Appendix

The supplementary materials are organized as follows:

- Appendix A: Provides more details on related works.
- Appendix B: Presents the full algorithm description of TAKFL.
- Appendix C: Presents formal theoretical statements, assumptions, and proofs supporting our method.
- Appendix D: Presents detailed experimental results including some additional experiments.
- Appendix E: Presents the ablation studies experiments.
- Appendix F: Presents hyper-parameters and implementation details.

A   More Detailed Related Works

(a) Partial Model Training   (b) Heterogeneous Device Prototypes
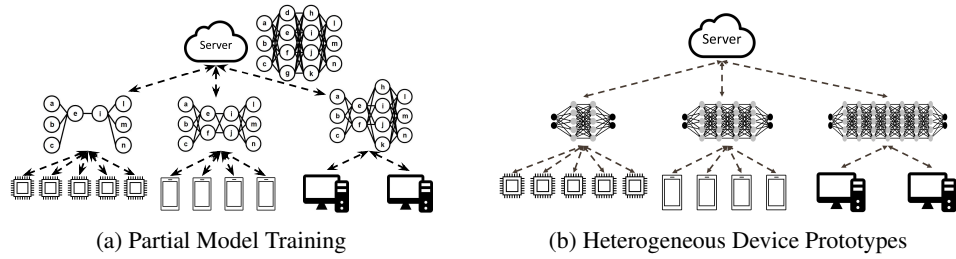
Figure 4: **Overview of Two Different Device Heterogeneous FL Settings.** (a) In the partial model training setting, the objective is to train a single global model where heterogeneous devices train a specific sub-model based on their computational resources. This approach necessitates device support for varying neural network architectures, which is impractical as devices typically have specialized architectures designed to match their hardware, software configurations, and underlying machine learning tasks. (b) In the heterogeneous device prototypes setting, device prototypes participate in FL to enhance the performance of their global model by transferring knowledge across prototypes. This setting is more feasible as it accommodates diverse device prototypes with their own specific configurations, including neural network architecture and dataset. However, establishing effective knowledge transfer between differently sized prototypes (like IoTs and workstations) and diverse configurations is challenging. In this paper, we address this issue.

Prior works on device heterogeneous FL have considered two distinct approaches with different objectives and settings. The first group of studies focuses on accommodating devices with varying compute resources, aiming to train a single global model [11, 3, 56, 52, 54]. Various partial model training techniques have been proposed for this setting, where devices are tasked with training a sub-model of a global model according to their compute resources. These include dropout-based [3], static [11, 18], and rolling-based sub-model extraction techniques [1]. Federated Dropout builds upon the concept of dropout [44] to extract smaller sub-models. Static sub-model extraction techniques like in HeteroFL [11] and FjORD [18] consistently extract designated portions of the global model, whereas FedRolex [1] introduces a more flexible rolling method for sub-model extraction. However, these approaches assume that devices can support various sub-model architectures for training, which does not fully reflect the real-world scenario. In practice, there exist a diverse spectrum of device prototypes such as IoT devices and smartphones each have unique and unhashable neural network architectures tailored to their specific hardware and software configurations and underlying machine learning tasks. Consequently, these device prototypes may not support training various neural network architectures, highlighting a significant limitation in accommodating the full spectrum of device heterogeneity in this setting.

The second array of studies tackles a more practical scenario where device prototypes with heterogeneous model architectures participate in FL to enhance their global model performance through mutual knowledge sharing. In this context, knowledge distillation techniques are employed to transfer knowledge among device prototypes [30, 6, 41]. Here, locally updated client models from various

1

device prototypes, collectively referred to as ensembles, serve as teachers to distill their knowledge into each server's student model using an unlabeled public dataset. For instance, FedDF [30] utilizes vanilla averaging of all ensemble logits as the distillation target for all server student models. In contrast, FedET [6] employs an uncertainty-weighted average of ensembles' logits as the distillation target for all server student models, complemented by a diversity regularization technique. However, methods like FedET rely on the neural networks' confidence scores for uncertainty estimates, overlooking the fact that neural networks are often poorly calibrated and prone to over-confidence, which compromises their ability to provide reliable uncertainty estimates [48, 15, 5, 53]. *These existing works typically focus on settings where device prototypes have similar capabilities, i.e. similar model and dataset sizes, thus neglecting the challenges presented in more diverse settings where device prototypes vary significantly in terms of model and dataset size. This oversight limits the effectiveness of these methods in truly diverse and heterogeneous environments. In this paper, we introduce TAKFL, which is designed to address the limitations of existing methods in these underexplored diverse device heterogeneous settings.*

Figure 1 illustrates the distinctions between these two different settings studied in the literature. For more information, we refer the reader to recent surveys [35, 26, 39, 4].

592

B   Full Algorithm Description of TAKFL

593 The full algorithm description of TAKFL is presented in Algorithm 1.

Algorithm 1 TAKFL Algorithm

Require: number of communication rounds (R), public unlabeled dataset $\mathbb{D}^{\text{public}}$, server training iterations I, heterogeneous device prototypes ($i \in \mathbb{M}$) with their associated clients ($\mathbb{C}^i$) and local datasets ($\{\mathbb{D}_k^i\}_{k \in \mathbb{C}^i}$), model architecture ($f^i$), local training iterations (I_{local}), local learning rate (η_{local}), server distillation iterations ($I_{distill}$), and server distillation learning rate ($\eta_{distill}$).

1: **Server Executes:**
2: Randomly initialize all device prototype's server model $\{\boldsymbol{\theta}_0^i\}_{i \in \mathbb{M}}$
3: **for** each round $r = 0, 1, \ldots, R-1$ **do**
4:     $\mathbb{C}_r^i \leftarrow$ (randomly select clients from each device prototype) $\forall i \in \mathbb{M}$
5:     **for** each client $k \in \mathbb{C}_r^i$, $\forall i \in \mathbb{M}$ **in parallel do**
6:         $\widehat{\boldsymbol{\theta}}_k^i \leftarrow \texttt{ClientUpdate}(k; \boldsymbol{\theta}_r^i)$
7:     **end for**
8:     $\boldsymbol{\theta}_{avg}^i = \sum_{k \in \mathbb{C}_r^i} \frac{|\mathbb{D}_k^i|}{\sum_{k \in \mathbb{C}_r^i} |\mathbb{D}_k^i|} \widehat{\boldsymbol{\theta}}_k^i$
9:     **for** each device prototype's server student $i = 1, 2, \ldots, M$ **in parallel do**
10:         **for** each device prototype's teacher ensembles $j = 1, 2, \ldots, M$ **in parallel do**
11:             $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{avg}^i$
12:             **for** each server distillation iteration $t = 0, 1, 2, \ldots, I_{distill}$ **do**
13:                 $\boldsymbol{x} \leftarrow$ sample a mini-batch of data from public dataset $\mathbb{D}^{\text{public}}$
14:                 $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta_{distill} \cdot \nabla \mathcal{L}_{\mathcal{S}}^{\mathcal{T}_i}$ defined in Eq. 6.
15:             **end for**
16:             $\tau_j \leftarrow \boldsymbol{\theta}^{I_{distill}} - \boldsymbol{\theta}_{avg}^i$
17:         **end for**
18:         $\boldsymbol{\theta}_{r+1}^i \leftarrow \boldsymbol{\theta}_{avg}^i + \sum_{j=1}^M \lambda_j \tau_j$
19:     **end for**
20:     $\boldsymbol{\theta}_{r+1}^i \leftarrow \boldsymbol{\theta}^i$
21: **end for**

22: **function** $\texttt{ClientUpdate}(k, \boldsymbol{\theta}_r^i)$
23:     $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_r^i$
24:     **for** each local update iteration $t = 0, 1, \ldots, I_{local} - 1$ **do**
25:         $\{\boldsymbol{x}, y\} \leftarrow$ sample a mini-batch of data from local dataset $\mathbb{D}_k^i$
26:         $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta_{local} \cdot \nabla \ell(f^i(\boldsymbol{x}; \boldsymbol{\theta}^t), y)$
27:     **end for**
28:     $\widehat{\boldsymbol{\theta}}_k^i \leftarrow \boldsymbol{\theta}^{I_{local}}$
29: **end function**

3

C Theoretical Results

C.1 Proofs of the Main Propositions

First we present the formal assumptions associated with our theoretical derivations.

Assumption 1. Local federated averaging is performed with perfect test accuracy, i.e.,

$$\underset{\boldsymbol{\theta}^j}{\operatorname{argmin}} \sum_{j=1}^{M}\sum_{k=1}^{N^j} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{D}_k^j}\left[\ell(f^j(\boldsymbol{x};\boldsymbol{\theta}^j),y)\right] = \underset{\boldsymbol{\theta}^j}{\operatorname{argmin}} \ \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}^j}\left[\ell(f^j(\boldsymbol{x};\boldsymbol{\theta}),y)\right] \qquad (9)$$

That is, the training error on the datasets $\{\mathbb{D}_k^j\}$ for the computed θ_{avg}^j is the same as the test error on the population distribution $\mathcal{D}^j$. Moreover assume that we can write $\mathcal{T}_i = \left\{\sum_{k=1}^{N^i} f^i(\cdot,\hat{\boldsymbol{\theta}}_k^i)|k\in\mathbb{C}^i\right\} = \{f^i(\cdot,\theta_{avg}^i)\}$. Finally, we assume that the same population distribution $\sum_j \omega_j \mathcal{D}^j$ is the same that the clients perform their testing on as the server performs distillation on.

These assumptions are made for mathematical practicality while at the same time not starkly unreasonable. The local FL the device prototypes perform is generically prone to imprecision, especially as the clients' data varies, but this discrepancy is bounded [16]. Similarly the difference in the average of logits and the logit of averages has a bounded difference norm [51]. Thus, violations of the Assumption add additional perturbations to quantities derived in the Theoretical analysis without having structural/qualitative effects, and thus would only present clutter in the presentation.

Notations. Now we present the notation defining the specific quantities we refer to in the derivations below. The set of important quantities is given in Table 3. Note that the formal definitions of the first two quantities are,

$$\boldsymbol{\Theta}^j := \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}^j}\left[\ell(f^j(\boldsymbol{x};\boldsymbol{\theta}),y\right], \ \boldsymbol{\Theta}^{j,k} := \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \ \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}^i}\left[\ell(f^j(\boldsymbol{x};\boldsymbol{\theta}),y\right]$$

Table 3: Notation and Definitions

Notation	Definition
$\boldsymbol{\Theta}^j$	Parameters in j's device model that minimize the loss on its population distribution
$\boldsymbol{\Theta}^{j,k}$	Parameters in j's device model that minimize the loss on i'th population distribution
$Q^j = \dim(\boldsymbol{\theta}^j)$	The total capacity of device prototype j
$\mathcal{Q}^j = \{e_k^j\}_{k=1,\dots,Q^j}$	Eigenbasis for the model of device prototype j
$W^j = \dim(\boldsymbol{\Theta}^j)$	Dimension of the solution submanifold $\boldsymbol{\Theta}^j$
$W^{j,k} = \dim(\boldsymbol{\Theta}^{j,k})$	Dimension of the solution submanifold $\boldsymbol{\Theta}^{j,k}$
$\mathcal{W}^j = \{e_k^j\}_{k=1,\dots,W^j}$	Eigenbasis the solution submanifold $\boldsymbol{\Theta}^j$
$\mathcal{W}^{j,k} = \{e_l^{j,k}\}_{l=1,\dots,W^{j,k}}$	Eigenbasis the solution submanifold $\boldsymbol{\Theta}^{j,k}$

We shall make use of the "Choose" combinatorial operator, defined to be $Ch(n,p) = \frac{n!}{p!(n-p)!}$. The standard $O(\cdot)$ notation indicates $a_k = O(b_k)$ to mean there exists K and C such that for $k\geq K$, $a_k \leq Cb_k$.

A recent finding that inspired the methodology in this work is the discovery of the weight disentanglement phenomenon underlying task arithmetic [37]. Indeed the *task arithmetic property* provides the ideal circumstance for federated knowledge transfer as we shall see below. Formally, adapting their definition to our notation:

(**Task Arithmetic Property**) holds for a set of vectors $\{\boldsymbol{\tau}_j\}$ if for all j it holds that,

$$f^j\left(\boldsymbol{x};\boldsymbol{\theta}_{avg}^j + \sum_{i\neq j}\lambda_i\boldsymbol{\tau}_i\right) = \begin{cases} f^j(\boldsymbol{x};\boldsymbol{\theta}_{avg}^j + \lambda_i\boldsymbol{\tau}_i) & \boldsymbol{x}\in\mathcal{D}^i \\ f^j(\boldsymbol{x};\boldsymbol{\theta}_{avg}^j) & \boldsymbol{x}\in\mathcal{D}^j\setminus\cup_{i\neq j}\mathcal{D}^i \end{cases} \qquad (10)$$

Let us define an important property of task arithmetic that we shall use in the sequel.

4

620 **(Weight disentanglement).**[37] A parametric function $f : \mathcal{X} \times \Theta \to \mathcal{Y}$ is weight disentangled with
621 respect to a set of task vectors $T = \{\boldsymbol{\tau}_j\}_{j \in \mathbf{T}}$ and the corresponding supports $\mathcal{D}_T := \{\mathcal{D}_j\}_{j \in \mathbf{T}}$ if

$$f(\boldsymbol{x}; \boldsymbol{\theta}_0 + \sum_{i \in \mathbf{T}}^{T} \alpha_i \boldsymbol{\tau}_i) = \sum_{i \in \mathbf{T}} g_j(\boldsymbol{x}; \alpha_i \boldsymbol{\tau}_i) + g_0(\boldsymbol{x}),$$

622 where $g_i(x; \alpha_i \boldsymbol{\tau}_i) = 0$ for $\boldsymbol{x} \notin \mathcal{D}_i$ and $i \in \mathbf{T}$, and $g_0(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \bigcup_{i \in \mathbf{T}} \mathcal{D}_i$.

623 We now present the formal statements as well as the proofs of the main propositions.

624 **Proposition 1.** *(Information Loss in VED). Consider the VED procedure in the form of solv-*
625 *ing (3). Consider two device prototypes with a device capacity and solution dimension of Q^1, Q^2*
626 *and W^1, W^2, respectively, and with associated eigenbases $\mathcal{Q}^i, \mathcal{W}^i$. Let the solution set of VED*
627 *with prototype i as student be $\hat{\Theta}^i_{VED}$ with $\dim(\hat{\Theta}^i_{VED}) = W^{v_i}$ with eigenbasis $\mathcal{W}^{v_i}$. In addition,*
628 *denote $W^{s,t}$, $s, t \in \{1, 2\}$ the dimension of the solution set for the student model trained on the*
629 *data from the teacher device's ensembles. We assume that self-distillation is executed appropriately,*
630 *e.g., $W^{1,1} = W^1$ and $W^{2,2} = W^2$.*

631 *1. **Case 1:** Assume that $Q^1 = Q^2$ and $W^1 = W^2 = W^{1,2} = W^{2,1}$. Then it holds that, in*
632 *expectation,*

$$\dim\left(\hat{\Theta}^1_{VED} \cap [\mathcal{Q}^1 \setminus \mathcal{W}^1]\right) = O\left(\frac{(Q^1 - W^1)(W^1)!(Q^1 - W^{1,2})!}{Q^1!(W^1)!(Q^1 - W^1)! + Q^1!W^{1,2}!(Q^1 - W^{1,2})!}\right)$$

633 *This corresponds to the expected capacity of prototype 1 that is taken up for fitting logits that are not in the*
634 *span of $\mathcal{W}^1$, that is, that do not fit the data corresponding to prototype 1.*
635 *2. **Case 2:** Assume that $Q^1 > Q^2$ and $W^1 = W^{1,2} > W^2$. Then the same quantity as for Case 1 holds.*
636 *Moreover,*

$$\dim\left(\hat{\Theta}_{VED} \cap [\mathcal{Q}^1 \setminus (\mathcal{W}^1 \cup \mathcal{W}^{1,2}]\right) = O\left(\frac{(Q^1 - W^1)(W^1!)(W^{1,2} - W^2)!}{Q^1!W^1!(Q^1 - W^1)! + Q^1!W^2!(W^{1,2} - W^2)!}\right)$$

637 *This corresponds to capacity of client 1 that has been allocated but fits, in the model of prototype 1, neither*
638 *the data of prototype 1, nor of the data of prototype 2.*

639 *Proof.* Formally,

$$\hat{\Theta}_{VED} := \operatorname*{argmin}_{\theta \in \mathcal{Q}^1} \mathcal{L}_{\text{ED}} = \operatorname*{argmin}_{\theta} \text{KL}\left[\sum_{i=1,2} \sigma\left(f^i(\boldsymbol{x}, \boldsymbol{\theta}^i_{avg})\right), \sigma\left(\mathcal{S}(\boldsymbol{x})\right)\right]$$

640 Since by assumption $\boldsymbol{\theta}^i_{avg}$ solves the training problem on the data associated with device prototype
641 i, the logit is accurate, and thus there is a map $\mathcal{O}(i, j) : \mathcal{T}_i \to \mathcal{T}^j_i \subseteq \mathcal{W}^{i,j}$. The self distillation, that
642 is, $\mathcal{S}_j$ defines a bijective map from $\mathcal{W}^j$ to $\mathcal{W}^j$ and thus does not affect the capacity allocation.

643 **Case 1:** In this case, generically (that is, measure zero on some non-atomic sampling on a dis-
644 tribution of operators) $\mathcal{O}(i, j)$ is bijective. Now let us compute the expectation of the number of
645 eigenvectors of, e.g. $\mathcal{W}^{1,2}$ that are in the complement of the span of $\mathcal{W}^1$. Assuming, for simplic-
646 ity, independence, this would correspond to counting the possible choices within the capacity of
647 $\mathcal{Q}^1 \setminus \mathcal{W}^1$ over the range of possible choices of filling the capacity of $\mathcal{Q}^1$ with vectors in $\mathcal{W}^1$ together
648 with choices of filling it with vectors in $\mathcal{W}^{1,2}$:

$$\sum_{i=1}^{Q^1 - W^1} i \frac{Ch(Q^1 - W^1, i)}{Ch(Q^1, W^1) + Ch(Q^1, W^{1,2})}$$

649 For, e.g., $Q^1 = 4$ and $W^1 = W^{1,2} = 2$ this is $\frac{1}{3}$.

650 To derive a scaling rate we can write:

$$\sum_i i \frac{\frac{(Q^1 - W^1)!}{i!(Q^1 - W^1 - i)!}}{\frac{Q^1!}{(W^1)!(Q^1 - W^1)!} + \frac{Q^1!}{W^{1,2}!(Q^1 - W^{1,2})!}} = O\left(\frac{(Q^1 - W^1)(W^1)!(Q^1 - W^{1,2})!}{Q^1!(W^1)!(Q^1 - W^1)! + Q^1!W^{1,2}!(Q^1 - W^{1,2})!}\right)$$

651 **Case 2:** In this case, it must be that, at best almost surely, $\mathcal{O}(2, 1)$ is injective, but not surjective.
652 This means that distilling from 2 to 1 does not fill the capacity of $\mathcal{W}^{1,2}$, and is thus a fundamentally

5

653 wasteful operation, that is $|\mathcal{T}_j^j| = W^2 < W^{1,2}$. Now let us compute the expectation of the number
654 of eigenvectors of, e.g. $\mathcal{W}^{1,2}$ that are in the complement of the span of $\mathcal{W}^1$. Since $\mathcal{W}^{1,2}$ are being
655 structurally allocated for fitting, the combinatorial expression is the same:

$$\sum_{i=1}^{Q^1-W^1} i \frac{Ch(Q^1-W^1,i)}{Ch(Q^1,W^1)+Ch(Q^1,W^{1,2})}$$

656 Thus for, e.g., $Q^1 = 4$ and $W^1 = W^{1,2} = 2$ this is, again, $\frac{1}{3}$. The scaling in this case is

$$O\left(\frac{(Q^1-W^1)(W^1!)(Q^1-W^{1,2})!}{Q^1!W^1!(Q^1-W^1)!+Q^1!W^2!(Q^1-W^2)!}\right)$$

657 However, we observe that there are vectors in the range of $\mathcal{W}^{1,2} \setminus \mathcal{O}(2,1)(\mathcal{W}^2)$ that have been
658 allocated by the VED but lie in neither $\mathcal{W}^1$ nor in $\mathcal{W}^{1,2}$, that is, are garbage. We can compute those
659 as the expected number of eigenvectors arising from allocating $\mathcal{W}^{1,2} \setminus \mathcal{O}(2,1)(\mathcal{W}^2)$ that intersect
660 with $\mathcal{Q}^1 \setminus \mathcal{W}^1$ (that is, the spare capacity not used for fitting data $\mathcal{D}^1$). This is, using similar
661 principles:

$$\sum_{i=1}^{W^{1,2}-W^1} i \frac{Ch(Q^1-W^1,i)}{Ch(Q^1,W^1)+Ch(Q^1,W^{1,2}-W^2)}$$

662 This is for, e.g., $Q^1 = 4$, $W^{1,2} = 2$ and $W^2 = 1$, this would be $\frac{3}{10}$

663 The scaling here is

$$O\left(\frac{(Q^1-W^1)(W^1!)(W^{1,2}-W^2)!}{Q^1!W^1!(Q^1-W^1)!+Q^1!W^2!(W^{1,2}-W^2)!}\right)$$

664 ∎

665 **Proposition 2.** *(Improve knowledge transfer with task arithmetic).* *Consider the TAKFL procedure*
666 *as in the form of computing* (8). *Consider two device prototypes with a device capacity and solution*
667 *dimension of Q^1, Q^2 and W^1, W^2, respectively, and with associated eigenbases $\mathcal{Q}^i, \mathcal{W}^i$. Let the*
668 *solution set of TAKFL with prototype i as student be $\hat{\Theta}_{TA}^i$ with $\dim(\hat{\Theta}_{TA}^i) = W^v$ with eigenbasis*
669 *$\mathcal{W}^v$. In addition, denote $W^{s,t}$, $s,t \in \{1,2\}$ dimension of the solution set for the student model*
670 *trained on the data from the teacher device's ensembles. . The following statements hold:*

671 *In the case that that $Q^1 \geq Q^2$ and $W^1 \geq W^2$, it holds that the TAKFL preserves that the*
672 *eigenbasis used to model the data $\mathcal{D}^1$'s accuracy for device prototype 1, that is for student 1*

$$\dim\left(\mathcal{W}^v \cap [\mathcal{Q}^1 \setminus \mathcal{W}^1]\right) = 0$$

673 ***Case 1:*** *Assume that $Q^1 = Q^2$ and $W^1 = W^2$. Then it holds that,*

$$\dim\left(\mathcal{W}^v \cap [\mathcal{Q}^1 \setminus (\mathcal{W}^1 \cup \mathcal{W}^{1,2})]\right) = 0$$

674 *Moreover, it holds that,*
$$\hat{\Theta}_{TA} \in Span(\mathcal{W}^1 \cap \mathcal{W}^{1,2})$$

675 *Thus, with equal capacity, no information is lost in Task Arithmetic aided knowledge ensemble distillation*
676 *and capacity is efficiently used to model the data from both prototype 1 and prototype 2.*
677 ***Case 2:*** *Assume that $Q^1 > Q^2$ and $W^1 > W^2$. Then it again holds that,*

$$\dim\left(\mathcal{W}^v \cap [\mathcal{Q}^1 \setminus (\mathcal{W}^1 \cup \mathcal{W}^{1,2})]\right) = 0$$

678 *However, while $\hat{\Theta}_{TA} \in Span(\mathcal{W}^1)$, it holds that $\dim\left(\mathcal{W}^v \cap \mathcal{W}^{1,2}\right) = W^2 < W^{1,2}$.*

679 *Proof.* We can see immediately from the weight disentanglement property of Task Arithmetic that,

$$f^1(\boldsymbol{x}; \boldsymbol{\theta}_{avg}^1 + \alpha_1\boldsymbol{\tau}_1 + \alpha_2\boldsymbol{\tau}_2) = g^{1,1}(\boldsymbol{x}; \alpha_1\boldsymbol{\tau}_1) + g^{1,2}(\boldsymbol{x}; \alpha_2, \boldsymbol{\tau}_2) + g^{1,0}(x)$$

680 with $g^{1,1}(\boldsymbol{x}; \alpha_1\boldsymbol{\tau}_1)$ for $\boldsymbol{x} \notin \mathcal{D}^1$, $g^{1,2}(\boldsymbol{x}; \alpha_2\boldsymbol{\tau}_2)$ for $\boldsymbol{x} \notin \mathcal{D}^2$ and $g^{1,0}(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \mathcal{D}^1 \cup \mathcal{D}^2$. From
681 this, we can immediately conclude the first statement of the Proposition as well as the expression

$$\dim\left(\mathcal{W}^v \cap [\mathcal{Q}^1 \setminus (\mathcal{W}^1 \cup \mathcal{W}^{1,2})]\right) = 0$$

6
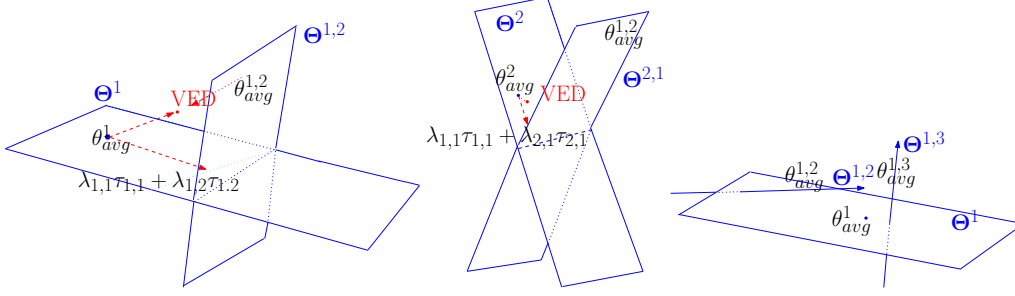
Figure 5: **Illustration of Geometric Intuition**. Each panel presents a different case example. The left and center panels present the geometric intuition of KD for vanilla ensemble distillation (VED) and TAKFL in the case where two different large device prototypes performing knowledge transfer. The planes represent the solution subspaces. The right panel presents a circumstance by which two small device prototypes (2, and 3) serve as teacher for transferring knowledge to a larger device prototype 1.

and also, in the case of $W^1 = W^{1,2} = W^2$ implies

$$\hat{\Theta}_{TA} \in Span(\mathcal{W}^1 \cap \mathcal{W}^{1,2})$$

For the last statement we observe again as in the second Case in the Proposition describing VED, $\dim\left(\mathcal{O}(2,1)(\mathcal{W}^2)\right) < W^{1,2}$ from which we can conclude that, generically

$$\dim\left(\{v : v \in \mathcal{O}(2,1)(\mathcal{W}^2) \subseteq \mathcal{W}^v, \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}^2}l(f^1(\boldsymbol{x};v),y) > 0\}\right) = W^{1,2} - W^2$$

proving the final statement. ∎

We observe that a key mechanism of the proof is the dimension of the target space of the teaching operator $\mathcal{O}(i,j)$. As an informative model, we can consider coefficients λ_j of task vectors as restricting the rank, relative to other teachers. For instance, in the previous Proposition, if $W^{1,2} = 2$ and $W^2 = 1$, then $\lambda_2 = 1/2$, so as to enforce one vector of $\mathcal{W}^{1,2}$ is a target for the map $\tilde{O}(2,1)$, would be appropriately sensible.

C.2 Geometric Intuition

In this section we aim to provide geometric intuition for the mechanism of VED and Task Arithmetic KD on three different cases. Figure 5 presents the geometry illustration for three different cases. We discuss each case in the following.

Case I: KD between two large prototypes with different data distributions. Consider Figure 5 left panel. This panel corresponds to a setting where two large device prototypes with similar total capacity, i.e. $Q^1 = Q^2 = 3$ perform knowledge transfer. We consider the solution dimensions of both prototypes to be the same, i.e. $W^1 = W^2 = 2$. These would correspond to planes in the ambient space. Therefore, one plane corresponds to the solution subspace of the prototype 1 trained on its own data, i.e. Θ^1 subspace in the panel, and the other corresponds to the (theoretical) solution subspace of this prototype trained on prototype 2's data, i.e. $\Theta^{(1,2)}$ in the panel. In this case, since the data distributions of the prototypes are fairly disparate, this has resulted into near orthogonal subspaces corresponding to these solutions. As we can see from the panel, VED will lead to point which is far away from either of the planes corresponding to optimal solution subspaces, and far from the optimal set of parameters, which is their intersection, suggesting a loss of knowledge. By contrast, the TAKFL approach, by customizing the merging coefficients and putting each to half, i.e. $\lambda_{1,1}\lambda_{1,2} = 0.5$, can traverse in the tangent space of zero loss surface and get into the intersection subspace which is exactly the optimal solution ($\boldsymbol{\theta}^*_{merged} = \boldsymbol{\theta}^1_{avg} + \lambda_{1,1}\boldsymbol{\tau}_{1,1} + \lambda_{1,2}\boldsymbol{\tau}_{1,2}$).

Case II: KD between two large prototypes with similar data distributions. Consider Figure 5 center panel. Similar to Case I, this panel corresponds to a setting where two large device prototypes with similar total capacity, i.e. $Q^1 = Q^2 = 3$ performing knowledge transfer. We consider the solution dimensions of both prototypes to be the same, i.e. $W^1 = W^2 = 2$. These would correspond

7

to planes in the ambient space. Therefore, one plane corresponds to the solution subspace of the prototype 1 trained on its own data, i.e. Θ^1 subspace in the panel, and the other corresponds to the (theoretical) solution subspace of this prototype trained on prototype 2's data, i.e. $\Theta^{(1,2)}$ in the panel. In this case, since the data distributions of the prototypes are fairly close, this has resulted into non-orthogonal solution subspaces. As we can see from the panel, while VED could still lead to some information loss, by and large we expect straightforward KD. Our task arithmetic (TA) approach, again by customizing the merging coefficients $\lambda_{1,1}\lambda_{1,2} = 0.5$, can traverse in the tangent space of zero loss surface on each plane and get into the intersection subspace, corresponding to the most efficient allocation of device prototype capacity for fitting simultaneously the logits corresponding to accurate modeling of device prototype 1 as well as device prototype 2's data distribution.

Case III: KD between two small prototypes and one large prototype. Now consider the right panel in Figure 5. This panel corresponds to a setting where two small prototypes serve as teachers and one large prototype is the student. The $\boldsymbol{\theta}_{avg}^1$ plane corresponds to the solution subspace of the large prototype 1 on its own data, $\mathcal{D}^1$. The line $\boldsymbol{\theta}_{avg}^{1,2}$ line corresponds to the subspace of solutions in prototype 1's parameter space projected into the capacity of the information transferred from device prototype 2. Finally, the line labelled $\boldsymbol{\theta}_{avg}^{1,3}$ corresponds to the subspace of solutions in prototype 1's parameter space projected into the capacity of the information transferred from device prototype 3. Here, we can see from the relative angle of the lines with respect to the plane that the distribution $\mathcal{D}^1$ is closer to the distribution $\mathcal{D}^2$ than to $\mathcal{D}^3$. Comparing this case to the previous cases, $\boldsymbol{\theta}_{avg}^{1,3}$ is like case I and $\boldsymbol{\theta}_{avg}^{1,2}$ is like case II. We can apply the same conclusions here as well regarding the performance of vanilla ensemble distillation and our adaptive task arithmetic approach. We can see from the geometric visualization that knowledge distillation towards $\boldsymbol{\theta}_{avg}^{1,3}$ has more margin of error for prototype 1. Therefore, with the TAKFL approach large prototype 1 can strategically select which prototype to learn more from, and since $\boldsymbol{\theta}_{avg}^{1,2}$ has closer data distribution to prototype 1, TAKFL will prioritizes this by putting a larger merging coefficient, i.e $\lambda_{1,2} > \lambda_{1,3}$. By contrast, VED lacks this customization and results in sub-optimal knowledge distillation.

The geometric intuition discussed here is consistent with our detailed experimental analysis in D.1.1.

C.3 Analytical Properties of Learning Dynamics

Here we provide additional insights from the literature as to the nature and properties of learning as it takes place on overparametrized models. Specifically, we comment on literature in the area of Stochastic Differential Equation (SDE) models for SGD training dynamics, and its correspondence to the results above. Overparametrization has been conjectured to be a significant factor in contributing to the (unexpected, by classical bias-variance tradeoffs) generalization ability of deep neural networks, from a number of perspectives [13].

Consider the diffusion model of SGD training for overparametrized NNs provided in [29]. Their analysis relies on the following two assumptions. For our purposes L is shorthand for a client group's loss, $L(\boldsymbol{\theta}) = \sum_{k \in \mathbb{C}^j} \mathbb{E}_{(\boldsymbol{x},y) \sim \mathbb{D}_k^j} \left[\ell(f^j(\boldsymbol{x}; \boldsymbol{\theta}), y) \right]$ for some j, which will be identified from the context.

Assumption 2. $L : \mathbb{R}^Q \to \mathbb{R}$ is C^3 and the solution set Γ is a W-dimensional C^2-submanifold of $\mathbb{R}^D$ for $0 \leq W \leq D$ and $\text{rank}(\nabla^2 L(\theta)) = Q - W$

Assumption 3. Assume that U is an open neighborhood of Γ satisfying that gradient flow starting in U converges to some point in Γ

From these, [29] derives Theorem 4.6. This Theorem decomposes the random process of the parameter weights driven by SGD after it has reached the solution manifold, e.g., the diffusive random walk of θ_{avg}^1 in Figure 5 along its respective solution manifold.

Theorem 1. *Given L, Γ and $\theta_\mu(0) = \theta(0) \in U$ as by Assumptions 2 and 3 the SDE modeling the optimization of F by SGD, that is, defining $\Phi(X)$ to be the gradient flow applied to the state random variable X, then for T as long as $\mathbb{P}[Y(t) \in U, \forall 0 \leq t \leq T] = 1$, $\theta_\eta(\lfloor T/\eta^2 \rfloor)$ converges in distribution to the stochastic process $Y(T)$ as $\eta \to 0$, with $Y(t)$ given as,*

$$
\begin{aligned}
dY(t) = {} & \Sigma_\parallel^{1/2}(Y)dW(t) - \tfrac{1}{2}\nabla^2 L(Y)^\dagger \partial^2(\nabla L)(Y)[\Sigma_\parallel(Y)]dt \\
& - \tfrac{1}{2}\partial\Phi(Y)\left(2\partial^2(\nabla L)(Y)\left[\nabla^2 L(Y)^\dagger \Sigma_{\perp,\parallel}(Y)\right] + \partial^2(\nabla L)(Y)\left[\mathcal{L}_{\nabla^2 L}^{-1}(\Sigma_\perp(Y))\right]\right)dt
\end{aligned}
\tag{11}
$$

8

762 *where* $\Sigma \equiv \sigma\sigma^T$ *and* $\Sigma_\parallel$, $\Sigma_\parallel$, $\Sigma_\parallel$ *are given as,*

$$\Sigma_\parallel = \partial\Phi\Sigma\partial\Phi, \ \Sigma_\perp = (I_D - \partial\Phi)\Sigma(I_D - \partial\Phi),$$
$$\Sigma_{\parallel,\perp} = \partial\Phi\sigma(I_D - \partial\Phi) \tag{12}$$

763 This theorem indicates that the asymptotic flow of SGD on the client training can be decomposed
764 into a covariance-driven random walk in the tangent space, drift to preserve the flow into the tangent
765 plane, the tangent-normal portion of the noise covariance and noise in the normal direction.

766 This analytical expression provides the probabilistic foundations for the more higher level theoretical
767 results above. In particular, local gradient dynamics, as employed by individual device prototypes
768 j using FedAvg on its local clients, yields a flow for the stochastic process defined by the weights.
769 At this point of learning, the weights are traversing the solution set, with noise predominantly in the
770 tangent directions. Thus knowledge distillation which preserves this noise structure is going to be
771 more effective as far as preserving accuracy across data.

D   Detailed Experimental Results

773 In this section we present a more detailed version of experimental results presented in the main paper
774 Section 7. Additional experimental results are also presented here.

D.1   Main Experimental Results on Computer Vision (CV) Task

776 **The experiments in this section complements the main experimental results in the main paper**
777 **Section 7.2.**

778 **Experimental Setup.** For the evaluation on CV task, we employ CIFAR-10 and CIFAR-100 [24]
779 datasets. For CIFAR-10, we use CIFAR-100 as the unlabeled public dataset, while ImageNet-100,
780 a subset of ImageNet [10] with 100 classes (see Appendix F.1.1), is used for CIFAR-100. We
781 distribute the training dataset among the device prototypes in a ratio of 1:3:6 for S, M, and L,
782 respectively. Each device prototype's data portion is further distributed among its clients using a
783 Dirichlet distribution. We apply two levels of data heterogeneity for a comprehensive evaluation:
784 low heterogeneity, i.e. Dir(0.3), and high heterogeneity, i.e. Dir(0.1). Additionally, we configure the
785 number of clients and their sampling rates as follows: 100 clients for S, 20 for M, and 4 for L, with
786 sampling rates set at 0.1, 0.2, and 0.5 respectively. To comprehensively evaluate, we use two distinct
787 architectural settings: the *"homo-family"* setting, where all device prototypes' architectures are from
788 the same family—employing ResNet8, ResNet14, and ResNet18 [17] for S, M, and L, respectively;
789 and the *"hetero-family"* setting, which diverse architectures are used—ViT-S [12] for S, ResNet14
790 for M, and VGG-16 [42] for L. All models are initialized from scratch, and the communication
791 round is set at 60 rounds. Further details regarding hyper-parameters can be found in Table 12.

792 **Overview of Performance Results.** Table 4 presents the performance of TAKFL across diverse
793 architecture settings on the CIFAR-10 and CIFAR-100 datasets. TAKFL consistently improves all
794 device prototypes of different sizes in various cases by a significant margin compared to the base-
795 lines, achieving SOTA performance. Notably, in the homo-family architecture setting with Dir(0.3)
796 on CIFAR-10, TAKFL improves average performance across all prototypes by 8%, and by 4% on
797 CIFAR-100. In the hetero-family settings with Dir(0.1) on CIFAR-10 and Dir(0.3) on CIFAR-100,
798 TAKFL enhances performance by ∼3% and 1%, respectively. Furthermore, we observe that our
799 self-regularization technique has successfully mitigated issues associated with the noisy and unsu-
800 pervised ensemble distillation process, thereby enhancing performance. Generally, the performance
801 gains from self-regularization are more pronounced in low data heterogeneity cases, where proto-
802 types' models perform better and possess higher quality self-knowledge. Thus, self-regularization
803 proves more effective as it preserves this higher quality self-knowledge.

D.1.1   Consistency Analysis: Experimental and Theoretical Correlations

805 In this part, we elaborate on our key experimental observations and their alignment with our theo-
806 retical findings.

807 **Insight 1:** From Table 4, it is evident that prior KD-based methods show inconsistent performance
808 across various device prototypes, particularly for the large (L) prototype. For instance, in the CIFAR-
809 10 homo-family setting with Dir(0.3), while small (S) and medium (M) prototypes see performance

9

Table 4: **Performance Results for CV task on CIFAR-10 and CIFAR-100.** Training data is distributed among S, M, and L device prototypes in a 1:3:6 ratio, subdivided among clients using Dirichlet distribution. Public datasets are CIFAR-100 for CIFAR-10 and ImageNet-100 for CIFAR-100. Client configurations include 100, 20, and 4 clients for S, M, and L, with sampling rates of 0.1, 0.2, and 0.5. In homo-family settings, architectures are ResNet8, ResNet14, and ResNet18; in hetero-family settings, they are ViT-S, ResNet14, and VGG-16. All models are trained from scratch for 60 rounds. See Appendix F.1 for more details.

| | | Homo-family Architecture Setting | | | | | | | |
| Dataset | Baseline | Low Data Heterogeneity | | | | High Data Heterogeneity | | | |
		S	M	L	Average	S	M	L	Average
CIFAR-10	FedAvg	$36.21_{\pm 2.24}$	$46.41_{\pm 2.33}$	$59.46_{\pm 6.17}$	47.36	$22.01_{\pm 0.78}$	$25.26_{\pm 3.89}$	$51.51_{\pm 3.52}$	32.93
	FedDF	$49.31_{\pm 0.15}$	$50.63_{\pm 0.73}$	$49.82_{\pm 0.98}$	49.92	$34.71_{\pm 1.48}$	$35.27_{\pm 4.74}$	$51.08_{\pm 4.04}$	40.35
	FedET	$49.21_{\pm 0.72}$	$55.01_{\pm 1.81}$	$53.60_{\pm 6.47}$	52.61	$29.58_{\pm 3.00}$	$30.96_{\pm 4.70}$	$45.53_{\pm 6.46}$	35.36
	TAKFL	$55.90_{\pm 1.70}$	$57.93_{\pm 3.49}$	$60.58_{\pm 2.35}$	58.14	$37.40_{\pm 1.68}$	$38.96_{\pm 0.17}$	$51.49_{\pm 6.15}$	42.61
	TAKFL+Reg	$\mathbf{56.37_{\pm 0.46}}$	$\mathbf{58.60_{\pm 0.43}}$	$\mathbf{65.69_{\pm 1.28}}$	**60.22**	$\mathbf{40.51_{\pm 1.05}}$	$\mathbf{40.12_{\pm 1.24}}$	$\mathbf{53.24_{\pm 2.51}}$	**44.62**
CIFAR-100	FedAvg	$13.22_{\pm 0.14}$	$21.39_{\pm 1.11}$	$29.47_{\pm 0.86}$	21.36	$11.86_{\pm 0.08}$	$14.63_{\pm 0.65}$	$26.25_{\pm 1.64}$	17.58
	FedDF	$19.54_{\pm 0.20}$	$24.32_{\pm 0.45}$	$29.29_{\pm 1.45}$	24.38	$16.09_{\pm 0.32}$	$19.80_{\pm 0.17}$	$26.59_{\pm 0.25}$	20.83
	FedET	$19.67_{\pm 0.35}$	$25.27_{\pm 0.66}$	$31.10_{\pm 1.53}$	25.35	$11.18_{\pm 1.68}$	$18.22_{\pm 0.35}$	$26.40_{\pm 0.65}$	18.60
	TAKFL	$24.48_{\pm 0.42}$	$27.60_{\pm 0.25}$	$29.84_{\pm 0.94}$	27.31	$\mathbf{22.90_{\pm 0.18}}$	$23.63_{\pm 0.72}$	$26.98_{\pm 0.13}$	24.50
	TAKFL+Reg	$\mathbf{27.18_{\pm 0.27}}$	$\mathbf{29.14_{\pm 0.20}}$	$\mathbf{31.15_{\pm 0.97}}$	**29.16**	$22.88_{\pm 0.37}$	$\mathbf{23.92_{\pm 0.57}}$	$\mathbf{28.01_{\pm 0.34}}$	**24.94**

| | | Hetero-family Architecture Setting | | | | | | | |
| Dataset | Baseline | Low Data Heterogeneity | | | | High Data Heterogeneity | | | |
		S	M	L	Average	S	M	L	Average
CIFAR-10	FedAvg	$27.53_{\pm 0.83}$	$47.30_{\pm 3.17}$	$55.10_{\pm 8.60}$	43.31	$20.93_{\pm 1.54}$	$25.62_{\pm 6.04}$	$36.80_{\pm 5.47}$	27.78
	FedDF	$34.15_{\pm 0.87}$	$54.06_{\pm 1.06}$	$69.07_{\pm 4.99}$	52.43	$24.20_{\pm 0.74}$	$34.07_{\pm 3.08}$	$39.81_{\pm 5.45}$	32.69
	FedET	$33.24_{\pm 1.27}$	$58.86_{\pm 0.94}$	$65.56_{\pm 3.49}$	52.55	$24.37_{\pm 1.26}$	$37.77_{\pm 4.71}$	$43.64_{\pm 3.36}$	35.26
	TAKFL	$33.29_{\pm 0.15}$	$57.64_{\pm 0.19}$	$68.44_{\pm 0.66}$	53.12	$24.92_{\pm 1.32}$	$38.07_{\pm 3.19}$	$48.01_{\pm 3.99}$	37.00
	TAKFL+Reg	$\mathbf{33.34_{\pm 3.36}}$	$\mathbf{59.01_{\pm 3.12}}$	$\mathbf{70.22_{\pm 4.40}}$	**54.19**	$\mathbf{25.10_{\pm 1.87}}$	$\mathbf{38.81_{\pm 5.36}}$	$\mathbf{50.26_{\pm 6.42}}$	**38.06**
CIFAR-100	FedAvg	$8.51_{\pm 0.37}$	$22.11_{\pm 0.58}$	$37.91_{\pm 2.60}$	22.84	$7.01_{\pm 0.47}$	$14.94_{\pm 0.96}$	$28.51_{\pm 1.46}$	16.82
	FedDF	$10.46_{\pm 0.17}$	$23.46_{\pm 0.65}$	$36.81_{\pm 0.82}$	23.58	$7.76_{\pm 0.40}$	$18.92_{\pm 0.39}$	$29.81_{\pm 1.09}$	18.83
	FedET	$11.16_{\pm 0.18}$	$25.40_{\pm 0.30}$	$37.38_{\pm 0.60}$	24.65	$8.20_{\pm 0.54}$	$20.66_{\pm 0.50}$	$28.95_{\pm 1.79}$	19.27
	TAKFL	$10.29_{\pm 0.11}$	$27.14_{\pm 0.89}$	$\mathbf{39.15_{\pm 0.88}}$	25.53	$7.88_{\pm 0.68}$	$21.41_{\pm 0.37}$	$31.31_{\pm 0.66}$	20.20
	TAKFL+Reg	$\mathbf{11.25_{\pm 0.37}}$	$\mathbf{27.86_{\pm 0.86}}$	$38.68_{\pm 0.45}$	**25.93**	$\mathbf{8.45_{\pm 0.20}}$	$\mathbf{22.16_{\pm 0.87}}$	$\mathbf{31.95_{\pm 1.13}}$	**20.85**

gains, the L prototype experiences up to a ∼10% performance decline compared to vanilla FedAvg, which lacks server-side knowledge distillation. This trend is consistent across other settings, such as CIFAR-10 Dir(0.1) homo-family and CIFAR-100 Dir(0.3) homo-family. These outcomes underline the dilution problem inherent in existing methods, where the valuable insights from larger, more capable device prototypes are overshadowed by less informative outputs from smaller devices, thereby degrading the performance of L prototypes. These empirical findings are supported by our theoretical insights as discussed in Remark 1. Specifically, Proposition 1 illustrates that vanilla ensemble distillation (VED) leads to knowledge dilution and inaccuracies due to misaligned device capacity allocations. Moreover, this issue becomes more significant when the smaller device prototype serve as teacher.

Insight 2: From Table 4, the suboptimality of existing KD-based methods is evident from the significant performance improvements of our method, especially for S and M prototypes across various settings. This underscores the ineffectiveness of the one-size-fits-all approach these methods employ, where a single averaged logits distillation target is used for all device sizes, proving to be suboptimal. Our experimental observations regarding the shortcomings of vanilla ensemble distillation methods align with our theoretical findings, as substantiated in Remark 1 and 2. It becomes evident that an efficient knowledge distillation process must allocate capacity in a manner that appropriately corresponds to the information value of the teacher ensemble prototypes.

Insight 3: Our experiments, detailed in Table 4, demonstrate TAKFL's adept handling of knowledge from various device prototypes under different data heterogeneity conditions. We observed consistent performance gains for small (S) and medium (M) prototypes across both low and high data heterogeneity, compared to vanilla FedAvg. However, in high heterogeneity settings, large (L) prototypes show less improvement, prompting the question: *What can smaller device prototypes offer to larger ones?*

In low heterogeneity scenarios, large prototypes significantly benefit from the collective knowledge, showing enhanced performance. Conversely, in conditions of extreme heterogeneity, where smaller models contribute less effectively, the performance improvements for larger devices are notably reduced. This pattern highlights TAKFL's ability to intelligently manage and utilize the available knowledge, selectively distilling information based on the intrinsic capacity and contributions of

10

Table 5: **Performance Results for CV task on TinyImageNet, STL-10, and CINIC-10 using pre-trained models.**

Private	Public	Baseline	S	M	L	Average
TinyImageNet	STL-10	FedAvg	8.97	13.03	15.12	12.37
		FedMH	15.08	17.10	17.83	16.67
		FedET	10.60	16.39	17.62	14.87
		TAKFL	16.10	17.60	19.03	17.58
		TAKFL+Reg	**16.55**	**17.98**	**19.74**	**18.09**
STL-10	CIFAR-100	FedAvg	26.01	34.47	42.88	34.45
		FedMH	28.64	34.55	39.25	34.15
		FedET	29.87	33.00	38.26	33.71
		TAKFL	29.57	37.57	42.53	36.56
		TAKFL+Reg	**30.78**	**37.89**	**43.38**	**37.35**
CINIC-10	CIFAR-100	FedAvg	44.87	55.49	51.33	50.56
		FedMH	45.52	55.75	53.48	51.58
		FedET	46.31	57.43	53.01	52.25
		TAKFL	**48.21**	**57.81**	52.74	**52.92**
		TAKFL+Reg	47.66	57.54	**53.25**	52.82

each prototype, and integrating only the most valuable knowledge from smaller devices when beneficial.

By contrast, our analysis of existing KD-based methods shows their failure to effectively discern and utilize the most informative knowledge across prototypes. These methods often overload the capacity of larger prototypes with suboptimal or irrelevant information, particularly in high heterogeneity environments, leading to not just stagnation but an accumulation of inefficiencies. These experimental observations align with our theoretical insights, as outlined in Remark 2, which emphasizes the crucial combinatorial constraint of capacity and diverse information. This further confirms the superiority of TAKFL's adaptive approach to knowledge distillation in diverse federated learning environments.

D.2 Additional Experimental Results on CV Task

Experimental Setup. For additional evaluation of the CV task, we conducted experiments on TinyImageNet [25], STL-10 [7], and CINIC-10 [9] datasets using pre-trained models. For TinyImageNet [25], we utilized STL-10 [7] as the unlabeled public dataset. STL-10 [7] and CINIC-10 [9] both employ CIFAR-100 [24] as their respective public datasets. We distributed the training datasets among the device prototypes in a 2:3:5 ratio for small (S), medium (M), and large (L) prototypes, respectively. The data portion for each prototype was further subdivided among its clients using a Dirichlet distribution: Dir(1.0) for TinyImageNet and Dir(0.3) for both STL-10 and CINIC-10. Client configurations were set with 4, 3, and 2 clients for S, M, and L, respectively, all with a sampling rate of 1.0. The architectures employed were MobileNetV3-Large [19] for S, Mobile-ViTV2 [34] for M, and ResNet-34 for L, sourced from the TIMM library.[1] The local training was conducted over 10 epochs using an Adam [23] optimizer with a learning rate of 1e-3 and weight decay of 1e-5. For server-side distillation, the epoch count was 10 for TinyImageNet and 1 for STL-10 and CINIC-10, with a batch size of 128, employing an Adam optimizer with a learning rate of 1e-5 and weight decay of 1e-5. For TinyImageNet, public dataset images from STL-10 were resized to 64×64, while for STL-10, images were resized to 32×32. No data augmentation was used. The communication rounds is fixed to 40. These experiments were conducted by only 1 trial. Table 14 details the configurations.

Performance Results. Table 5 presents the results. The superiority of TAKFL's performance across these challenging datasets using pre-trained models is evident here as well.

D.3 Additional Experimental Results on Natural Language Processing (NLP) Task

The experiments in this section complements the main experimental results in the main paper Section 7.2.

---

[1] https://github.com/huggingface/pytorch-image-models

11

Table 6: **Performance Results for NLP Task on 4 Datasets.**. Training data is distributed among S, M, and L device prototypes in a 1:3:6 ratio, subdivided among clients using Dir(0.5). Client configurations are 8, 4, and 2 clients for S, M, and L, with sample rates of 0.3, 0.5, and 1.0, respectively. Architectures include Bert-Tiny, Bert-Mini, and Bert-Small for S, M, and L, initialized from pre-trained parameters and fine-tuned for 20 communication rounds. See Appendix F.2 for more details.

Private	Public	Baseline	S	M	L	Average
MNLI	SNLI	FedAvg	$36.15_{\pm0.46}$	$54.47_{\pm2.48}$	$57.51_{\pm2.79}$	49.37
		FedDF	$54.21_{\pm0.15}$	$60.44_{\pm1.91}$	$66.71_{\pm1.09}$	60.45
		FedET	$48.03_{\pm6.32}$	$50.33_{\pm7.87}$	$53.80_{\pm6.18}$	50.72
		TAKFL	$57.43_{\pm0.21}$	$63.58_{\pm0.31}$	$68.74_{\pm0.12}$	63.25
		TAKFL+Reg	$\mathbf{57.61}_{\pm0.89}$	$\mathbf{63.91}_{\pm1.05}$	$\mathbf{68.96}_{\pm1.10}$	**63.49**
SST2	Sent140	FedAvg	$54.98_{\pm1.81}$	$74.71_{\pm8.22}$	$86.69_{\pm0.06}$	72.13
		FedDF	$74.41_{\pm2.62}$	$80.71_{\pm1.63}$	$84.35_{\pm1.66}$	79.82
		FedET	$66.63_{\pm9.14}$	$65.89_{\pm16.35}$	$70.05_{\pm15.83}$	67.52
		TAKFL	$74.73_{\pm0.55}$	$82.17_{\pm0.31}$	$86.93_{\pm0.42}$	81.28
		TAKFL+Reg	$\mathbf{74.88}_{\pm0.43}$	$\mathbf{82.40}_{\pm0.83}$	$\mathbf{87.33}_{\pm0.63}$	**81.54**
MARC	Yelp	FedAvg	$33.76_{\pm1.13}$	$49.08_{\pm1.28}$	$59.26_{\pm1.43}$	47.36
		FedDF	$53.01_{\pm1.24}$	$55.37_{\pm0.87}$	$56.81_{\pm0.99}$	55.06
		FedET	$52.63_{\pm2.29}$	$54.28_{\pm2.31}$	$56.11_{\pm2.61}$	54.34
		TAKFL	$55.70_{\pm2.08}$	$58.64_{\pm1.75}$	$59.39_{\pm1.16}$	57.91
		TAKFL+Reg	$\mathbf{55.96}_{\pm1.66}$	$\mathbf{59.18}_{\pm1.13}$	$\mathbf{59.61}_{\pm1.89}$	**58.25**
AG-News	DBPedia	FedAvg	$83.64_{\pm3.51}$	$83.47_{\pm2.35}$	$91.48_{\pm2.22}$	86.20
		FedDF	$85.97_{\pm2.45}$	$89.10_{\pm1.85}$	$91.37_{\pm1.10}$	88.81
		FedET	$75.27_{\pm3.85}$	$81.13_{\pm3.21}$	$83.19_{\pm4.58}$	79.86
		TAKFL	$87.37_{\pm1.31}$	$90.11_{\pm1.56}$	$92.48_{\pm1.12}$	89.99
		TAKFL+Reg	$\mathbf{87.66}_{\pm1.83}$	$\mathbf{90.30}_{\pm2.05}$	$\mathbf{92.61}_{\pm1.72}$	**90.19**

Experimental Setup. For the evaluation of NLP tasks, we utilize four datasets: MNLI [50], SST-2 [43], MARC [22], and AG-news [58]. The corresponding unlabeled public datasets are SNLI [2] for MNLI, Sentiment140 [14] for SST-2, Yelp [59] for Amazon, and DBPedia [57] for AG-News. The training data is distributed among the device prototypes in a ratio of 1:3:6 for small (S), medium (M), and large (L) categories, respectively, with each portion further subdivided among its clients using a Dirichlet distribution (Dir(0.5)). The client configurations and their sampling rates are set as follows: 8, 4, and 2 clients for S, M, and L categories, respectively, with sampling rates of 0.3, 0.5, and 1.0. The architectures employed for each prototype size are BERT [45] -Tiny, -Small, and -Mini, respectively, each initialized from pre-trained parameters and tested over 20 communication rounds. Additional details regarding hyper-parameters and datasets are presented in Appendix F.2 and Table 16.

Performance on NLP Task. Table 6 presents the results on four different datasets: MNLI, SST-2, MARC, and AG-News. Similar to the CV task, TAKFL has consistently improved performance across all device prototypes of varying sizes, achieving state-of-the-art results. On MNLI, it has enhanced average performance across all prototypes by 3%, on SST-2 by ∼2%, on MARC by 3%, and on AG-News by ∼1.50%. As observed in the CV task, the suboptimality of existing KD-based methods is also evident here. Notably, FedET exhibits very poor performance compared vanilla FedAvg, failing to achieve satisfactory results on all datasets except for the MARC dataset. Particularly, the performance of the L prototype has consistently decreased across all datasets compared to vanilla FedAvg. This behavior can be attributed to FedET's reliance on neural network confidence scores for uncertainty estimates in its uncertainty-weighted distillation. However, neural networks, especially pretrained language models (PLMs), are often poorly calibrated and prone to overconfidence, which compromises their ability to provide reliable uncertainty estimates [48, 15, 5, 53].

D.4 Scalability Evaluation

This section complements the experimental results in the main paper Section 7.3.

Experimental Setup. To evaluate the effectiveness and scalability of our method across a broad spectrum of device prototypes, ranging from very small to very large sizes, we conduct experiments involving 3 to 7 different prototypes. Our objective is to assess how effectively our method adapts from a uniform array of small-size prototypes (3 device prototypes) to a diverse mix that

12

includes prototypes ranging from extremely small (XXS) to extremely large (XXL) (7 device proto-types). These experiments involve training image classification models from scratch on the CINIC-10 dataset, using CIFAR-100 as the unlabeled public dataset. We randomly distribute the dataset among prototypes with dataset ratios set to 1:2:3:4:5:6:7 from XXS to XXL. Each dataset portion is further distributed among clients using a Dirichlet distribution (Dir(0.5)). The number of clients ranges from 35 to 5 from XXS to XXL, respectively. Client sample rates are set at 0.1, 0.1, 0.15, 0.15, 0.2, 0.3, and 0.6 from XXS to XXL. We use a series of ResNet architectures—ResNet10-XXS, ResNet10-XS, ResNet10-S, ResNet10-M, ResNet10, ResNet18, and ResNet50—scaled appropri-ately for each prototype. The local training epochs are set at 2, 2, 2, 5, 10, 10, and 20 from XXS to XXL to account for resource constraints, with fewer epochs assigned to smaller devices. We employ the Adam optimizer with a learning rate of 1e-3 and a weight decay of 5e-5 for local training. For XL and XXL, a step learning rate scheduler reduces the learning rate by a factor of 0.1 at half epoch. Server-side distillation employs a fixed batch size of 128, using the Adam optimizer with learning rate of 1e-3 and weight decay of 5e-5. The softmax temperature is set at 3 for ensemble distillation and 20 for self-regularization. The number of communication rounds is fixed at 30. These exper-iments are conducted over 3 trials with different random seeds, and the average performance with standard deviation is reported. The entire device prototypes configurations are given in Table 15.

The detailed results are presented in Tables 7, 8, and 9.

Table 7: **Scalability Evaluation.** Detailed performance results for 7 device prototypes case.

Baseline	XXS	XS	S	M	L	XL	XXL	Average
FedAvg	$23.17_{\pm1.26}$	$30.66_{\pm0.14}$	$32.81_{\pm0.21}$	$31.77_{\pm0.21}$	$37.69_{\pm0.08}$	$41.78_{\pm0.05}$	$50.52_{\pm0.01}$	35.49
FedDF	$27.98_{\pm0.66}$	$\mathbf{37.47_{\pm0.33}}$	$40.61_{\pm0.01}$	$40.26_{\pm0.18}$	$43.83_{\pm0.22}$	$45.58_{\pm0.18}$	$52.18_{\pm0.12}$	41.13
FedET	$26.75_{\pm0.98}$	$36.99_{\pm0.31}$	$40.51_{\pm0.19}$	$41.60_{\pm0.16}$	$46.12_{\pm0.31}$	$48.39_{\pm0.11}$	$52.71_{\pm0.09}$	41.87
TAKFL	$27.30_{\pm0.08}$	$36.93_{\pm0.16}$	$43.31_{\pm0.42}$	$40.88_{\pm0.01}$	$48.52_{\pm0.15}$	$50.95_{\pm0.04}$	$54.27_{\pm0.43}$	43.17
TAKFL+Reg	$\mathbf{29.28_{\pm0.16}}$	$37.10_{\pm0.45}$	$\mathbf{43.96_{\pm1.65}}$	$\mathbf{41.83_{\pm0.73}}$	$\mathbf{48.77_{\pm0.37}}$	$\mathbf{51.43_{\pm0.46}}$	$\mathbf{54.63_{\pm0.84}}$	**43.86**

Table 8: **Scalability Evaluation.** Detailed performance results for 5 device prototypes case.

Baseline	XXS	XS	S	M	XL	Average
FedAvg	$24.19_{\pm1.03}$	$21.04_{\pm0.76}$	$33.62_{\pm0.88}$	$38.91_{\pm0.74}$	$46.93_{\pm0.05}$	32.94
FedDF	$\mathbf{28.31_{\pm0.61}}$	$34.66_{\pm0.00}$	$39.91_{\pm0.07}$	$38.24_{\pm0.36}$	$46.81_{\pm0.11}$	37.59
FedET	$26.88_{\pm0.95}$	$34.11_{\pm0.27}$	$\mathbf{41.15_{\pm0.29}}$	$40.81_{\pm0.87}$	$48.14_{\pm0.06}$	38.22
TAKFL	$27.91_{\pm0.12}$	$37.09_{\pm0.11}$	$40.46_{\pm0.34}$	$41.06_{\pm0.02}$	$49.02_{\pm0.35}$	39.11
TAKFL+Reg	$28.24_{\pm0.46}$	$\mathbf{37.30_{\pm1.10}}$	$40.76_{\pm0.94}$	$\mathbf{43.09_{\pm0.27}}$	$\mathbf{50.86_{\pm0.22}}$	**40.05**

Table 9: **Scalability Evaluation.** Detailed performance results for 3 device prototypes case.

Baseline	XXS	S	M	Average
FedAvg	$24.19_{\pm1.03}$	$33.62_{\pm0.88}$	$\mathbf{38.91_{\pm0.74}}$	32.24
FedDF	$27.85_{\pm0.10}$	$\mathbf{37.83_{\pm0.12}}$	$37.74_{\pm0.41}$	34.47
FedET	$26.04_{\pm0.67}$	$36.87_{\pm0.68}$	$37.66_{\pm0.09}$	33.52
TAKFL	$26.62_{\pm0.16}$	$37.32_{\pm0.40}$	$38.13_{\pm0.58}$	34.02
TAKFL+Reg	$\mathbf{27.90_{\pm0.98}}$	$37.63_{\pm0.87}$	$38.20_{\pm0.91}$	**34.58**

E   Ablation Studies

E.1   Understanding Merging Coefficient

In this section, we conduct an ablation study to further understand how TAKFL customizes knowl-edge integration and understand how the merging coefficients λ_i are achieving this. This experiment aims to further understand the trade-offs between customized knowledge integration approach from the one-size-fits-all strategy employed in vanilla ensemble distillation and prior works.

Experimental Setup. Our experimentation focuses on two device prototypes: XXS and XXL, se-lected from the scalability evaluation detailed in Section 7.3, Appendix D.4, and Table 15. We employ the image classification task on the CINIC-10 [9] dataset, starting from scratch. Each proto-type receives a randomly selected, non-overlapping subset of the training dataset—3.57% for XXS and 25% for XXL—distributed among their clients in a non-i.i.d. manner using Dir(0.5). Both pro-totypes have three clients each. The architectures used are ResNet10-XXS for the XXS prototype and ResNet-50 for the XXL prototype. To focus solely on the evaulation of the server-side distil-lation process and its evolution with varying λ, we pre-train each prototype using standard FedAvg

13

for 10 communication rounds, with a sample rate of 1.0. Local training involves 20 epochs for XXS and 20 epochs for XXL using an Adam optimizer with a learning rate of 1e-3 and weight decay of 5e-5. The XXL prototype employs a step learning rate scheduler that reduces the rate by a factor of 0.1 at local epoch 10. For server-side distillation, we utilize a batch size of 128 and an Adam optimizer with a learning rate of 1e-5 and weight decay of 5e-5. CIFAR-100 [24] serves as the unlabeled public dataset. We save the final updated client and server models from both prototypes for further experimentation, focusing on the impact of merging coefficients without self-regularization in TAKFL. The merging coefficient λ is varied linearly from 0 to 1 in increments of 0.05. For simplicity, the XXS prototype is referred to as the small (S) prototype and XXL as the large (L) prototype.

Discussion. Figure 6 illustrates the significant impact of customized knowledge integration on the performance of both small and large device prototypes compared to the one-size-fits-all approach typical of vanilla ensemble distillation in the prior works, at different distillation epochs. Here, TAKFL adeptly manages customization for both small and large prototypes by controlling the merging coefficient λ. The merged model for both the small and large student prototypes is obtained using the formula $\boldsymbol{\theta}_{merged} = \boldsymbol{\theta}_{avg} + \big((1 - \lambda)\boldsymbol{\tau}_S + \lambda\boldsymbol{\tau}_L\big)$. Notably, the performance is benchmarked at $\lambda \approx 0.5$ in all cases, reflecting similar results to vanilla ensemble distillation (FedDF), where no customization in knowledge transfer occurs. This baseline performance is critical for understanding the effects of further customization.

In small distillation epochs ($I_{distill} < 10$), minimal benefit is observed from customized knowledge integration, as both small and large prototypes achieve optimal performance at the non-customized $\lambda \approx 0.5$. However, as the distillation process progresses beyond 10 epochs, the influence of λ becomes increasingly pronounced. For $\lambda > 0.5$, the knowledge from the large prototype's ensembles predominates, enhancing their impact, while for $\lambda < 0.5$, integration is more influenced by the small prototype's ensembles. This pattern suggests that increased distillation epochs enable more effective distillation of each prototype's unique knowledge for extreme cases of extremely small and large prototypes, thereby making the customization benefits evident. In scenarios with small distillation epochs, the absence of significant unique knowledge results in optimal performance at $\lambda \approx 0.5$. Conversely, as the number of distillation epochs rises ($I_{distill} \geq 20$), the one-size-fits-all strategy proves suboptimal, underscoring the importance of tailored knowledge integration strategies. Optimal performance increasingly occurs at $\lambda > 0.5$, indicating effective leveraging of each prototype's strengths to maximize overall performance. These findings confirm the necessity for customized knowledge integration in environments with significant prototype size variations and support our theoretical insights as detailed in Remark 1 and 2.

E.2 Impact of Public Dataset

In this section, we explore the influence of the public dataset on the performance of TAKFL and existing KD-based methods when the public dataset used for server-side distillation is less similar to the private dataset, which is the actual learning objective. For this analysis we employ the same experimental setup previously outlined in Section 7.2 and Appendix D.1, using the CIFAR-10 homo-family architecture. To measure dataset similarity, we compute cosine similarity between the averaged features of datasets, extracted using an off-the-shelf pre-trained CLIP model [40] (CLIP ViT-B/32) available from the official GitHub repository.[2]

Discussion. Table 10 presents our results, highlighting a significant observation: the performance of existing methods drastically deteriorates as the similarity between the public dataset and private datasets decreases. In contrast, TAKFL exhibits robustness, suffering much less performance degradation under the same conditions. This demonstrates TAKFL's practical utility in real-world scenarios where the server typically lacks knowledge of the private datasets to select a closely aligned public dataset for distillation. Notably, FedET underperforms significantly when using a less similar public dataset, performing worse than vanilla FedAvg in both low and high data heterogeneity scenarios. A similar pattern was observed with FedET in the NLP tasks discussed in Section 7.2 and Appendix D.3. This issue is likely due to FedET's dependence on the overconfident and poorly calibrated confidence scores from neural networks [15, 53] for uncertainty estimates in its uncertainty-weighted distillation approach.

---

[2] https://github.com/OpenAI/CLIP

14
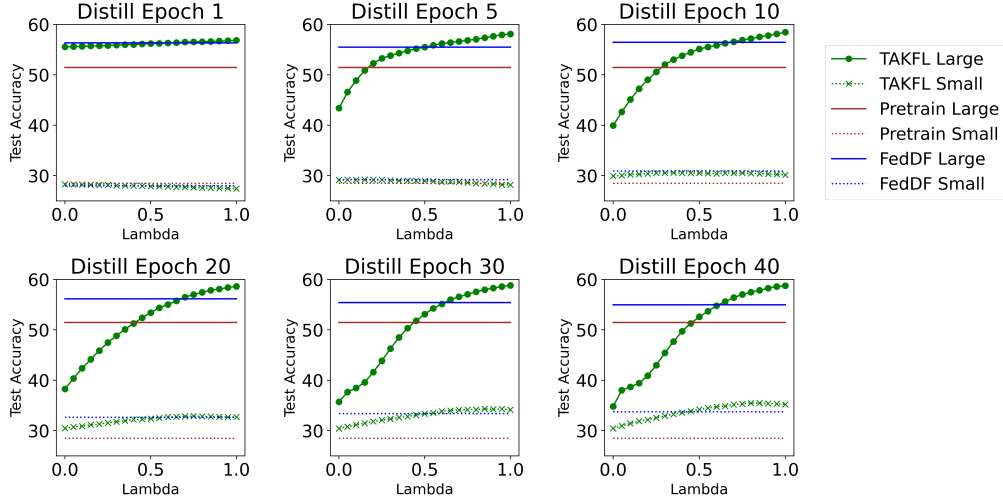
Figure 6: **Understanding the Impact of Merging Coefficients.** This figure showcases server-side knowledge distillation between two device prototypes, XXS and XXL, referred to as small and large, respectively, utilizing CIFAR-100 as the unlabeled public dataset. Both prototypes were pre-trained from scratch using standard FedAvg for 10 communication rounds. The CINIC-10 dataset was distributed between the small and large prototypes in ratios of 3.57% and 25%, respectively, and further subdivided non-i.i.d. among the clients using Dir(0.5). Each prototype has three clients with a sample rate of 1.0. The small prototype utilizes a ResNet10-XXS architecture, while the large prototype employs a ResNet-50.

Table 10: **Impact of Public Dataset on performance results.** Same experimental setting described in Section 7.2 and Appendix D.1 on CIFAR-10 homo-family setting is used for this experiment. The numbers in parentheses represent the similarity scores between private and public datasets, obtained using a pre-trained CLIP ViT-B/32 model.

Public Dataset	Baseline	Low Data Heterogeneity (Dir(0.3))				High Data Heterogeneity (Dir(0.1))			
		S	M	L	Average	S	M	L	Average
—	FedAvg	$36.21_{\pm2.24}$	$46.41_{\pm2.33}$	$59.46_{\pm6.17}$	47.36	$22.01_{\pm0.78}$	$25.26_{\pm3.89}$	$51.51_{\pm3.52}$	32.93
CIFAR-100 (0.99)	FedDF	$49.31_{\pm0.15}$	$50.63_{\pm0.73}$	$49.82_{\pm0.98}$	49.92	$34.71_{\pm1.48}$	$35.27_{\pm4.74}$	$51.08_{\pm4.04}$	40.35
	FedET	$49.21_{\pm0.72}$	$55.01_{\pm1.81}$	$53.60_{\pm6.47}$	52.61	$29.58_{\pm3.00}$	$30.96_{\pm4.70}$	$45.53_{\pm6.46}$	35.36
	TAKFL	$55.90_{\pm1.70}$	$57.93_{\pm3.49}$	$60.58_{\pm2.35}$	58.14	$37.40_{\pm1.68}$	$38.96_{\pm0.17}$	$51.49_{\pm6.15}$	42.62
	TAKFL+Reg	$\mathbf{56.37}_{\pm0.46}$	$\mathbf{58.60}_{\pm0.43}$	$\mathbf{65.69}_{\pm1.28}$	**60.22**	$\mathbf{40.51}_{\pm1.05}$	$\mathbf{40.12}_{\pm1.24}$	$\mathbf{53.24}_{\pm2.51}$	**44.62**
TinyImagenet (0.92)	FedDF	$49.37_{\pm1.58}$	$49.41_{\pm4.21}$	$55.06_{\pm6.71}$	51.28	$31.41_{\pm6.61}$	$30.73_{\pm7.77}$	$39.82_{\pm5.16}$	33.99
	FedET	$33.95_{\pm0.92}$	$37.26_{\pm1.64}$	$39.77_{\pm3.44}$	36.99	$24.12_{\pm1.84}$	$24.58_{\pm2.13}$	$28.91_{\pm1.09}$	25.87
	TAKFL	$55.20_{\pm0.07}$	$56.36_{\pm0.40}$	$60.71_{\pm0.22}$	57.42	$40.08_{\pm0.19}$	$40.26_{\pm0.04}$	$43.56_{\pm1.10}$	41.30
	TAKFL+Reg	$\mathbf{56.28}_{\pm0.09}$	$\mathbf{57.14}_{\pm0.03}$	$\mathbf{60.90}_{\pm0.22}$	**58.11**	$\mathbf{40.88}_{\pm0.11}$	$\mathbf{41.10}_{\pm1.15}$	$\mathbf{46.25}_{\pm5.95}$	**42.74**
Celeb-A (0.77)	FedDF	$\mathbf{48.99}_{\pm0.37}$	$50.06_{\pm0.43}$	$55.12_{\pm4.95}$	51.39	$29.80_{\pm0.39}$	$32.28_{\pm4.41}$	$44.0_{\pm4.60}$	35.36
	FedET	$28.56_{\pm3.00}$	$28.80_{\pm1.00}$	$37.20_{\pm2.78}$	31.52	$15.28_{\pm1.75}$	$19.00_{\pm3.43}$	$23.29_{\pm5.04}$	19.19
	TAKFL	$45.65_{\pm2.72}$	$54.53_{\pm1.72}$	$58.13_{\pm0.13}$	52.77	$\mathbf{31.02}_{\pm0.68}$	$\mathbf{36.76}_{\pm1.58}$	$48.33_{\pm0.53}$	38.70
	TAKFL+Reg	$46.93_{\pm0.67}$	$\mathbf{56.67}_{\pm1.26}$	$\mathbf{60.13}_{\pm1.38}$	**54.58**	$30.88_{\pm3.51}$	$35.95_{\pm5.40}$	$\mathbf{52.68}_{\pm1.90}$	**39.84**

15

F Hyper-parameters and Implementation

In this section we bring the details of the hyper-parameters we used and our implementation. We implement our entire code in PyTorch [38] and release it anonymously at `https://anonymous.4open.science/r/TAKFL-DD28/README.md`. We use two NVIDIA RTX 3090 gpus to conduct the entire experimentation in this paper.

F.1 Computer Vision (CV) Task

For comprehensive evaluation of our method, we consider federated learning image classification training from scratch.

F.1.1 Datasets

Datasets. We experiment with several image classification datasets: CIFAR-10, CIFAR-100, CINIC-10, TinyImageNet, STL-10. The details of each dataset is the following:

- **CIFAR-10** [24]: CIFAR-10 consists of 60,000 images of size 32×32 RGB across 10 classes, with each class containing 6,000 images.
- **CIFAR-100** [24]: CIFAR-100 comprises 60,000 images of size 32×32 RGB distributed across 100 classes, with 500 images per class.
- **CINIC-10** [9]: CINIC-10 has 270,000 images of size 32×32 RGB across 10 classes, each class containing 27,000 images.
- **TinyImageNet** [25]: TinyImageNet contains 100,000 images of size 64×64 RGB across 200 classes.
- **STL-10** [7]: STL-10 has 100,000 unlabeled images and 13,000 labeled images of size 96×96 RGB across 10 classes.

The ImageNet-100 as the unlabeled public dataset is constructed by randomly selecting 100 classes from the ImageNet [10] dataset.

F.1.2 Architectures

Experiments in Section 7.2 and Appendix D.1. The architecture that we use are two distinct architectural settings: the *"homo-family"* setting, where all device prototypes' architectures are from the same family, and the *"hetero-family"* setting, where architectures do not necessarily belong to the same family. For the homo-family scenario, we employ ResNet-8 for S, ResNet-14 for M, and ResNet-18 for L. For the hetero-family scenario, we use ViT-S for S, ResNet-14 for M, and VGG-16 for L. All models are initialized from random initialization.

For the ResNet architecture configuration, we utilize the standard 'BasicBlock' as the building block. This consists of a convolutional block, followed by four residual block stages, an adaptive average pooling layer, and a classifier layer. The models within this family differ in terms of the number of repetitions of the residual block and the number of filters in each stage. The configurations for different capacities are detailed below:

- *ResNet-18* is configured with [64, 128, 256, 512] filters and repeats the 'BasicBlock' [2, 2, 2, 2] times.
- *ResNet-14* is configured with [64, 128, 256, 512] filters and repeats the 'BasicBlock' [1, 2, 2, 1] times.
- *ResNet-8* is configured with [64, 128, 256] filters, corresponding to the first three stages, and repeats the 'BasicBlock' [1, 1, 1] times.

For VGG-16 [42], we use the standard architecture which includes convolutional layers followed by max-pooling layers. The configuration of filters for each layer is as follows:

VGG-16: [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 'M', 512, 512, 512, 'M', 512, 512, 512, 'M']

16

The final classification head consists of two linear layers with a hidden size of 512, followed by a ReLU activation, and a final linear classifier layer.

For ViT-S, we adopt the standard Vision Transformer [12] architecture implementation from Github.[3] We configure ViT-S with 6 attention blocks, each with 16 heads and a hidden dimension of 64. The final MLP dimension is set to 256. In our experiments with ViT-S, we set the patch size to 4, and the input image size is 32×32.

Experiment in Appendix D.2. The pre-trained MobileNetV3-Large [19], MobileViTV2 [34], and ResNet34 [49] were instantiated using the TIMM library.[4]

Scalability Experiments in Section 7.3. The architectures in these experiments are inspired by [20]. The details of architectures are as following:

- *ResNet10-XXS* is configured with [8, 8, 16, 16] filters and repeats the 'BasicBlock' [1, 1, 1, 1] times.
- *ResNet10-XS* is configured with [8, 16, 32, 64] filters and repeats the 'BasicBlock' [1, 1, 1, 1] times.
- *ResNet10-S* is configured with [16, 32, 64, 128] filters and repeats the 'BasicBlock' [1, 1, 1, 1] times.
- *ResNet10-M* is configured with [8, 16, 32, 64] filters and repeats the 'BasicBlock' [1, 1, 1, 1] times.
- *ResNet10* is configured with [64, 128, 256, 512] filters and repeats the 'BasicBlock' [1, 1, 1, 1] times.
- *ResNet18* is configured with [64, 128, 256, 512] filters and repeats the 'BasicBlock' [1, 1, 1, 1] times.
- *ResNet50* is configured with [64, 128, 256, 512] filters and repeats the 'BasicBlock' [3,4,6,3] times.

F.1.3 FL configuration and Hyper-parameters

Base Hyper-parameters. The following hyperparameter values apply to all CV experiments unless stated otherwise. We set the diversity regularizer coefficient of FedET to 0.1 for our entire experimentation per the original paper [6]. We use the Adam optimizer with a learning rate of 1e-5, weight decay value of 5e-5, and a batch size of 128 for distillation. The softmax distillation temperature is set to 3, the distillation epoch to 1, and the self-regularizer softmax temperature to 20 for both CV and NLP experiments. Table 12 details the hyper-parameters.

Experiments in Section 7.2 and Appendix D.1. For tables 1 and 4, there are 100 clients with the S device prototype, 20 clients with the M device prototype, and 4 clients with the L device prototype. For each round, 10, 4, and 2 clients from each S, M, and L prototype are randomly sampled respectively for participation. 10% of the data goes to the S prototype, 30% to M, and 60% to L. The data is distributed to each client among each prototype in a Dirichlet distribution. Table 13 details the FL configuration.

Experiments in Appendix D.2 and Appendix E.2. For tables 5 and 10, there are 4, 3, and 2 clients for S, M, and L device prototypes, respectively. Each round, every client participates in FL. 20% of the data is distributed to prototype S, 30% to prototype M, and 50% to prototype L. Table 14 details the FL configuration.

Scalability Experiments in Section 7.3 and D.4. For tables 7, 8, and 9, there are 35, 30, 25, 20, 15, 10, and 5 clients for the prototypes XXS, XS, S, M, L, XL, and XXL, respectively. The sample rate is set to 0.1, 0.1, 0.15, 0.15, 0.2, 0.3, and 0.6 from XXS to XXL. The data is distributed for each prototype in the ratio 1:2:3:4:5:6:7 from XXS to XXL. Table 15 details the hyper-parameters and configuration.

Validation Set. For TAKFL, the validation set used for the heuristic method (see F.3) is 5% of the training dataset. The validation set and the private dataset does not overlap.

---

[3] https://github.com/lucidrains/vit-pytorch
[4] https://github.com/huggingface/pytorch-image-models

17

F.2 Natural Language Processing (NLP) Task

For the NLP task, we fine-tune federated learning text classification task using pretrained models.

F.2.1 Datasets.

All NLP datasets were provided by Hugging Face. [5]

- **MNLI** [50]: MNLI contains 433K sentence pairs, each sentence pair labeled as one of 'entailment,' 'neutral,' and 'contradiction.'

- **SNLI** [2]: SNLI is similar to MNLI, with 570K sentence pairs each labeled one of 3 labels.

- **SST2** [43]: SST2 consists of 67K phrases, each labeled as sentiment 'positive' or 'negative.'

- **Sentiment140** [14]: Sentiment140 is a dataset of 1.6M Twitter messages each labeled with one of 2 sentiment values.

- **MARC** [22]: MARC (Multilingual Amazon Reviews Corpus) is a dataset with online reviews in multiple languages from the Amazon delivery service website. Each review has a label which is one of 1-5 stars. We only use the English reviews from this dataset, which results in 260,000 English reviews total.

- **Yelp** [59]: The Yelp reviews dataset contains 700K reviews each labeled 1-5 stars from the Yelp service which publishes public reviews of businesses.

- **AG News** [58]: AG News contains 127,600 news article titles. Each article is one of four classifications of news articles.

- **DBpedia** [57]: The DBpedia dataset consists of 630K DBpedia article summaries each labeled one of 14 categorizations.

F.2.2 Architectures

Experiments in Section 7.2 and Appendix D.3. We use three variations of the BERT architecture: BERT-Tiny, BERT-Mini, and BERT-Small from [45]. The weights were pre-trained on the BookCorpus dataset and extracted text from Wikipedia. Further details regarding each model are described extensively on Github.[6] The tokenizer used for these transformer models are the same ones provided by the authors of [45].

- *BERT-Tiny* contains 2 transformer layers and an embedding size of 128.
- *BERT-Mini* contains 4 transformer layers and an embedding size of 256.
- *BERT-Small* contains 4 transformer layers and and embedding size of 512.

F.2.3 FL configuration and hyper-parameters

Base Hyper-parameters. For distillation, we use the Adam optimizer with a learning rate of 3e-5, no weight decay, and batch size of 32. The distillation epoch is set to 1, the ensemble distillation softmax temperature to 3, and the self-regularizer softmax temperature to 20 for all NLP experiments. Table 16 details the hyper-parameters.

Experiments in Section 7.2 and Appendix D.3. For tables 2 and 6, we limit the private dataset to 100,000 samples, randomly sampled from the original dataset i.i.d. The public dataset is limited to 30,000 examples sampled i.i.d as well. There are 8, 4, and 2 clients for the S, M, and L prototypes. The private data is split across each prototype in the following proportions: 0.1, 0.3, 0.6. Table 17 details the FL configuration.

Validation Set. The validation dataset used for TAKFL is 5,000 samples taken from the original training dataset that does not overlap with the 100,000 private dataset.

---

[5]https://github.com/huggingface/datasets
[6]https://github.com/google-research/bert

18

F.3 Hyper-parameters of TAKFL

Merging Coefficients. We conducted extensive experiments with different merging coefficients on the main 3-device prototype setting of small (S), medium (M), and large (L) discussed in Section 7.2 and Appendix D.1. We empirically observed that the small (S) prototype typically achieves the best performance using a uniformly increasing merging coefficient, where the larger the prototype, the larger the merging coefficient, i.e., $\lambda_S \leq \lambda_M \leq \lambda_L$. As we move towards larger prototypes, they benefit more from increasingly skewed merging coefficients towards the larger ones. In the extreme case of the large (L) prototype, highly skewed merging coefficients generally led to better performance, i.e., $\lambda_S \ll \lambda_M \ll \lambda_L$. This pattern is intuitive as small prototypes can benefit from everyone while gaining more from the larger, more informative prototypes. However, larger prototypes benefit less from smaller ones, as they typically offer less information, especially in high data heterogeneity cases. Notably, in high data heterogeneity cases, more skewed merging coefficients seemed to be more advantageous as the smaller prototypes (S and M) possess lower quality knowledge.

Based on these observations, we designed a simple and cost-effective heuristic method that randomly instantiates merging coefficients following this intuition. Our heuristic method, presented in 1, leverages these observations by generating candidate merging coefficients that incorporate both uniformly increasing and different degrees of skewed merging coefficients. This dual approach enables us to explore a wide range of merging strategies and identify the most effective configurations for different prototypes. The optimal merging coefficient candidate is determined using the performance on the held-out validation set.

```
1  import numpy as np
2  def heuristic(num_devices=3, n_candidates=10):
3      candidates = [[1/num_devices for _ in range(num_devices)]]
4      for exponent in [1, 5, 10]:
5          for i in range(n_candidates):
6              candidate = np.random.beta(a=1, b=100, size=num_devices)
7              candidate = candidate ** exponent
8              candidate = np.sort(candidate)
9              candidate = candidate / np.sum(candidate)
10             candidates.append(candidate)
11     return candidates
```

Listing 1: Implementation of the heuristic method for merging coefficients in Python. The exponent term controls the degree of skewness or peaking in the merging coefficients.

Furthermore, we experiment with manually determining the merging coefficients and fixating them throughout the federation. We achieved similar results with this approach compared to adaptively finding the coefficients using the heuristic method and a small held-out validation set. We present the merging coefficient candidates that performed reasonably well during our experiments in Table 11.

Table 11: Details of the experimentally determined merging coefficients for the 3-device prototype setting discussed in Section 7.2 and Appendix D.1. Coefficients are ordered as $[\lambda_S, \lambda_M, \lambda_L]$.

Merging Coefficient Candidate	Small Prototype	Medium Prototype	Large Prototype
1	[0.2, 0.3, 0.5]	[0.1, 0.2, 0.7]	[0.1, 0.2, 0.7]
2	[0.3, 0.3, 0.4]	[0.05, 0.15, 0.8]	[0.01, 0.09, 0.99]
3	[0.2, 0.3, 0.5]	[0.1, 0.2, 0.7]	[0.05, 0.2, 0.75]
4	[0.05, 0.1, 0.85]	[0.01, 0.19, 0.8]	[0.01, 0.09, 0.90]
5	[0.1, 0.15, 0.75]	[0.05, 0.15, 0.8]	[0.01, 0.09, 0.90]
6	[0.05, 0.1, 0.85]	[0.05, 0.05, 0.9]	[0.001, 0.009, 0.99]
7	[0.05, 0.15, 0.80]	[0.05, 0.2, 0.75]	[0.001, 0.009, 0.99]
8	[0.05, 0.15, 0.80]	[0.05, 0.1, 0.85]	[0.001, 0.009, 0.99]
9	[0.3, 0.35, 0.35]	[0.2, 0.3, 0.5]	[0.1, 0.2, 0.7]

Self-Regularization Coefficient. Extensive experiments were conducted on the self-regulation coefficients for different device prototypes and settings. Although no consistent pattern emerged, we experimentally determined that optimal performance for the small prototype was achieved with self-

19

regulation coefficients $\gamma_S \in 0.1, 0.01, 0.001$. For the medium prototype, the coefficients were $\gamma_M \in 0.5, 0.1, 0.01, 0.001, 0.0001$, and for the large prototype, $\gamma_L \in 1.0, 0.8, 0.5, 0.1, 0.01, 0.001, 0.0001$ yielded the best results.

Table 12: Details of hyper-parameters for CV task in Section 7.2, Appendix D.1, and Appendix E.2.

Local/Server	Hyperparameter	Small Prototype	Medium Prototype	Large Prototype
	Training epochs	20	80	100
	Batch Size	64	64	64
	Optimizer	Adam	Adam	Adam
Local Training	Learning Rate	1e-3	1e-3	1e-3
	Weight Decay	5e-5	5e-5	5e-5
	LR scheduler	None	None	StepLR(step_size = 10, gamma = 0.1)
	Optimizer	Adam	Adam	Adam
	Learning Rate	1e-5	1e-5	1e-5
	Weight Decay	5e-5	5e-5	5e-5
Server KD Training	Batch Size	128	128	128
	Training Epochs	1	1	1
	Ensemble Distillation Softmax Temperature	3	3	3
	Self-Regularizer Softmax Temperature	20	20	20

Table 13: Details of Architecture parameters and FL configuration for CV task in Section 7.2 and Appendix D.1.

Architecture Setting	Device Prototype	Architecture	CIFAR-10 Parameters	CIFAR-100 Parameters	Dataset Portion	Clients	Sample Rate
	Prototype S	ResNet8	1.23M	1.25M	0.1	100	0.1
Homo-Family	Prototype M	ResNet14	6.38M	6.43M	0.3	20	0.2
	Prototype L	ResNet18	11.17M	11.22M	0.6	4	0.5
	Prototype S	ViT-S	1.78M	1.79M	0.1	100	0.1
Hetero-Family	Prototype M	ResNet14	6.38M	6.43M	0.3	20	0.2
	Prototype L	VGG16	15.25M	15.30M	0.6	4	0.5

Table 14: Details of Architecture parameters and FL configuration for CV task experiment using pre-trained models in Appendix D.2.

Device Prototype	Architecture	STL-10/CINIC-10 Parameters	TinyImageNet Parameters	Dataset Portion	Clients	Sample Rate
Prototype S	mobilenetv3-large-100	4.21M	4.45M	0.2	4	1.0
Prototype M	mobilevitv2-175	13.36M	13.53M	0.3	3	1.0
Prototype L	ResNet34	21.28M	21.38M	0.5	2	1.0

Table 15: Details of Architecture parameters for Scalability Section 7.3, and Appendix D.4.

Device Prototype	CINIC-10					
	Architecture	Parameters	Dataset Portion	Clients	Sample Rate	Local Epochs
Prototype XXS	ResNet10-XXS	11K	0.0357	35	0.1	2
Prototype XS	ResNet10-XS	78K	0.0714	30	0.1	2
Prototype S	ResNet10-S	309K	0.1071	25	0.15	2
Prototype M	ResNet10-M	1.2M	0.1428	20	0.15	5
Prototype L	ResNet10	4.9M	0.1785	15	0.2	10
Prototype XL	ResNet18	11M	0.2142	10	0.3	10
Prototype XXL	ResNet50	24M	0.25	5	0.6	20

20

Table 16: Details of hyper-parameters for NLP task experiments in Section 7.2 and Appendix D.3.

Local/Server	Hyperparameter	Prototype S	Prototype M	Prototype L
Local Training	training epochs	1	1	1
	batch size	32	32	32
	optimizer	Adam	Adam	Adam
	Learning rate	3e-5	3e-5	3e-5
	Weight decay	0	0	0
	lr scheduler	None	None	None
Server KD Training	optimizer	Adam	Adam	Adam
	learning rate	3e-5	3e-5	3e-5
	weight decay	3e-5	3e-5	3e-5
	batch size	32	32	32
	training epochs	1	1	1
	Ensemble distillation softmax temperature	3	3	3
	Self-regularizer softmax temperature	20	20	20

Table 17: Details of Architecture parameters and FL configuration for NLP task experiment using pre-trained models in Section 7.2 and Appendix D.3.

Device Prototype	Architecture	Parameters	Clients	Dataset Portion	Sample Rate
Prototype S	BERT-Tiny	4.39M	8	0.1	0.4
Prototype M	BERT-Mini	11.17M	4	0.3	0.5
Prototype L	BERT-Small	28.77M	2	0.6	1.0

21