shown in blue. The figure hence offers some interpretability as to how Ree's feature modulation affects the self-attention module in the backbone model especially in the earlier layers.

parameters of θ_i will be exactly ϕ and ψ. Following previous works (Kim et al., 2023; Ilhan et al., 2023), each client, i, aims to minimize the following loss:

$$\mathcal{L}_i = \sum_{e=1}^{E_{r_i}} \mathcal{L}^{ce}_e + \eta \mathcal{L}^{kl} \tag{5}$$

where E_{r_i} is the maximum exit within the client's budget, $\mathcal{L}^{ce}_e = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathcal{L}^{ce}(\hat{y}^e_j, y_j)$ is the cross-entropy (CE) loss of the e-th exit, $\mathcal{L}^{kl}$ is the Kullback–Leibler (KL) loss denoting knowledge transfer between sub-models with its corresponding hyperparameter η. At the end of each round, participated clients send their locally trained parameters back to the server where these parameters will be aggregated using FedAvg (McMahan et al., 2017). As the clients have different resource budgets, we weight the overlapping parameters accordingly to the number of data samples used to train each parameter.

ReeFL's Knowledge Distillation. In many prior works, the teacher and student sub-models in $\mathcal{L}^{kl}$ are manually defined, *e.g.* from a bigger model to a smaller model (Horvath et al., 2021; Ilhan et al., 2023) or the use of deep mutual learning (Zhang et al., 2018) where all sub-models learn from one another (Kim et al., 2023). A major limitation of these approaches is that the teacher sub-model selected might be under-performing, especially in depth-scaling methods where the deepest sub-model is trained with partial data and the shallowest sub-model might not be sufficiently parameterized to learn good representations. The best-performing sub-model is hence dependent on the client, the sub-model, and the dataset on which it is optimized. Hence, instead of manually picking a teacher sub-model, ReeFL uses the best training loss per-client to select the teacher sub-model and compute the KL loss as follows:

$$\mathcal{L}^{kl} = \frac{1}{N_i}\sum_{e\neq\tilde{e}}^{E_{r_i}}\sum_{j=1}^{N_i}(\sigma(\frac{\hat{y}^{\tilde{e}}_j}{\tau})\log\frac{\sigma(\frac{\hat{y}^{\tilde{e}}_j}{\tau})}{\sigma(\frac{\hat{y}^{e}_j}{\tau})})\tau^2 \tag{6}$$

where $\tilde{e} = \arg\min \bar{\mathcal{L}}^{ce} = \arg\min[\mathcal{L}^{ce}_1, \ldots, \mathcal{L}^{ce}_{E_{r_i}}]$ is the exit with the lowest training CE loss, σ is the softmax function and τ is the temperature. In practice, as the training loss per mini-batch of data is noisy, we take the running estimate

of the training loss per client: $\bar{\mathcal{L}^{ce}} = (1-\zeta)*\bar{\mathcal{L}^{ce}}+\zeta*\bar{\mathcal{L}^{ce}_{new}}$ where ζ is a hyperparameter and $\bar{\mathcal{L}^{ce}_{new}}$ is the training CE loss of all exits within budget computed on a new mini-batch of data.

4. Evaluation

4.1. Experimental Setup

4.1.1. Datasets

We conduct experiments on classification tasks using standard FL vision & speech benchmarks of differing degree of data heterogeneity in both feature and label distributions[1].

CIFAR-100 (Krizhevsky et al., 2009). We use the default partitions for train and test. Following prior works (Karimireddy et al., 2020; Wang et al., 2020a), we set the number of clients to 100 and partition the data using the latent Dirichlet allocation (LDA) method: $y \sim Dir(\alpha)$ for each client. Hence, the lower the α, the greater the degree of data heterogeneity in label distributions.

FEMNIST (Caldas et al., 2018a). We use the LEAF benchmark (Caldas et al., 2018a)'s natural partition, each client corresponds to its own handwriting, hence non-IID in both feature and label distributions. We use a total of 381 clients.

SpeechCommandV2 (Warden, 2018). We adopt the setup from Lee et al. (2023), sample 250 speakers from the training set, and split each speaker's data into 80%/20% train/test sets. However, instead of adopting the simpler 12-classes version, we use the full version comprising of 35 classes. Each speaker corresponds to its own voice, resulting in a challenging setup with both non-IID features and labels.

4.1.2. Model & Client Heterogeneity

Recent works showed that starting with a pre-trained model as opposed to a randomly initialized model leads to better stability and performance (Nguyen et al., 2023; Chen et al., 2023). Hence, we start with a pre-trained model, using the smaller variant of DeiT (Touvron et al., 2021), DeiT-S[2], which is pre-trained on ImageNet (Russakovsky et al., 2015). Besides training the entire model (Full), we also include freezing the backbone (Frozen) and training with a wide range of popular PEFT methods, namely Serial Adapter (SA) (Houlsby et al., 2019), Parallel Adapter (PA) (He et al., 2022), LoRA (Hu et al., 2022), and SSF (Lian et al., 2022). We use LN followed by a linear layer for all classifiers.

As previous works (Diao et al., 2021; Horvath et al., 2021; Kang et al., 2023; Liu et al., 2022; Ilhan et al., 2023) typically divide the model into 3-5 submodels, we evaluate on 4 submodels, an exit every 3 DeiT-S blocks apart from ScaleFL which selects where to place the exits via a grid search. Additionally, we also adopt a more challenging scenario where we evaluate on 12 submodels, an exit every block, in order to accommodate a wider range of end-devices. The same number of clients is allocated to each submodel, e.g. for 4 exits, 25 clients out of a total of 100 are given a max budget corresponding to each exit.

4.1.3. Baselines

We compare with recent depth-based FL approaches: DepthFL (Kim et al., 2023) and InclusiveFL (Liu et al., 2022), a recent width & depth-based approach ScaleFL (Ilhan et al., 2023), as well as a popular naive baseline, ExclusiveFL, where clients with insufficient budget to train the full model are excluded during training. For fair comparisons, we use the same classifier architecture in all baselines. Details of each baseline can be found in Appendix Section. B.

4.1.4. Hyperparameters

We run each experiment 3 times for 1k rounds, sampling 10% of the total number of clients per round, and report the mean performance of each exit, as well as the mean and standard deviation (SD) of the mean performance of all exits[3], on the full test dataset. Each sampled client in a FL round trains its local parameters with its local dataset using SGD for a single epoch using batch size of 32. We ran a simple grid search to pick the highest performing learning rate (LR) $[1e^{-1}, 5e^{-2}, 1e^{-2}, 5e^{-3}, 1e^{-3}]$, weight decay $[0, 1e^{-2}, 1e^{-3}, 1e^{-4}]$, minimum LR after LR decay using a cosine annealing LR schedule $[1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}]$, for each baseline. More details, along with the hyperparameters of different baselines, aggregation, and PEFT methods, can be found in Appendix Section A.

4.2. Comparison with Baselines

Performance Evaluation. Tables 1 & 2 show the mean and SD of the mean test accuracy of all 4 & 12 exits respectively across 3 runs. ReeFL outperforms all baselines in all scenarios except one particular scenario where we use SA for finetuning with the CIFAR-100 dataset on clients with IID data ($\alpha = 1000$) for 4 exits. ReeFL also outperforms baselines in most exits as shown in Fig. 2 and Appendix Fig. 10 & 8 where we show the accuracy for each exit for both 4 and 12 exits scenarios.

We also observe that ScaleFL, although being highly efficient, often struggles to beat the naive baseline ExclusiveFL, especially for Frozen and PEFT scenarios. With

[1]Code is available at https://github.com/royson/reefl.

[2]Experiments with other pre-trained models can be found in Appendix Section. F

[3]We also include the ensemble performance in Appendix Section. H.

Table 1. Mean and standard deviation (SD) of the mean performance of all **4 exits** across 3 runs and the mean communication cost per round for each approach.

Finetuning	Approach	CIFAR-100			FEMNIST	SpeechCmds	Comm. Cost (MB)
		α=1000	α=1.0	α=0.1			
Full	ExclusiveFL	67.29±0.06	66.66±0.18	60.91±0.11	84.07±0.09	73.69±0.05	53.31
	InclusiveFL	61.04±0.03	60.92±0.26	54.97±0.41	84.3±0.05	77.87±1.0	52.33
	ScaleFL	57.84±0.1	56.83±0.06	48.35±0.05	82.56±0.09	71.96±0.09	34.91
	DepthFL	57.52±0.4	55.25±0.03	45.79±0.27	81.24±0.37	78.44±0.38	53.31
	ReeFL (ours)	**76.42±0.12**	**75.69±0.17**	**72.58±0.33**	**86.13±0.08**	**84.47±0.26**	53.98
Frozen	ExclusiveFL	48.02±0.03	47.04±0.08	41.2±0.08	48.09±0.03	17.26±0.19	1.12
	InclusiveFL	53.99±0.03	53.23±0.02	49.08±0.01	63.0±0.02	26.53±0.03	0.15
	ScaleFL	28.5±0.04	27.9±0.01	25.18±0.04	28.51±0.03	10.8±0.1	0.89
	DepthFL	51.27±0.01	49.07±0.05	28.63±0.54	45.49±0.84	24.37±0.09	1.12
	ReeFL (ours)	**67.52±0.1**	**66.48±0.03**	**61.36±0.12**	**82.33±0.04**	**66.09±0.08**	1.79
LoRA	ExclusiveFL	67.46±0.03	66.4±0.03	58.78±0.06	82.98±0.09	69.93±0.1	2.53
	InclusiveFL	69.37±0.02	69.03±0.07	63.55±0.13	83.76±0.09	76.79±0.1	1.56
	ScaleFL	45.68±0.04	44.6±0.07	37.02±0.02	70.69±0.02	43.04±0.2	2.00
	DepthFL	71.56±0.29	70.18±0.27	64.33±0.05	82.35±0.21	77.15±0.13	2.53
	ReeFL (ours)	**73.9±0.03**	**73.37±0.07**	**69.29±0.1**	**84.69±0.03**	**79.91±0.08**	3.20
PA	ExclusiveFL	67.42±0.13	66.48±0.19	59.0±0.04	82.35±0.16	69.82±0.02	2.55
	InclusiveFL	66.81±0.04	66.57±0.11	61.2±0.31	83.12±0.0	76.47±0.08	1.58
	ScaleFL	46.48±0.01	45.16±0.06	38.02±0.08	72.83±0.04	43.53±0.24	2.01
	DepthFL	72.01±0.05	70.79±0.08	64.92±0.23	82.54±0.17	74.46±0.24	2.55
	ReeFL (ours)	**72.33±0.08**	**71.44±0.05**	**66.92±0.04**	**84.2±0.04**	**78.51±0.34**	3.22
SA	ExclusiveFL	68.2±0.02	67.39±0.07	59.71±0.09	82.07±0.04	68.98±0.2	2.55
	InclusiveFL	67.5±0.18	67.01±0.14	61.88±0.27	82.57±0.11	75.01±0.15	1.58
	ScaleFL	41.88±0.01	40.48±0.01	33.6±0.15	58.51±0.03	26.62±0.03	2.01
	DepthFL	**72.63±0.18**	71.3±0.02	64.73±0.04	81.92±0.17	74.7±0.23	2.55
	ReeFL (ours)	72.15±0.07	**71.48±0.1**	**66.52±0.25**	**84.07±0.09**	**78.39±0.25**	3.22
SSF	ExclusiveFL	66.06±0.02	65.29±0.01	57.94±0.04	79.27±0.07	57.03±0.46	1.39
	InclusiveFL	67.6±0.04	67.35±0.02	62.23±0.1	82.11±0.02	71.43±0.04	0.40
	ScaleFL	40.12±0.09	39.3±0.04	33.4±0.06	52.68±0.06	27.01±0.14	1.10
	DepthFL	45.61±0.03	42.87±0.11	28.38±0.42	74.66±0.22	66.22±0.12	1.39
	ReeFL (ours)	**70.12±0.07**	**69.54±0.02**	**64.77±0.11**	**83.42±0.04**	**73.6±0.04**	2.04

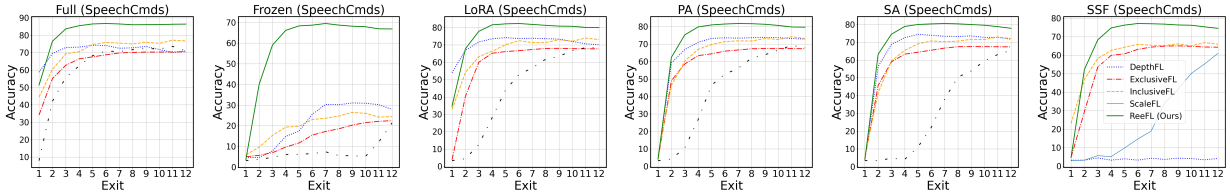


Figure 2. Mean accuracy of each exit across 3 runs on SpeechCommands. More results can be found in the Appendix.

full fine-tuning and PEFT, ScaleFL often performs better than ExclusiveFL and InclusiveFL at lower compute and memory bounds as shown in Figure. 4 and the Appendix Section. D. However, their performance is inferior in deeper layers, resulting in the low average performance shown in Tables 1 & 2. As observed in many previous works such as DepthFL and InclusiveFL, width-based scaling requires adequate retraining of these pruned channels. Since deeper layers are fine-tuned with fewer data, ScaleFL fails to beat the other baselines in most higher compute and memory regimes. This observation can also be seen with a frozen backbone, where ScaleFL fails to outperform all baselines in the accuracy and memory trade-off.

Lastly, for both 4 and 12 exits scenarios, full finetuning of the backbone model along with DepthFL or InclusiveFL led to worse performance than using a PEFT method on CIFAR-100 and FEMNIST datasets. Although PEFT outperforming full finetuning is a common phenomenon (Hu et al., 2022; Zhang et al., 2023; Zhao et al., 2024; Basu et al., 2023), its cause has not been thoroughly investigated. We hypothesize that since the model is pre-trained with ImageNet, the domain gap is smaller as compared to domain gap to SpeechCommands, hence, the model can easily overfit on the clients' small local dataset and lose its generalizability.

ReeFL's Consistent Performance. The baselines in Tables 1 & 2 show different performance depending on the finetuning method. For instance, DepthFL, in most cases, is the second best-performing baseline. Nonetheless, it performs poorly on the SSF PEFT method and fails to converge on more challenging scenarios, *e.g.* 12 exits with high data heterogeneity on some PEFT methods including SA, PA, and SSF. ReeFL, on the other hand, is more consistent across different fine-tuning methods. This is due to four main reasons: 1) Ree is shared and trained on the full dataset, 2) Ree utilizes features from multi-layers, 3) ReeFL's knowledge distillation is dynamic, and 4) ReeFL uses FedAvg which is more robust than FedDyn in highly heterogeneous data and resource scenarios. In existing baselines, classifiers rely

Table 2. Mean and standard deviation (SD) of the mean performance of all **12 exits** across 3 runs and the mean communication cost per round for each approach.

Finetuning	Approach	CIFAR-100			FEMNIST	SpeechCmds	Comm. Cost (MB)
		α=1000	α=1.0	α=0.1			
Full	ExclusiveFL	39.47±0.48	38.26±0.21	31.11±0.08	77.07±0.06	64.71±0.18	46.39
	InclusiveFL	42.35±2.18	44.05±0.13	33.42±0.43	80.33±0.01	70.93±0.25	45.57
	ScaleFL	35.63±0.09	34.29±1.33	23.41±0.17	74.68±0.05	61.57±0.25	26.40
	DepthFL	46.66±0.05	43.6±0.52	33.69±0.4	78.34±0.23	71.23±0.34	46.39
	ReeFL (ours)	**66.83±0.24**	**65.89±0.19**	**57.55±0.78**	**82.98±0.03**	**82.34±0.27**	47.21
Frozen	ExclusiveFL	39.19±0.01	37.6±0.02	28.95±0.02	42.1±0.12	14.75±0.06	0.97
	InclusiveFL	43.53±0.03	41.6±0.04	32.28±0.05	53.35±0.03	20.22±0.05	0.15
	ScaleFL	16.66±0.02	15.4±0.01	11.77±0.01	20.98±0.05	7.32±0.1	0.70
	DepthFL	45.22±0.06	42.93±0.18	25.2±1.89	41.29±0.1	21.3±0.02	0.97
	ReeFL (ours)	**58.74±0.05**	**58.1±0.09**	**51.33±0.36**	**75.36±0.02**	**59.5±0.5**	1.79
LoRA	ExclusiveFL	48.04±0.05	46.22±0.13	36.34±0.18	74.9±0.11	59.1±0.08	2.19
	InclusiveFL	55.9±0.08	54.54±0.15	41.53±0.15	78.21±0.15	66.18±0.58	1.37
	ScaleFL	35.14±0.09	33.05±0.02	22.62±0.1	59.57±0.05	44.23±0.14	1.58
	DepthFL	62.74±0.09	61.3±0.11	51.9±0.15	78.97±0.15	70.66±0.07	2.19
	ReeFL (ours)	**65.7±0.09**	**65.01±0.03**	**58.85±0.05**	**81.06±0.06**	**75.96±0.18**	3.01
PA	ExclusiveFL	45.63±0.14	43.57±0.27	32.94±2.19	73.1±0.02	59.06±0.12	2.21
	InclusiveFL	52.27±0.12	50.97±0.06	39.41±0.03	73.04±0.03	63.05±0.23	1.39
	ScaleFL	33.38±0.1	31.48±0.04	21.72±0.22	62.0±0.05	44.08±0.07	1.60
	DepthFL	61.23±0.23	59.8±0.07	6.88±5.88	4.95±0.2	65.69±0.41	2.21
	ReeFL (ours)	**62.35±0.07**	**61.47±0.12**	**55.67±0.14**	**77.62±0.04**	**72.37±0.02**	3.03
SA	ExclusiveFL	47.69±0.04	46.18±0.19	31.54±0.53	73.06±0.05	58.93±0.13	2.21
	InclusiveFL	53.37±0.11	52.02±0.06	40.06±0.1	72.54±0.07	61.8±0.19	1.39
	ScaleFL	31.33±0.07	28.97±0.05	19.27±0.02	53.08±0.01	31.65±0.11	1.60
	DepthFL	62.66±0.09	60.79±0.11	1.02±0.01	4.82±0.27	65.68±0.59	2.21
	ReeFL (ours)	**63.28±0.13**	**61.94±0.2**	**54.99±0.1**	**77.43±0.04**	**71.52±0.13**	3.03
SSF	ExclusiveFL	49.09±0.02	47.69±0.08	18.66±1.7	69.43±0.03	54.94±0.22	1.21
	InclusiveFL	54.43±0.04	52.92±0.07	40.63±0.01	74.48±0.02	59.6±0.09	0.37
	ScaleFL	30.01±0.01	27.99±0.05	19.24±0.03	46.63±0.02	25.17±0.12	0.87
	DepthFL	47.31±0.2	44.01±0.14	1.82±0.71	5.01±0.19	3.72±0.44	1.21
	ReeFL (ours)	**61.85±0.02**	**61.25±0.17**	**54.42±0.13**	**78.05±0.03**	**67.44±0.24**	2.01

Table 3. Ablation study on proposed aggregation strategies and knowledge distillation of depth-based scaling methods on 4 exits. Results on 12 exits can be found in the Appendix.

Finetuning	Approach	Distillation	Aggregation	CIFAR-100			FEMNIST	SpeechCmds
				α=1000	α=1.0	α=0.1		
Frozen	InclusiveFL	-	FedAvg	47.58±0.04	46.47±0.1	41.79±0.07	49.5±0.03	16.95±0.21
	InclusiveFL	-	FedAdam	53.99±0.03	53.23±0.02	49.08±0.01	63.0±0.02	26.53±0.03
	DepthFL	✗	FedAvg	48.88±0.04	48.15±0.07	45.01±0.27	49.22±0.03	17.16±0.34
	DepthFL	✗	FedDyn	51.42±0.04	49.31±0.25	38.27±4.84	53.26±0.08	24.57±0.45
	DepthFL	✓	FedAvg	49.38±0.03	48.57±0.01	44.88±0.07	49.19±0.07	18.45±0.11
	DepthFL	✓	FedDyn	51.27±0.01	49.07±0.05	28.63±0.54	45.49±0.84	24.37±0.09
	ReeFL	✗	FedAvg	67.04±0.09	66.32±0.11	61.14±0.18	81.86±0.05	65.9±0.03
	ReeFL	✓	FedAvg	**67.52±0.1**	**66.48±0.03**	**61.36±0.12**	**82.33±0.04**	**66.09±0.08**
LoRA	InclusiveFL	✗	FedAvg	65.78±0.01	65.34±0.04	58.8±0.04	81.19±0.03	67.24±0.06
	InclusiveFL	✗	FedAdam	68.04±0.12	67.97±0.0	61.89±0.29	84.12±0.19	77.81±0.1
	InclusiveFL	✓	FedAvg	68.18±0.04	67.99±0.09	61.05±0.2	82.26±0.05	69.0±0.15
	InclusiveFL	✓	FedAdam	69.37±0.02	69.03±0.07	63.55±0.13	83.76±0.09	76.79±0.1
	DepthFL	✗	FedAvg	71.22±0.02	70.8±0.11	66.35±0.14	83.76±0.02	74.78±0.14
	DepthFL	✗	FedDyn	72.89±0.1	71.83±0.08	66.53±0.26	82.2±0.02	77.49±0.29
	DepthFL	✓	FedAvg	69.84±0.1	69.28±0.02	63.31±0.07	83.67±0.06	74.88±0.02
	DepthFL	✓	FedDyn	71.56±0.29	70.18±0.27	64.33±0.05	82.35±0.21	77.15±0.13
	ReeFL	✗	FedAvg	73.47±0.01	72.76±0.07	68.61±0.2	84.53±0.09	79.21±0.21
	ReeFL	✓	FedAvg	**73.9±0.03**	**73.37±0.07**	**69.29±0.1**	**84.69±0.03**	**79.91±0.08**

Table 4. Mean Accuracy with and without ReeFL's feature modulation on 4 exits. Results on 12 exits can be found in the Appendix.

Finetuning	Modulation	CIFAR-100			FEMNIST	SpeechCmds
		α=1000	α=1.0	α=0.1		
Frozen	✗	53.89±0.0	53.24±0.02	44.85±0.49	61.01±0.02	20.14±0.21
	✓	**67.52±0.1**	**66.48±0.03**	**61.36±0.12**	**82.33±0.04**	**66.09±0.08**
LoRA	✗	73.69±0.07	72.98±0.14	69.14±0.12	84.51±0.07	79.72±0.02
	✓	**73.9±0.03**	**73.37±0.07**	**69.29±0.1**	**84.69±0.03**	**79.91±0.08**
PA	✗	71.53±0.1	70.83±0.11	66.1±0.05	84.03±0.03	77.42±0.04
	✓	**72.33±0.08**	**71.44±0.05**	**66.92±0.04**	**84.2±0.04**	**78.51±0.34**
SA	✗	71.6±0.01	71.04±0.02	66.44±0.02	83.73±0.01	75.96±0.08
	✓	**72.15±0.07**	**71.48±0.1**	**66.52±0.25**	**84.07±0.09**	**78.39±0.25**
SSF	✗	67.32±0.02	66.88±0.02	62.49±0.08	81.94±0.03	71.01±0.18
	✓	**70.12±0.07**	**69.54±0.02**	**64.77±0.11**	**83.42±0.04**	**73.6±0.04**

on features from a single layer, deep classifiers are trained on partial data, and exits are manually selected as teacher exits for knowledge distillation. The performance of these baselines is more dependent on the given scenario and is hence less consistent.

For instance, due to the low data regime at deeper exits, these exits are more sensitive to the fine-tuning method used, e.g. full full-tuning leads to overfitting. ReeFL counteracts this drawback by 1) utilizing features from earlier exits in addition to the features of the current exit and 2) learning the fusion, through Ree, and classification of these features on the full dataset. As another example, as different fine-tuning methods result in different performance for each exit, manually picking the teacher exits is only advantageous

Table 5. Injecting Ree only at exit layers as opposed to all layers leads to similar or worse performance.

Finetuning	CIFAR-100			FEMNIST	SpeechCmds
	α=1000	α=1.0	α=0.1		
Frozen	60.63±0.02 (-6.89)	59.82±0.06 (-6.66)	54.58±0.02 (-6.78)	68.93±0.03 (-13.4)	45.42±0.07 (-20.67)
LoRA	73.86±0.04 (-0.04)	73.21±0.0 (-0.16)	69.30±0.09 (0.01)	84.54±0.03 (-0.15)	79.98±0.02 (0.07)
PA	71.89±0.13 (-0.44)	71.2±0.01 (-0.24)	66.17±0.06 (-0.75)	84.19±0.01 (-0.01)	77.93±0.01 (-0.58)
SA	71.66±0.02 (-0.49)	71.22±0.07 (-0.26)	66.4±0.13 (-0.12)	83.82±0.03 (-0.25)	76.79±0.05 (-1.6)
SSF	68.75±0.1 (-1.37)	68.09±0.03 (-1.45)	63.06±0.04 (-1.71)	82.89±0.01 (-0.53)	73.05±0.16 (-0.55)

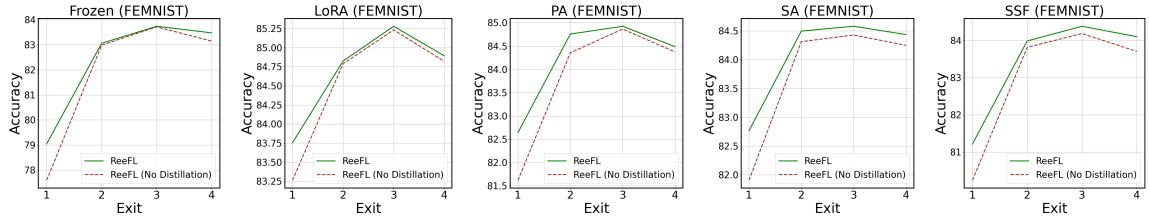


Figure 3. Impact of ReeFL's proposed knowledge distillation on FEMNIST (4 exits). See Appendix for more results.

in scenarios where the teacher exits are high-performing. Ree, on the other hand, dynamically selects this teacher exit, resulting in more consistent gains.

Lastly, in the case of DepthFL, FedDyn is the recommended choice for parameter aggregation. FedDyn dynamically modifies the local loss functions such that local models converge to a consensus that is consistent with a stationary point of the global loss. While FedDyn often leads to performance gains over FedAvg as seen in Table. 3 as well as other existing works that adopt FedDyn, we observe that in highly heterogeneous scenarios, e.g. CIFAR-100 $\alpha = 0.1$, this consensus either leads to a sub-optimal stationary point or is not found, leading to divergence, as seen in Table. 2. A more detailed discussion on the performance gap among these methods can be found in our ablation study (Section. 4.3) and in Appendix Section. G.

Training Costs. A comparison between different FL algorithms remains incomplete unless metrics accounting for communication, compute, and memory costs undergone by the clients during training are collected. To this end, we measure the average communication cost per round shown in Tables. 1 & 2 for 4 & 12 exits respectively. Additionally, we measure MACs (multiply-accumulate), using a single 224×224 input, as proxy metric of the amount of compute needed to train a model; and, we measure the memory peak at training time, using a batch size of 32, to quantify the minimum amount of memory needed by a client to train the largest sub-model within its resource budget.

As seen in Fig. 4, ReeFL outperforms existing baselines in most cases, achieving the best accuracy given similar MACs and peak memory constraints. Regarding communication costs, ReeFL incurs, on average, an additional $\sim$1MB more than the second best performing baseline, DepthFL. Although ReeFL has marginally fewer number of parameters, $\sim$2K parameters fewer, in the max resource budget scenario compared to the other depth-based scaling baselines, including InclusiveFL, DepthFL, and ExclusiveFL, its communication cost is constant and does not scale with the number of exits. As a result, every client has to send the shared Ree & classifier every round, leading to the slight increase in communication costs. However, we argue that this additional cost is marginal in most real-world scenarios, taking a small fraction of the average bandwidth supported by both broadband and mobile networks globally (Cisco Systems, 2020). Moreover, in realistic FL deployment scenarios, the eligible clients are connected to power and to an unmetered connection (i.e. WiFi), hence the main bottleneck is no longer the upstream communication, but the device resource capacity (Bonawitz et al., 2019). Hence, we place a greater emphasis on the benefit-cost ratio of on-device training.

4.3. Ablation

Aggregation & Knowledge Distillation. We weight the contribution of the aggregation strategy and knowledge distillation proposed in existing depth-based scaling works, namely DepthFL and InclusiveFL, as well as in our approach, ReeFL. We show results for Frozen and the best-performing PEFT approach, LoRA, in Table. 3 and Appendix Table. 10 for 4 and 12 exits respectively.

InclusiveFL proposed distilling from deeper backbone transformer blocks to shallower backbone blocks and hence not applicable for a frozen backbone; this distillation leads to a boost in performance in most cases for 4 exits but often lead to a drop in performance in the more challenging setup of 12 exits. Their proposed aggregation, FedAdam (Reddi et al., 2021), outperforms InclusiveFL+FedAvg by a considerable margin in all cases. DepthFL's proposed aggregation, FedDyn (Acar et al., 2021), on the other hand, boosts DepthFL's performance in most cases for 4 exits and some cases for 12 exits. Their proposed distillation, deep mutual distillation, however, often leads to a slight drop in performance as

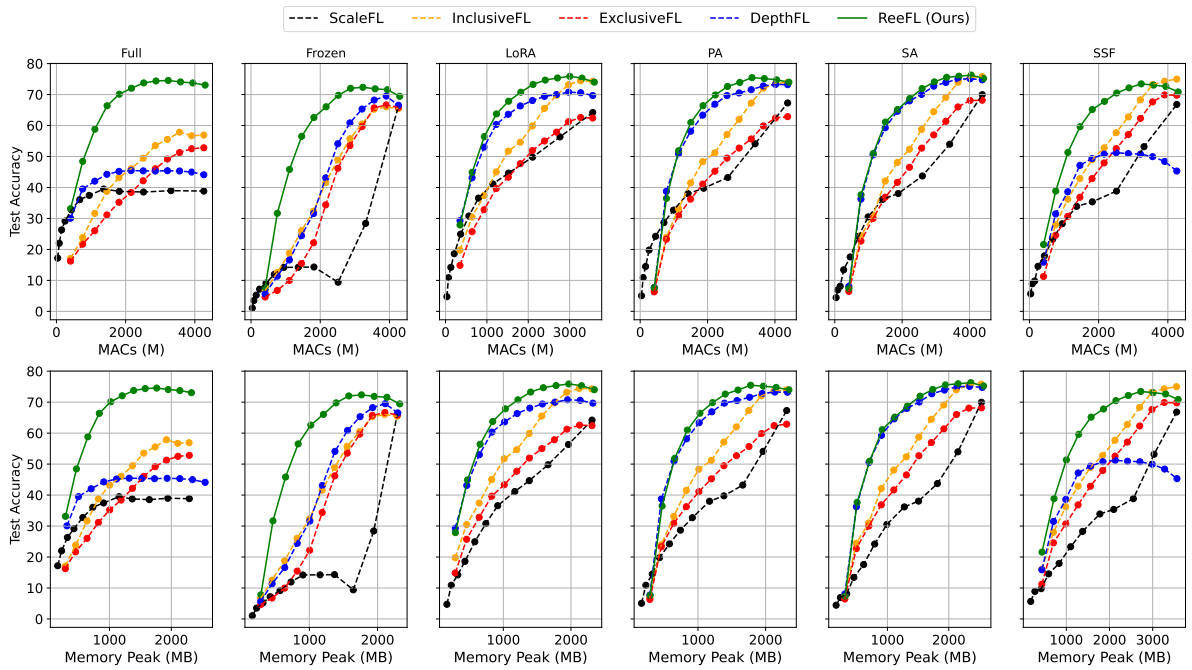


Figure 4. Quantifying training costs for each exit for CIFAR-100, $\alpha = 1.0$ for 12 exits, where each dot along each line represents an exit. Similar results on other datasets and scenarios can be found in the Appendix.

weaker sub-models are also selected as teacher models. In contrast, ReeFL distillation approach often picks the better performing sub-models as the teacher (Appendix Section C), resulting in a slight performance boost in most scenarios - also illustrated per-exit in Fig. 3. Most importantly, our approach still outperforms both DepthFL and InclusiveFL by a significant margin when all three approaches forgo knowledge distillation and use the same aggregation strategy, FedAvg.

ReeFL Feature Modulation. In Section 3.2, we detail that ReeFL aids feature learning through modulating the feature representations of the backbone model using the modulated class tokens (Eq. 4). In this ablation, we further elucidate its benefits by comparing to its unmodulated counterpart where each block in the backbone model uses the class token from the previous block. In Fig. 1(b), we illustrate the effects of feature modulation through visualizing the attention maps between the class token and other tokens before and after applying Ree. We observe that this helps the backbone model to build visually sensible feature representation especially in early layers. In Table. 4 and Appendix Table. 11, we show that using the modulated class tokens as inputs to the backbone model helps to boost performance. This observation is especially evident in the scenario where the backbone model is frozen on the SpeechCommands dataset where the pretrained model fails to reuse pretrained image features for speech-based prediction without ReeFL's feature modulation. The contribution of ReeFL's feature modulation is smaller when using PEFT methods as these methods explicitly learns new features to adapt to the new domain. Nonetheless, our feature modulation technique aids feature learning in most scenarios without additional costs.

Injecting Ree Only at Exit Layers. Instead of recurrently sharing Ree among all layers, we inject Ree only at the exits and present results for 4 exits, including the mean performance difference with sharing Ree in all layers, in Table. 5. As shown in the table above, sharing Ree only at exit layers leads to either similar performance or a drop in performance. Notably, if we freeze the backbone model, there is a considerable drop in accuracy, showing that Ree benefits from utilizing feature representations from all layers.

5. Conclusion

In this paper, we propose ReeFL, a radically distinct approach that leverages recurrent early exits to better handle client heterogeneity, offering superior performance, training efficiency and scalability in federated fine-tuning. By learning to weight and fuse feature representations from sub-models of varying depth, we can utilize a single shared classifier for all clients. Additionally, we show that these fused feature respresentations can modulate the backbone model to improve feature learning and subsequent predictions. Coupled with our best-teacher distillation, we are able to boost the accuracy of underperforming sub-models. As a future work, our approach can be extended to different modalities (e.g. language) and/or to include differential privacy noise for efficient private learning.

Acknowledgements

This work was supported by Samsung AI and the European Research Council via the REDIAL project (Grant Agreement ID: 805194).

Impact Statement

This paper presents work whose goal is to advance the field of federated learning (FL) when applied to heterogeneous devices. The goal of FL is to enable the training of machine learning models while protecting the privacy of end-users. There are many potential threats and attacks that threaten this privacy, none which we feel must be specifically highlighted here as our work does not focus on mitigating these attacks nor introduce additional vulnerabilities and is orthogonal to many of the current security measures in place.

References

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Basu, S., Massiceti, D., Hu, S. X., and Feizi, S. Strong baselines for parameter efficient few-shot fine-tuning. *arXiv preprint arXiv:2304.01917*, 2023.

Bonawitz, K. et al. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems (MLSys)*, 2019.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018a.

Caldas, S., Konečny, J., McMahan, H. B., and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018b.

Chen, H.-Y., Tu, C.-H., Li, Z., Shen, H. W., and Chao, W.-L. On the importance and applicability of pre-training for federated learning. In *International Conference on Learning Representations*, 2023.

Cisco Systems, I. Cisco annual internet report (2018–2023) white paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020. Accessed: 2024-03-30.

Diao, E., Ding, J., and Tarokh, V. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

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

Dudziak, L., Laskaridis, S., and Fernandez-Marques, J. Fedoras: Federated architecture search under system heterogeneity. *arXiv preprint arXiv:2206.11239*, 2022.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning. *International Conference on Learning Representations*, 2022.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Hong, J., Wang, H., Wang, Z., and Zhou, J. Efficient split-mix federated learning for on-demand and in-situ customization. In *International Conference on Learning Representations*, 2022.

Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., and Lane, N. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 2022.

Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., and Weinberger, K. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

Huba, D., Nguyen, J., Malik, K., Zhu, R., Rabbat, M., Yousefpour, A., Wu, C.-J., Zhan, H., Ustinov, P., Srinivas, H., et al. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832, 2022.

Ilhan, F., Su, G., and Liu, L. Scalefl: Resource-adaptive federated learning with heterogeneous clients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24532–24541, 2023.

Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K., and Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Kang, H., Cha, S., Shin, J., Lee, J., and Kang, J. Nefl: Nested federated learning for heterogeneous clients. *arXiv preprint arXiv:2308.07761*, 2023.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Kim, M., Yu, S., Kim, S., and Moon, S.-M. DepthFL : Depthwise federated learning for heterogeneous clients. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=pf8RIZTMU58.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lai, F., Zhu, X., Madhyastha, H. V., and Chowdhury, M. Oort: Efficient federated learning via guided participant selection. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pp. 19–35. USENIX Association, July 2021. ISBN 978-1-939133-22-9. URL https://www.usenix.org/conference/osdi21/presentation/lai.

Laskaridis, S., Venieris, S. I., Kim, H., and Lane, N. D. Hapi: Hardware-aware progressive inference. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2020.

Lee, R., Kim, M., Li, D., Qiu, X., Hospedales, T., Huszár, F., and Lane, N. Fedl2p: Federated learning to personalize. *Advances in Neural Information Processing Systems*, 30, 2023.

Li, H., Zhang, H., Qi, X., Yang, R., and Huang, G. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1891–1900, 2019.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Lian, D., Zhou, D., Feng, J., and Wang, X. Scaling & shifting your features: A new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35:109–123, 2022.

Liu, R., Wu, F., Wu, C., Wang, Y., Lyu, L., Chen, H., and Xie, X. No one left behind: Inclusive federated learning over heterogeneous devices. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3398–3406, 2022.

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

Mei, Y., Guo, P., Zhou, M., and Patel, V. Resource-adaptive federated learning with all-in-one neural composition. *Advances in Neural Information Processing Systems*, 35: 4270–4284, 2022.

Nguyen, J., Wang, J., Malik, K., Sanjabi, M., and Rabbat, M. Where to begin? on the impact of pre-training and initialization in federated learning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Mpa3tRJFBb.

Phuong, M. and Lampert, C. H. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1355–1364, 2019.

Qu, L., Zhou, Y., Liang, P. P., Xia, Y., Wang, F., Adeli, E., Fei-Fei, L., and Rubin, D. Rethinking architecture design for tackling data heterogeneity in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10061–10071, 2022.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252, 2015.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

Song, G. and Chai, W. Collaborative learning for deep neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Sun, G., Mendieta, M., Yang, T., and Chen, C. Conquering the communication constraints to enable large pre-trained models in federated learning. *arXiv*, 2022.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jegou, H. Training data-efficient image transformers & distillation through attention. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/touvron21a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020a.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020b.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., and Zhang, W. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2):513–535, 2023.

Yao, D., Pan, W., O'Neill, M. J., Dai, Y., Wan, Y., Jin, H., and Sun, L. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization. *arXiv preprint arXiv:2111.14655*, 2021.

Yoon, J., Park, G., Jeong, W., and Hwang, S. J. Bitwidth heterogeneous federated learning with progressive weight dequantization. In *International Conference on Machine Learning*, pp. 25552–25565. PMLR, 2022.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.

Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3713–3722, 2019.

Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4320–4328, 2018.

Zhang, Z., Yang, Y., Dai, Y., Wang, Q., Yu, Y., Qu, L., and Xu, Z. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pp. 9963–9977. Association for Computational Linguistics (ACL), 2023.

Zhao, W., Chen, Y., Lee, R., Qiu, X., Gao, Y., Fan, H., and Lane, N. D. Breaking physical and linguistic borders: Multilingual federated prompt tuning for low-resource languages. In *International Conference on Learning Representations*, 2024.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model, 2024.

A. Training & Implementation Details

In Section 4.1, we provide a summary of our experimental setup. In this section, we detail all hyperparameters and configurations used in all our experiments. Code is available at https://github.com/royson/reefl.

Baseline Hyperparameters Following the original works, we set $\beta = 0.2$ for InclusiveFL's momentum distillation. For DepthFL, we consistently ramp up the weight of the KL loss, η, for 300 rounds till $\eta = 1.0$. For ScaleFL, we use $\eta = 0.05$ and set the softmax temperature, $\tau = 3.0$ - $\tau = 1.0$ for all other works. For 4 exits, we follow ScaleFL's depth-scaling of adding classifiers to the 4-th, 6-th, 9-th and last transformer block and width-scaling ratios of $[0.4, 0.55, 0.75, 1]$. For 12 exits, we add a classifier for every block and scale the width ratio with [0.25,0.3,0.35,0.4,0.48,0.55,0.61,0.69,0.75,0.84,0.92,1.].

ReeFL Hyperparameters We scale ReeFL to be similar in parameters than baselines for a fair comparison. The number of multi-attention heads in Ree is set to 8. The number of bottleneck features for MSA is set to 16 and the number of hidden features in the MLP is set to $1.35\times$ the input features. Following DepthFL, we consistently ramp up η for 300 rounds. Our running estimate hyperparameter ζ is set to 0.2.

Local Training Hyperparameters Each client trains its local parameter using SGD with momentum set to 0, batch size set to 32, for a single epoch. The other hyperparameters are selected using a simple grid search to pick the highest performing learning rate (LR) $[1e^{-1}, 5e^{-2}, 1e^{-2}, 5e^{-3}, 1e^{-3}]$, weight decay $[0, 1e^{-2}, 1e^{-3}, 1e^{-4}]$, minimum LR after LR decay using a cosine annealing LR schedule $[1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}]$. For FedAvg and FedAdam, we pick $5e^{-2}$ as the initial LR with a minimum LR of $1e^{-3}$ after the aforementioned LR decay. For FedDyn, we pick $1e^{-1}$ as the initial LR with a minimum LR of $1e^{-2}$ and a weight decay value of $1e^{-3}$. We also clip gradients by value $[-1, 1]$ for all experiments for better stability.

Server & Aggregation Hyperparameters We set the total number of rounds to 1K, sampling 10% of the total number of clients for each round. Following recommended settings, we set FedDyn's $\alpha = 0.1$, not to be confused with α in the Dirichlet Distribution used in the main paper. For FedAdam, we set $\beta_1 = 0.9$, $\beta_2 = 0.999$, and the learning rate to $1e^{-3}$.

Backbone Model Hyperparameters We use a vision transformer model, pre-trained DeiT-S, as our backbone for all experiments. Our backbone, hence, has 12 blocks, a hidden dimension of 384, 6 multi-attention heads, and divides the input image into 16×16 patches. Instead of experimenting with different models, we experiment with different PEFT fine-tuning methods which inserts different learnable adapters to the backbone model.

PEFT Hyperparameters Rank of LoRA, Parallel Adapter (PA), Serial Adapter (SA) is set to 32. LoRA's α, not to be confused with α in the Dirichlet Distribution used in the main paper, is set to 64. Following their original works, LoRA is applied to the query and value projection matrices, PA & SA are inserted in each MSA and MLP block, and SSF is inserted to every learnable parameter except the last Linear layer in each classifier.

Data Hyperparameters To utilize the pretrained backbone model, we resize all inputs to 224×224 using bilinear interpolation. During training, we augment CIFAR-100 and FEMNIST images by randomly cropping and randomly flipping the images horizontally. Following best practices, we normalize all images with the mean and SD of the respective dataset.

B. Baseline Details.

In this section, we present further details of each baseline that we compare with. ExclusiveFL is a standard baseline commonly used in existing works where clients with insufficient compute or memory are left out. InclusiveFL and DepthFL are depth-based approaches, which prune the deeper layers of the model to accommodate resource-constrained clients, showing considerable gains over previous width-based scaling methods. As the model is pruned by depth, these methods deploy a separate classifier at each exit.

Two key differences between InclusiveFL and DepthFL include how local optimization is performed and how knowledge transfer is utilized. In InclusiveFL, each client trains its transformer-based sub-model and the sub-model's deepest classifier. On the server, InclusiveFL distills knowledge from deeper transformer layers to earlier transformer layers by injecting a gradient momentum distillation term that applies an extent of the average gradients of the deeper layers to the earlier layers. In contrast, each client in DepthFL trains all classifiers in its sub-model, along with the sub-model itself, and employ mutual self-distillation where each classifier acts as a teacher for the other classifiers. DepthFL also manually hand-picked different classifiers for different exits, with earlier exits having larger classifiers. As mentioned in Section. 4.1.3, we keep the classifier architecture fixed for all exits in ReeFL and the considered baselines for fair comparisons; the number of parameters of the global model for ReeFL, DepthFL, and InclusiveFL is similar.

ScaleFL, on the other hand, does both width-based and depth-based scaling, often resulting in models that are more efficient than depth-based scaling approaches. Besides additionally pruning channels, ScaleFL employs a grid-search to decide where to prune the global model for each sub-model

as opposed to pruning the global model uniformly in ReeFL, DepthFL, and InclusiveFL.

C. Teacher Sub-model Selection

In ReeFL, we select the best performing teacher sub-model to distill from per client based on the running estimate of the training loss of the respective client (Section. 3.2). In Figure. 7, we show the average test performance of ReeFL for each exit and the number of times each exit is selected as the teacher sub-model across all **max budget** clients for a single round using the running estimate of the training loss versus using the training loss of each mini-batch. We observe that *1)* the best performing exit gets selected the most often across clients and *2)* utilizing the training loss without the running estimate often leads to picking sub-optimal teacher sub-models.

D. Extended Results

The following figures and tables extend the results in the main paper: Fig. 5 & 6 show the accuracy and cost comparisons on CIFAR-100 $\alpha = 1000$ and $\alpha = 0.1$ respectively. Fig. 10 & 8 show the mean accuracy per exit for 4 & 12 exits respectively. Fig. 9 shows the impact of our proposed knowledge distillation for 12 exits. Lastly, Table. 10 shows an ablation on the aggregation strategies and knowledge distillation, comparing our ReeFL with previous depth-based scaling methods for 12 exits, and Table. 11 highlights the benefits of ReeFL's feature modulation for 12 exits.

E. Attention Map Visualization

To understand the "exploration" and "exploitation" impact of Ree on class tokens, z^l_{cls}, for $l = 1, \ldots, L$, we compare three attention maps:

$$\text{attn}^l_x := \text{attentionMap}^l(z^{l-1}_{\text{cls}}, z^{l-1}_{1:n}) \tag{7}$$

$$\text{attn}^l_m := \text{attentionMap}^l(m^l_l, z^{l-1}_{1:n}) \tag{8}$$

$$\text{attn}^l_c := \text{attentionMap}^l(m^l_0 + z^l_{\text{cls}}, z^{l-1}_{1:n}) \tag{9}$$

where $\text{attentionMap}^l(a, b_{1:n})$ is the first row without the first element (i.e., attention between a and a) of the attention map of $\text{MSA}^l(\text{LN}^l_1([a, b_1, \ldots, b_n]))$ of shape $(n+1, n+1, \#heads)$ after taking a mean operation over the multi-head dimension. We show a few examples for CIFAR-100 images in Fig 11.

F. Different Pretrained Models

In this section, we further run experiments on the much larger DeiT-B model (Touvron et al., 2021) and some additional preliminary experiments with a recent state space model (SSM), Vision Mamba (Vim) (Zhu et al., 2024). We then discuss the extendibility, limitation, and potential directions of applying Ree to other architectures.

DeiT-B. We show results for 4 exits for Frozen and the best performing PEFT approach, LoRA, across 3 runs, in Table. 8. Comparing Table. 8 and Table. 1, utilizing the bigger DeiT-B results in a drop in performance compared with DeiT-S. This is primarily due to overfitting, fitting a much larger model on each client's small dataset. Nonetheless, the relative performance ranking among these approaches stays the same across benchmarks for both DeiT-S and DeiT-B.

Vim. Besides transformers, we adopt pretrained Vim-T, the tiny variant of Vim which is much smaller than DeiT-S. As this is a different architecture, we run the same grid search detailed in Appendix Section. A and pick the best performing hyperparameters for each baseline. We show results for 4 exits comparing ReeFL with the second best-performing baseline, DepthFL, on both a frozen backbone and full fine-tuning in Table. 9.

Comparing the Table. 9 with Table. 1, utilizing features from a smaller frozen backbone model leads to a considerable drop in performance in all cases. Notably, for SpeechCommands, both ReeFL and DepthFL fail to utilize the pretrained features. For full fine-tuning, we observe that using the smaller Vim-T mitigates the overfitting issue in DeiT-S+DepthFL, resulting in a gain in performance. Nonetheless, ReeFL still outperforms DepthFL and the performance gap between Vim-T and DeiT-S for ReeFL is consistent with the results shown in (Zhu et al., 2024). Note that as vision SSMs are a new addition to the literature, there is no consensus on how to effectively apply existing PEFT approaches to it. We leave the exploration of PEFT methods on SSM as a potential future work.

In general, ReeFL can be applied, out-of-the-box, to any model that has a uniform state size at every layer. Extending to architectures with a non-uniform state size, for example ResNets, however, is non-trivial as Ree's feature modulation requires this state size to be uniform across layers. One possible future direction is to generate scaling and shifting parameters for feature modulation.

G. Additional Discussion

In this section, we supplement existing insights found in Section 4 by further discussing the performance gap found between ReeFL and the second best-performing baseline DepthFL. In Section. 4.3, we show that both knowledge distillation and feature modulation can help improve performance in ReeFL. Nonetheless, in some cases, removing knowledge distillation and feature modulation in ReeFL still leads to a better performance than the second best perform-

ing baseline, DepthFL. This performance gap comes with the use of the Ree which fuses features from multiple layers for each exit and is trained on the full dataset in contrast with DepthFL which uses separate classifiers for each exit, where deeper classifiers are trained on partial data.

To elucidate this, we focus on the same parameter aggregation (FedAvg) without knowledge distillation for both ReeFL and DepthFL given a particular scenario (CIFAR-100 $\alpha = 1000$, LoRA, 4 exits). In this scenario, DepthFL has a mean accuracy of 71.22 and ReeFL has a mean accuracy of 73.47 as shown in Table. 3. We remove 1) both the knowledge distillation and modulation of ReeFL which results in a mean accuracy of 73.21 and 2) remove the Ree module entirely and use a single shared classifier for different exits which results in a mean accuracy of 70.74. DepthFL outperforms the use of a single shared classifier (71.22 vs 70.74) as it uses multiple classifiers per exit, despite deeper classifiers being partially trained, to better handle the different feature representations of different layers. With the inclusion of the Ree module, these multi-layer features are learnt to be aggregated to improve downstream classification, outperforming DepthFL (73.21 vs 71.22). We hypothesize that using FedDyn with ReeFL may potentially lead to higher accuracy gains in some cases as it did with DepthFL and we leave that as a possible direction for future work.

We would also like to point out that in this scenario, simply using a shared classifier outperforms training with 25% of the whole dataset: ExclusiveFL (70.74 vs 67.46). It also outperforms the case where each classifier is trained with separate subsets of the dataset: InclusiveFL (70.74 vs 69.37). This is not only because the shared classifier is trained on the full dataset but also because the features of the backbone model are fine-tuned jointly with the classifier via LoRA. If we, instead, consider a frozen backbone (CIFAR-100 $\alpha = 1000$, Frozen, 4 exits), using a shared classifier results in a mean accuracy of 41.74 which is significantly lower than ExclusiveFL (48.02) and InclusiveFL (53.99), as well as DepthFL, due to the differences in features among layers. Including the Ree module can better handle these differences, achieving better performance over these baselines.

H. Ensemble Performance

We adopt the same ensemble method used in DepthFL: taking the average logits of all exits. Specifically, we pick the best performing run in Table. 1 and Table. 2, for 4 and 12 exits respectively, and compute the ensemble performance for both DepthFL and ReeFL (Table. 6 and Table. 7). Unsurprisingly, ensembles lead to an improvement in performance in most cases, with a drop in performance only in the case where the majority of the exits has poor performance.

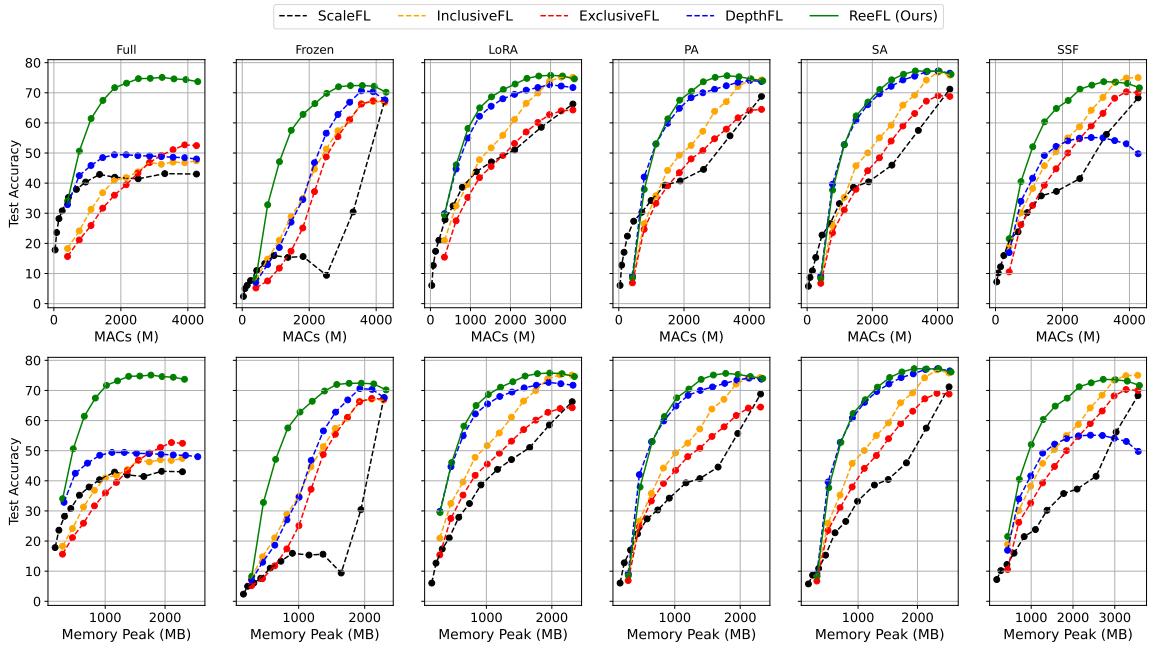


Figure 5. Quantifying training costs for each exit for CIFAR-100, $\alpha = 1000$ for 12 exits, where each dot along each line represents an exit.

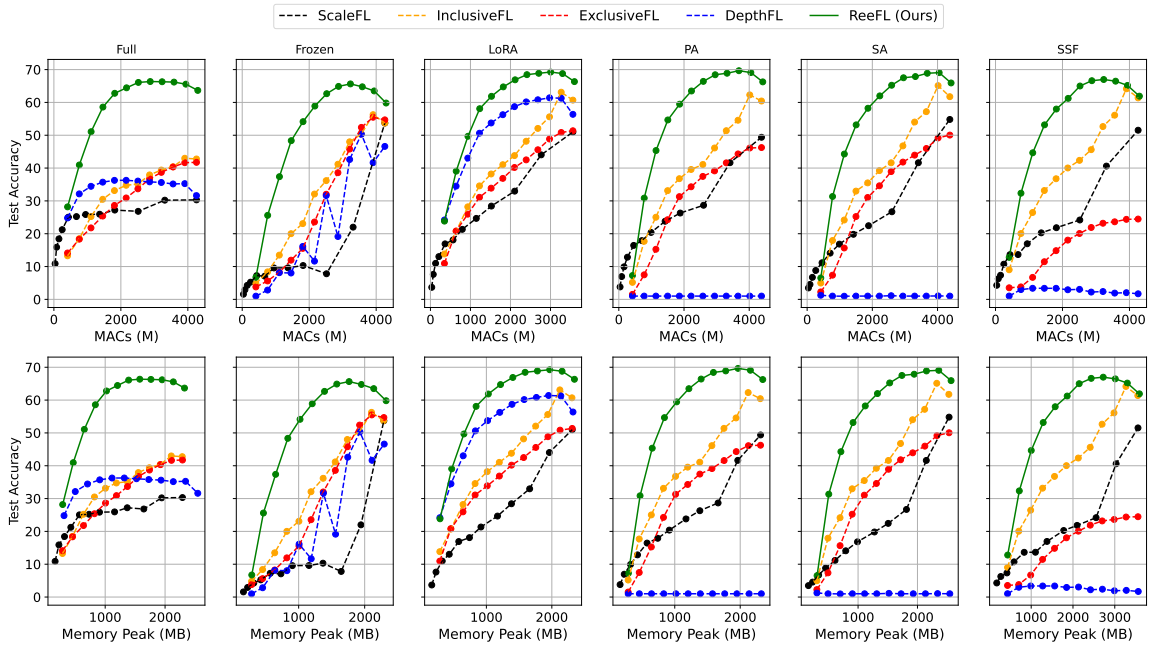


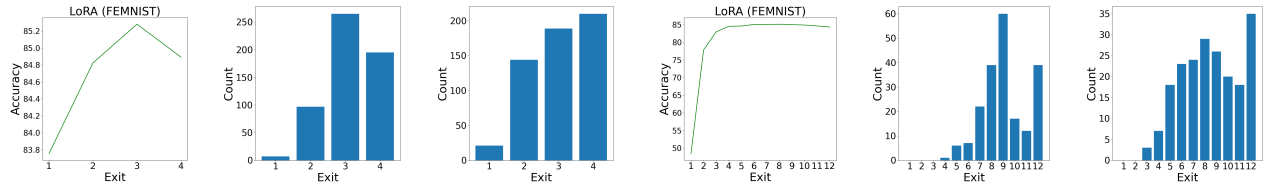
Figure 6. Quantifying training costs for each exit for CIFAR-100, $\alpha = 0.1$ for 12 exits, where each dot along each line represents an exit.

Table 6. **4 exits** ensemble performance for both DepthFL and ReeFL.

Finetuning	Approach	CIFAR-100			FEMNIST	SpeechCmds
		α=1000	α=1.0	α=0.1		
Full	DepthFL	56.74	54.66	46.13	81.77	78.33
	ReeFL (ours)	**81.16**	**80.69**	**79**	**86.71**	**85.06**
Frozen	DepthFL	66.67	61.97	21.73	53.53	24.65
	ReeFL (ours)	**74.86**	**73.58**	**69.66**	**83.69**	**66.16**
LoRA	DepthFL	73.53	72.08	66.37	81.61	76.66
	ReeFL (ours)	**80.32**	**79.75**	**76.05**	**85.38**	**79.97**
PA	DepthFL	75.21	73.81	69.02	82.26	73.64
	ReeFL (ours)	**80.35**	**79.34**	**75.35**	**84.15**	**78.72**
SA	DepthFL	71.46	73.61	67.16	82.52	74.66
	ReeFL (ours)	**72.25**	**78.51**	**73.86**	**84.6**	**78.12**
SSF	DepthFL	44.53	41.94	31.36	74.88	65.04
	ReeFL (ours)	**70.17**	**76.72**	**72.4**	**84.37**	**74.36**

Table 7. **12 exits** ensemble performance for both DepthFL and ReeFL.

Finetuning	Approach	CIFAR-100			FEMNIST	SpeechCmds
		α=1000	α=1.0	α=0.1		
Full	DepthFL	47.26	45.96	35.3	80.53	70.18
	ReeFL (ours)	**74.9**	**73.8**	**66.08**	**85.75**	**81.95**
Frozen	DepthFL	65.74	61.75	23.15	53.35	21.32
	ReeFL (ours)	**71.72**	**71.5**	**63.58**	**84.35**	**60.01**
LoRA	DepthFL	69.26	67.82	56	79.53	69.83
	ReeFL (ours)	**76.03**	**75.63**	**69.01**	**85.12**	**75.48**
PA	DepthFL	71.26	70.49	1.02	5.01	65.21
	ReeFL (ours)	**75.91**	**75.44**	**68.56**	**77.66**	**72.32**
SA	DepthFL	73.88	71.92	1	4.11	65.44
	ReeFL (ours)	**76.52**	**74.86**	**68.04**	**85.25**	**71.86**
SSF	DepthFL	52.5	48.62	1.02	5.21	3.11
	ReeFL (ours)	**72.65**	**72.49**	**65.81**	**84.54**	**67.24**



(a) 4 Exits (b) 12 Exits

Figure 7. LoRA finetuning on FEMNIST with ReeFL mean accuracy per exit (left), number of times each exit is selected as the teacher sub-model for knowledge distillation across **max resource budget clients** while taking the **running estimate** of the training loss for a single round (middle), and selecting the teacher sub-model by simply using the training loss of each mini-batch (right).

Table 8. Experiments on DeiT-B. Mean and standard deviation (SD) of the mean performance of all **4 exits** across 3 runs.

Finetuning	Approach	CIFAR-100			FEMNIST	SpeechCmds
		α=1000	α=1.0	α=0.1		
Frozen	ExclusiveFL	46.36±0.03	45.56±0.05	40.92±0.03	49.03±0.01	19.62±0.05
	InclusiveFL	52.97±0.11	51.97±0.01	48.24±0.04	64.37±0.03	30.14±0.01
	ScaleFL	29.96±0.0	28.92±0.0	26.43±0.0	29.5±0.01	11.95±0.01
	DepthFL	46.67±0.21	42.75±0.37	28.41±0.1	35.2±1.0	24.68±0.33
	ReeFL (ours)	**67.6±0.03**	**66.47±0.08**	**62.36±0.14**	**81.06±0.03**	**65.37±0.24**
LoRA	ExclusiveFL	65.24±0.06	64.22±0.04	57.23±0.08	81.68±0.06	67.04±0.06
	InclusiveFL	66.77±0.01	66.32±0.01	61.23±0.04	83.3±0.01	76.04±0.13
	ScaleFL	51.11±0.03	50.27±0.06	43.49±0.12	66.74±0.04	39.25±0.11
	DepthFL	70.4±0.11	69.04±0.14	61.85±0.05	80.78±0.71	76.38±0.19
	ReeFL (ours)	**71.92±0.01**	**71.29±0.1**	**66.94±0.01**	**83.85±0.16**	**77.07±0.05**

Table 9. Experiments on Vim comparing ReeFL with DepthFL. Mean and standard deviation (SD) of the mean performance of all **4 exits** across 3 runs.

Finetuning	Approach	CIFAR-100			FEMNIST	SpeechCmds
		α=1000	α=1.0	α=0.1		
Frozen	DepthFL	37.23±0.01	36.53±0.08	27.82±0.42	45.41±0.65	**18.92±0.02**
	ReeFL (ours)	**46.63±0.48**	**45.43±1.28**	**38.53±0.3**	**74.17±0.37**	11.95±1.45
Full	DepthFL	64.36±0.11	63.87±0.1	56.93±0.18	80.0±0.25	70.07±0.68
	ReeFL (ours)	**74.08±0.1**	**73.79±0.1**	**70.68±0.23**	**85.42±0.0**	**81.87±0.28**

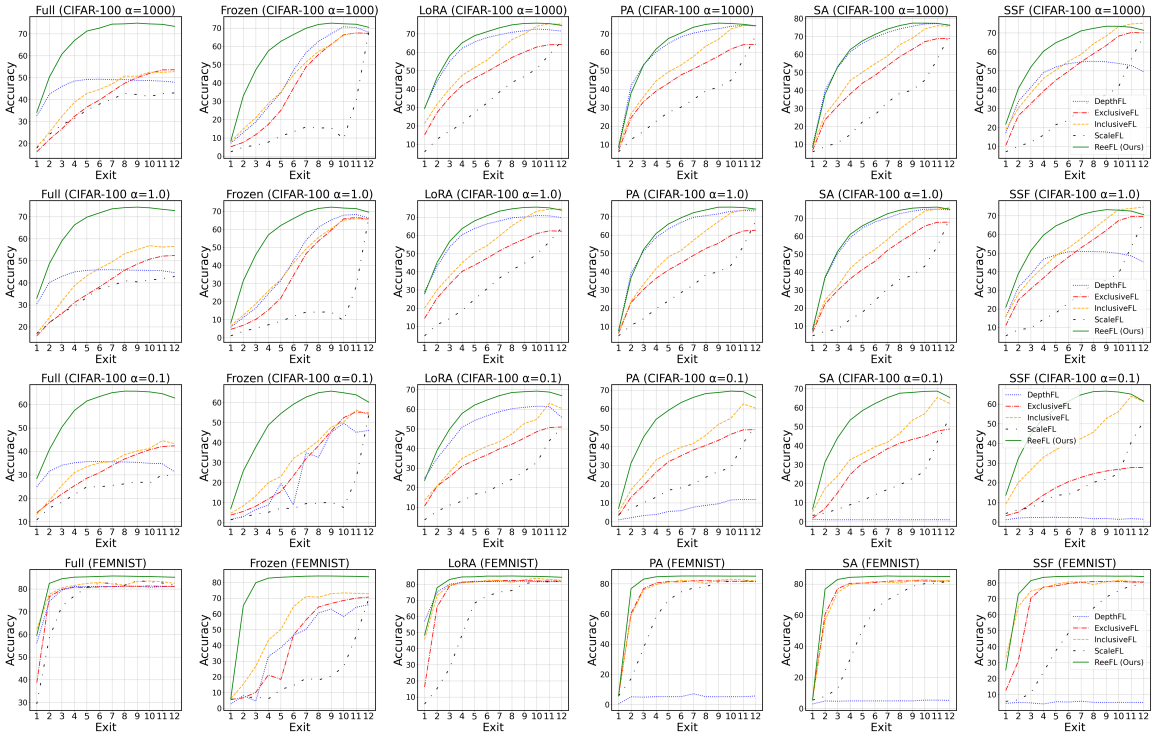


Figure 8. Fig 2's extended results. Mean accuracy of each exit across 3 runs.

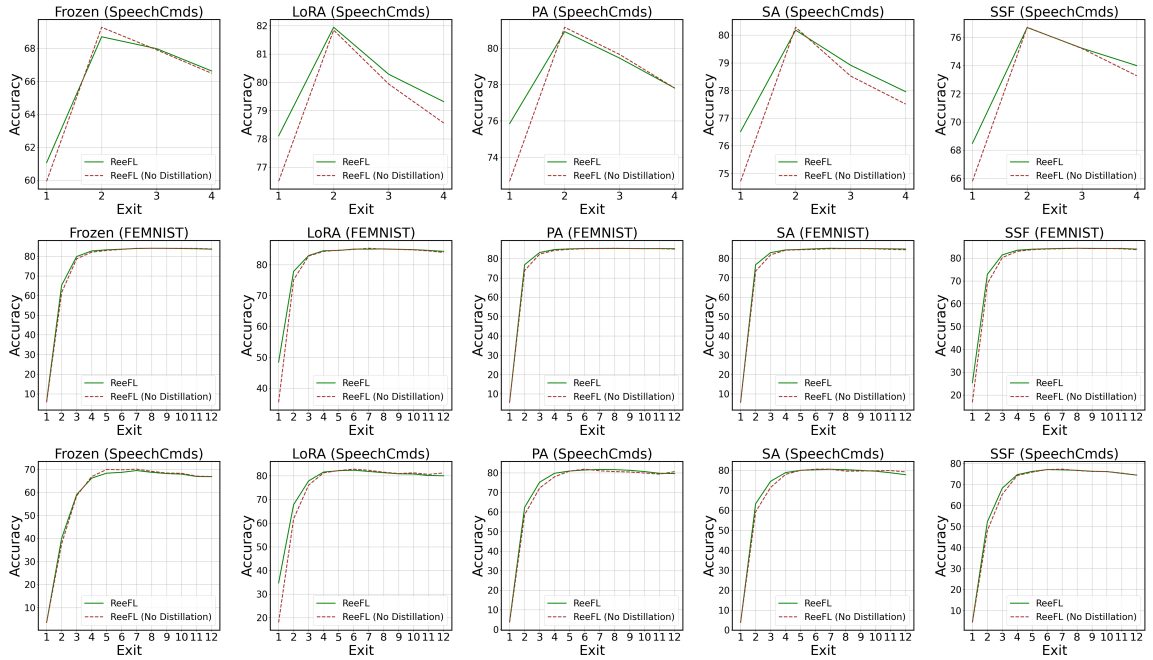


Figure 9. Impact of ReeFL's proposed knowledge distillation on FEMNIST and SpeechCommands (4 & 12 exits).

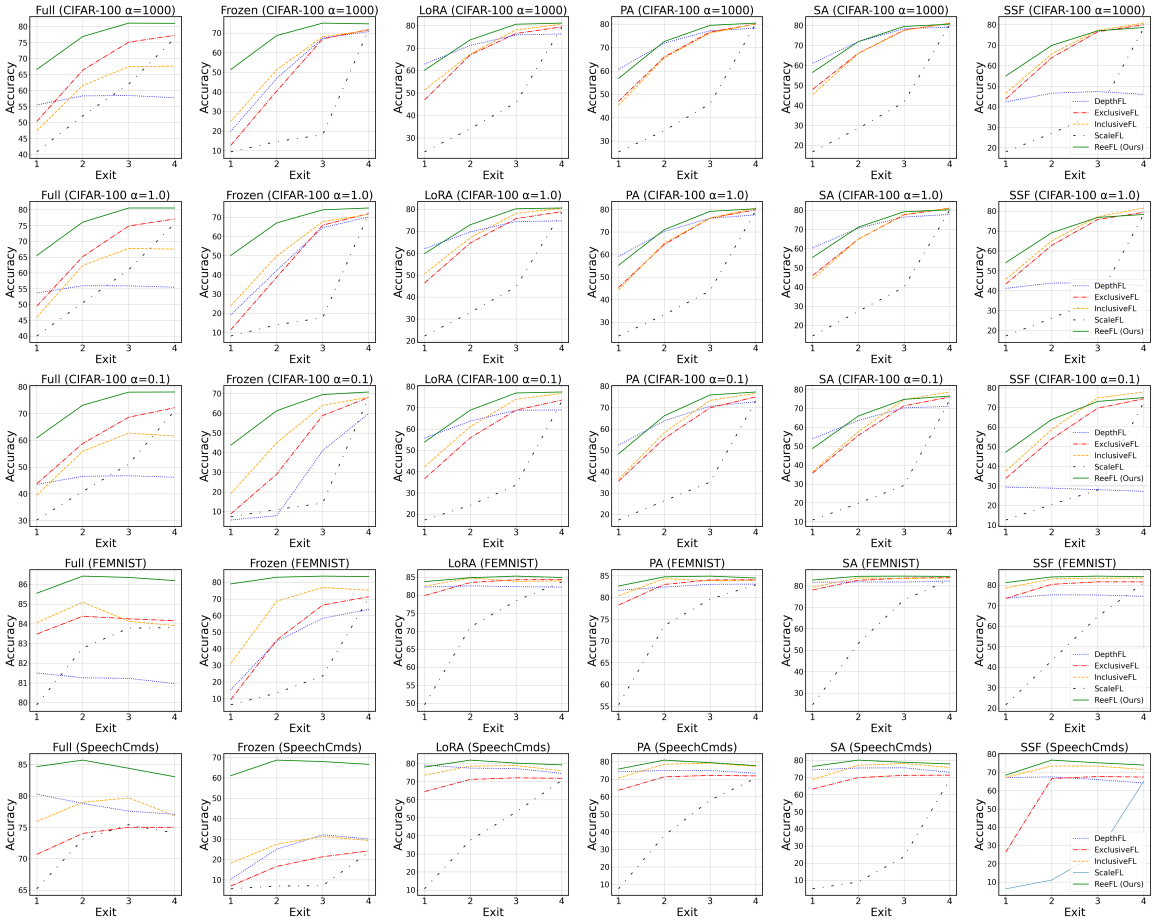


Figure 10. Fig 2's extended results. Mean accuracy of each exit across 3 runs.

Table 10. Ablation study on different aggregation strategies and knowledge distillation of depth-based scaling methods on 12 exits.

Finetuning	Approach	Distillation	Aggregation	CIFAR-100			FEMNIST	SpeechCmds
				α=1000	α=1.0	α=0.1		
Frozen	InclusiveFL	-	FedAvg	36.98±0.01	34.59±0.02	26.44±0.02	40.88±0.03	13.93±0.05
	InclusiveFL	-	FedAdam	43.53±0.03	41.6±0.04	32.28±0.05	53.35±0.03	20.22±0.05
	DepthFL	✗	FedAvg	42.6±0.02	41.87±0.02	36.62±0.01	43.11±0.17	15.47±0.03
	DepthFL	✗	FedDyn	45.12±0.02	43.4±0.16	36.09±0.39	48.52±0.23	21.19±0.17
	DepthFL	✓	FedAvg	43.31±0.0	42.55±0.0	36.75±0.13	43.25±0.01	16.01±0.2
	DepthFL	✓	FedDyn	45.22±0.06	42.93±0.18	25.2±1.89	41.29±0.1	21.3±0.02
	ReeFL (ours)	✗	FedAvg	58.28±0.03	57.7±0.03	51.6±0.32	74.87±0.15	**59.69±0.57**
	ReeFL (ours)	✓	FedAvg	**58.74±0.05**	**58.1±0.09**	**51.33±0.36**	**75.36±0.02**	59.5±0.5
LoRA	InclusiveFL	✗	FedAvg	55.79±0.03	54.17±0.01	40.35±0.06	73.24±0.01	55.01±0.14
	InclusiveFL	✗	FedAdam	55.9±0.08	54.54±0.15	41.53±0.15	78.21±0.15	66.18±0.58
	InclusiveFL	✓	FedAvg	52.17±0.01	50.24±0.07	37.61±0.05	69.22±0.07	53.53±0.01
	InclusiveFL	✓	FedAdam	54.13±0.19	52.69±0.1	39.32±0.09	78.32±0.14	67.49±0.03
	DepthFL	✗	FedAvg	63.13±0.08	62.79±0.07	56.69±0.23	77.55±0.1	67.47±0.23
	DepthFL	✗	FedDyn	62.79±0.11	61.55±0.17	53.5±0.15	78.87±0.27	70.91±0.2
	DepthFL	✓	FedAvg	62.73±0.0	62.17±0.01	54.9±0.01	77.54±0.0	66.62±0.08
	DepthFL	✓	FedDyn	62.74±0.09	61.3±0.11	51.9±0.15	78.97±0.15	70.66±0.07
	ReeFL (ours)	✗	FedAvg	65.16±0.04	64.2±0.09	58.69±0.09	79.68±0.19	74.18±0.25
	ReeFL (ours)	✓	FedAvg	**65.7±0.09**	**65.01±0.03**	**58.85±0.05**	**81.06±0.06**	**75.96±0.18**

Table 11. Mean Accuracy with and without ReeFL's feature modulation on 12 exits.

Finetuning	Modulation	CIFAR-100			FEMNIST	SpeechCmds
		α=1000	α=1.0	α=0.1		
Frozen	✗	43.4±0.1	41.88±0.29	25.6±17.4	49.59±0.37	17.52±0.07
	✓	**58.74±0.05**	**58.1±0.09**	**51.33±0.36**	**75.36±0.02**	**59.5±0.5**
LoRA	✗	64.99±0.12	64.68±0.1	58.69±0.15	80.75±0.07	**76.5±0.27**
	✓	**65.7±0.09**	**65.01±0.03**	**58.85±0.05**	**81.06±0.06**	75.96±0.18
PA	✗	60.5±0.14	60.28±0.07	53.97±0.1	77.01±0.05	71.13±0.08
	✓	**62.35±0.07**	**61.47±0.12**	**55.67±0.14**	**77.62±0.04**	**72.37±0.02**
SA	✗	61.83±0.05	60.95±0.12	**55.19±0.12**	76.75±0.06	70.69±0.23
	✓	**63.28±0.13**	**61.94±0.2**	54.99±0.1	**77.43±0.04**	**71.52±0.13**
SSF	✗	58.18±0.09	57.46±0.13	51.33±0.64	75.79±0.08	65.02±0.23
	✓	**61.85±0.02**	**61.25±0.17**	**54.42±0.13**	**78.05±0.03**	**67.44±0.24**

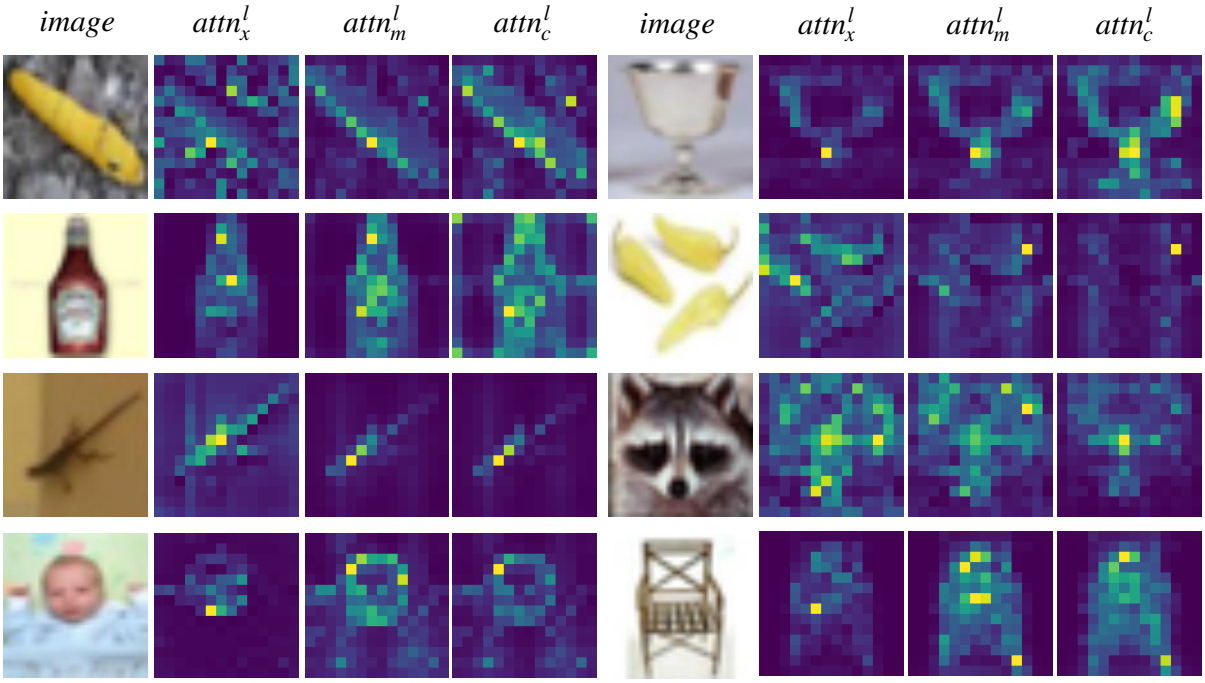


Figure 11. Attention map comparison between attn_x^l, attn_m^l and attn_c^l. We select to visualize the earliest block whose classification is correct.