Appendix

A Derivation of Equation (7)

From Eq (5), we know that:

$$\frac{d\mathcal{L}_{i,T}(\theta_i^*, \lambda)}{d\theta} = 0 \tag{11}$$

Based on the implicit functional theorem (IFT), we get that if we have a function $F(x, y) = c$, we can derive that $y'(x) = -F_x/F_y$. Therefore, plus the Eq (11) into the theorem, we can get:

$$\frac{d\theta^*}{d\lambda} = -\frac{\partial_\theta\left(\frac{d\mathcal{L}_{i,T}(\theta_i^*,\lambda)}{d\theta}\right)}{\partial_\lambda \frac{d\mathcal{L}_{i,T}(\theta_i^*,\lambda)}{d\theta}} = -\left(\partial_\theta^2 \mathcal{L}_T(\theta, \lambda)\right)^{-1} \partial_{\lambda\theta}\mathcal{L}_T(\theta, \lambda) \tag{12}$$

B Positioning of FedL2P.

Table 5 shows the positioning of FedL2P against existing literature. Note that this list is by no means exhaustive but representative to highlight the position of our work. All existing approaches obtained personalized models using a personalized policy and local data, often through a finetuning-based approach. This personalized policy can either 1) be fixed, *e.g.* hand-crafting hyperparameters, layers to freeze, selecting number of mixture components, number of clusters or 2) learned, *e.g.* learning a hypernetwork to generate weights or meta-nets to generate hyperparameters. These approaches are also grouped based on whether this personalized policy is dependent on the local data during inference, *e.g.* meta-nets require local client meta-data to generate hyperparameters.

In order to adapt to per-dataset per-client scenarios, many works rely on storing per-client personalized layers, which are trained only on each client's local data. Unfortunately, the memory cost of storing these models scales with the number of clients, C, restricting previous works to small scale experiments. We show that these works are impractical in our CIFAR-10 setup of 1000 clients in Appendix. C. Moreover, most of existing methods rely on a fixed personalized policy, such as deriving shared global hyperparameters for all clients in FLoRA, or do not dependent on local data, such as FedEx which randomly samples per-client hyperparameters from learned categorical distributions. Hence, these methods do not adapt as well to per-dataset per-client scenarios and are ineffective at targeting both label and feature shifts. Lastly, although pFedHN targets both label and feature shift cases, it does not scale well to our experiments as shown in Appendix. C.

Table 5: Positioning of FedL2P with existing FL approaches. C is the total number of clients, M is the number of layers in the model, B is the number of BN layers in the model, H is the number of hidden layers in the hypernetwork.

FL Approach	Learns Shared Model(s)	Personalized Layers	Personalized Policy Obtained by?	Personalized Policy Data Dependent?	Targets Label Shift	Targets Feature Shift	Memory Cost	Scale to Large Networks
FedProx [32] PerFedAvg [15] pFedMe [57] Ditto [33] MOON [31] FedBABU [46]	Yes	No	Fixed	No	Yes	No	O(M)	✓
PerFedMask [52]	Yes	No	Fixed	No	Yes	No	O(M)	✓
FedBN [34]	Yes	Yes	Fixed	No	No	Yes	O(CB)	✓
FedPer [2] FedRep [11] APFL [14] LG-FedAvg [36] IFCA [16]	Yes	Yes	Fixed	No	Yes	No	O(CM)	✗
FLoRA [61]	Yes	No	Fixed	No	Yes	No	O(M)	✓
FedEx [29]	Yes	Supported	Learned	No	Yes	No	O(M)	✓
FedEM [41]	Yes	No	Fixed	No	Yes	No	O(M)	✓
FedFOMO [59] FedMe [42]	No	Yes	Fixed	No	Yes	No	O(CM)	✗
pFedHN [53]	No	Yes	Learned	Yes	Yes	Yes	O(CMH)	✗
FedL2P (Ours)	No	Supported	Learned	Yes	Yes	Yes	O(M)	✓

Most importantly, all existing FL approaches shown in Table 5 can use finetuning either to personalize shared global model(s) or as a complementary personalization strategy to further adapt their personalized models. Since our proposed FedL2P focus on better personalizing shared global model(s) by learning better a personalized policy which leverages the clients' local data statistics relative to the given pretrain model(s), our approach is complementary to all existing FL solutions that learn shared model(s); we showed improvements over a few of these works in Table 1.

Use Cases of FedL2P Mainstream FL focuses on training from scratch, but we focus on federated learning of a strategy to adapt an existing pre-trained model (whether obtained by FL or not) on an unseen group of clients with heterogeneous data. There are several scenarios where this setup and our solution would be useful:

1. Scenarios where it's expensive to train from scratch for a new group of clients, e.g. adopting FedEx [29] from scratch for a new group of clients would require thousands of rounds to retrain the model and HP while our method takes hundreds to learn two tiny meta-nets (Appendix. C).
2. Scenarios where there is a publicly available pre-trained foundation model that can be exploited. This is illustrated in Section 4.4 where we adapt a publicly available pretrained model trained using ImageNet on domain generalization datasets.
3. Scenarios where it's important to also maintain a global model with high initial accuracy - often neglected by previous personalized FL works.

Note that our approach also does not critically depend on the global model's performance. Even in the worst case where the input statistics derived from the global model are junk (e.g., they degenerate to a constant vector, or are simply a random noise vector), then it just means the hyperparameters learned are no longer input-dependent. In this case, FedL2P would effectively learn a constant vector of layer-wise learning rates + BN mixing ratio, as opposed to a function that predicts them. Thus, in this worst case we would lose the ability to customize the hyperparameters differently to different heterogeneous clients, but we would still be better off than the standard approach where these optimization hyperparameters are not learned. In the case where our global-model derived input features are better than this degenerate worst case, FedL2P's meta-nets will improve on this already strong starting point.

C Cost of FedL2P

Computational Cost of Hessian Approximation. We compare with hessian-free approaches, namely first-order (FO) MAML and hessian-free (HF) MAML, both of which are used by PerFedAvg, and measure the time it took to compute the meta-gradient after fine-tuning. Specifically, we run 100 iterations of each algorithm and report the mean of the walltime. Our proposed method takes 0.24 seconds to compute the hypergradient, 0.12 seconds of which is used to approximate the Hessian. In comparison, FO-MAML took 0.08 seconds and HF-MAML took 0.16 seconds to compute the meta-gradient. Hence, our proposed method is not a significant overhead relative to simpler non-Hessian methods. It is also worth noting that the number of fine-tuning epochs would not impact the cost of computing the hypergradient.

Memory Cost. In our CIFAR10 experiments, the meta-update of FedL2P has a peak memory usage of 1.3GB. In contrast, existing FL methods that generate personalized policies require an order(s) of magnitude more memory and hence only evaluated in relatively small setups with smaller networks. For instance, pFedHN [53] requires in a peak memory usage of 17.93GB in our CIFAR10 setup as its user embeddings and hypernetwork scale up with the number of clients and model size. Moreover, they fail to generate reasonable client weights as these techniques do not scale to larger ResNets used in our experiments. APFL [14], on the other hand, requires each client to maintain three models: local, global, and mixed personalized. Adopting APFL in our CIFAR10 setup of 1000 clients requires over 134GB of memory just to store the models per experiment, which is infeasible.

Communication Cost. For each FL round, we transmit the parameters of the meta-nets, which are lightweight MLP networks to from server to client and vice versa. Note that transmitting the global pretrained model to each new client is a one-time cost. Office-Caltech-10 and DomainNet setup typically takes less than 100 communication rounds to obtain a learned λ that leads to the lowest validation loss. CIFAR-10 and CIFAR-10-C, on the other hand, can take hundreds of rounds up to a maximum of 500 rounds. In contrast, joint model and hyperparameter optimization typically takes thousands of rounds [29], having to transmit both the model and the hyperparameter distribution across the network. Although FedL2P incurs additional costs on top of conventional fine-tuning, FedL2P forgoes the cost of federatedly learning a model from scratch and can be advantageous in certain scenarios as listed in Section. B.

Inference Cost. During the fine-tuning stage, given the learned meta-nets, FedL2P requires 2 forward pass of the model per image and one forward pass of each meta-net to compute the personalized hyperparameters. This equates to 0.55GFLOPs per image and would incur a minor additional cost of 4.4% more than the regular finetuning process of 15 finetune epochs.

D Ablation Study

To elucidate the individual impact of BNNet & LRNet, we run an ablation study of all of the datasets used in our experiments and present the results in Table. 6, where CIFAR10 adopts the pretrained model trained using FedAvg. As BNNet learns to weight between client's BN statistics (**BN C**) and pretrained model's BN statistics (**BN G)**, running FedL2P with BNNet alone leads to either better or similar performance to the better performing baseline. Running LRNet as a standalone, on the other hand, can result in further gains, surpassing the use of both BNNet and LRNet on some benchmarks. Nonetheless, it requires prior knowledge of the data feature distribution of the client in order to set a suitable $\boldsymbol{\beta}$, of which $\boldsymbol{\beta} = 1$ uses **BN C** and $\boldsymbol{\beta} = 0$ uses **BN G**. Our approach assumes no knowledge of the client's data and learns an estimated $\boldsymbol{\beta}$ per-scenario and per-client using BNNet.

E Pretrained Model and Setup Details

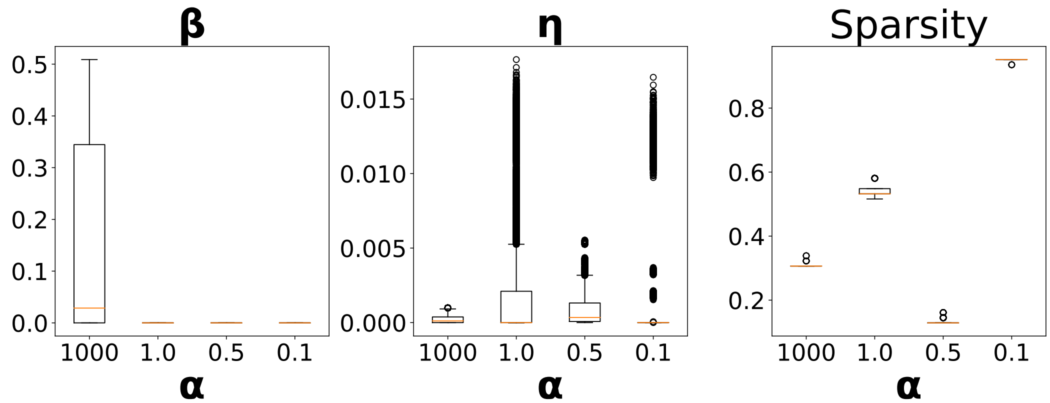
We use the Flower federated learning framework [6] and 8 NVIDIA GeForce RTX 2080 Ti GPUs for all experiments. ResNet-18 [19] is adopted with minor differences in the various setups:

CIFAR-10. We replaced the first convolution with a smaller convolution 3×3 kernel with stride$= 1$ and padding$= 1$ instead of the regular 7×7 kernel. We also replaced the max pooling operation with the identity operation and set the number of output features of the last fully connected layer to 10. The model is pretrained in a federated manner using FedAvg [44] or FedBABU [46] or PerFedAvg(HF) [15] with a starting learning rate of 0.1 for 500 communication rounds. For PerFedAvg, we adopted the recommended hyperparameters used by the authors to meta-train the model. The fraction ratio is set to $r = 0.1$; 100 clients, each of who perform a single epoch update on its own local dataset before sending the updated model back to the server, participate per round. We dropped the learning rate by a factor of 0.1 at round 250 and 375. This process is repeated for each $\alpha = 1000, 1.0, 0.5, 0.1$, resulting in a pretrained model for each group of clients. We experiment with various fine-tuning learning rates $\{1.0, 0.1, 0.01, 1e-3, 1e-4, 1e-5\}$ and pick the best-performing one, $1e-3$ for all experiments; the initial value of $\tilde{\boldsymbol{\eta}}$ in FedL2P is also set at $1e-3$.

CIFAR-10-C. We adopted the pretrained model trained in CIFAR-10 for $\alpha = 1000$ and used the same fine-tuning learning rate for all experiments.

Table 6: Ablation study for FedL2P with $e = 15$.

α	Dataset	+FT (BN C)	+FT (BN G)	+FedL2P (BNNet)	+FedL2P (LRNet) β=1	+FedL2P (LRNet) β=0	+FedL2P
1000	CIFAR-10	63.04±0.02	59.85±0.04	62.35±0.24	62.62±0.21	65.11±0.02	**65.13±0.02**
(↓ heterogeneity)	CIFAR-10-C	59.58±0.03	57.03±0.08	59.57±0.13	**60.09±0.02**	59.30±0.11	59.97±0.22
	Caltech-10	80.97±0.33	36.02±25.21	88.12±1.18	85.50±5.76	42.59±22.87	**88.85±0.89**
	DomainNet	52.17±1.55	30.55±1.07	53.39±0.85	**55.59±2.76**	44.43±3.46	54.38±0.45
1.0	CIFAR-10	61.42±0.13	63.23±0.15	63.75±0.04	64.67±0.06	64.61±0.49	**65.76±0.31**
	CIFAR-10-C	67.37±0.08	66.45±0.03	68.1±0.07	68.62±0.07	67.82±0.1	**68.83±0.15**
	DomainNet	62.27±0.58	44.15±0.11	62.73±0.51	63.69±0.43	diverge	**63.77±0.44**
0.5	CIFAR-10	62.34±0.14	67.4±0.06	67.59±0.15	**68.81±0.05**	68.01±0.29	68.45±0.50
	CIFAR-10-C	74.92±0.08	75.24±0.17	76.36±0.08	**76.86±0.06**	76.11±0.07	76.82±0.19
	DomainNet	71.39±0.97	49.81±1.98	70.99±1.15	**72.74±0.51**	diverge	72.64±0.30
0.1	CIFAR-10	79.15±0.07	78.97±0.07	79.47±0.2	80.24±0.09	**80.39±0.15**	80.28±0.07
(↑ heterogeneity)	CIFAR-10-C	87.25±0.06	88.5±0.02	88.6±0.1	89.08±0.04	89.14±0.13	**89.23±0.15**
	DomainNet	86.03±0.47	69.41±1.95	85.87±1.31	85.78±0.6	diverge	**86.36±0.45**



(a) CIFAR10 ($e = 5$)

Figure 4: Locality, spread, and skewness of FedL2P's learned hyperparameters, $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$, of different layers across clients and the model's sparsity of all clients for each personalization scenario.

Office-Caltech-10 & DomainNet. We adopted a Resnet-18 model that was pretrained on ImageNet [13] and provided by torchvision [39]. We replaced the number of output features of the last fully connected layer to 10. Similar to CIFAR-10 setup, we experiment with the same set of learning rates and pick the best-performing one, $1e-2$ for our experiments.

F Architecture & Initialization Details

We present the architecture of our proposed meta-nets, BNNet and LRNet. Both networks are 3-layer MLP models with 100 hidden layer neurons and ReLU [1] activations in-between layers. BNNet and LRNet clamp the output to a value of $[0, 1]$ and $[0, 1000]$ respectively and use a straight-through estimator [4] (STE) to propagate gradients. We also tried using a sigmoid function for BNNet which converges to the same solution but at a much slower pace. We initialize the weights of BNNet and LRNet with Xavier initialization [17] using the normal distribution with a gain value of 0.1. To control the starting initial value of BNNet and LRNet, we initialize the biases of BNNet and LRNet with constants 0.5 and 1.0, resulting in initial values of ~ 0.5 and ~ 1.0 respectively. We also tried experimenting BNNet with different initializations by setting the biases to $[0.2, 0.5, 0.8]$ and got similar results.

G Additional Results

Relative Clustered Distance Maps. We present an extension of Fig. 3 for both the inputs, $\boldsymbol{\xi}$ & $\boldsymbol{x}$, and outputs, $\boldsymbol{\beta}$ & $\boldsymbol{\eta}$, of the meta-nets in Fig. 5.

Learned Personalized Hyperparameters. We present the learned hyperparameters for the other client groups not shown in Fig. 2 in Fig. 4.