A NETWORK ARCHITECTURES

Each experiment in this paper used four different types of architectures split among the different clients plus an additional small architecture for the stress test. There are ten different types of nodes (layers) in each architecture. Figure 5 shows the architectures used in the CIFAR-10/100 experiments. The types of convolutional layers are denoted by " c<channels_in>_<channels_ out>_<kernel_size>_<stride>". In the chest x-ray experiment, where the input images are grayscale and of size 224×224 the first convolutional layer type is "c1_64_k7_s2" type.

CIFAR-10, CIFAR-100, and Chest X-rays, respectively, use different types of linear layers with 10, 100, and 14 output dimensions.

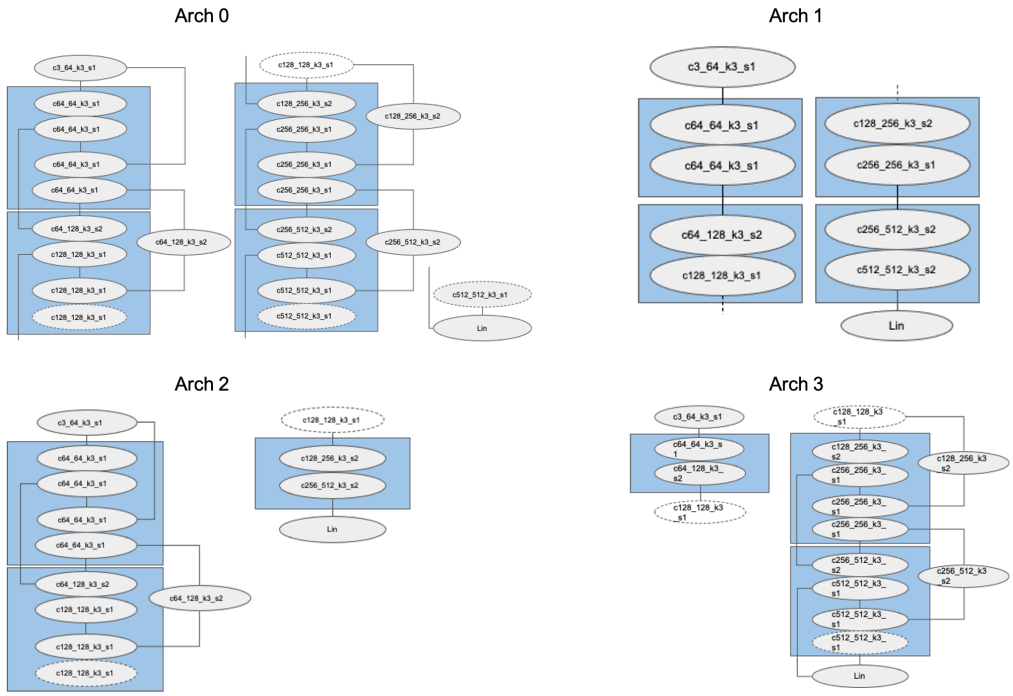

Figure 5: The four client architectures used in CIFAR-10/100 experiments.

B NON PARAMETRIC LAYERS

Since our main architectures all use residual connections and same activation type, the graph connectivity suffices to express the architecture. However, in general, different non-parametric layers might be desired. For that, we implemented three versions of ResNet in which several different residual layers were replaced by concatenation, resulting in a hybrid addition-concatenation. To that end, we included two new non-parametric types of nodes in our layer type set: "add" and "concat". Participating in message passing, these nodes produce latent embedding, depending on where they reside in the graph. No additional parameters are required for them. We trained these 3 architectures together with the vanilla ResNet with four and eight clients with the CIFAR10 and CIFAR100 datasets. On CIFAR10, the average results by architecture type are: 88.96, 89, 89, 89.2 for 4 clients and 88.6, 88.1, 88.3, 86.8 for 8 clients. On CIFAR100, the results are: 56, 56.2, 56.5, 48.200 for 4 clients and 50.6, 50.4, 47.7, 46.0 for 8 clients. The results are on par with those shown in Table 1. In particular, the performance obtained by the vanilla ResNet architecture on CIFAR10/100 with 4 and 8 clients respectively is: 88.96, 88.6, 56 and 50.6, whereas its performance under the FLHA-distillation baseline is 90, 85.7, 51.1, 44.2. In this example, FLHA-GHN shows success in training with two commonly used non-parametric layers.

C HYPERNETWORK INITIALIZATION

As described in Section 4.2 of the main paper, a proper initialization of the hypernetwork weights is instrumental to a successful training of the client networks. In figure 6 this is shown on an example convolutional layer with dimensions: $3 \times 3 \times 64 \times 128$. In both plots, the blue colored histogram shows the distribution of the desired Kaiming weight initialization He et al. (2015b). When the weights are generated by a hypernetwork, a standard initialization of the hypernetwork would generate the orange histogram shown on the left. We show what that histogram looks like after our initialization scheme on the right.

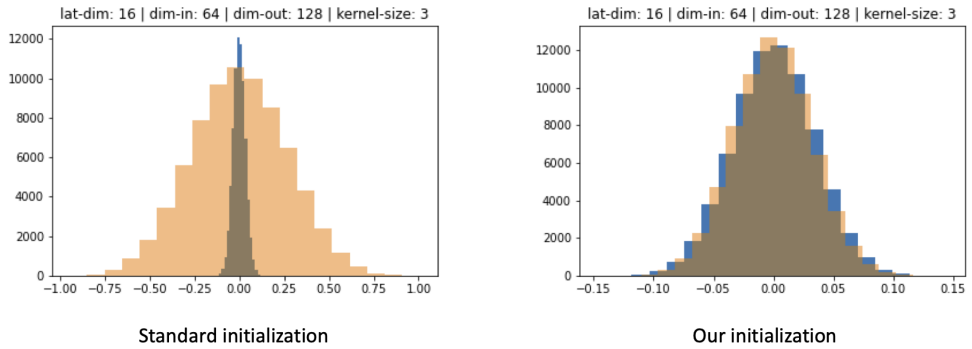

Figure 6: Hypernetwork weight initialization. We compare (blue:) a direct Kaiming weight initialization He et al. (2015b) of a convolutional layer with (orange:) the resulting weight initialization by the hypernetwork, without (left) and with (right) our initialization scheme.

D HYPERARAMETER SEARCH FOR OUR FLHA

We ran an extensive hyperparameter search using Biewald (2020) for a 4 architecture setup using a fixed number of 500 epochs , with 3 different GNN types: GraphConv(Morris et al., 2019), GatedGraphConv(Li et al., 2017), and GraphSAGE(Hamilton et al., 2018); number of GNN layers T from 1 to 8 with latent dimensions between 16 and 128; hypernetwork H_l bottleneck dimension between 16 and 64; learning rates between 1e-4 and 0.1; SGD (with and without cosine scheduler) and Adam(Kingma & Ba, 2014) optimizers with weight decay values between 5e-4 and 5e-6.

E IMPLEMENTATION DETAILS: LOCAL DISTILLATION

Baseline distillation from a teacher model trained via standard (same-architecture) FL is done using a distillation loss Hinton et al. (2015): $(1-\alpha)\text{CE}(y_{pred}, y) + \alpha\text{KL}(y_{pred}, y_{teacher}) \times 2T^2$ where CE and KL denote Cross-Entropy and Kullback Leibler, respectively. The softmax in the KL loss is taken with respect to a temperature $T = 20.0$ and $\alpha = 0.7$. We trained distillation as well as the main FL models for 200 epochs with SGD. The learning rate for training the teacher network (with same-architecture FL) is set to 0.1 with cosine scheduling. When distilling from the teacher network to the student, we use a learning rate to 0.01.

F IMPLEMENTATION DETAILS: PFEDHN

We found this architecture to be quite sensitive to hyperparameters, hence unlike our method which uses the same set of parameters in all experiments, here we chose the best performing parameters per setting. Hyperparameters sweep on the following parameters: learning rate, number of hidden layers in shared mlp, latent dimension size, optimizer (adam and sgd), and weight decay.

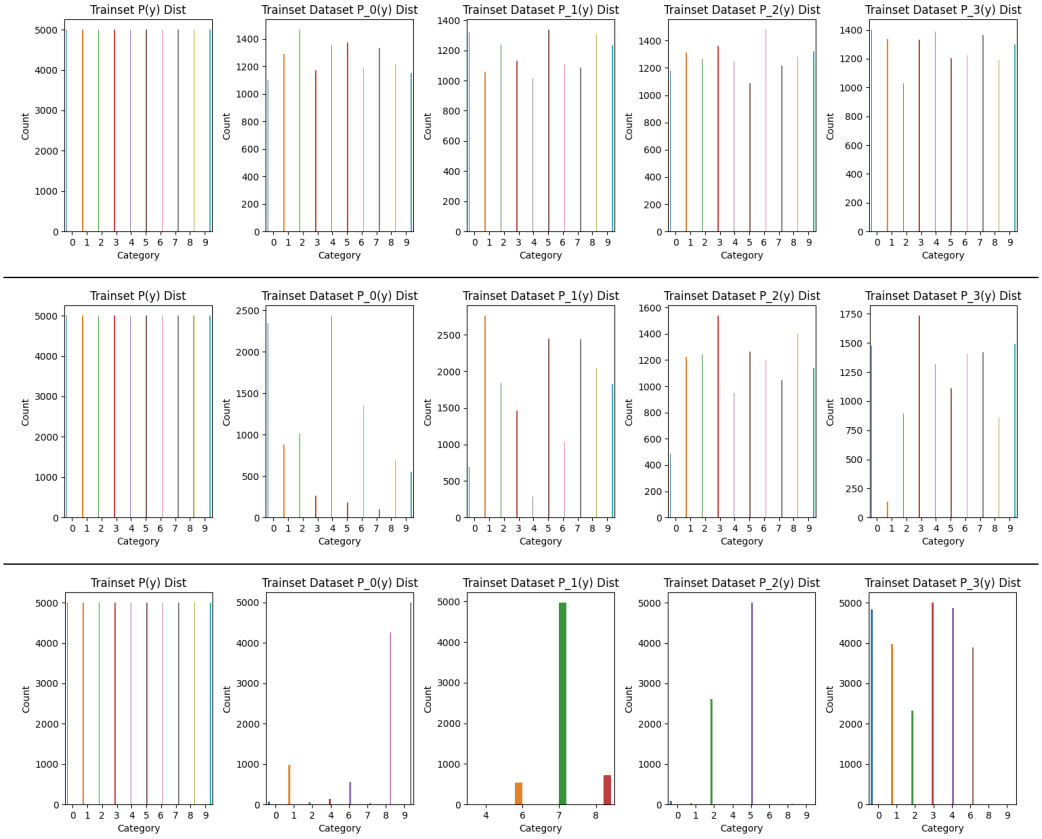

Table 4: Each row in the table shows the unbalanced class distribution for 4 clients, with the original balanced distribution of the left. Three different α values are shown: (top) $\alpha = 100$, (mid) $\alpha = 1$, (bottom) $\alpha = 0.1$

G UNBALANCED DISTRIBUTION

In collaborative training between different entities, clients' data may be distributed unevenly. In medical data, for example, this may occur when clinics specialize in certain diseases or use different sensors. Thus, in addition to architectural differences, FLHA can also have data imbalance. Here we study the behavior of FLHA under such unbalanced data distributions.

We follow Yurochkin et al. (2019); Hsu et al. (2019); Lin et al. (2020) and use the (symmetric) Dirichlet distribution, parameterized by a concentration parameter α to split the CIFAR-10 training and test sets between the different clients. Table 4 shows the resulting per-client class distributions. As can be seen, the smaller $alpha$ is, the less balanced the distribution is.

Table 5 shows the performance on CIFAR-10 under 3 different α values: $0.1, 1, 100$. The smaller α is the more unbalanced the distribution is. Class distribution under the different α values are shown in Table 4.

When the training data is unevenly distributed, performance can either be measured in a similar distributed test set or a balanced test set. In the former case, a client would like high performance on local, biased samples. Suppose a hospital specializes in a specific disease and hopes to improve model performance related to that disease through FL. We refer to that a "unbalanced" metric. Alternatively, a client might be interested in balancing its model bias, in which case the performance on the full ("balanced") test set is of interest. We compare our FLHA-GHN against a local training. We also include the usual upper bound performance of standard FL with all clients using the same architecture. Table 5 shows that FLHA-GHN outperforms local training in both "unbalanced" and "balanced" tasks and across all $alpha$ values. Specifically, it can be seen that local training achieves high performance only on test sets of similar distribution, but severely sacrifices performance on sets of balanced

		Client 0		Client 1		Client 2		Client 3	
α	method	unbal.	bal.	unbal.	bal.	unbal.	bal.	unbal.	bal.
100	FLHA-GHN	89.7	89.5	87.6	87.6	87.0	86.2	88.3	88.1
	Local	81.8	82.4	75.1	74.2	80.5	80.5	81.1	80.3
	Standard FL	92.9	93.7	93.2	93.7	94.1	93.7	94.5	93.7
1	FLHA-GHN	91.1	87.1	87.4	85.5	84.9	84.6	85.0	85.5
	Local	87.8	75.8	87.3	83.5	84.4	84.0	84.4	83.5
	Standard FL	93.8	93.0	92.9	93.0	92.6	93.0	92.9	93.0
0.1	FLHA-GHN	94.6	62.2	98.3	28.7	92.3	34.0	92.4	61.0
	Local	93.6	51.1	96.6	27.8	90.9	25.2	90.7	54.1
	Standard FL	69.3	53.8	58.2	53.8	42.3	53.8	49.6	53.8

Table 5: FLHA with unbalanced distribution. In the table, α corresponds to the level of unbalanced,e.g. α=100 (almost uniform), α=0.1 (extremelly unbalanced). unbal. and bal. are short for unbalanced and balanced and correspond to the distribution of the test set with unbalanced being the same distribution of each client's training set.

	GHN Init	From scratch
Arch 1 (original)	86.1	83.9
Arch 2 (No skip)	84.2	81.3
Arch 3 (Skip first)	84.3	83.7
Arch 4 (Skip last)	85.6	83.6

Table 6: Generalization to unseen architectures: leave-one-architecture-out experiment on CIFAR-10. Each row is the accuracy on a held-out architecture while training on the other architectures.

distributions. The same architecture FL also shows a trade-off. Despite being the most performant on the balanced test set, it sacrifices accuracy on local distributions. FLHA-GHN's capability to improve on local training in both tasks can be attributed to the inherent personalization of the network. That is, beyond its ability to adapt to new architectures, FLHA-GHN can also learn a personalized weight prediction according to the client distributions. This result aligns well with the observation of Shamsian et al. (2021).

H GENERALIZATION – ADDITIONAL RESULTS

Table 7 shows the performance of the 4 architectures used in our experiments, and how they are influenced when each is replaced by a smaller architecture. The average drop in performance of 2.2 ± 1.4 keep the performance well above the local-training alternative.

Replaced	Arch 1	Arch 2	Arch 3	Arch 4
None	90.4	89.0	87.3	88.5
Arch 4	88.2	86.9	86.4	80.3
Arch 3	88.8	87.1	79.6	88.2
Arch 2	88.6	81.6	85.4	86.2
Arch 1	77.5	83.8	85.4	83.6

Table 7: Training with a much smaller architecture shows an average performance drop by 2.2 ± 1.4 pts. However, this is well above the local training alternative.

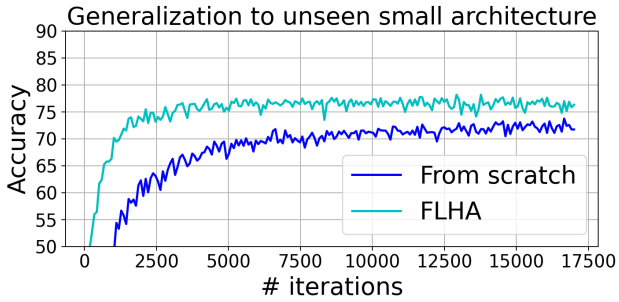

Figure 7: Generalization to a smaller 4 layer CNN architecture. Our method (light blue) quickly ramps up to high performance and maintains a considerable gap compared to training from scratch until convergence(dark blue).