DarkDistill: Difficulty-Aligned Federated Early-Exit Network Training on Heterogeneous Devices

Lehao Qu*
SKLCCSE Lab
Beihang University
Beijing, China
lehaoqv@buaa.edu.cn

Shuyuan Li*
Department of Data Science
City University of Hong Kong
Hong Kong, China
shuyuan.li@cityu.edu.hk

Zimu Zhou
Department of Data Science
City University of Hong Kong
Hong Kong, China
zimuzhou@cityu.edu.hk

Boyi Liu
SKLCCSE Lab
Beihang University
Beijing, China
boyliu@buaa.edu.cn

Yi Xu
SKLCCSE Lab
Beihang University
Beijing, China
xuy@buaa.edu.cn

Yongxin Tong
SKLCCSE Lab
Beihang University
Beijing, China
yxtong@buaa.edu.cn

Abstract

Early-exit networks (EENs), which adapt their computational depths based on input samples, are widely adopted to accelerate inference in edge computing applications. The effectiveness of EENs relies on difficulty-aware training, which tailors shallow exits for simple samples and deep exits for complex ones. However, existing difficulty-aware training schemes assume centralized environments with sufficient data, which become invalid with real-world edge devices. In this paper, we explore difficulty-aware training in a federated manner, where EENs are collaboratively trained on heterogeneous devices. We observe the cross-model exit unalignment phenomenon, a unique problem when aggregating local EENs into a cohesive global model. To address this problem, we design a novel Difficulty-Aligned Reverse Knowledge Distillation scheme named DarkDistill that preserves the difficulty-specific specialization for aggregating heterogeneous local models. Instead of direct parameter averaging, it trains difficulty-conditional data generators, and selectively transfers generated knowledge of specific difficulty among matched exits of heterogeneous EENs. Evaluations show that DarkDistill outperforms the state-of-the-arts in both full-parameter and parameter-efficient fine-tuning of EENs.

CCS Concepts

• **Computing methodologies** → **Learning paradigms**.

Keywords

Federated Learning on Heterogeneous Devices; Early-Exit Networks; Knowledge Distillation

*Equal contribution

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3736902

ACM Reference Format:
Lehao Qu, Shuyuan Li, Zimu Zhou, Boyi Liu, Yi Xu, and Yongxin Tong. 2025. DarkDistill: Difficulty-Aligned Federated Early-Exit Network Training on Heterogeneous Devices. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3736902

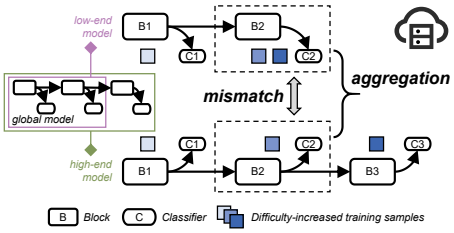


Figure 1: Cross-model Exit Unalignment.

1 Introduction

Deep neural networks (DNNs) are increasingly deployed on edge devices for *continuous*, *real-time* inference tasks such as traffic analysis [2], autonomous driving [38], and well-being monitoring [34]. Given the varied difficulty across input samples, a promising approach to accelerate inference is to adaptively allocate the model's computation [10]: more for complex samples and less for simple ones. Early-exit networks (EENs) [36, 44] exemplify this adaptive inference strategy via adjustable model *depth* by attaching intermediate classifiers (*i.e.*, early exits) to each block of the backbone. These exits enable samples with high-confidence predictions to terminate their inference at shallow layers, effectively reducing latency.

Despite their benefits, EENs require dedicated training strategies compared to single-exit network. Particularly, the exits are often jointly optimized to resolve cross-exit interference and boost accuracy [5, 11, 23, 26, 42, 54]. An emerging strategy for synergetic exit training is *difficulty-aware training* [5, 11, 54], which tailors *shallow* exits for *simple* samples and *deep* exits for *complex* samples. This paradigm encourages exit specialization by either directing samples

misclassified by shallow exits to deep ones [54], or increasing the weight of complex samples on training deep exits [5, 11].

Difficulty-aware training of EENs implicitly assume *centralized* environments with ample data availability. Yet in edge computing applications, training data are often decentralized, with each device holding a limited dataset [12, 52]. This urges adapting difficulty-aware training to a *federated learning* (FL) setup [30, 51], where multiple devices collaboratively learn an EENs via iterative *local training* and *model aggregation* coordinated by a server.

Enabling *difficulty-aware federated training* on *heterogeneous* devices faces unique challenges. Ideally, shallow (deep) exits should be trained with simple (complex) samples, *both* during local training and model aggregation. Enforcing gradually increased sample difficulty along exits in local models is achievable using existing solutions [5, 11, 54]. Yet this condition fails when aggregating local models from heterogeneous devices, as their exits at equivalent depths may handle samples from disparate difficulty ranges. This mismatch, termed as *cross-model exit unalignment*, calls for new *aggregation* strategies to align exits across local models, thereby injecting difficulty-awareness into federated learning (see Fig. 1).

To address the cross-model exit unalignment problem, we draw on recent studies [59] that advocate knowledge distillation (KD) [13] over direct parameter averaging for aggregating single-exit network. Building upon this principle, we leverage KD for *multi-exit* network *aggregation* in a *difficulty-aligned* manner. Although EENs and KD are often coupled in federated learning with heterogeneous clients [7], existing methods [18, 20, 25, 29] cannot solve our problem for two reasons. *(i)* Their objective is not to train EENs for *input-adaptive* inference. Some [18, 29] use early exits to align features among heterogeneous local models but discard them after training, resulting in *static* inference. Others [20, 25] target at adapting to *runtime resources* rather than input samples at inference. Accordingly, their KD is *agonistic* to sample difficulties, let alone align difficulty ranges across exits. *(ii)* They [18, 20, 25] mainly conduct *client-side* KD among exits *within* the same local model, whereas the *cross-model* exit unalignment arises during the aggregation of these models at the *server*.

In this paper, we propose DarkDistill, a novel Difficulty-Aligned Reverse Knowledge Distillation scheme for federated EENs training. Given multiple EENs of varied depths, each locally trained to specialize on increasing difficulty ranges [5, 11, 54], DarkDistill aggregates them into a global EEN that preserves this difficulty-specific specialization. Achieving this objective confronts two challenges: *(i)* knowledge transfer among unaligned exits of heterogeneous local EENs and *(ii)* server-side distillation without local data. DarkDistill tackles these challenges with two designs.

- *Selective, progressive, and reverse KD.* Knowledge is only transferred between exits of *matched* difficulty ranges, which is key to difficulty-aligned model aggregation. Distillation proceeds progressively from *shallow* models to *deep* ones of *adjacent* depths. This enables effective knowledge transfer because *(i)* shallow models are locally trained with more data in FL with heterogeneous clients [32], and *(ii)* it avoids distillation between models of serious capacity gaps [31].
- *Difficulty-conditional data generation.* We train a data generator that produces pseudo data at specified difficulty to support the above KD scheme. In standard KD [13], students imitate the responses of teachers to the target dataset, which is inaccessible at the server in FL. Inspired by data-free KD [56, 59], we develop a lightweight data generator to produce pseudo data of controllable difficulty.

The main contributions of this paper are summarized as follows:

- To our best knowledge, this is the first work that identifies and addresses the cross-model exit misalignment problem for federated EEN training on heterogeneous devices. It enables the deployment of input-adaptive DNN to practical edge computing applications.
- We propose DarkDistill, a new data-free KD framework for difficulty-aligned aggregation of heterogeneous local EENs. It advances EEN training by extending difficulty-awareness to the federated context.
- Extensive evaluations on image and speech classification show that our solution outperforms state-of-the-arts baselines [18, 20, 25, 29] in both full-parameter and parameter-efficient fine-tuning across varied resource distributions.

2 Related Work

Early-Exit Network. EEN [44] is a subclass of dynamic neural network [10] to enhance inference efficiency by tailoring their computational depths to inputs. Research on EEN [24, 40] spans three main areas: architecture design [8, 16], exit policies [17, 27], and training techniques, with training being critical yet under-explored. Typically, exits are jointly trained [5, 11, 16, 54] because isolated training tends to yield lower accuracy [24, 40]. We focus on difficulty-aware training [5, 11, 54], an emerging joint-exit training strategy to mitigate train-test exit mismatches, by aligning each exit's training with the input types it would expect during inference. Yet existing difficulty-aware training methods [5, 11, 54] are designed for centralized settings. In contrast, we focus on difficulty-aware training in federated environments.

Heterogeneous Federated Learning. Different from the traditional challenges of devices collaboration such as the dynamic spatiotemporal information [43, 45, 46], the asynchronous demand and supply of task [3, 9, 28] and the adaptive task assignment [37, 55], federate learning on edge devices has faced the unique effective training challenge across devices with diverse computational resources [7]. To involve all devices into training, a prevailing strategy is to assign sub-models tailored to each device's computational power by scaling the global model in widths [1, 7, 14], depths [20, 25, 29, 41], or both [4, 18]. EENs, with their natural depth-scaling capabilities, have been utilized in recent depth-based FL proposals [20, 25, 29] to align feature representations of exits at the same depth across sub-models. However, these exits are discarded after training [18], leading to static models at inference. Although some studies [20, 25] have evaluated the use of these exits for adaptive inference, they focus on resource adaptation rather than input-adaptive inference. In contrast, we prioritize input-adaptive inference through difficulty-aware training and transform its specific knowledge between matched exits across varied depth models.

Knowledge Distillation. KD [13] is an effective knowledge transfer strategy between teacher and student models or within a single model, *i.e.*, self-distillation [57]. KD has been applied in FL as an

alternative *model aggregation* scheme over direct parameter averaging [30] to aggregate heterogeneous, single-exit models [59]. It has also been utilized in EEN-based FL [18, 20, 25] as self-distillation of local EEN during *local training* at clients. In contrast, we leverage KD for a new problem, *i.e.*, aggregating EENs at the server in a difficulty-aligned manner. Our solution is also inspired by data-free distillation [50, 56, 59], which maintains a data generator for server-side KD without access to local datasets. We extend the idea by training data generators conditioned on difficulty of samples.

3 Problem Statement

Early-Exit network. Early-exit network parameterized by θ consists of M blocks and the classifiers connected after each block, and each classifier can be considered as an exit. EENs terminate the inference of an input sample at an early exit when its prediction exceeds a confidence threshold [44].

Difficulty-Aware Training. The exits in EENs are often jointly trained to enhance accuracy of all exits together [16, 44]:

$$\mathcal{L}(\theta; D) = \sum_{m=1}^{M} \omega^m \mathcal{L}^m(\theta; D) = \sum_{m=1}^{M} \omega^m \sum_{i=1}^{|D|} l_i^m \tag{1}$$

where D is the training dataset of size $|D|$, and $\omega^m, \mathcal{L}^m$ denote the weight and the loss of exit m, respectively. For the loss of i-th sample at exit m, $l_i^m = \mathrm{CE}(v(h^m(x_i;\theta)), y_i)$, where $v(\cdot)$ is the softmax function, which will output the prediction score of x_i based on logits $h^m(x_i;\theta)$, and $\mathrm{CE}(\cdot)$ is the cross-entropy loss function.

Difficulty-aware training [5, 11, 54] is an emerging joint training strategy that aims to match EEN training to its inference. Since easy samples tend to terminate their inference at shallow exits, difficulty-aware training emulates this phenomenon by assigning samples of increasing difficulty to exits based on their depth during training. This can be achieved by learning sample- and exit-specific weights ω_i^m based on corresponding losses [5, 11] or replacing the single-exit logits $h^m(x_i;\theta)$ with accumulative ones (over exits) [54].

Difficulty-Aware Training in EEN-based FL. We focus on difficulty-aware training in standard EEN-based FL setups [7]. Assume K clients with local training datasets $\{D_1, ..., D_K\}$ and local EENs parameterized by $\{\theta_g[:m_1], \ldots, \theta_g[:m_K]\}$, where $\theta_g[:m_k]$ represents model parameters till exit m_k in the global EEN θ_g. The K clients have heterogeneous resources, and m_k denotes the maximum number of exits that client k can afford for local training.

$$\mathcal{L}(\theta_g; D_1, ...D_K) = \sum_{m=1}^{M} \sum_{k \in S_m} \frac{|D_k|}{|D_m|} \mathcal{L}_k^m(\theta_g[:m_k]; D_k) \tag{2}$$

where $\mathcal{L}_k^m$ is the loss of client k at exit m on local dataset D_k as defined in Eq. (1), S_m is the set of clients with m exits, and $|D_m| = \sum_{k \in S_m} |D_k|$ is the sum of the quantities of the local training samples across client set S_m.

Although $\mathcal{L}_k$ can be optimized via existing difficulty-aware training schemes [5, 11, 54], directly averaging parameters of local EENs with different depths as standard FedAvg [30] would mix exits locally trained for different difficulty ranges (as explained in Fig. 1). Such *cross-model exit unalignment* phenomenon is unique when incorporating difficulty-awareness in EEN-based FL. It calls for new *model aggregation* strategies, which motivates our design.

Discussions. We make the following notes on our problem setups.

- We quantify the *difficulty* of sample i as its training loss l_i^1 at the first exit of an EEN: $d_i' = l_i^1$. The training loss is a widely utilized indicator for sample difficulty in self-paced learning [19, 22] and difficulty-aware training [5, 11]. To reduce communication cost and protect privacy of local datasets, we further quantize d_i' as follows:

$$d_i = d^*, \ if \ d_i' \in [d^*, d^* + \frac{ln\ c}{r}) \tag{3}$$

  where $\{d^*\}$ are r discrete values uniformly dividing the difficulty range $[0, ln\ c]$ of all samples, and c is the number of classes. We use the quantized difficulty d_i in our designs.
- We assess the performance of EENs in two settings [11, 16, 54]. *(i) Anytime Inference*: It processes each testing sample independently, and the inference may terminate at any time due to *e.g.*, sudden changes in resource availability. *(ii) Budgeted Inference*: It processes a batch of testing samples given a throughput budget, where the EEN maximizes the throughput of correct predictions.

4 Method

4.1 DarkDistill Overview

Key Idea. DarkDistill addresses the cross-model exit unalignment problem via a *difficulty-aligned, data-free* knowledge distillation scheme to aggregate the local EENs into a global one (see Fig. 2). It measures the difficulty ranges of local datasets via a lightweight *difficulty assessment* module (Sec. 4.2), which assists in training a *difficulty-conditional data generator* (Sec. 4.3) that produces pseudo data for sever-side knowledge distillation. The pseudo data is utilized to align exits and transfer knowledge among matched exits of local EEN, and DarkDistill adopts a *progressive reverse knowledge distillation* strategy (Sec. 4.4) for effective knowledge transfer. In addition to such progressive distillation, we also provide a parallel variant (Sec. 4.5) to accelerate model aggregation.

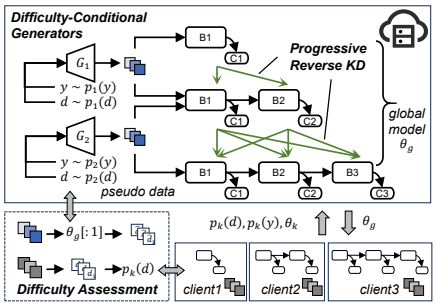


Figure 2: DarkDistill overview.

Workflow. DarkDistill adopts an iterative training framework in typical client-server based FL [30]. There are three steps in each communication round (see Algorithm 1).

- Step 1: *Local Processing*. After receiving the global model parameters θ, client k takes the parameters of the first m_k exits as its local model θ_k. Client k trains θ_k via existing (centralized) difficulty-aware schemes. Meanwhile, it measures

the difficulty distribution $p_k(d)$ and class distribution $p_k(y)$ of its local dataset D_k.

- Step 2: *Generator Training*. The server utilizes $\{p_k(d)\}$ uploaded by clients to train difficulty-conditional generators, which produce pseudo data with controllable difficulty levels. The data will assist in knowledge distillation between intermediate models in the server-side.
- Step 3: *Model Aggregation*. The server first aggregates the local models of the same number of exits into intermediate models via standard parameter averaging [30], where θ_m denotes an intermediate model with m exits. The sever then transfers knowledge of pseudo data among exits with matched difficulty ranges from an intermediate model with m exits to one with $m+1$ exits, starting from the shallowest intermediate model. After difficulty-aligned distillation, we aggregate all the intermediate models exit-wise [20] as the global model to be disseminated to clients in the next round.

We explain the unique modules in DarkDistill over standard FedAvg, *i.e.*, difficulty assessment, difficulty-conditional data generator, progressive reverse knowledge distillation and its acceleration below.

Algorithm 1: ServerExecute

Input: global model parameters θ^0, Generators parameters $\{\phi^0\}$, learning rate γ, β

for *communication round* $q = 1, ..., Q$ **do**
  $\theta^{q+1}, \{\theta_m^q\} \leftarrow 0$
  // Step1: Local Processing
  for *selected client* k **do**
    $\theta_k^{q+1}, p_k(d), p_k(y) \leftarrow$ **ClientExecute**(θ^q, D_k)
    $\theta_{m_k}^q \leftarrow \theta_{m_k}^q + \frac{|D_k|}{|D_{m_k}|}\theta_k^{q+1}$
  // Step2: Generator Training
  Attain $\{p_m(d)\}, \{p_m(y)\}$ based on $\{p_k(d)\}, \{p_k(y)\}$
  for $m = 1, ..., M-1$ **do**
    $\phi_m^{q+1} \leftarrow \phi_m^q - \beta \nabla_{\phi_m^q} \mathcal{L}_G(\phi_m^q, \theta_m^{q+1})$ ▷ By Eq. (5)
  // Step3: Model Aggregation
  for $m = 1, ..., M-1$ **do**
    Conditions $y \sim p_m(y), d \sim p_m(d), \epsilon \sim \mathcal{N}(0, I)$
    Generate $\xi_m \sim G_m(y, d, \epsilon; \phi_m)$ based on y, d, ϵ
    $g_{m+1} \leftarrow \nabla_{\theta_{m+1}^q} \mathcal{L}_{KD}(\theta_m^q, \theta_{m+1}^q; \xi_m)$ ▷ By Eq. (7)
  for $m = M, ..., 1$ **do**
    $\theta_m^{q+1} \leftarrow \theta_m^q - \gamma g_m$
    $\theta^{q+1}[m] \leftarrow \theta^{q+1}[m] + \sum_{m_1=m}^{M} \frac{|D_{m_1}|}{\sum_{m_2=m}^{M}|D_{m_2}|}\theta_{m_1}^{q+1}[m]$
  return θ^{q+1}

4.2 Difficulty Assessment of Local Datasets

This module measures the difficulty distribution of clients' local datasets. In edge computing scenarios, clients have not only heterogeneous compute capabilities but also *non-IID* datasets. Accordingly, the difficulty distributions of local datasets vary across clients, and should be assessed with a *unified* scale for difficulty-aligned KD.

Algorithm 2: ClientExecute

Input: Local model parameters θ^q, Local dataset D_k, Local epoch E, local model learning rate γ

$\theta_k^{q,0} \leftarrow \theta^q[:m_k]$
for *local epoch* $t = 1, ..., T$ **do**
  $\theta_l^{q,t} \leftarrow \theta_l^{q,t} - \gamma \nabla \mathcal{L}(\theta_k^{q,t-1}; D_k)$ ▷ By Eq. (1)
  $L^t \leftarrow \{l_i^1, ..., l_i^{m_k}\}_{i=1}^{|D_k|}$ on $\theta_k^{q,t}$
Attain $p_k(d)$ based on L^T as Eq. (3)
$p_y(d) \leftarrow \frac{n_k^y}{|D_k|}$, $\theta_k^{q+1} \leftarrow \theta_k^{q,T}$
return $\theta_k^{q+1}, p_k(d), p_k(y)$

We normalize the difficulty d_i of sample i by performing inference on the *global* EEN as the unified scale, and takes the loss at the first layer as the difficulty estimate (see Eq. (3)). Then the difficulty distribution of client k is calculated as $p_k(d) = \frac{n_k^d}{|D_k|}$, where n_k^d is the number of samples of difficulty d in client k. The above difficulty assessment should be performed every communication round, since the global model used for inference on local datasets is updated in every round.

To avoid per-round EEN inference on clients, we leverage the loss each sample derived from the *last local training epoch* as approximated loss in Eq. (3), since the local model $\theta_k^{q,T}$ may not notably deviate from the initial $\theta^q[:m_k]$ between rounds.

In addition to $p_k(d)$, client k also uploads its class distribution $p_k(y) = \frac{n_k^y}{|D_k|}$, where n_k^y is the number of samples for y class in D_k.

4.3 Difficulty-Conditional Data Generator

This module trains generators that output *difficulty-conditioned* pseudo data for indeterminate models. Specifically, we train one generator G_m *independently* for each intermediate model θ_m to produce pseudo data $\tilde{x}$, which will be used for knowledge distillation in Sec. 4.4. The $\tilde{x}$ is referred to as follows:

$$\tilde{x} \sim G_m(y, d, \epsilon; \phi_m) \tag{4}$$

where ϕ_m is model parameters of G_m, ϵ is a Gaussian noise vector from $\mathcal{N}(0, I)$, and $\tilde{x}$ is the pseudo data based on given difficulty d and class y, sampled from $p_m(d)$ and $p_m(y)$, which are the difficulty distribution and class distribution of θ_m. The difficulty distribution $p_m(d)$ of θ_m is defined as $p_m(d) = \frac{\sum_{k \in S_m} p_k(d) \times |D_k|}{\sum_{k \in S_m} |D_k|}$, where $p_m(d)$ is the difficulty distribution of local dataset for m exits clients, denoted by S_m. The class distribution $p_m(y)$ of θ_m is calculated similarly based on $\{p_k(y)\}_{k \in S_m}$. Note that it is common to upload meta information like the distributions *i.e.*, $p_k(d)$ and $p_k(y)$ of local datasets in data-free KD based federated learning [50, 56, 59].

4.3.1 Generator Architecture. The generator is based on MLP, it takes one-hot label vector y, difficulty d and noise ϵ as the input, and after passing through a hidden layer with dimension d_h, it outputs outputs a feature representation with dimension d_r at last. We set the noise dimension d_ϵ as the classes of dataset, and $d_h = 1000$. Notably, Deit model is based on Transfermer [48] architecture, which has a embedding layer before the first block, since we set $d_r =$

197×192, where 197 is the number of tokens and 192 is the latent representation of Deit-Tiny. Benefits from that, the output can be directly input to the first block (encoder) to avoid the computation of the embedding layer and simulate raw data.

4.3.2 Generator Training. Our generator differs from those in previous data-free KD [59] in two unique constraints.

- **Difficulty Simulation.** It should simulate datasets conditioned on *difficulty*. The measured difficulty vector $\hat{d}$ of pseudo data $\tilde{x}$ should resemble d sampled from $p(d)$. It can be enforced by minimizing the difference $|d - \hat{d}|$.

$$\mathcal{L}_{dif}(\phi_m, \theta_m) = \mathbb{E}_{\tilde{x} \sim G_m(y,d,\epsilon;\phi_m)} |d - \hat{d}|$$

- **Classification.** It should represent knowledge extracted from *multi-exit* networks (*i.e.*, intermediate models in our case). In other words, pseudo data $\tilde{x}$ must be correctly classified by all exits of the targeted intermediate model.

$$\mathcal{L}_{ce}(\phi_m, \theta_m) = \mathbb{E}_{\tilde{x} \sim G_m(y,d,\epsilon;\phi_m)} \sum_{i=1}^{m} \mathrm{CE}(v(h^i(\tilde{x}; \theta_m)), y)$$

We simply weights of above loss functions as 1, and a generator with parameters ϕ_m simulates difficulty-conditioned knowledge from intermediate model θ_m by optimizing the following objective.

$$\min_{\phi_m} \mathcal{L}_G(\phi_m, \theta_m) = \mathcal{L}_{dif}(\phi_m, \theta_m) + \mathcal{L}_{ce}(\phi_m, \theta_m) \tag{5}$$

4.4 Progressive Reverse Knowledge Distillation

This section presents a novel reverse knowledge distillation scheme among intermediate models. As mentioned in Sec. 4.1, distillation proceeds from an intermediate model with m exits to one with $m+1$ exits, while knowledge is selectively transferred among exits based on their difficulty ranges.

Distillation between Adjacent Intermediate Models. We first explain how to transfer knowledge from a teacher model $\mathcal{T}$ with m exits, to a student model $\mathcal{S}$ with $m+1$ exits, in a *difficulty-aligned* and *effective* manner. We enable *difficulty-aligned* distillation via the following designs.

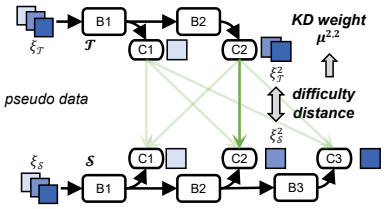


Figure 3: Adaptive weight based on difficulty distance.

- *Exit Pairing*. Due to different model depths between $\mathcal{T}$ and $\mathcal{S}$, there is no exact match in difficulty among their exits. Hence, we connect all the $m(m+1)$ exit pairs to maximize distillation flexibility and rely on dataset matching and KD weights for difficulty-aligned distillation.
- *Dataset Matching*. Since each exit of θ_m targets at a distinct difficulty range, we associate each exit to a unique dataset for distillation. Ideally, this corresponds to the local datasets that θ_m is trained on, which are inaccessible at the server. We approximate the *training* data for exit i of θ_m by performing *inference* on the pseudo data set $\xi = \{\tilde{x}, y, d\}$ generated for θ_m, since the local difficulty-aware training minimizes train-test mismatch [18, 20, 25]. Specifically, for exit i of θ_m, its dataset for distillation is subset $\xi^i_{\theta_m}$, which are samples in ξ that terminate their inference at exit i.
- *Exit-Wise KD Weight*. Shown as Fig. 3, after identifying the intended difficulty ranges for exit i of $\mathcal{T}$ and exit j of $\mathcal{S}$, *i.e.*, by their associated datasets $\xi^i_{\mathcal{T}}$ and $\xi^j_{\mathcal{S}}$, we assign an exit-wise weight for distillation from exit i of $\mathcal{T}$ to exit j of $\mathcal{S}$ proportional to the similarity between their targeted difficulty ranges as follows:

$$\mu^{i,j} = v\Big(\sum_{d_{\mathcal{T}} \in \xi^i_{\mathcal{T}}} \sum_{d_{\mathcal{S}} \in \xi^j_{\mathcal{S}}} |d_{\mathcal{T}} - d_{\mathcal{S}}|\Big) \tag{6}$$

where we sum over the differences in sample-wise difficulty and normalize it via softmax.

We further ensure *effective* reverse distillation by *(i)* transferring *feature and relation*, and *(ii)* imposing *model-wise* distillation weights. The final distillation objective is calculated as:

$$\mathcal{L}_{KD}(\theta_m, \theta_{m+1}; \xi_{\mathcal{T}}) = w_m \sum_{i=1}^{m} \sum_{j=1}^{m+1} \mu^{i,j} \psi(\mathcal{F}^i_{\mathcal{T}}, \mathcal{F}^j_{\mathcal{S}})$$
$$\mathcal{F}^i_{\mathcal{T}}, \mathcal{F}^j_{\mathcal{S}} = \{f^i(\tilde{x}; \theta_m) | \tilde{x} \in \xi^i_{\mathcal{T}}\}, \{f^j(\tilde{x}; \theta_{m+1}) | \tilde{x} \in \xi^i_{\mathcal{T}}\} \tag{7}$$

where $f^i(\tilde{x}; \theta_m)$ is the feature of $\tilde{x}$ which input into i-th classifier of intermediate model θ_m, $\mathcal{F}^i_{\mathcal{T}}$ ($\mathcal{F}^j_{\mathcal{S}}$) is the set of feature for $\xi^i_{\mathcal{T}}$ at exit i (j) of $\mathcal{T}$ ($\mathcal{S}$), and we optimize $m+1$ exits of $\mathcal{S}$ jointly on $\xi_{\mathcal{T}}$ by the relation-based knowledge distillation loss $\psi(\cdot)$ [35]. The model-wise weight $w_m = \frac{\sum_{k \in S_m} |D_k|}{\sum_{k \notin S_M} |D_k|}$ accounts for the differences in numbers of training samples across models.

Progressive Distillation. The above distillation from θ_m to θ_{m+1} starts from the shallowest and proceeds incrementally to the deepest. Such *sequential* distillation may cause error accumulation [49]. That is, if improper distillation occurs at a certain model and its parameters are updated immediately, the error will propagate to subsequent distillation among deeper models. To mitigate error accumulation, we calculate gradients g_{m+1} of the distillation loss for student intermediate model $\mathcal{S}$ without updating its parameters; and utilize the gradients to update all intermediate models together after distillation to the deepest model (lines 15-16 in Algorithm 1).

4.5 Parallel Variant for Acceleration

This section introduces DarkDistill-PL, a variant of DarkDistill that accelerates the server-side KD (see Fig. 4). Unlike DarkDistill, which *sequentially* transfers knowledge from one intermediate model to another, DarkDistill-PL *simultaneously* distills the ensemble knowledge of all immediate knowledge to the global model $\mathcal{S}$ parameterized by θ in an *exit-wise* manner. The workflow of DarkDistill-PL is presented in Algorithm 3.

Exit-Wise Parallel Distillation. We approximately align the difficulty of exits by matching their *locations* rather than their training datasets as Sec. 4.4 to simplify the design. Specifically, we organize the M intermediate models as M teachers $\{\mathcal{T}_m\}_{m=1}^{M}$ by exits. That is,

$\mathcal{T}_m$ corresponds to the m-th exits of intermediate models. Its ensemble knowledge corresponds to features accumulated over $\{\theta_m\}$ that have at least m exits, *i.e.*, $F^m(\tilde{x};\{\theta_m\}) = \sum_{i=m}^{M} \frac{n_i}{\sum_{i'=m}^{M} n_{i'}} f^m(\tilde{x};\theta_i)$, where $n_i = \sum_{k\in S_i} |D_k|$. The ensemble knowledge of $\mathcal{T}_m$ on pseudo datasets $\{\xi_m\}$ is distilled to exit m of the global model (denoted by $\mathcal{S}_m$) by optimizing the objective below:

$$\mathcal{L}_{KD}(\{\theta_m\},\theta;\{\xi_m\}) = \frac{1}{M}\sum_{m=1}^{M}\psi(\mathcal{F}_{\mathcal{T}_m},\mathcal{F}_{\mathcal{S}_m}),$$
$$\mathcal{F}_{\mathcal{T}_m},\mathcal{F}_{\mathcal{S}_m} = \{F^m(\tilde{x};\{\theta_m\})|\tilde{x}\in\xi_m\},\{f^m(\tilde{x};\theta)|\tilde{x}\in\xi_m\} \tag{8}$$

Difficulty-Increased Data Generators. The parallel distillation above needs a new generator design. As shown in Fig. 4, DarkDistill-PL assigns one generator for each $\mathcal{T}_m$, and they should generate pseudo data with increasing difficulty. That is, pseudo data produced by the generator for $\mathcal{T}_{m+1}$ is more difficult than that for $\mathcal{T}_m$. This is implemented by maximizing the difference in difficulty of generators for adjacent exits as follows:

$$\mathcal{L}_{dif}(\phi_{m+1},\phi_m) = \mathbb{E}_{\tilde{x}_{m+1}\sim G_{m+1}(y;\phi_{m+1})}\hat{d}_{m+1} - \mathbb{E}_{\tilde{x}_m\sim G_m(y;\phi_m)}\hat{d}_m \tag{9}$$

where $\tilde{x}_{m+1}$ is the pseudo data for $\mathcal{T}_{m+1}$, and $\hat{d}_{m+1}$ is the difficulty of $\tilde{x}_{m+1}$, measured as Eq. (3) based on its ensemble logits [56], which is larger than $\hat{d}_m$ to ensure increased difficulty. Classification loss of DarkDistill-PL is similar to the second term of Eq. (5) but also utilizes ensemble logits $H^m(\tilde{x})$, where $H^m(\tilde{x}) = \sum_{i=m}^{M} \frac{n_i}{\sum_{i'=m}^{M} n_{i'}} h^m(\tilde{x};\theta_i)$. The objective of ϕ_{m+1} in DarkDistill-PL is presented as follows:

$$\min_{\phi_{m+1}} \mathcal{L}_G(\phi_m,\phi_{m+1},\{\theta_m\}) = \mathcal{L}_{dif}(\phi_m,\phi_{m+1},\{\theta_m\}) + \mathcal{L}_{ce}(\phi_{m+1},\theta_{m+1}) \tag{10}$$

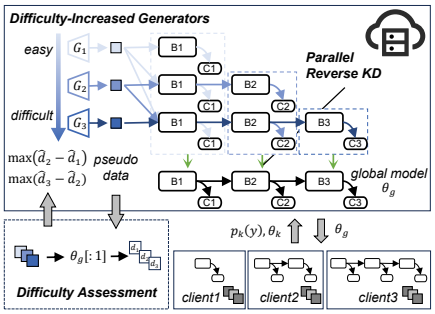


Figure 4: Variant of DarkDistill with parallel distillation.

4.6 Convergence Analysis

Under assumptions in Appendix A, our proposed federated EEN training algorithm converges in FL with heterogeneous clients (see Theorem 1). We extend the analysis in [58] from single-exit network to multi-exit network. All proofs are in Appendix A.

THEOREM 1. *If the learning rate γ of local training satisfies $\frac{1}{T\sqrt{Q}} \le \gamma < \frac{1}{6M^2LT}$, our solution coverages to a neighborhood of a stationary point of standard FL as follows:*

$$\frac{1}{Q}\sum_{q=1}^{Q}\mathbb{E}\|\nabla\mathcal{L}(\theta^q)\|^2 \le \frac{G_0}{\sqrt{Q}} + V_0 + \frac{H_0}{T} + \frac{I_0}{\sqrt{Q}}\sum_{q=1}^{Q}\mathbb{E}\|\theta^q\|^2 \tag{11}$$

Algorithm 3: ServerExecute-PL

Input: global model parameters θ^0, generators parameters $\{\phi_m^0\}$, learning rate γ,β

for *communication round* $q = 1,...,Q$ **do**
  $\theta^q, \{\theta_m^q\} \leftarrow 0$
  // Step1: Local Processing
  for *client* k *selected* **do**
    $\theta_k^{q+1}, p_k(d), p_k(y) \leftarrow$ **ClientExecute**(θ^q, D_k)
    $\theta_{m_k}^q \leftarrow \theta_{m_k}^q + \frac{|D_k|}{|D_{m_k}|}\theta_k^{q+1}$
  // Step2: Generator Training
  Attain $p_m(d), p_m(y)$ based on $\{p_k(d), p_k(y)\}_{k\in S_m}$
  for $m = 1,...,M$ **do**
    Conditions $y \sim p_m(y), d \sim p_m(d), \epsilon \sim \mathcal{N}(0,I)$
    Generate $\xi_m \sim G_m(y,d,\epsilon)$ based on y,d,ϵ
    $\phi_m^{q+1} \leftarrow \phi_m^q - \beta\nabla_{\theta_m^q}\mathcal{L}_G(\phi_m,\{\theta_m^q\})$ ▷ By Eq. (10)
  // Step3: Model Aggregation
  for $m = 1,...,M$ **do**
    $\theta^q[m] \leftarrow \theta^q[m] + \sum_{m_1=m}^{M} \frac{|D_{m_1}|}{\sum_{m_2=m}^{M}|D_{m_2}|}\theta_{m_1}^q[m]$
  Generate pseudo latent $\{\xi_m\}_{m\in M}$ with $\{G_m\}_{m\in M}$
  $\theta^{q+1} \leftarrow \theta^q - \gamma\nabla_{\theta^q}\mathcal{L}_{KD}(\{\theta_m^q\},\theta^q;\{\xi_m\})$ ▷ By Eq. (8)
  return θ^{q+1}

where $G_0 = 4\mathbb{E}[\mathcal{L}(\theta^0)]$, $V_0 = \frac{KG}{36\Gamma^*}$, $H_0 = \frac{MK\sigma^2}{(\Gamma^*)^2} + \frac{KG}{18\Gamma^*M}$ and $I_0 = \frac{L^2\delta^2K(2M+1)}{\Gamma^*\sqrt{Q}}$ are constants related to the initial model parameters θ^0, assumption bounds L,σ,δ,G, and federated configurations Γ^*, M, T, Q, K. Concretely, Γ^* is the occurrence for the parameter of the last block and its classifier in local models, M is the largest exit number of local models, T is the local training epoch, Q is the total communication round, K is the number of clients.

5 Experiments

5.1 Experimental Setup

Baselines. We compare DarkDistill and DarkDistill-PL against FL methods with heterogeneous clients that utilize EEN. For fair comparison, we extend them for difficulty-aware training by applying existing centralized difficulty-awareness schemes to their local training. We consider the following schemes for local training.

- **None** [44]: Jointly exit training without difficulty-awareness.
- **BoostNet** [54]: Difficulty-aware training via boosted gradient and accumulative logits.
- **L2W-DEN** [11]: Difficulty-aware training by meta learning of sample-adaptive exit weights. It requires computation-intensive meta learning on clients.

The above local training strategies are integrated with the following FL schemes for evaluation.

- **ExclusiveFL**: Naive baseline that only involves clients that can train the global model in FL.
- **InclusiveFL** [29]: Generic FL that transfers momentum knowledge from large models to small ones.

- **ScaleFL** [18]: EEN-based FL method that scales both the widths and depths of local models. It uses self-distillation within local models at clients, transferring knowledge from the last exit to shallower one.
- **DepthFL** [20]: EEN-based FL that scales model depths. It also uses self-distillation within local models at clients, but manually transfers knowledge between exits.
- **ReeFL** [25]: EEN-based FL that devises a shared classifier for all exits and selects the exit with the smallest loss to teach other exits at clients.
- **DarkDistill**: Our difficulty-aligned model aggregation strategy with progressive KD.
- **DarkDistill-PL**: Our difficulty-aligned model aggregation strategy with parallel KD.

Datasets & Model. We experiment with CIFAR-100 [21] and SVHN [33] for image classification, and SpeechCommandsV2 [53] for speech classification. We finetune a pre-trained, transformer-based model, *i.e.*, a variant of tiny DeiT [47], in both full-parameter (Full) and parameter-efficient (LORA [15]) setups. The model is configured with 4 exits (including the last one) by default. Details of how to add early exits are in Appendix B.1.

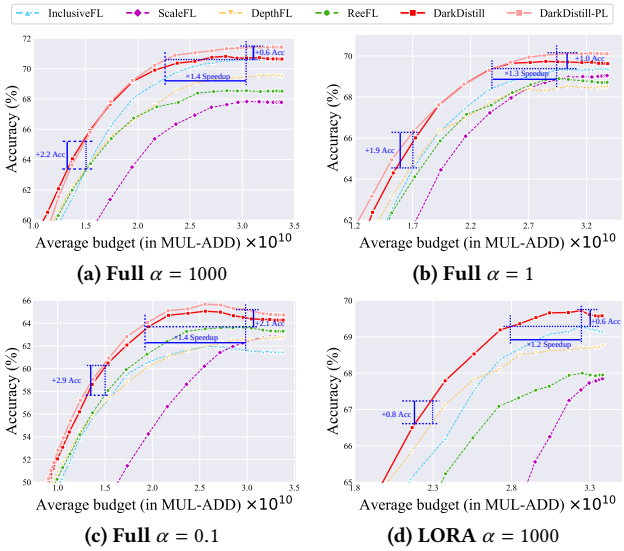
Client Heterogeneity. Following previous FL with heterogeneous clients studies [20, 25], we set 100 clients, divided into 4 levels with increasing compute capabilities, with 4 sizes of local model. We simulate three client heterogeneity settings by varying the distributions of high- and low-end clients (details in Appendix B.1.4). We also use Dirichlet distribution Dir(α) on label ratios to simulate the non-IID data among clients, where a smaller α represents higher data heterogeneity. The training hyperparameters for baselines and our works are detailed in Appendix B.1.

5.1.1 Metrics. We assess the accuracy of the global EEN in two settings. *(i) Anytime inference*: It measures the accuracy of each exit assuming sufficient budgets. *(ii) Budget inference*: It measures the accuracy of a batch samples within given budgets.

5.2 Main Results

Performance of Anytime Inference. Table. 1 summarizes the mean and standard deviation of test accuracy across all 4 exits of global EEN. Our solutions achieve the highest accuracy in both fine-tuning settings. Comparing the columns with and without difficulty-awareness (BoostNet) for local training, we observe increased accuracy in most FL baselines for both Full and LORA, except for ReeFL [25], which is caused by shared exit. This validates the necessity of difficulty-aware training in federated EEN training. Comparisons between local difficulty-aware training schemes *i.e.*, BoostNet vs. L2W-DEN are deferred to ablation studies (see Sec. 5.3). In full-parameter fine-tuning, DarkDistill and DarkDistill-PL with BoostNet are the top 2 on all datasets, which have about 2% increase. Meanwhile, the larger SD across exits accuracy presents DarkDistill can specialize exits on difficulty ranges. For LORA, DarkDistill is still the best. We exclude DarkDistill-PL for LORA since it aims to improve efficiency when training large numbers of parameters and thus is more suited for the full-parameter case. Additionally, our work exhibits a larger standard deviation across exits, which confirms that these exits are specialized to specific difficulty ranges.

Performance of Budgeted Inference. Fig. 5 plots the test accuracy given different computation budgets (measured in amounts of Mul-Adds), where the global EEN is finetuned with various EEN-based FL methods together with BoostNet for local difficulty-aware training on CIFAR-100 with various α. As is shown, our DarkDistill and DarkDistill-PL can improve the accuracy over the baselines at various computation budgets. For example, in Full with $\alpha = 1000$, DarkDistill-PL improves the accuracy by 2.2% when evoking 15M Mul-Adds, and achieves the same peak accuracy of the baselines yet is 1.4 times faster; in Full when $\alpha = 0.1$, DarkDistill-PL increases the accuracy by 2.9% and 2.1% at 13M and 25M Mul-Adds, respectively, and is 1.4 times faster than baselines to achieve the same accuracy of 63.7%. For LORA, DarkDistill also achieves higher accuracy and faster inference than the baselines.



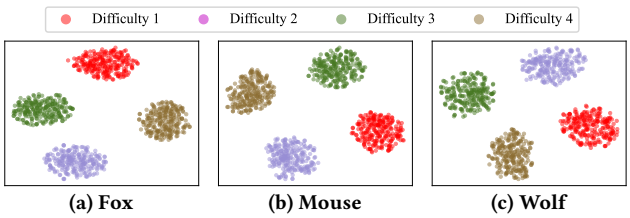
(a) **Full** $\alpha = 1000$ (b) **Full** $\alpha = 1$

(c) **Full** $\alpha = 0.1$ (d) **LORA** $\alpha = 1000$

Figure 5: Budgeted inference: performance on CIFAR-100 with varied α in Full and LORA setups.

5.3 Ablation Study

Effectiveness of Generator. Fig. 6 visualizes the feature space of pseudo data produced by the generators of DarkDistill for three classes. As is shown, the pseudo data is divided into 4 clusters with various difficulty levels $\{1, 2, 3, 4\}$. It implies that the generators can produce pseudo data at specified difficulty for a given class.



(a) Fox (b) Mouse (c) Wolf

Figure 6: Visualization of pseudo data generated by DarkDistill with difficulty-increased samples.

Table 1: Anytime inference performance to finetune a 4-exit DeiT-tiny model (best accuracy in bold; second best underlined).

Finetune	Difficulty-aware	Approach	CIFAR-100 [21]			SVHN [33]	SpeechCmds [53]
			$\alpha = 0.1$	$\alpha = 1$	$\alpha = 1000$		
Full	None	ExclusiveFL	$26.60_{\pm 3.10}$	$49.96_{\pm 11.48}$	$41.58_{\pm 7.01}$	$85.28_{\pm 2.97}$	$87.00_{\pm 2.88}$
		InclusiveFL [29]	$40.10_{\pm 2.03}$	$58.83_{\pm 6.98}$	$61.40_{\pm 7.01}$	$82.95_{\pm 0.34}$	$91.90_{\pm 1.42}$
		ScaleFL [18]	$54.99_{\pm 10.61}$	$63.21_{\pm 9.14}$	$63.82_{\pm 9.87}$	$88.24_{\pm 0.78}$	$92.56_{\pm 0.26}$
		DepthFL [20]	$40.70_{\pm 1.57}$	$59.01_{\pm 5.18}$	$61.71_{\pm 5.73}$	$83.45_{\pm 0.43}$	$92.05_{\pm 0.60}$
		ReeFL [25]	$59.24_{\pm 8.00}$	$63.37_{\pm 7.72}$	$63.90_{\pm 8.68}$	$88.37_{\pm 1.27}$	$93.12_{\pm 1.14}$
	BoostNet [54]	ExclusiveFL	$48.68_{\pm 13.66}$	$57.57_{\pm 15.12}$	$58.65_{\pm 15.31}$	$87.30_{\pm 2.89}$	$91.07_{\pm 2.58}$
		InclusiveFL [29]	$57.10_{\pm 7.21}$	$62.96_{\pm 8.12}$	$64.01_{\pm 8.24}$	$87.86_{\pm 1.66}$	$92.91_{\pm 1.10}$
		ScaleFL [18]	$52.74_{\pm 13.82}$	$60.55_{\pm 11.93}$	$60.73_{\pm 10.80}$	$87.91_{\pm 0.77}$	$92.03_{\pm 0.37}$
		DepthFL [20]	$58.15_{\pm 6.73}$	$63.81_{\pm 6.34}$	$64.19_{\pm 6.73}$	$87.74_{\pm 1.01}$	$92.72_{\pm 0.64}$
		ReeFL [25]	$59.01_{\pm 7.98}$	$63.08_{\pm 9.03}$	$63.66_{\pm 7.31}$	$88.39_{\pm 1.28}$	$93.01_{\pm 1.18}$
		DarkDistill	$\underline{60.48}_{\pm 7.93}$	$\underline{64.50}_{\pm 7.97}$	$\mathbf{65.67}_{\pm 7.48}$	$\underline{88.41}_{\pm 1.46}$	$\underline{93.31}_{\pm 1.13}$
		DarkDistill-PL	$\mathbf{61.05}_{\pm 8.19}$	$\mathbf{65.12}_{\pm 7.02}$	$\underline{65.49}_{\pm 7.88}$	$\mathbf{88.48}_{\pm 1.57}$	$\mathbf{93.42}_{\pm 0.98}$
LORA [15]	None	ExclusiveFL	$44.44_{\pm 18.61}$	$52.33_{\pm 18.56}$	$52.88_{\pm 18.17}$	$83.78_{\pm 4.43}$	$88.72_{\pm 3.15}$
		InclusiveFL [29]	$44.82_{\pm 23.36}$	$54.26_{\pm 21.38}$	$54.76_{\pm 21.37}$	$85.16_{\pm 5.31}$	$89.58_{\pm 3.04}$
		ScaleFL [18]	$22.17_{\pm 23.36}$	$30.85_{\pm 30.58}$	$32.58_{\pm 31.96}$	$76.42_{\pm 13.14}$	$58.82_{\pm 34.80}$
		DepthFL [20]	$52.17_{\pm 14.16}$	$57.09_{\pm 14.78}$	$57.63_{\pm 14.48}$	$85.69_{\pm 2.71}$	$90.11_{\pm 2.04}$
		ReeFL [25]	$52.32_{\pm 9.83}$	$57.74_{\pm 11.78}$	$58.16_{\pm 11.69}$	$85.54_{\pm 3.00}$	$89.56_{\pm 2.62}$
	BoostNet [54]	ExclusiveFL	$50.34_{\pm 13.63}$	$55.68_{\pm 15.33}$	$56.48_{\pm 15.33}$	$84.48_{\pm 3.88}$	$88.51_{\pm 2.26}$
		InclusiveFL [29]	$54.25_{\pm 11.78}$	$59.66_{\pm 11.72}$	$59.81_{\pm 11.66}$	$85.96_{\pm 2.50}$	$90.38_{\pm 2.10}$
		ScaleFL [18]	$40.46_{\pm 22.36}$	$47.18_{\pm 24.11}$	$48.26_{\pm 24.17}$	$81.70_{\pm 3.14}$	$80.19_{\pm 4.83}$
		DepthFL [20]	$55.85_{\pm 9.45}$	$60.95_{\pm 9.20}$	$61.45_{\pm 9.04}$	$79.90_{\pm 1.61}$	$90.93_{\pm 1.29}$
		ReeFL [25]	$51.57_{\pm 9.82}$	$58.04_{\pm 11.88}$	$58.62_{\pm 11.91}$	$85.44_{\pm 2.92}$	$89.40_{\pm 2.44}$
		DarkDistill	$\mathbf{57.32}_{\pm 11.91}$	$\mathbf{61.24}_{\pm 11.06}$	$\mathbf{61.74}_{\pm 11.42}$	$\mathbf{86.11}_{\pm 2.16}$	$\mathbf{91.06}_{\pm 2.08}$

Robustness of Generator. The experiments in Table. 2 prove that DarkDistill, DarkDistill-PL are *robust* across different generator architectures. For full parameters fine-tuning, our work always outperforms SOTA on CIFAR-100 with varied combinations of noise dimension d_ϵ and hidden dimension d_h.

Table 2: Effects of the Generator Network Structure on CIFAR-100 (DarkDistill is left; DarkDistill-PL is right).

d_ϵ	d_h			
	64	128	256	512
2	64.79|64.88	65.05|64.93	65.32|65.25	65.10|65.11
16	65.38|64.91	65.20|65.09	65.37|64.65	65.18|65.51
32	65.05|64.79	65.06|64.79	65.28|65.08	65.60|64.90
64	65.15|64.92	65.74|64.93	64.91|64.55	65.06|65.05
SOTA	$64.19_{\pm 6.73}$			

Impact of Local Difficulty-Aware Training. Our main results adopt BoostNet for local difficulty-aware training. This experiment tests L2W-DEN, another local difficulty-aware training scheme. Table. 3 shows the anytime inference accuracy on CIFAR-100 for LORA using the two methods for local training. We choose the LORA setting because L2W-DEN involves on-device meta-learning, which imposes drastic computation burden for full-parameter training at clients. Aligned with previous studies [5, 11], L2W-DEN improves the accuracy over BoostNet (yet with larger computation overhead). Our DarkDistill still outperforms the baselines, which implies it functions with other difficulty-aware training strategies.

Table 3: Impact of local difficulty-aware strategy for LORA.

Approch	$\alpha = 1$		$\alpha = 1000$	
	BoostNet	L2W-GEN	BoostNet	L2W-GEN
InclusiveFL [29]	$59.66_{\pm 11.72}$	$59.99_{\pm 10.34}$	$59.81_{\pm 11.66}$	$60.82_{\pm 10.48}$
ScaleFL [18]	$47.18_{\pm 24.11}$	$52.67_{\pm 18.52}$	$48.26_{\pm 24.17}$	$53.61_{\pm 18.31}$
DepthFL [20]	$60.95_{\pm 9.20}$	$61.92_{\pm 8.15}$	$61.45_{\pm 9.04}$	$62.42_{\pm 8.65}$
ReeFL [25]	$58.04_{\pm 11.88}$	$57.87_{\pm 12.17}$	$58.62_{\pm 11.91}$	$59.09_{\pm 12.87}$
DarkDistill	$\underline{61.42}_{\pm 11.06}$	$\mathbf{62.08}_{\pm 11.26}$	$\underline{61.74}_{\pm 11.42}$	$\mathbf{62.90}_{\pm 10.31}$

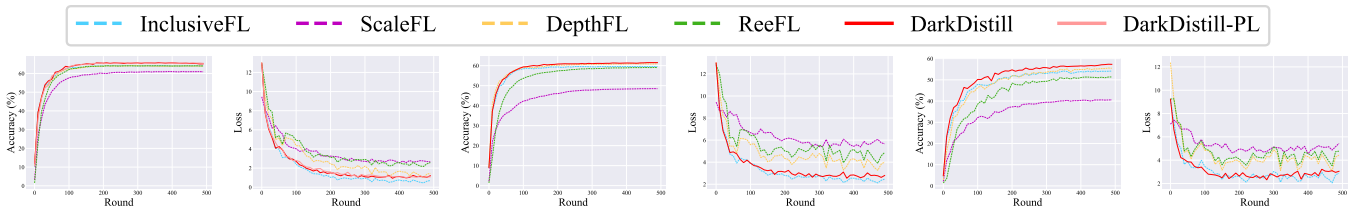
Contributions of Individual Modules. Table. 4 decomposes the contributions of knowledge distillation (KD) and difficulty condition of data generator (G) in DarkDistill, measured by their anytime inference accuracy on CIFAR-100. For G, ✗ means traditional generator without difficulty condition. Our KD scheme improves 0.5%, and the generator with difficulty condition further boosts the accuracy by 0.5%. This proves that the two are mutually reinforcing.

Table 4: Contributions of KD and Generator of DarkDistill.

Finetune	KD	G	Exit-1	Exit-2	Exit-3	Exit-4	Avg
Full	✗	✗	53.67	64.89	70.54	70.30	64.85
	✓	✗	53.70	65.49	70.80	69.94	64.98
	✓	✓	**55.10**	**65.67**	**71.26**	**70.64**	**65.67**
LORA [15]	✗	✗	44.84	62.19	68.33	68.50	60.97
	✓	✗	44.87	62.73	69.15	69.30	61.51
	✓	✓	**45.28**	**62.71**	**69.43**	**69.52**	**61.74**

Table 5: Table. 1's Extension: SST-2 on 4-exit Bert model (best accuracy in bold; second best underlined).

Difficulty-aware	ExclusiveFL	InclusiveFL	ScaleFL	DepthFL	ReeFL	DarkDistill	DarkDistill-PL
None	$78.07_{\pm 0.20}$	$80.13_{\pm 0.33}$	$80.36_{\pm 0.39}$	$80.02_{\pm 0.11}$	$81.79_{\pm 0.14}$	/	/
BoostNet	$79.07_{\pm 0.27}$	$81.59_{\pm 0.14}$	$81.82_{\pm 0.15}$	$81.33_{\pm 0.14}$	$81.76_{\pm 0.30}$	$\underline{82.11}_{\pm 0.42}$	$\mathbf{83.02}_{\pm 0.39}$



(a) Full Acc $\alpha = 1000$ **(b) Full Loss** $\alpha = 1000$ **(c) LORA Acc** $\alpha = 1000$ **(d) LORA Loss** $\alpha = 1000$ **(e) LORA Acc** $\alpha = 0.1$ **(f) LORA Loss** $\alpha = 0.1$

Figure 7: Federated training convergence and performance of CIFAR-100 with Full and LORA.

Table 6: Impact of client heterogeneity on accuracy, tested on CIFAR-100 (best accuracy in bold; second best underlined).

Finetune	Major Devices	Low-end [0.4,0.3,0.2,0.1]			Normal [0.25,0.25,0.25,0.25]			High-end [0.1,0.2,0.3,0.4]		
	Approach	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 1000$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 1000$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 1000$
Full	InclusiveFL [29]	$48.69_{\pm 3.22}$	$57.93_{\pm 6.36}$	$60.04_{\pm 6.42}$	$57.10_{\pm 7.21}$	$62.96_{\pm 8.12}$	$64.01_{\pm 8.24}$	$60.00_{\pm 9.61}$	$65.84_{\pm 8.77}$	$66.43_{\pm 9.55}$
	ScaleFL [18]	$40.54_{\pm 11.12}$	$54.18_{\pm 8.76}$	$56.53_{\pm 8.39}$	$52.74_{\pm 13.82}$	$60.55_{\pm 11.93}$	$60.73_{\pm 10.80}$	$58.73_{\pm 14.73}$	$63.42_{\pm 13.34}$	$63.45_{\pm 13.01}$
	DepthFL [20]	$53.36_{\pm 4.64}$	$61.54_{\pm 4.88}$	$61.96_{\pm 4.63}$	$58.15_{\pm 6.73}$	$63.81_{\pm 6.34}$	$64.19_{\pm 6.73}$	$61.43_{\pm 8.42}$	$65.40_{\pm 7.61}$	$65.71_{\pm 7.09}$
	ReeFL [25]	$53.79_{\pm 5.95}$	$60.00_{\pm 5.59}$	$61.02_{\pm 7.16}$	$59.01_{\pm 7.98}$	$63.08_{\pm 9.03}$	$63.66_{\pm 7.31}$	$61.77_{\pm 8.61}$	$65.15_{\pm 9.13}$	$65.87_{\pm 9.63}$
	DarkDistill	$\underline{55.89}_{\pm 5.63}$	$\underline{61.81}_{\pm 5.66}$	$\underline{62.32}_{\pm 5.48}$	$\underline{60.48}_{\pm 7.93}$	$\underline{64.50}_{\pm 7.97}$	$\mathbf{65.67}_{\pm 7.48}$	$\underline{62.88}_{\pm 10.41}$	$\underline{66.01}_{\pm 8.85}$	$\underline{67.02}_{\pm 9.16}$
	DarkDistill-PL	$\mathbf{56.42}_{\pm 4.88}$	$\mathbf{62.11}_{\pm 5.19}$	$\mathbf{62.98}_{\pm 6.58}$	$\mathbf{61.05}_{\pm 8.19}$	$\mathbf{65.12}_{\pm 7.02}$	$\underline{65.49}_{\pm 7.88}$	$\mathbf{63.86}_{\pm 10.45}$	$\mathbf{66.54}_{\pm 8.31}$	$\mathbf{67.44}_{\pm 9.98}$
LORA [15]	InclusiveFL [29]	$46.98_{\pm 6.48}$	$55.41_{\pm 9.57}$	$56.90_{\pm 9.80}$	$54.25_{\pm 11.78}$	$59.66_{\pm 11.72}$	$59.81_{\pm 11.66}$	$58.59_{\pm 14.45}$	$62.00_{\pm 13.49}$	$62.20_{\pm 13.01}$
	ScaleFL [18]	$31.78_{\pm 31.78}$	$43.71_{\pm 21.42}$	$46.15_{\pm 21.22}$	$40.46_{\pm 22.36}$	$47.18_{\pm 24.11}$	$48.26_{\pm 24.17}$	$45.44_{\pm 25.09}$	$49.01_{\pm 26.83}$	$49.12_{\pm 26.73}$
	DepthFL [20]	$50.71_{\pm 7.51}$	$58.55_{\pm 6.37}$	$59.31_{\pm 8.81}$	$55.85_{\pm 9.45}$	$60.95_{\pm 9.20}$	$61.45_{\pm 9.04}$	$58.58_{\pm 10.67}$	$62.51_{\pm 9.96}$	$62.72_{\pm 10.07}$
	ReeFL [25]	$45.92_{\pm 7.12}$	$54.39_{\pm 10.03}$	$55.58_{\pm 10.54}$	$51.57_{\pm 9.82}$	$58.04_{\pm 11.88}$	$58.62_{\pm 11.91}$	$54.00_{\pm 10.76}$	$60.06_{\pm 12.84}$	$60.73_{\pm 12.90}$
	DarkDistill	$\mathbf{53.27}_{\pm 8.48}$	$\mathbf{58.88}_{\pm 5.66}$	$\mathbf{59.33}_{\pm 8.81}$	$\mathbf{57.32}_{\pm 11.91}$	$\mathbf{61.24}_{\pm 11.06}$	$\mathbf{61.74}_{\pm 11.42}$	$\mathbf{60.02}_{\pm 14.34}$	$\mathbf{62.77}_{\pm 13.24}$	$\mathbf{62.88}_{\pm 13.16}$

Different pretrained model. We conduct extension experiments on SST-2 [39] for GLUE benchmark with Bert [6] model, which has 12 layers and 128 hidden dimension. We show the anytime inference with 4 exits for full parameter fine-tuning in Table. 5. These experiments show that our work is also *robust* across pertrained models with different architectures.

Intermediate Results of Training. As shown in Fig. 7, all approaches reach the convergence on CIFAR-100 with varied α and fine-tuning setups. The accuracy of DarkDistill outperforms all baselines, with almost the lowest training loss on the valid dataset.

Impact of Client Heterogeneity. Table. 6 lists the anytime inference accuracy on CIFAR-100 under three client heterogeneity distributions. For example, "high-end" means high-end devices dominate, where 10%, 20%, 30%, and 40% clients can train local models with 1, 2, 3, and 4 exits, respectively. Our DarkDistill and DarkDistill-PL always outperform the baselines with the three distributions.

6 Conclusion

This paper introduces DarkDistill, a novel heterogeneous federated learning scheme dedicated for early-exit networks (EENs) and its parallel variant DarkDistill-PL for acceleration. We identify the cross-model exit unalignment problem, an unexplored challenge when extending difficulty-aware EEN training to federated contexts. We develop a difficulty-conditional generator training strategy and a difficulty-aligned reverse distillation scheme to aggregate EENs of varying depths into a global model that retains its difficulty-specific specialization. Extensive experiments on image and speech classification benchmarks show that DarkDistill outperforms existing heteronomous federated learning solutions in both full-parameter and parameter-efficient fine-tuning settings. We envision DarkDistill as a critical step for training dynamic deep neural network on edge devices with heterogeneous resources.

Acknowledgements

We are grateful to anonymous reviewers for their constructive comments. This work was partially supported by National Key Research and Development Program of China under Grant No. 2023YFF0725103, National Science Foundation of China (NSFC) (Grant Nos. 62425202, U21A20516, 62336003), the CityU APRC grant (No. 9610633), the Beijing Natural Science Foundation (Z230001), the Fundamental Research Funds for the Central Universities No. JK2024-03, the Didi Collaborative Research Program and the State Key Laboratory of Complex & Critical Software Environment (SKLCCSE). Yi Xu and Yongxin Tong are the corresponding authors.

References

[1] Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. 2022. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *NeurIPS* 35 (2022).

[2] Romil Bhardwaj, Gopi Krishna Tummala, Ganesan Ramalingam, Ramachandran Ramjee, and Prasun Sinha. 2018. Autocalib: Automatic traffic camera calibration at scale. *ACM Transactions on Sensor Networks* 14, 3-4 (2018), 1–27.

[3] Caleb Chen Cao, Yongxin Tong, Lei Chen, and HV Jagadish. 2013. Wisemarket: a new paradigm for managing wisdom of online social users. In *KDD*.

[4] Yun-Hin Chan, Rui Zhou, Running Zhao, Zhihan JIANG, and Edith CH Ngai. 2023. Internal Cross-layer Gradients for Extending Homogeneity to Heterogeneity in Federated Learning. In *ICLR*.

[5] Joud Chataoui, Mark Coates, et al. 2023. Jointly-Learned Exit and Inference for a Dynamic Neural Network. In *ICLR*.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Enmao Diao, Jie Ding, and Vahid Tarokh. 2020. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In *ICLR*.

[8] Biyi Fang, Xiao Zeng, Faen Zhang, Hui Xu, and Mi Zhang. 2020. Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *SEC*.

[9] Dawei Gao, Yongxin Tong, Jieying She, Tianshu Song, Lei Chen, and Ke Xu. 2016. Top-k Team Recommendation in Spatial Crowdsourcing. In *WAIM*. 191–204.

[10] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2021. Dynamic neural networks: A survey. *TPAMI* 44, 11 (2021), 7436–7456.

[11] Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. 2022. Learning to Weight Samples for Dynamic Early-Exiting Networks. In *ECCV*. 362–378.

[12] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[14] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. 2021. Efficient Split-Mix Federated Learning for On-Demand and In-Situ Customization. In *ICLR*.

[15] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

[16] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *ICLR*.

[17] Jiaming Huang, Yi Gao, and Wei Dong. 2024. Unlocking the Non-deterministic Computing Power with Memory-Elastic Multi-Exit Neural Networks. In *WWW*.

[18] Fatih Ilhan, Gong Su, and Ling Liu. 2023. Scalefl: Resource-adaptive federated learning with heterogeneous clients. In *CVPR*. 24532–24541.

[19] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. 2014. Easy samples first: Self-paced reranking for zero-example multimedia search. In *MM*. 547–556.

[20] Minjae Kim, Sangyoon Yu, Suhyun Kim, and Soo-Mook Moon. 2022. DepthFL: Depthwise federated learning for heterogeneous clients. In *ICLR*.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[22] M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *NeurIPS* 23 (2010).

[23] Assaf Lahiany and Yehudit Aperstein. 2022. PTEENet: Post-Trained Early-Exit Neural Networks Augmentation for Inference Cost Optimization. *IEEE Access* 10 (2022), 69680–69687.

[24] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D Lane. 2021. Adaptive inference through early-exit networks: Design, challenges and directions. In *EMDL*.

[25] Royson Lee, Javier Fernandez-Marques, Shell Xu Hu, Da Li, Stefanos Laskaridis, Łukasz Dudziak, Timothy Hospedales, Ferenc Huszár, and Nicholas Donald Lane. 2024. Recurrent Early Exits for Federated Learning with Heterogeneous Clients. In *ICML*.

[26] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. 2019. Improved techniques for training adaptive deep networks. In *ICCV*. 1891–1900.

[27] Xiangjie Li, Chenfei Lou, Yuchi Chen, Zhengping Zhu, Yingtao Shen, Yehan Ma, and An Zou. 2023. Predictive exit: Prediction of fine-grained early exits for computation-and energy-efficient inference. In *AAAI*, Vol. 37. 8657–8665.

[28] Boyi Liu, Yiming Ma, Zimu Zhou, Yexuan Shi, Shuyuan Li, and Yongxin Tong. 2024. CASA: Clustered Federated Learning with Asynchronous Clients. In *KDD*.

[29] Ruixuan Liu, Fangzhao Wu, Chuhan Wu, Yanlin Wang, Lingjuan Lyu, Hong Chen, and Xing Xie. 2022. No one left behind: Inclusive federated learning over heterogeneous devices. In *KDD*.

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*. 1273–1282.

[31] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *AAAI*, Vol. 34. 5191–5198.

[32] Sahar Almahfouz Nasser, Nihar Gupte, and Amit Sethi. 2024. Reverse knowledge distillation: Training a large model using a small one for retinal image matching on limited data. In *WACV*.

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. 2011. Reading digits in natural images with unsupervised feature learning. *NeurIPS* 2011 (2011).

[34] Xiaomin Ouyang, Xian Shuai, Jiayu Zhou, Ivy Wang Shi, Zhiyuan Xie, Guoliang Xing, and Jianwei Huang. 2022. Cosmo: contrastive fusion learning with small data for multimodal human activity recognition. In *MobiCom*. 324–337.

[35] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational knowledge distillation. In *CVPR*. 3967–3976.

[36] Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020. The Right Tool for the Job: Matching Model and Instance Complexities. In *ACL*. 6640–6651.

[37] Jieying She, Yongxin Tong, Lei Chen, and Tianshu Song. 2017. Feedback-Aware Social Event-Participant Arrangement. In *SIGMOD*. 851.

[38] Shuyao Shi, Neiwen Ling, Zhehao Jiang, Xuan Huang, Yuze He, Xiaoguang Zhao, Bufang Yang, Chen Bian, Jingfei Xia, Zhenyu Yan, Raymond W. Yeung, and Guoliang Xing. 2024. Soar: Design and Deployment of A Smart Roadside Infrastructure System for Autonomous Driving. In *MobiCom*. 139–154.

[39] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *ACL*. 1631–1642.

[40] Max Sponner, Bernd Waschneck, and Akash Kumar. 2024. Adapting Neural Networks at Runtime: Current Trends in At-Runtime Optimizations for Deep Learning. *Comput. Surveys* 56, 10 (2024), 1–40.

[41] Shangchao Su, Bin Li, and Xiangyang Xue. 2025. Fedra: A random allocation strategy for federated tuning to unleash the power of heterogeneous clients. In *ECCV*. 342–358.

[42] Yi Sun, Jian Li, and Xin Xu. 2022. Meta-GF: Training Dynamic-Depth Neural Networks Harmoniously. In *ECCV*. 691–708.

[43] Qian Tao, Yuxiang Zeng, Zimu Zhou, Yongxin Tong, Lei Chen, and Ke Xu. 2018. Multi-worker-aware task planning in real-time spatial crowdsourcing. In *Database Systems for Advanced Applications: 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, May 21-24, 2018, Proceedings, Part II 23*. 301–317.

[44] Surat Teerapittayanon and Bradley McDanel. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*. 2464–2469.

[45] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. 2016. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*. 49–60.

[46] Yongxin Tong, Dingyuan Shi, Yi Xu, Weifeng Lv, Zhiwei Qin, and Xiaocheng Tang. 2021. Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale. *TKDE* 35, 10 (2021), 9812–9823.

[47] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*. 10347–10357.

[48] A Vaswani. 2017. Attention is all you need. *NeurIPS* (2017), 5998–6008.

[49] Hui-Po Wang, Sebastian Stich, Yang He, and Mario Fritz. 2022. ProgFed: Effective, communication, and computation efficient federated learning by progressive training. In *ICML*. 23034–23054.

[50] Shuai Wang, Yexuan Fu, Xiang Li, Yunshi Lan, Ming Gao, et al. 2024. DFRD: Data-Free Robustness Distillation for Heterogeneous Federated Learning. *NeurIPS* 36 (2024).

[51] Yansheng Wang, Yongxin Tong, Dingyuan Shi, and Ke Xu. 2021. An efficient approach for cross-silo federated learning to rank. In *ICDE*. 1128–1139.

[52] Yansheng Wang, Yongxin Tong, Zimu Zhou, Ziyao Ren, Yi Xu, Guobin Wu, and Weifeng Lv. 2022. Fed-LTD: Towards cross-platform ride hailing via federated learning to dispatch. In *KDD*.

[53] Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).

[54] Haichao Yu, Haoxiang Li, Gang Hua, Gao Huang, and Humphrey Shi. 2023. Boosted dynamic neural networks. In *AAAI*, Vol. 37. 10989–10997.

[55] Chen Jason Zhang, Yongxin Tong, and Lei Chen. 2014. Where to: Crowd-aided path selection. *Proceedings of the VLDB Endowment* 7, 14 (2014), 2005–2016.

[56] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. 2022. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *CVPR*. 10174–10183.

[57] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*. 3713–3722.

[58] Hanhan Zhou, Tian Lan, Guru Prasadh Venkataramani, and Wenbo Ding. 2024. Every parameter matters: Ensuring the convergence of federated learning with dynamic heterogeneous models reduction. *NeurIPS* 36 (2024).

[59] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. 2021. Data-free knowledge distillation for heterogeneous federated learning. In *ICML*. 12878–12889.

A Proof of Theorems

We extend the analysis in [58] from single-exit network to multi-exit network. Specifically, we first introduce Assumption 1, which is unique in EEN training. We then derive several unique lemmas for EEN training, which extend the three lemmas in [58] for federated EEN training (*i.e.*, Lemma 5, Lemma 6, Lemma 7). Finally, we prove Theorem 1 based on these lemmas.

Assumption 1 (EEN Training). *For an EEN with M exits, it's training loss function consists of M exits, we have:*

$$\mathcal{L} = \sum_{m=1}^{M} \mathcal{L}^m$$

Assumption 2 (Smoothness). *Loss functions $\{\mathcal{L}_k^m\}_{k\in K, m\in M}$ are all L-smooth, where $\mathcal{L}_k^m$ is the loss at exit m for client k. For any θ, θ' and any k, m, we assume that there exists $L > 0$ s.t.:*

$$\left\|\nabla\mathcal{L}_k^m(\theta) - \nabla\mathcal{L}_k^m(\theta')\right\| \le L\left\|\theta - \theta'\right\|$$

Assumption 3 (Server KD Noise). *For $\delta^2 \in [0,1)$ and any k, the global model knowledge distillation and reduction noise in each communication round is bounded as follows:*

$$\|\theta - \theta_{KD}[:m_k]\| \le \delta^2\|\theta\|$$

where $\theta_{KD} = \theta - \gamma\nabla\mathcal{L}_{KD}(\theta)$ represents the parameters optimized by server KD, and $\theta_{KD}[:m_k]$ is the model of client k after reduction.

Assumption 4 (Bound Gradient). *The expected squared norm of stochastic gradients is bounded uniformly. That is, in local epoch t, for constant $G > 0$ and any k, we have:*

$$\mathbb{E}_{\xi_k^t}\left\|\nabla\mathcal{L}_k^m(\theta_k^t;\xi_k^t)\right\|^2 \le G$$

Assumption 5 (Gradient Noise). *In local dataset, for any k, t, we assume a bounded gradient estimate:*

$$\mathbb{E}_{\xi_k^t}\left\|\nabla\mathcal{L}_k^m(\theta_k^t;\xi_k^t) - \nabla\mathcal{L}^m(\theta_k^t)\right\|^2 \le \sigma^2$$

Lemma 1. *The gradient of each region's parameters, calculated by the EEN training loss, can be represented by the gradient of each exit. For any k, i, we have:*

$$\nabla\mathcal{L}_k^{(i)} = \sum_{m=i}^{M}\nabla\mathcal{L}_k^{(i),m}$$

where M exits ENN has M regions. For region i, it consists of block i and classifier i, and $\nabla\mathcal{L}_k^{(i)}$ is the gradient of region i.

Proof. We first expand $\mathcal{L}_k$ to $\sum_{m=1}^{M}\mathcal{L}_k^M$ based on Assumption 1 and then utilize $\nabla(f+g) = \nabla f + \nabla g$:

$$\nabla\mathcal{L}_k = \nabla\left(\sum_{m=1}^{M}\mathcal{L}_k^m\right) = \sum_{m=1}^{M}\nabla\mathcal{L}_k^m \tag{12}$$

We attain $\sum_{j=1}^{M}\nabla\mathcal{L}_k^{(i),j}$ based on Eq. (12) at first, and then we remove losses of exits which can't train parameters of block i:

$$\nabla\mathcal{L}_k^{(i)} = \sum_{m=1}^{M}\nabla\mathcal{L}_k^{(i),m} = \sum_{m=i}^{M}\nabla\mathcal{L}_k^{(i),m} \tag{13}$$

□

Lemma 2. *Under Lemma 1 and Assumption 4, $\|\nabla\mathcal{L}_k(\theta_k;\xi)\|^2$ is the squared norm for gradient of region i's parameters in client k for training samples ξ, which can be bounded as follows:*

$$\|\nabla\mathcal{L}_k(\theta_k;\xi)\|^2 \le M^2G$$

Proof. This lemma quantifies the expected squared norm of the stochastic gradient for the joint training strategy of local EENs.

$$\text{LEFT} = \left\|\sum_{j=1}^{M}\nabla\mathcal{L}_k^j(\theta_k;\xi)\right\|^2 \le M\sum_{j=1}^{M}\left\|\nabla\mathcal{L}_k^j(\theta_k;\xi)\right\|^2 \le M^2G \tag{14}$$

In the first step, the joint training loss of the EEN is converted based on Eq. (12); the second step employs the inequality $|\sum_{i=1}^{n}a_i|^2 \le n\sum_{i=1}^{n}|a_i|^2$; and the final step is derived from the gradient noise inequality, as specified in Assumption 5. □

Lemma 3. *Under Lemma 1 and Assumption 2, we have:*

$$\sum_{i=1}^{M}\mathbb{E}\left\|\nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1}) - \nabla\mathcal{L}_k^{(i)}(\theta^q)\right\|^2 \le M^2L^2\mathbb{E}\left\|\theta_k^{q,t-1} - \theta^q\right\|^2$$

where in communication round q, θ^q denotes the global model parameters, $\theta_k^{q,t-1}$ is the local model parameters of client k in local epoch $t-1$, and $\theta_k^{q,0} = \theta_{KD}^q[:m_k]$ is the initial parameters of client k.

Proof. This lemma quantifies the difference between a local gradient and a stochastic gradient by summing the gradients of all parameters as follows.

$$\begin{aligned}\text{LEFT} &= \sum_{i=1}^{M}\mathbb{E}\left\|\sum_{j=i}^{M}\nabla\mathcal{L}_k^{(i),j}(\theta_k^{q,t-1}) - \nabla\mathcal{L}_k^{(i),j}(\theta^q)\right\|^2 \\ &\le M\sum_{j=1}^{M}\mathbb{E}\left\|\nabla\mathcal{L}_k^j(\theta_k^{q,t-1}) - \nabla\mathcal{L}_k^j(\theta^q)\right\|^2 \le M^2L^2\mathbb{E}\left\|\theta_k^{q,t-1} - \theta^q\right\|^2\end{aligned} \tag{15}$$

In the first step, we utilize Lemma 1 to simplify the gradients of region i, utilize the inequality $|\sum_{i=1}^{n}a_i|^2 \le n\sum_{i=1}^{n}|a_i|^2$ and relax $M-i+1$ to M. The second step converts the parameter-wise gradients to exit-wise gradients. Finally, we utilize Assumption 2, which allows us to consider the difference in parameters. □

Lemma 4. *Under Lemma 1 and Assumption 5, we have:*

$$\sum_{i=1}^{M}\mathbb{E}\left\|\nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1};\xi_k^{t-1}) - \nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1})\right\|^2 \le M^2\sigma^2$$

Proof. We quantify the difference between a local gradient and a stochastic gradient using the gradient noise assumption.

$$\begin{aligned}\text{LEFT} &= \sum_{i=1}^{M}\mathbb{E}\left\|\sum_{j=i}^{M}\left[\nabla\mathcal{L}_k^{(i),j}(\theta_k^{q,t-1};\xi_k^{t-1}) - \nabla\mathcal{L}_k^{(i),j}(\theta_k^{q,t-1})\right]\right\|^2 \\ &\le M\sum_{j=1}^{M}\mathbb{E}\left\|\nabla\mathcal{L}_k^j(\theta_k^{q,t-1};\xi_k^{t-1}) - \nabla\mathcal{L}_k^j(\theta_k^{q,t-1})\right\|^2 \le M^2\sigma^2\end{aligned} \tag{16}$$

The first step is similar to the proof of Lemma 4. And then we replace the gradients from parameter-wise to exit-wise. Finally, we use Assumption 5 to relax the gradient estimate of all exits objectives to $M\sigma^2$. □

Extension. Now, based on Lemma 2, Lemma 3 and Lemma 4 we proved above, we extend three lemmas in [58] to federated EEN learning as Lemma 5, Lemma 6 and Lemma 7 as follows.

Lemma 5 (Extension for Lemma 1 in [58]). *Under Lemma 2, We bounds the squared norm for federated EEN training as follows:*

$$\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left\|\theta_k^{q,t-1}-\theta^q\right\|^2 \le M^2\gamma^2T^3KG+\delta^2KT\cdot\mathbb{E}\left\|\theta^q\right\|^2$$

Proof. The squared norm of the difference between $\theta_k^{q,t-1}$ and θ^q can be bounded by $\mathbb{E}\|\theta_k^{q,t-1}-\theta_k^{q,0}\|^2+\mathbb{E}\|\theta_k^{q,0}-\theta_k^{q}\|^2$. For the first term, we apply Lemma 2 in the third step as follows:

$$\text{FIR}\le\sum_{t=1}^{T}\sum_{k=1}^{K}(t-1)\sum_{j=1}^{t-1}\mathbb{E}\left\|-\gamma\nabla\mathcal{L}_k\left(\theta_k^{q,j-1};\xi_k^{j-1}\right)\right\|^2\le\frac{M^2\gamma^2T^3KG}{3} \tag{17}$$

For the second term, the parameters changed by knowledge distillation in server is bounded by δ based on Assumption 3:

$$\text{SEC}=\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left\|\theta_{KD}^{q}[:m_k]-\theta^q\right\|^2\le\delta^2KT\cdot\mathbb{E}\left\|\theta^q\right\|^2 \tag{18}$$

□

Lemma 6 (Extension for Lemma 2 in [58]). *Under Lemma 3, we attain the upperbound for federated EEN training as follows:*

$$\sum_{i=1}^{M}\mathbb{E}\left\|\frac{1}{T\Gamma_q^{(i)}}\sum_{k\in S_i}\sum_{t=1}^{T}\left[\nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1})-\nabla\mathcal{L}^{(i)}(\theta^q)\right]\right\|^2 \le\frac{M^4L^2\gamma^2TKG}{\Gamma^*}+\frac{M^2L^2\delta^2K}{\Gamma^*}\mathbb{E}\left\|\theta^q\right\|^2$$

Proof. The difference of the gradient calculated by local model and global model can be bounded by Lemma 5.

$$\begin{aligned}\text{LEFT}&\le\sum_{i=1}^{M}\frac{1}{\Gamma_q^{(i)}T}\sum_{t=1}^{T}\sum_{k\in S_i}\mathbb{E}\left\|\nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1})-\nabla\mathcal{L}_k^{(i)}(\theta^q)\right\|^2\\&\le\frac{M^4L^2\gamma^2TKG}{\Gamma^*}+\frac{M^2L^2\delta^2K}{\Gamma^*}\mathbb{E}\left\|\theta^q\right\|^2\end{aligned} \tag{19}$$

The first step utilizes Lemma 2 from [58] and extends single-exit to multi-exit based on Lemma 3 and Lemma 5. □

Lemma 7 (Extension for Lemma 3 in [58]). *Under Lemma 4:*

$$\sum_{i=1}^{M}\mathbb{E}\left\|\frac{1}{\Gamma_q^{(i)}T}\sum_{k\in S_i}\sum_{t=1}^{T}\left[\nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1},\xi_k^{t-1})-\nabla\mathcal{L}_k^{(i)}(\theta_k^{q,t-1})\right]\right\|^2\le\frac{M^2K\sigma^2}{T(\Gamma^*)^2}$$

Proof. We use Lemma 3 in [58] for the first step and Lemma 4 for the second step as follows:

$$\text{LEFT}\le\frac{1}{T(\Gamma^*)^2}\sum_{t=1}^{T}\sum_{k=1}^{K}M^2\sigma^2\le\frac{M^2K\sigma^2}{T(\Gamma^*)^2} \tag{20}$$

□

Our Analysis Conclusion. We extend Lemma 1-3 in [58] to Lemma 5, Lemma 6 and Lemma 7 correspondingly, and then we adopt a similar procedure to get our convergence result for federated EEN training as follows:

Theorem 1. *If $\frac{1}{T\sqrt{Q}}\le\gamma<\frac{1}{6M^2LT}$, our solution coverages to a neighborhood of a stationary point of standard FL:*

$$\frac{1}{Q}\sum_{q=1}^{Q}\mathbb{E}\left\|\nabla\mathcal{L}(\theta^q)\right\|^2\le\frac{G_0}{\sqrt{Q}}+V_0+\frac{H_0}{T}+\frac{I_0}{\sqrt{Q}}\sum_{q=1}^{Q}\mathbb{E}\left\|\theta^q\right\|^2$$

where $G_0=4\mathbb{E}[\mathcal{L}(\theta^0)]$, $V_0=\frac{KG}{36\Gamma^*}$, $H_0=\frac{MK\sigma^2}{(\Gamma^*)^2}+\frac{KG}{18\Gamma^*M}$ and $I_0=\frac{L^2\delta^2K(2M+1)}{\Gamma^*\sqrt{Q}}$.

B Experiments Implementation Details

B.1 Training Details Hyperparameters

B.1.1 Baseline Hyperparameters. We reproduce previous methods in two aspects: difficulty-aware training strategies and federated learning with heterogeneous clients.

Difficulty-aware training strategies. We set the ensemble weight to 0.2 for BoosteNet. For L2W-DEN, we set the hidden size of the meta net to 500 and use the Adam optimizer with a weight decay of $1e^{-4}$, an initial learning rate (LR) of $1e^{-4}$, and $p=30$ for the budgeted exit policy during meta training. One meta training process is conducted in each local epoch.

FL with heterogeneous clients. These heterogeneous clients are divided into 4 categories based on their resources, and submodels, one exit after 3-layers for deit model with 12 layers, are assigned to them. For federated learning with heterogeneous clients, we set $\beta=0.2$ for the momentum distillation of InclusiveFL. For DepthFL, we set the temperature $\tau=1$ for self-KD. For ReeFL, we set $\tau=1$ for its dynamic self-KD, use a normalized linear layer as the shared classifier, and configure the hyperparameters of the accumulator as specified for ReeFL. For ScaleFL, we follow the depth-scale method to add classifiers at the 4-th, 7-th, 10-th, and last transformer blocks with widths of $[\frac{3}{4},\frac{6}{7},\frac{9}{10},\frac{12}{12}]$ for each block, to ensure that the number of model parameters is similar to the others, and set the temperature $\tau=3$.

B.1.2 Ours Hyperparameters. In each communication round, we iterate 5 knowledge distillation steps after updating the generator once, which involves conducting 2 epochs on the server. The architecture and training strategies of the generator are detailed in Sec. 4.3.1. For knowledge distillation, the training configuration (learning rate, optimizer, weight decay) of the student model is the same as that used for local training as follows.

B.1.3 Local Training. Using difficulty-aware training strategies [11, 54], each client trains its local model using the SGD optimizer with the following settings: a batch size of 32, a momentum of 0.9, a weight decay of $1e^{-4}$, an initial learning rate (LR) of $5e^{-2}$, and an LR decay of 0.99.

B.1.4 Server & Aggregation. There are 100 clients, divided into 4 levels with increasing resources, which train 4 sizes local model with their private dataset. We set three scenarios to simulate heterogeneous clients by varying the majority resources of clients, low-end: ratio of 4 level clients is $[0.1, 0.2, 0.3, 0.4]$; high-end: the ratio is $[0.4, 0.3, 0.2, 0.1]$; normal: the ratio is $[0.25, 0.25, 0.25, 0.25]$.

Additionally, the total number of communication rounds is 500, and in each round, 10% of the clients are sampled for local training. We use FedAvg [30] to aggregate parameters of local models.