Decentralized Learning with Multi-Headed Distillation

Supplementary Materials

A. Analysis and Discussion of Our Method

A.1. Analysis of Multi-Headed Distillation

As described in the main text, the multi-headed distillation involves simultaneous training of multiple model heads that communicate with each other. Rigorous theoretical analysis of this process in the most general case is very complicated. However, by making several assumptions, here we find an approximate solution for the weights of the heads of rank k being given the weights of the heads of rank $k-1$. In our future work, we hope to study this model for different prediction aggregation techniques and compare conclusions obtained from this simple model with those obtained empirically in realistic systems.

Let X be the input space and $L = \mathbb{R}^d$ be the logit space, where d is the number of classes. The logits $f(x)$ for a model $f : X \to L$ are then converted to label assignment probabilities $p(y|x)$ via softmax, i.e., $p(y|x) = \mathrm{softmax}(f(x))$.

Consider a single client $h_i : X \to L$ distilling information from some model head $h : X \to L$. The corresponding distillation loss $\mathcal{L}[h_i; h]$ admits many possible choices, but we will assume that

$$\mathcal{L} \equiv \mathbb{E}_{x\sim\mathcal{D}}\, D\left[h_i(y|x;\psi_i) \,\|\, p_h(y|x)\right]$$

with $\mathcal{D}$ being the shared (proxy) dataset and D being some divergence (more general f-divergence or KL-divergence as some examples). The distillation is then carried out by performing optimization of $\mathcal{L}$, for example via gradient descent:

$$\Delta\psi_i = -\gamma\frac{\partial\mathcal{L}}{\partial\psi_i}.$$

Notice that the components of ψ_i corresponding to the model backbone may receive updates from multiple heads reusing the same model embedding.

In our system, we assume that there is a set of heads $\{h_i^{(1)}, h_i^{(2)}, \ldots, h_i^{(n)}\}$ for each client i. For simplicity, let us first consider distillation procedure independent of prediction confidence. In this case, the loss for head k of the client i may look like:

$$\mathcal{L}_i^{(k)} \equiv \sum_{j=1}^{N} \rho_{ij}\, \Gamma\left[h_i^{(k)} \,\middle\|\, h_j^{(k-1)}\right],$$

where

$$\Gamma\left[h_i^{(k)} \,\middle\|\, h_j^{(k-1)}\right] \equiv \mathbb{E}_{x\sim\mathcal{D}}\, D\left[p_i^{(k)}(y|x) \,\middle\|\, p_j^{(k-1)}(y|x)\right],$$

$p_i^{(k)}(y|x)$ is a shorter notation for $p_{h_i^{(k)}}(y|x)$ and ρ_{ij} is some distribution defining the probability of picking a particular client for distillation. Again, here we assume that ρ_{ij} does not depend on the sample confidence and is simply fixed.

While we talked about $h^{(k)}$ distilling to $h^{(k-1)}$, we have not yet discussed the "main head" $h^{(1)}$. This head is normally trained locally on the client's private data. For simplicity, in the following we thus assume that its behavior is known, i.e., $h_i^{(1)}$ is a specified function of the training step. Furthermore, in the following, we start analyzing the problem by assuming that all $h_i^{(1)}$ already converged and are all generally different due to some differences in the client's private data. The behavior of all other heads is then defined by the losses outlined above.

Let us first consider the simplest case of $\rho_{ij} = \delta_{ij}$. In other words, each head only distills from the same client's "prior" head. The choice of $h_i^{(n)} = \cdots = h_i^{(1)}$ would obviously minimize all losses $\mathcal{L}_i^{(k)}$ since all corresponding $\Gamma[\cdot]$ values vanish. But as soon as we introduce a small correction $\rho_{ij} = \delta_{ij} + \nu_{ij}$ with $\sum_j \nu_{ij} = 0$, this trivial solution is no longer optimal. Instead, each client's head is now optimizing:

$$\mathcal{L}_i^{(k)} = \Gamma\left[h_i^{(k)} \,\middle\|\, h_i^{(k-1)}\right] + \sum_{j=1}^{N} \nu_{ij}\, \Gamma\left[h_i^{(k)} \,\middle\|\, h_j^{(k-1)}\right].$$

Notice that if Γ was a metric in the h space, we could interpret this optimization objective geometrically as a minimization of the head's distance to its lower-order state (towards $h_i^{(k-1)}$) coupled with a weak ($\sim \nu$) attraction towards a number of other heads ($h_j^{(k-1)}$). See Figure 7 for illustration.

Here we have to make yet another simplifying assumption and consider a prescribed model backbone (and corresponding embedding) that we are not optimizing or updating with backpropagated gradients. Doing so, we disentangle individual heads and can treat their optimization as independent tasks. For sufficiently small ν_{ij} it will hold that $p_i^{(k)} = p_i^{(k-1)} + O(\nu)$ and we can therefore write:

$$\mathcal{L}_i^{(k)} = \mathbb{E}_{x\sim\mathcal{D}}\left\{ D\left[p_{h_i^{(k-1)}+\kappa_i^{(k)}} \,\middle\|\, p_{h_i^{(k-1)}}\right] + \right.$$
$$\left. + \sum_{j=1}^{N} \nu_{ij} D\left[p_{h_i^{(k-1)}+\kappa_i^{(k)}} \,\middle\|\, p_{h_j^{(k-1)}}\right]\right\},$$

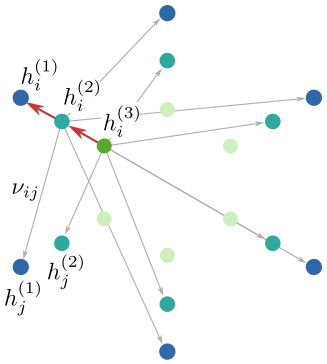

Figure 7. Illustration of multi-head distillation as discussed in Appendix A.1. Large red arrow shows strong distillation of $h_i^{(k)}$ to $h_i^{(k-1)}$ and smaller gray arrows indicate attraction towards $h_j^{(k-1)}$ with effective "strength" ν_{ij}.

where $h_i^{(k)} \equiv h_i^{(k-1)} + \kappa_i^{(k)}$ and $\kappa_i^{(k)} \sim O(\nu)$. Introducing $\delta_i^{(k)} \equiv p_i^{(k)} - p_i^{(k-1)}$, we obtain:

$$\mathcal{L}_i^{(k)} = \mathbb{E}_{x\sim\mathcal{D}}\Bigg\{ D\left[p_i^{(k-1)} + \delta_i^{(k)} \,\middle\|\, p_i^{(k-1)}\right] + \\ + \sum_j \nu_{ij}\, D\left[p_i^{(k-1)} + \delta_i^{(k)} \,\middle\|\, p_j^{(k-1)}\right]\Bigg\}.$$

Noticing that the first term needs to be decomposed near the global minimum and the second term permits linear expansion, we obtain:

$$\mathcal{L}_i^{(k)} \approx \mathbb{E}_{x\sim\mathcal{D}}\Bigg\{ D''\left[p_i^{(k-1)} \,\middle\|\, p_i^{(k-1)}\right]\frac{\delta_i^{(k)}\delta_i^{(k)}}{2} + \\ + \sum_j \nu_{ij}\, D'\left[p_i^{(k-1)} \,\middle\|\, p_j^{(k-1)}\right]\delta_i^{(k)}\Bigg\},$$

where D'' and D' are the derivatives of D with respect to the first argument. Recalling that $\delta_i^{(k)} \in \mathbb{R}^d$ we can rewrite the loss function as:

$$\mathcal{L}_i^{(k)} \approx \mathbb{E}_{x\sim\mathcal{D}}\left[\delta^\top \mathrm{A}\delta + b^\top\delta\right],$$

where $\delta \equiv \delta_i^{(k)}$ for brevity,

$$\mathrm{A} \equiv D''\left[p_i^{(k-1)} \,\middle\|\, p_i^{(k-1)}\right]/2$$

is effectively a matrix and

$$b \equiv \sum_j \nu_{ij}\, D'\left[p_i^{(k-1)} \,\middle\|\, p_j^{(k-1)}\right] \in \mathbb{R}^d$$

can be thought of as a column vector.

At this point we can connect the probability distribution perturbation δ to the logit perturbation $\kappa \equiv \kappa_i^{(k)}$ using the fact that $p_m \equiv e^{h_m}/Z$, where $Z \equiv \sum_k e^{h_k}$ (we omit this simple calculation here):

$$p_i^{(k)} = p_{h_i^{(k-1)}+\kappa_i^{(k)}} = p_i^{(k-1)} + \delta = \\ = p_i^{(k-1)} + \kappa * p_i^{(k-1)} - (\kappa\cdot p_i^{(k-1)})p_i^{(k-1)},$$

where $\boldsymbol{a} * \boldsymbol{b}$ is an element-wise product of two vectors and therefore:

$$\delta = \kappa * p_i^{(k-1)} - (\kappa\cdot p_i^{(k-1)})p_i^{(k-1)} \equiv \mathrm{C}\kappa, \qquad (6)$$

where C is a matrix constructed from the components of $p_i^{(k-1)}(x) \in \mathbb{R}^d$. Notice that $\sum_m \delta_m = 0$, which agrees with δ being the perturbation of the normalized probability distribution.

Finally, remember that κ itself is a perturbation of model logits. Given the sample embedding $\xi_i(x) \in \mathbb{R}^t$, the sample logits are constructed as $\mathrm{W}_i\xi_i(x)$ with W_i being a $d\times t$ matrix. The perturbation κ transforming $\mathrm{W}_i^{(k-1)}\xi_i$ into $\mathrm{W}_i^{(k)}\xi_i$ can thus be characterized by the logit weight perturbation $\mu \equiv \mu_i^{(k)} := \mathrm{W}_i^{(k)} - \mathrm{W}_i^{(k-1)}$ and we get $\kappa = \mu\xi(x)$. Combining everything together, we see that the loss function transforms to:

$$\mathcal{L}_i^{(k)} \approx \mathbb{E}_{x\sim\mathcal{D}}\left[\xi(x)^\top\mu^\top\mathrm{C}^\top\mathrm{AC}\mu\xi(x) + b^\top\mathrm{C}\mu\xi(x)\right], \qquad (7)$$

where A, C and $b \sim \nu$ all depend on the sample x via $p_i^{(k-1)}(x)$ and ξ is a function of x, while μ is effectively an unknown sample-independent matrix that we need to tune with the goal of minimizing $\mathcal{L}_i^{(k)}$. The optimum can be identified by taking a derivative with respect to $\mu_{\alpha\beta}$ and setting it to 0:

$$\mathbb{E}_{x\sim\mathcal{D}}\left[2(\xi(x)^\top\mu^\top\mathrm{C}^\top\mathrm{AC})_\alpha\xi_\beta(x) + (b^\top\mathrm{C})_\alpha\xi_\beta(x)\right] = 0.$$

This is a linear equation on $\mu \sim \nu$ that can be solved in a closed form to give us a logit weight perturbation μ as a complex nonlinear function of ν_{ij} and $\{p_\ell^{(k-1)}\}$.

Note that since μ is only a small perturbation, we can introduce $\mathrm{W}_i^{(k)}$ as a function of a *continuous* parameter k and approximate $d\mathrm{W}_i^{(k)}/dk$ with a finite difference $\mathrm{W}_i^{(k)} - \mathrm{W}_i^{(k-1)} = \mu$ leaving us with a differential equation (the approximation is valid in the first order in ν):

$$\frac{d\mathrm{W}_i(k)}{dk} = G\left[\nu, \{\mathrm{W}_\ell(k)\}\right]$$

with G being a linear function with respect to ν, but very complex nonlinear function with respect to $\{\mathrm{W}_\ell\}$. If ν_{ij} is localized around $i = j$ (which would be the case for communication patterns with partial connectivity, like in the

case of long chains), this differential equation resembles a complex nonlinear diffusion equation defining the spread of information across the clients as we look at deeper and deeper heads (with the head rank k essentially playing the role of time).

It is also worth noting here that if ν was not fixed, but was itself a function of model confidence (while still remaining small), our conclusions would not change except that ν itself would now itself be a complex nonlinear function of $\{\mathrm{W}_\ell(k)\}$ and x. In our future work, we hope to study the effect that this confidence-dependent aggregation has on head dynamics and the final stationary state.

Finally, let us look at the stationary state of system dynamics. Equation (7) suggests that $\mu = 0$ is a local optimum when $b^\top \mathrm{C} = 0$, or

$$\sum_{i,j} \nu_{ij} D' \left[p_i \| p_j\right] \mathrm{C}_{ik} = 0,$$

or after noticing that $D'[p_i\|p_j] = p_j/p_i$ and recalling that C is defined by Eq. (6) we obtain for every k:

$$\mathbb{E}_{x\sim\mathcal{D}} \left[\sum_{i,j} \nu_{ij} p_j \left(\delta_{ik} - p_k\right) \right] = 0. \tag{8}$$

Since $\sum_j \nu_{ij} = 0$, the trivial solution of this system of equations is the case of identical models, i.e., $p_1 = \cdots = p_n$, but since generally the models might have different embeddings and cannot be made identical, the solution of Eq. (8) restricts the system stationary state.

A.2. Value of $p(y|x)$ as Classifier Confidence

In our model distillation approach, we need to combine predictions of multiple different model heads. If all predictions $p_k(y|x)$ (by heads $\{h_k\}$) come with reliable error estimates, this information can be taken into account. For example, if we know that for the true distribution $p(y|x)$ and every prediction $p_k(y|x)$ it holds that $D[p_k(y|x)\|p(y|x)] \leq e_k(x)$ with D being some divergence, the true $p(y|x)$ belongs to the intersection of "balls"[3] $\mathcal{B}_k \equiv \{p' | D[p'\|p] \leq e_k\}$. We can then choose any point in this set and compute a prediction error as a maximum distance from a chosen distribution to any point in the intersection. Unfortunately, however such reliable measures of classifier error are not generally available and even approximating them can be quite difficult.

Instead we choose a very simple approach based on estimating classifier confidence and picking the most confident model, effectively ignoring other predictions. The confidence measure itself is chosen as a value of the largest component of the classifier prediction $o(x) \equiv \mathrm{softmax}(f(x;\theta))$ with $f(x;\theta) = \mathrm{W}\xi(x;\theta)$ and ξ being the embedding vector.

[3] note that D is not generally a metric

Why choose $\max_k o_k$. This value has a number of trivial properties that can actually make it a useful measure of classifier uncertainty. First is that after seeing a supervised training sample (x, y), the value of $o_y(x)$ is increased. Second is that if the class prototypes in the embedding space are nearly orthogonal for different classes, then updates for samples of different classes would not "interfere" with each other and high-confidence predictions would not generally be disrupted by observing unrelated samples with different labels. For a simple logits layer $\mathrm{W}\xi(x)$ trained with cross-entropy loss, both of these properties trivially follow from the following expression for $\Delta o_k(x')$ after training on a sample (x, y):

$$\Delta o_k(x') = \lambda o_k(x')[\xi(x)\cdot\xi(x')]\times \\ \times \sum_i \left(\delta_{k,i} - o_i(x')\right)\left(\delta_{y,i} - o_i(x)\right).$$

Drawbacks. But while $\max_k o_k(x)$ has these useful properties, it is not guaranteed to be a reliable measure of classifier confidence for out-of-distribution samples and the training objective never explicitly optimizes for that[4]. A density model $\rho(x)$ would allow detecting such out-of-distribution samples, but could also reveal information about the client samples in their private dataset. Combining classification models with locally-trained density models, or adopting other existing similar techniques could be a logical extension of our present work.

A.3. Distillation as Revelation of Some Information about Model Weights

The canonical version of FedAvg combines the knowledge of individual clients by periodically aggregating their weight snapshots. Distillation-based techniques are instead based on communicating model predictions on datasets accessible to all participants. While these two approaches appear to be different, communication in distillation-based methods can of course also be viewed as a way of revealing incomplete information about model weights.

The amount of revealed information can be defined as follows. Assuming the knowledge of the prior $p(\theta)$ on the model weights and model predictions $(y_1, \ldots, y_n)$ on a public dataset $\mathcal{D}_* = (x_1, \ldots, x_n)$, one can compare the difference of entropies for the original $p(\theta)$ and $p(\theta|y_1, \ldots, y_n)$ with

$$p(\theta|y_1,\ldots,y_n) = \frac{p(y_1,\ldots,y_n|\theta)p(\theta)}{\int d\theta\, p(y_1,\ldots,y_n|\theta)p(\theta)}.$$

[4] Contrast this to the energy-based models, for example, where the energy update and the MCMC sampling are explicitly contributing to the model "awareness" of what in-distribution samples are and are not.

While generally intractable, it might be possible to obtain the lower bound on the amount of the revealed information by training a model that predicts the weights θ from $(y_1, \ldots, y_n)$.

Deeper understanding of this question can have an impact on the optimal choice of the public dataset $\mathcal{D}_*$ that would allow us to retrieve the knowledge of interest from a trained model using only a small number of samples. Ongoing research on dataset distillation [7, 27, 36] is very closely related to this question.

B. Additional Experiments and Experimental Data

B.1. Effect of Distilling to Self and Same-Level Heads

In Section 4.2.2 we reported that including a head into the list of its own distillation targets ("self") improved the model accuracy, but the gain was still smaller than that of a model with multiple auxiliary heads. Here we explore what happens if we use the head as its own potential distillation target, while also using a number of auxiliary heads. Furthermore, what if we modify our method to include distillations to other heads of the same rank (see Figure 9)?

We conducted a set of experiments with a heterogeneous dataset with $s = 100$, $\nu_{\text{emb}} = 1$, $\nu_{\text{aux}} = 3$, four auxiliary heads and 250 randomized labels per each of 8 clients. The results of experiments using different combinations of distillation targets and both $\Delta = 1$ and $\Delta = 2$ (choosing two other clients at a time as potential distillation targets) are presented in Table 3. We observed that using same-level heads and "self" targets *separately* provides noticeable benefit only for earlier heads. But when used together, these two techniques result in $\sim 1\%$ accuracy improvement and this improvement is realized for the 2^{nd} auxiliary head. Also, not unexpectedly, using two clients to distill to ($\Delta = 2$) instead of one, leads to a noticeable 1.5% accuracy improvement. Combined together, all these techniques, in conjunction with using the entire ImageNet as the public dataset improve the accuracy to 59.4% if trained for 60k steps, or 65.7% if trained for 180k steps.

B.2. Dependence on the Public Dataset Size

In a separate set of experiments, we trained 8 clients with 4 auxiliary heads, $s = 100$, $\nu_{\text{emb}} = 1$, $\nu_{\text{aux}} = 3$ and 250 randomly assigned "private" labels and "private" samples drawn from 70% of the ImageNet training data. The remaining 30% of ImageNet samples were fully or partly used as a public dataset, i.e., $\gamma_{\text{pub}} \leq 30\%$. As one would expect, increasing the size of the "public" dataset while fixing the amount of "private" training data has a positive impact on the final model performance (see Table 4).

B.3. Training Algorithms

The proposed distributed learning algorithm is summarized in Algorithm 1. In our experiments, we used a modified version of this technique, communicating model checkpoints instead of directly communicating model predictions (see Algorithm 2).

Algorithm 1 Core Distributed Learning Algorithm

Input: Private datasets $\{\mathcal{D}_i\}_{i=1}^K$, public dataset $\mathcal{D}_*$
Output: Trained models $\{\mathcal{M}_i\}_{i=1}^K$

for client C_i (run concurrently) **do**
  for iteration $t \in [0, T]$ **do**
    Sample a private mini-batch $(x_{\text{priv}}, y_{\text{priv}}) \sim \mathcal{D}_i$
    Sample a public mini-batch $(x_{\text{pub}}, y_{\text{pub}}) \sim \mathcal{D}_*$
    Communicate with clients from $e_t(i)$ to get their embeddings and/or predictions[5] for $(x_{\text{pub}}, y_{\text{pub}})$
    Do a single backpropagation step updating $\mathcal{M}_i$ to optimize $\mathcal{L}_i$ given by Eqs. (1) and (2), (4)

Algorithm 2 Distributed Learning Algorithm with Communication via Model Checkpoints

Input: Private datasets $\{\mathcal{D}_i\}_{i=1}^K$, public dataset $\mathcal{D}_*$
Output: Trained models $\{\mathcal{M}_i\}_{i=1}^K$

for client C_i (run concurrently) **do**
  Initialize $\mathcal{P}_i \leftarrow \{\}$
  for iteration $t \in [0, T]$ **do**
    if $t \ \% \ S_{\mathcal{P}} = 0$ **then**
      Update $\mathcal{P}_i$ adding a new or replacing a random existing checkpoint (if up to capacity $N_{\mathcal{P}}$) with a new random checkpoint from $e_t(i)$
    Sample a private mini-batch $(x_{\text{priv}}, y_{\text{priv}}) \sim \mathcal{D}_i$
    Sample a public mini-batch $(x_{\text{pub}}, y_{\text{pub}}) \sim \mathcal{D}_*$
    Use Δ random checkpoints from $\mathcal{P}_i$ to compute embeddings and/or predictions for $(x_{\text{pub}}, y_{\text{pub}})$
    Do a single optimization step updating $\mathcal{M}_i$ to minimize $\mathcal{L}_i$ given by Eqs. (1) and (2), (4)

C. Additional Tables and Figures

C.1. Raw Experimental Data

Tables 5 and 6 contain raw values used for producing Figure 3, while Tables 7 and 8 complement Figure 4.

[5] can send sample IDs rather than samples themselves

Experiment	$\beta_{\text{priv}}^{\text{Main}}$	$\beta_{\text{sh}}^{\text{Aux1}}$	$\beta_{\text{sh}}^{\text{Aux2}}$	$\beta_{\text{sh}}^{\text{Aux3}}$	$\beta_{\text{sh}}^{\text{Aux4}}$
Base	70.9%	46.7%	51.8%	53.9%	**54.6%**
$\Delta = 2$	71.1%	50.9%	55.1%	**56.1%**	**56.0%**
SL	70.8%	48.6%	53.6%	**54.7%**	**54.7%**
SF	71.3%	48.1%	53.4%	**54.9%**	**54.8%**
SL+SF	70.3%	53.0%	**55.5%**	53.9%	52.4%
All	70.8%	53.5%	**55.8%**	54.5%	52.9%
All+	72.7%	56.5%	**59.4%**	57.9%	56.1%
All+, 180k steps	76.2%	62.3%	**65.7%**	65.0%	64.0%

Table 3. Experimental results exploring the usage of different distillation heads trained for 60k steps. Here "*Base*" is the original experiment with $\Delta = 1$ and conventional heads as described in Sec. 4.2.2; "*SL*" adds same-level heads to distillation targets; "*SF*" adds the distilled head ("self") as a potential target; "*All*" combines same-level and "self" heads and $\Delta = 2$ (each step distilling to two other clients), "*All+*" is the same as *All*, but also uses the entire ImageNet as the public dataset.

Public DS fraction	10%	20%	30%	*All*
main head β_{priv}	70.1%	71.1%	70.9%	71.9%
last aux head β_{sh}	52.4%	53.9%	54.1%	55.3%

Table 4. The dependence of the main head "private" accuracy β_{priv} and the "shared" accuracy of the 4^{th} auxiliary head on the size of the public dataset (fraction of ImageNet training set). Experiments were conducted for a system of 8 clients with 4 auxiliary heads, $s = 100$, $\nu_{\text{emb}} = 1$, $\nu_{\text{aux}} = 3$ and 250 randomly assigned "private" labels. Private training samples were drawn from 70% of the ImageNet training set in all experiments. "*All*" column shows the accuracy attained by using the entire ImageNet training set as a public dataset (while still using only 70% of it as private data).

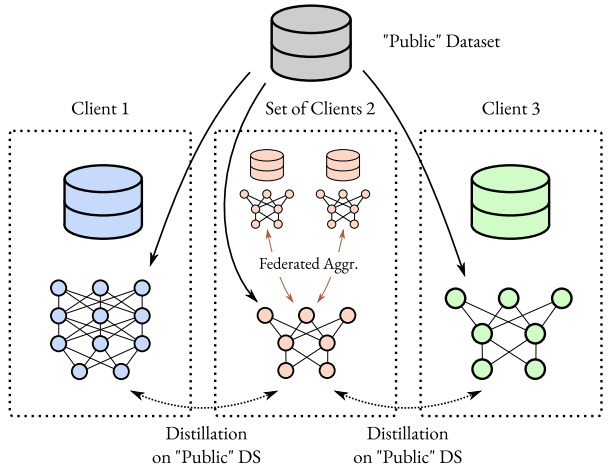

Figure 8. Conceptual diagram of a distillation in a distributed system. Clients use a "public" dataset to distill knowledge from other clients, each having their primary private dataset. Individual clients may have different architectures and different objective functions. Furthermore, some of the "clients" may themselves be collections of models trained using federated learning.

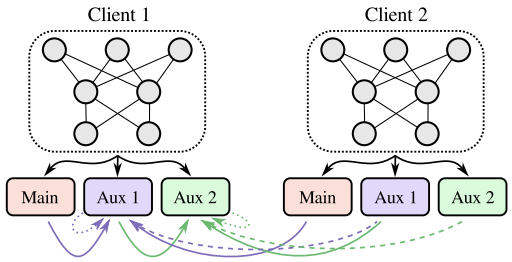

Figure 9. A pattern used for distilling multiple auxiliary heads with two additional types of distillation targets: (a) distilling to heads of the same "rank" (dashed), (b) distilling to "self" (dotted). Here distilling to the same "rank" means, for example, that *Aux1* head is distilled to the most confident of *Main* heads, or *Aux1* heads of adjacent clients. Distilling to "self" means that the samples on which the distilled head is already most confident will effectively be ignored.

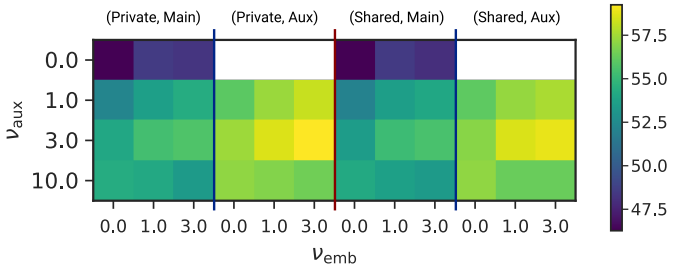

Figure 10. Alternative visualization of $s = 0$ results from Figure 3.

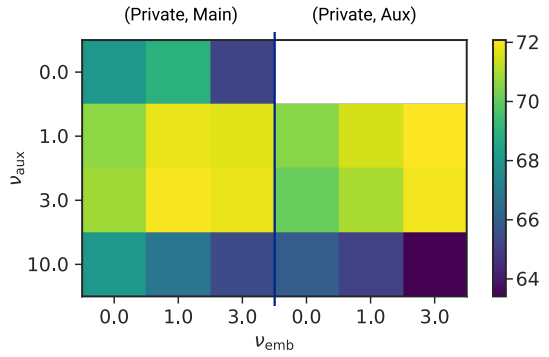

Figure 11. Alternative visualization of $s = 100$ *private* accuracies from Figure 3.

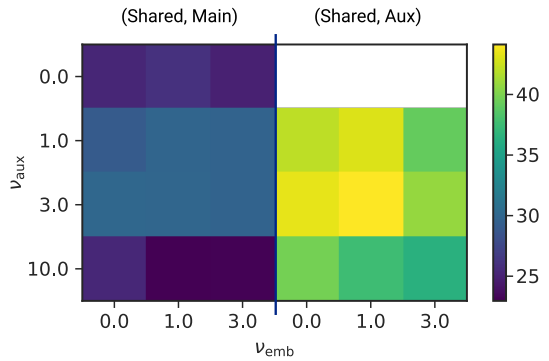

Figure 12. Alternative visualization of $s = 100$ *shared* accuracies from Figure 3.

ν_{emb}	ν_{aux}	$\beta_{\text{priv}}^{(\text{m})}$	$\beta_{\text{sh}}^{(\text{m})}$	$\beta_{\text{priv}}^{(\text{aux})}$	$\beta_{\text{sh}}^{(\text{aux})}$
0.0	0.0	46.3%	46.3%	0.1%	0.1%
	1.0	52.2%	52.0%	56.0%	56.0%
	3.0	54.1%	53.5%	57.3%	57.0%
	10.0	54.3%	54.1%	57.1%	57.1%
1.0	0.0	48.5%	48.5%	0.1%	0.1%
	1.0	53.6%	53.5%	57.3%	57.2%
	3.0	55.4%	55.2%	58.6%	58.5%
	10.0	54.1%	53.6%	56.9%	56.3%
3.0	0.0	48.3%	48.0%	0.1%	0.1%
	1.0	54.3%	54.0%	58.2%	57.6%
	3.0	55.7%	55.5%	59.3%	58.8%
	10.0	53.3%	53.4%	56.5%	56.3%

Table 5. Results for 8-client experiments with 250 random classes per client, $s = 0$ and a varying values of ν_{emb} and ν_{aux}.

ν_{emb}	ν_{aux}	$\beta_{\text{priv}}^{(\text{m})}$	$\beta_{\text{sh}}^{(\text{m})}$	$\beta_{\text{priv}}^{(\text{aux})}$	$\beta_{\text{sh}}^{(\text{aux})}$
0.0	0.0	68.0%	25.2%	0.1%	0.1%
	1.0	70.6%	29.1%	70.5%	42.0%
	3.0	70.9%	30.0%	70.1%	43.3%
	10.0	68.0%	25.3%	66.0%	39.7%
1.0	0.0	69.0%	26.0%	0.1%	0.1%
	1.0	71.8%	29.8%	71.5%	43.0%
	3.0	72.0%	29.9%	71.0%	44.1%
	10.0	66.8%	23.0%	65.1%	37.5%
3.0	0.0	65.2%	24.9%	0.1%	0.1%
	1.0	71.7%	29.7%	72.1%	39.1%
	3.0	71.8%	29.7%	71.9%	40.8%
	10.0	65.4%	23.1%	63.4%	36.4%

Table 6. Results for 8-client experiments with 250 random classes per client, $s = 100$ and a varying values of ν_{emb} and ν_{aux}.

Heads	1	2	3	4
$\beta_{\text{priv}}^{(\text{m})}$	56.2%	56.1%	55.8%	55.9%
$\beta_{\text{sh}}^{(\text{m})}$	56.1%	55.8%	55.8%	55.5%
$\beta_{\text{priv}}^{(1)}$	59.6%	59.6%	59.4%	59.4%
$\beta_{\text{sh}}^{(1)}$	59.4%	59.5%	59.6%	59.4%
$\beta_{\text{priv}}^{(2)}$		60.0%	59.7%	59.7%
$\beta_{\text{sh}}^{(2)}$		59.7%	59.9%	59.5%
$\beta_{\text{priv}}^{(3)}$			59.5%	59.1%
$\beta_{\text{sh}}^{(3)}$			59.5%	59.1%
$\beta_{\text{priv}}^{(4)}$				58.7%
$\beta_{\text{sh}}^{(4)}$				58.6%

Table 7. Results for 8-client experiments with 250 random classes per client, $s = 0$, $\nu_{\text{emb}} = 1$, $\nu_{\text{aux}} = 3$ and a varying number of auxiliary heads (separate columns).

Heads	1	2	3	4
$\beta_{\text{priv}}^{(\text{m})}$	72.5%	71.6%	71.1%	72.5%
$\beta_{\text{sh}}^{(\text{m})}$	30.5%	32.5%	33.1%	32.7%
$\beta_{\text{priv}}^{(1)}$	71.4%	70.6%	70.7%	71.4%
$\beta_{\text{sh}}^{(1)}$	44.7%	46.6%	46.9%	46.4%
$\beta_{\text{priv}}^{(2)}$		68.5%	68.1%	68.7%
$\beta_{\text{sh}}^{(2)}$		51.6%	52.0%	51.6%
$\beta_{\text{priv}}^{(3)}$			66.1%	66.1%
$\beta_{\text{sh}}^{(3)}$			53.8%	53.6%
$\beta_{\text{priv}}^{(4)}$				63.4%
$\beta_{\text{sh}}^{(4)}$				54.5%

Table 8. Results for 8-client experiments with 250 random classes per client, $s = 100$, $\nu_{\text{emb}} = 1$, $\nu_{\text{aux}} = 3$ and a varying number of auxiliary heads (separate columns).

Notation	Meaning
$\mathcal{D}_*$	Public dataset
K	Total number of clients
$C = \{C_1, \ldots, C_K\}$	A set of clients
$\mathcal{D}_i$	Private dataset of the client C_i
$\mathcal{M}_i$	Private model of C_i
$\mathcal{T}_i$	Private task (typically unique label distribution) of C_i
t	Global training step
$\mathcal{G}_t$	Communication graph at step t
$\mathcal{E}_t$	Set of edges of $\mathcal{G}_t$
$e_t(i)$	Set of clients connected to C_i via a set of outgoing edges from $\mathcal{E}_t$
$\mathcal{L}_i$	Local objective optimized by C_i
$\mathcal{L}_{i,\mathrm{CE}}$	Cross-entropy loss of C_i (on private data $\mathcal{D}_i$)
$\mathcal{L}^{\alpha}_{\mathrm{dist}}$	Collection of different *distillation losses* enumerated by α
$\psi^{\alpha}_i(x)$	Local activations computed by C_i for sample x (for distillation)
$\Phi^{\alpha}_{t,i}(x) \equiv \{\phi^{\alpha}_j(x) \vert j \in e_t(i)\}$	Remote activations computed by $e_t(i)$ for sample x (for distillation)
$\phi^{\alpha}_j(x)$	Remote activations computed by C_j for sample x
$\xi_i(x)$	Intermediate embedding produced by C_i for sample x
ν_{emb}	Regularization weight for embedding distillation
ρ	Monotonically growing function used for embedding distillation
$\psi^{\mathrm{norm}}_i(x)$	Normalized embedding of x (for C_i)
$\boldsymbol{h}_i(\xi_i(x))$	Main head of the model $\mathcal{M}_i$
$\boldsymbol{h}^{\mathrm{aux}}_i(\xi_i(x))$	Auxiliary head of the model $\mathcal{M}_i$
$\mathcal{L}^{\mathrm{aux}}_{\mathrm{dist}}[\boldsymbol{h}^{\mathrm{aux}}, \boldsymbol{h}]$	Prediction distillation loss
ν_{aux}	Auxiliary loss weight
$\Lambda(\boldsymbol{h}(x))$	"Confidence" of the classifier prediction $\boldsymbol{h}(x)$
Q	Function used in a particular form of prediction distillation loss
H	Confidence of all related distillation targets
$\{\boldsymbol{h}^{\mathrm{aux},1}, \ldots, \boldsymbol{h}^{\mathrm{aux},m}\}$	Collection of multiple auxiliary heads
$\mathcal{D}$	Underlying labeled dataset used to generate experimental $\{\mathcal{D}_i\}_{i=1}^{K}$
S	Set of all samples from $\mathcal{D}$
ℓ_i	Subset of all labels treated as *primary labels* for C_i
γ_{pub}	Fraction of public set samples
s	Dataset "skewness"
$\mathcal{P}_i$	A rolling pool of model checkpoints for C_i
$N_{\mathcal{P}}$	Number of checkpoints in each pool
Δ	Number of random checkpoints from $\mathcal{P}_i$ picked at every step for distillation
$S_{\mathcal{P}}$	Number of steps between pool updates
β_{priv}	Model $\mathcal{M}_i$ accuracy on the private dataset $\mathcal{D}_i$
β_{sh}	Model $\mathcal{M}_i$ accuracy on the public dataset $\mathcal{D}_*$

Table 9. Summary of notation.