A PROOFS

A.1 PROOF OF THEOREM 3.1

We already know Θ_k^* is the unique minimizer of $\mathcal{R}(\Theta)$ under the constrain that $\mathbf{rank}(\Theta) = k$. We just need to show it is achievable. Given the fact that $\tilde{G}$ being invertible, we could easily have $WV = \Theta_k^* G^{-1}$ which gives $\Theta = \Theta_k^*$.

A.2 PROOF OF THEOREM 3.2

For pFedHN, it will be unable to recover the client representations for all the m lost clients. Thus it will suffered from an error $\mathcal{R}(\Theta) = \frac{1}{n}\sum_{i=1}^{m}\mathcal{R}(\theta_i) = \frac{1}{n}\|W^* v_i^* - W^* v_i\|_F^2$. In expectation, $\mathbb{E}[\|W^* v_i^* - W^* v_i\|_F^2] = \mathbb{E}[\mathrm{tr}((v_i^* - v_i)^\top W^{*\top} W^* (v_i^* - v_i))] = 2dk$. Thus $\mathcal{R}(\Theta) = \frac{2dkm}{n} = 2dk\alpha$.

A.3 PROOF OF THEOREM 3.3

The full loss on all clients can be decomposed as the sum of the loss on attacked clients and the loss on unattacked clients, i.e., $\mathcal{R}(\Theta) = \mathcal{R}_{\texttt{atk}}(\Theta) + \mathcal{R}_{\texttt{uatk}}(\Theta)$ where $\mathcal{R}_{\texttt{atk}}(\Theta) := \frac{1}{n}\|(\Theta^* - \Theta)_{1:m}\|_F^2$ and $\mathcal{R}_{\texttt{uatk}}(\Theta) := \frac{1}{n}\|(\Theta^* - \Theta)_{(m+1):n}\|_F^2$. Recall that we use the subscripts to denote the column indices. For simplicity, we further denote $G_{(m+1:n)}$ and $G_{1:m}$ as P, Q. During training, the unattacked loss get minimized. Using the assumption of client embeddings's consistency with the graph, i.e., $V^*G = V^*$, we have $\mathcal{R}_{\texttt{uatk}}(\Theta) \propto \|W^*(V^* - V)G_{1:m}\|_F^2$. It is easy to see $\mathcal{R}_{\texttt{uatk}}(\Theta)$ is minimized to zero when $\|(V^* - V)P\|_F^2 = 0$. The solution might not unique depending on the rank of P. However, we use SGD to optimize V which has an implicit regularization of the deviation of the final solution $\hat{V}_{\texttt{gnn}}$ and the initialization V_0. Thus, we can have an unique solution $\hat{\Theta}_{\texttt{gnn}} = W^*\hat{V}_{\texttt{gnn}}$ with $\hat{V}_{\texttt{gnn}} := V_0 + (V^* - V_0)PP^\dagger$ where $\dagger$ indicates the Moore-Penrose inverse. The finish loss $\mathcal{R}(\hat{\Theta}_{\texttt{gnn}}) = \mathcal{R}_{\texttt{atk}}(\hat{\Theta}_{\texttt{gnn}}) = \frac{1}{n}\|W^*(V^* - \hat{V}_{\texttt{gnn}})Q\|_F^2$, which in expectation equals the following

$$\mathbb{E}[\mathcal{R}(\hat{\Theta}_{\texttt{gnn}})] = \frac{1}{n}\mathbb{E}[\mathrm{tr}(Q^\top(V^* - \hat{V}_{\texttt{gnn}})^\top \mathbb{E}[W^{*\top} W^*](V^* - \hat{V}_{\texttt{gnn}})Q)] \tag{9}$$

$$= \frac{d}{n}\mathbb{E}[\|(V^* - \hat{V}_{\texttt{gnn}})Q\|_F^2] = \frac{d}{n}\mathbb{E}[\|(V^* - V_0)(I - PP^\dagger)Q\|_F^2] \tag{10}$$

$$= \frac{2dk}{n}\|(I - PP^\dagger)Q\|_F^2 = \frac{2dk}{n}\|(I - G_{(m+1):n}G_{(m+1):n}^\dagger)G_{1:m}\|_F^2 \tag{11}$$

Derivation from equation (10) to equation (11) is rooted from the Guassianality of V^* and V_0 and the independence between V^* and V_0.

A.4 PROOF OF LEMMA 3.1

We first recall the definition of the graph aggregation operation $\tilde{G}$ and the multi-layer aggregation $G = \tilde{G}^L$. We then point out a few properties holds for both $\tilde{G}$ and G. Finally, we prove the necessary and sufficient condition for the lowerbound and upperbound.

Definition 1 (Graph aggregation operation)**.** *$\tilde{G}$ is a simple mean aggregation which averages embeddings according the graph adjacent matrix $\mathbf{A}$, i.e., $[\tilde{G}]_{i,j} = \frac{1}{N(i)+1}1_{[j\in N(i)\cup\{i\}]}$ where $N(i) := \{j|\mathbf{A}_{ij} = 1\}$. $G := \tilde{G}^L$ is operation of aggregating L times.*

Property 1. *(Value range) By definition, all elements in $\tilde{G}$ and G are non-negative and in the range of $[0, 1]$.*

Property 2. *(Column summation) By definition, summation of any column of $\tilde{G}$ is one, i.e., $\mathbf{1}^\top\tilde{G} = \mathbf{1}^\top$. The property also holds for G since $\mathbf{1}^\top G = \mathbf{1}^\top\tilde{G}\cdots\tilde{G} = \mathbf{1}^\top$.*

Property 3. *(Norm) We denote G's columns as g_i, i.e., $G = [g_1, \cdots, g_n]$. We have $\|g_i\|_2^2 \in [1/n, 1]$.*

Proof of property 3. For the norm of g_i is lower bounded via Jensen's inequality as

$$\|g_i\|^2 = \sum_{i,j} g_{i,j}^2 \geq n(\sum_j g_{i,j}/n)^2 = 1/n \tag{12}$$

For the norm of g_i is upper bounded due to the non-negativity of g_i's elements,

$$\|g_i\|^2 = \sum_{i,j} g_{i,j}^2 \leq (\sum_j g_{i,j})^2 = 1 \tag{13}$$

□

Proof of the lower-bound in Lemma. First, it is trivial that $\|(I - G_{(m+1):n}G^{\dagger}_{(m+1):n})G_{1:m}\|_F^2 \geq 0$. Second, $\|(I - G_{(m+1):n}G^{\dagger}_{(m+1):n})G_{1:m}\|_F^2 = 0 \iff G_{1:m} = G_{(m+1):n}G^{\dagger}_{(m+1):n}G_{1:m} \iff \mathbf{col}(G_{1:m}) \subset \mathbf{col}(G_{(m+1):n})$. □

Proof of the upper-bound in Lemma. Because, $G_{(m+1):n}G^{\dagger}_{(m+1):n}G_{1:m}$ is the projection of $G_{1:m}$ in the column space $G_{(m+1):n}G^{\dagger}_{(m+1):n}$, we have $\|(I - G_{(m+1):n}G^{\dagger}_{(m+1):n})G_{1:m}\|_F^2 \leq \|G_{1:m}\|_F^2$. The equality holds when $G_{1:m}^{\top}G_{(m+1):n} = \mathbf{0}$. By property 3, we know $\|g_i\|_2^2 \leq 1$ for any $i \in [m]$. Thus $\|G_{1:m}\|_F^2 \leq m$. The equality holds when each g_i is one-hot vector. Further since $g_{i,i} > 0$ by definition, g_i has to equal e_i which is the i'th standard basis vector. □

B Algorithm Details

Communication cost. Via the chain rules, the gradient of the hypernetwork w.r.t the local task loss can be composed as the following:

$$\nabla_{\psi}\mathcal{L}_t = (\nabla_{\psi}\theta_t)^{\intercal}\nabla_{\theta_t}\mathcal{L}_t. \tag{14}$$

Equation 14 shows that the client will only need to transmit $\nabla_{\theta_t}\mathcal{L}_t$ to the server for updating the graph hypernetwork parameters. Therefore, the communication cost is determined by the size of the local model parameter changes $\triangle\theta_t$ uploaded from the clients to the server and the size of θ_t sent from the server to the clients. It means that our Panacea causes no additional communication cost compared with traditional federated learning methods such as FedAvg McMahan et al. (2017).

C Dataset Details

Car classification dataset. *Comprehensive Cars* (CompCars) Yang et al. (2015) contains 136,726 images of cars with labels including 4 car types (MPV, SUV, sedan, and hatchback), 5 viewpoints (front (F), rear (R), side (S), front-side (FS), and rear-side (RS)), and years of manufacture (YOMs, ranging from 2009 to 2014). We follow the data splitting from Xu et al. (2021), and each client has car images only from one viewpoint and one YOM. The task is to predict the car type based on the image. Two clients are connected if either their viewpoints or YOMs are identical/nearby. For example, client A and B are connected if A's YOM is 2009, and B's is 2010.

Road network traffic datasets. *PEMS-BAY* and *METR-LA* Li et al. (2018); Xu et al. (2021) are two datasets constructed based on the traffic data collected by sensors in the road network. Specifically, *PEMS-BAY* contains the traffic speed readings from 325 sensors in the Bay Area over six months, from January 1st, 2017, to May 31st, 2017. *METR-LA* contains the traffic speed readings from 207 loop detectors in Los Angeles County over 4 months from March 1st, 2012, to June 30th, 2012. To visualize the statistical heterogeneity, we show the histograms of traffic speed from different sensors of PEMS-BAY in Figure 3. We also list the statistics of PEMS-BAY and METR-LA in Table 3.

Table 3: Statistics of *PEMS-BAY* and *METR-LA*.

Dataset	# of Clients	# of Edges	# of Train Seq. Per Client	# of Val Seq. Per Client	# of Test Seq. Per Client
BPEMS-BAY	325	2,369	36,465	5,209	10,419
METR-LA	207	1,515	23,974	3,425	6,850

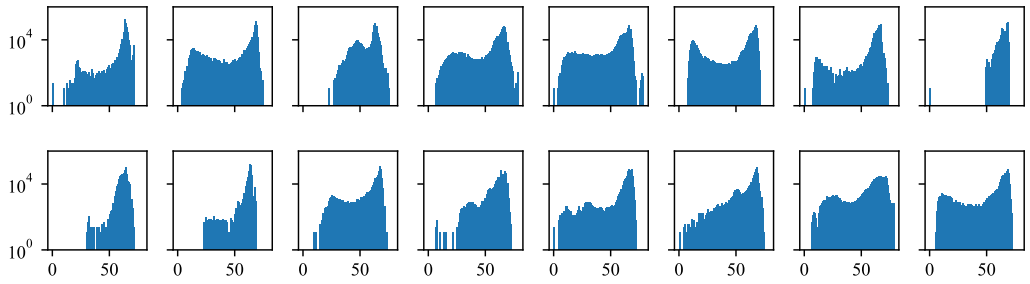


Figure 3: Illustration of data heterogeneity on road network traffic dataset PEMS-BAY. We randomly selected 16 sensors and visualized the distributions of their data values.

D Implementation Details

D.1 Implementation of Baselines

In the following, we introduce the implementation details of the baselines compared in Section 4. We use the original implementations released by the authors if available; otherwise, we re-implement them by referring to the original paper.

FedAvg McMahan et al. (2017). We re-implemented FedAvg by referring to the original paper. On the client side, the clients pull the global model from the server and train it using their own local data. The updated model weights are then uploaded back to the server. On the server side, a subset of clients is randomly selected to perform local training, and their model weight updates are aggregated to generate the new global model.

Per-FedAvg Fallah et al. (2020b). We utilized a third-party's PyTorch implementation at `https://github.com/KarhouTam/Per-FedAvg`, which claims to be implemented based on the source code shared by the original authors. On the client side, Per-FedAvg trains the local model with Hessian-Free MAML (Model-Agnostic Meta-Learnin). On the server side, it performs the same aggregation process as FedAvg.

pFedMe T Dinh et al. (2020). We use the original implementation at `https://github.com/CharlieDinh/pFedMe`. Different from other baselines, pFedMe maintains a personalized model on the client side and uses the Moreau envelope function to help decompose the personalized model optimization from global model learning. On the server side, pFedMe also adopts the same average aggregation as FedAvg.

SFL Chen et al. (2022). We use the original implementation at `https://github.com/dawenzi098/SFL-Structural-Federated-Learning`. To prevent the local model from deviating significantly from the global model, SFL introduces an additional regularization term during the client's local training process. For server aggregation, SFL aggregates the clients' weights from their neighbors and generates the global model by averaging the aggregated weights of all clients.

pFedHN Shamsian et al. (2021). We use the original implementation at `https://github.com/AvivSham/pFedHN`. In pFedHN, the client performs the local training in the same way as in FedAvg. However, the model weights that clients pull from the server are generated by a hypernetwork. On the server side, the hypernetwork is trained to share the knowledge between clients while also personalizing the model for each individual client. Unlike Panacea, pFedHN treats each client equally and independently and realizes the hypernetwork with an MLP.

D.2 Implementation of Panacea

The implementations of the graph hypernetworks. Our framework consists of three modules: a GNN encoder, an MLP, and a graph generator (see Figure 1 in Sec. 3). For evaluation, we use a

3-layer GNN encoder with hidden dimension 100 and a 3-layer MLP for generating local model parameters. The graph generator is instantiated with an inner-product operator. The client embedding dimension was fixed to 100 for all datasets.

The implementations of the target networks. For the binary classification/regression tasks, we realize the target network model with a 3-layer MLP for each client, where the model's hidden dimension is 16. For the forecasting tasks with traffic datasets, we reuse the gated recurrent unit (GRU) model from Cho et al. (2014); Meng et al. (2021), which has 63K parameters and it is a 1-layer GRU with hidden dimension 100. For the image classification task with CompCars dataset, we use the ResNet18 He et al. (2016) model and the weights pre-trained with ImageNet dataset from torchvision maintainers & contributors (2016) as the initial model.

D.3 Hyperparameters and Hardware

Hyperparamters. In all methods, we use a batch size of 64. Unless otherwise stated, we applied a grid search for following hyperparameters: the learning rate of graph hypernetworks was tuned amongst $\{0.001, 0.003, 0.01, 0.03, 0.1\}$, the learning rate of target networks was tuned amongst $\{0.001, 0.003, 0.01, 0.03, 0.1\}$, and the coefficient λ_d was searched in $\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\}$. We used the SGD optimizer Kingma & Ba (2015) for all client local updates. In each communication round, the number of client local steps K_c is set as 50, and the number of server local steps K_s is set as 10. For all algorithms, we limit the training process to 100 server-client communication rounds for CompCars dataset, and 800 rounds for other datasets. At each round, we randomly select 5 clients to participate in the training. For the baseline approaches, we set their hyperparameters as reported in the corresponding paper Shamsian et al. (2021); Chen et al. (2022); McMahan et al. (2017); T Dinh et al. (2020); Fallah et al. (2020b). Without specification, we report the results under the hyperparameters with the best performance overall.

Hardware. We implemented our framework and all baseline approaches in PyTorch 1.12. All clients and servers are simulated on a workstation with 2 x AMD Milan 7413 @ 2.65 GHz CPU, 4 x NVIDIA A100 GPU with 40GB memory, and 3.84TB SSD.

E Results

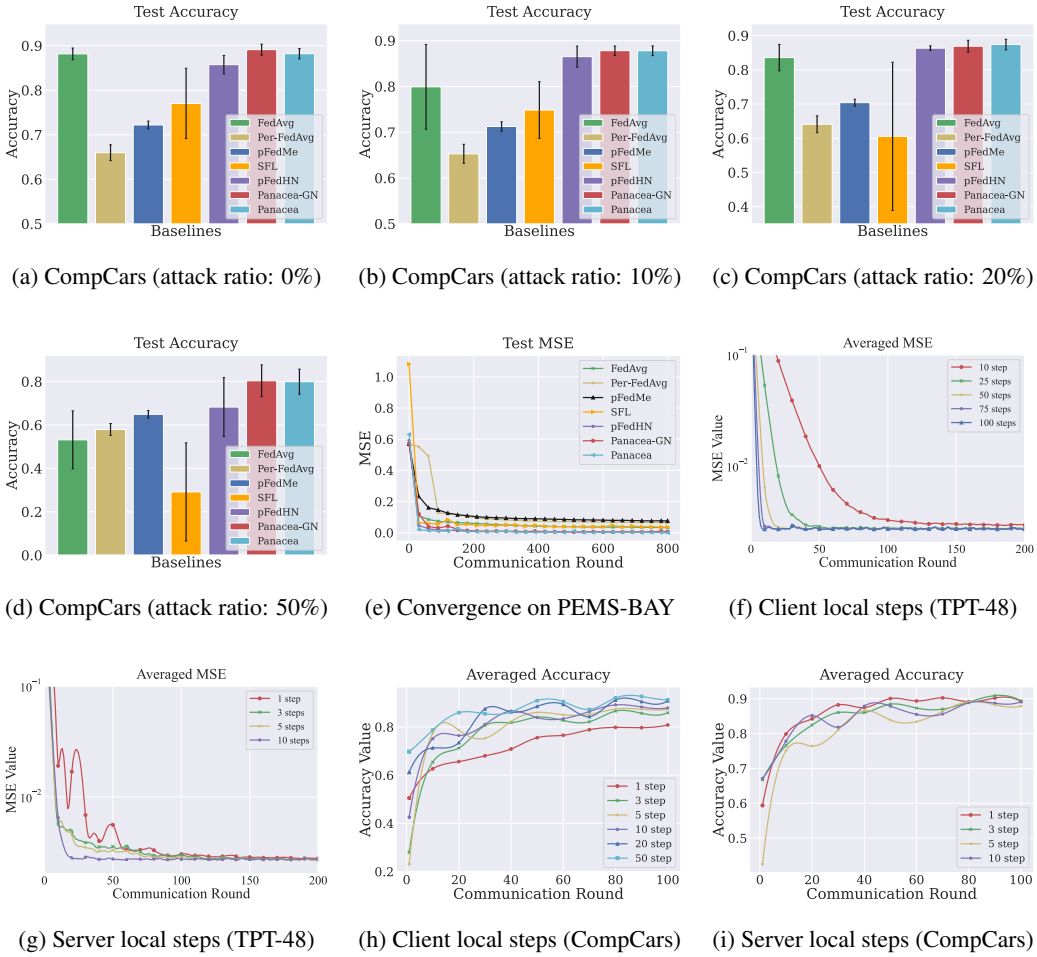
Convergence and performance on PEMS-BAY. Figure 4 (e) shows the task performance during training with different personalized federated learning algorithms on PEMS-BAY. We can observe that PEMS-BAY shows a similar trend as METR-LA, illustrated in the main paper (see Section 4).

Ablation on attack ratio. We conduct robustness evaluation over two classification tasks over FL and CompCars, respectively. Figure 4 (a)-(d) depicts the performance of all approaches under 0%, 10%, 20% and 50% malicious clients, respectively. We can observe that our method, *Panacea* and its variant *Panacea*-GN consistently outperform all the baselines in all settings, showing our robustness to malicious attacks.

Ablation on local steps of clients/server. We also examine the effect of performing local optimization steps of the clients and server. Figure 4 (f)-(i) show the performance on TPT-48 and CompCars throughout the training process for both clients and the server. Specifically, Figure 4 (f) and (h) compare training using the chain rule with various K_c while fixing the server step $K_s = 10$. Using our proposed update rule, i.e., making multiple local update steps yields significant improvements in convergence speed and final accuracy when setting $K_c = 50$. Figure 4 (g) and (i) compare training the graph hypernetwork using the chain rule with various K_s while fixing the client step $K_c = 50$. When setting $K_s = 1$, the performance is poor. We observe that *Panacea* achieves the best in convergence speed and predictive performance when $K_s = 10$. As stated above, we set $K_c = 50$ and $K_s = 10$ while comparing with the baselines.

F Limitations and Potential Negative Societal Impacts

Limitations. Our work sheds light on and facilitates the development of personalized federated learning based on graph hypernetworks. However, *Panacea* assumes that the graph relation of clients



(a) CompCars (attack ratio: 0%) (b) CompCars (attack ratio: 10%) (c) CompCars (attack ratio: 20%)

(d) CompCars (attack ratio: 50%) (e) Convergence on PEMS-BAY (f) Client local steps (TPT-48)

(g) Server local steps (TPT-48) (h) Client local steps (CompCars) (i) Server local steps (CompCars)

Figure 4: (a)-(d) compare final performance under various attack ratios on CompCars; (e) compares convergence speed and final performance on PEMS-BAY; (f)-(i) ablation studies of the number of local optimization steps on the client/server sides (on TPT-48 and CompCars).

is prior; one limitation is that graph relation may not be explicitly observed. Moreover, though our framework can enhance the predictive performance, generalization capability, and robustness in one go, how incorporating graph relation into the federated learning procedure impact the fairness metrics of learned models is unknown. Specifically, how do clients with biased datasets in our framework impact the experienced fairness of other clients on their local distributions? We leave the above issues as our future work.

Potential negative societal impacts. Federated Learning (FL) is a framework intended to uphold the privacy of sensitive personal data by categorically preventing raw data from being disseminated to other users or organizations. Our work illuminates and propels progress in a previously untapped field of personalized federated learning using graph hypernetworks. A potential drawback of our system, named *Panacea*, is that it necessitates knowledge of the graph relation of clients, potentially leading to unintended privacy breaches due to the implications of client adjacency. The client relationships inherently hint at data similarities, thereby posing a privacy risk.

Furthermore, FL demands the active involvement of numerous clients/participants during the learning process. Numerous end-user devices, including mobile phones, smartwatches, and other IoT devices, are utilized as /participants. However, a surge in the number of federated learning participants also escalates the communication frequency between them. This escalation can culminate

in prohibitive communication costs and potentially exacerbate global environmental issues, such as global warming, due to the increased energy consumption.

G Related Literature

Federated learning. The federated learning (FL) setting assumes a federation of distributed devices/clients, each with its local dataset Kairouz et al. (2021). The goal of FL is to collaboratively train statistical models over distributed clients with data confidentiality and communication efficiency. Perhaps the standard and the most known algorithm is FedAvg McMahan et al. (2017) which learns a global model by aggregating local models trained with IID data McMahan et al. (2017). In practice, the clients' data distributions typically exhibit various degrees of heterogeneity Mothukuri et al. (2021); Wang et al. (2020). Many variants have been proposed to tackle statistical heterogeneity issues, such as FedMA Wang et al. (2020). However, training a unique model in FL settings has been shown to be vulnerable to statistical heterogeneity over distributed clients in the literature Kairouz et al. (2021); Tan et al. (2022). More specifically, FL shows poor convergence on highly heterogeneous data, and the statistical heterogeneity deteriorates the performance of the global model on individual clients Li et al. (2019; 2020); Zhao et al. (2018); Karimireddy et al. (2020).

Model-agnostic meta-learning approaches are proposed to learn meta-models for local personalization Fallah et al. (2020b); Chen et al. (2018); Jiang et al. (2019); Khodak et al. (2019); Fallah et al. (2020a); T Dinh et al. (2020). Existing MAML-based approaches usually are computationally intensive as they require computing the Hessian matrix. Recently, multi-task learning (MTL) has emerged as an alternative solution for PFL, where each client was treated as a learning task Smith et al. (2017); Marfoq et al. (2021); Li et al. (2021). Specifically, MOCHA Smith et al. (2017) is proposed to generalize the distributed optimization methods Ma et al. (2015), attempting to address the statistical challenges in federated settings. Marfoq *et al.* Marfoq et al. (2021) proposed federated surrogate optimization algorithms by assuming each local data distribution is a mixture of unknown underlying distributions. However, these works do not explicitly model the inherent dependencies among clients, leading to sub-optimal solutions.

Some prior works implicitly model the statistical dependency among clients by enforcing the client relationship in regularizing model weights Smith et al. (2017). Nevertheless, these works bear the limitation of regularization-based methods due to the assumption that graphs only encode the similarity of clients, and they can not operate in settings where only a fraction of devices are observed during training. Instead, we propose to learn a parametric function approximation referred to as graph hypernetworks, which attempts to generate the personalized model weights directly. The most related to us is pFedHN Shamsian et al. (2021), which used a multi-layer perceptron to generate personalized heterogeneous models. Considering the participants equally limits their ability to deal with the interdependency among the devices. To the best of our knowledge, the work proposed here is the first federated learning framework to leverage hypernetwork to generate personalized models considering the inherent dependency between devices.

Hypernetworks. Hypernetwork refers to a framework in which a neural network is trained to predict the weights of another neural network that performs the tasks of interest. The term "hypernetwork" was first coined in Ha et al. (2017), and it has shown strong performance in large-scale language modeling and image classification tasks. In the literature, hypernetworks have been widely used in many learning tasks, such as neural network architecture search (NAS) Zhang et al. (2019); Brock et al. (2018), molecule property prediction Nachmani & Wolf (2020). Specifically, in NAS, hypernetworks were proposed to generate target networks conditioned on neural network architectures Zhang et al. (2019). In the context of personalized federated learning, hypernetworks could be a feasible solution for generating personalized local models conditioned on the clients' heterogeneous conditions, such as statistical heterogeneity.

The most relevant to ours is the use of Graph Hypernetworks (GHNs) — hypernetworks that take graphs as input. For example, Nachmani *et al.* used graph hypernetworks for molecule property prediction Nachmani & Wolf (2020); Zhang *et al.* leveraged graph hypernetworks for neural architecture search Zhang et al. (2019). To the best of our knowledge, ours is the first work to use graph hypernetworks for generating personalized local models in federated learning settings. In this work, we show that our graph hypernetwork can explicitly model the clients' statistical relationship to en-

sure effective knowledge sharing among clients. By incorporating a graph generator, it can preserve clients' uniqueness to ensure personalized performance.

REFERENCES

Climate Change in the Contiguous United States. URL https://github.com/washingtonpost/data-2C-beyond-the-limit-usa. 2020.

Sawsan AbdulRahman, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. A Survey on Federated Learning: The Journey from Centralized to Distributed On-Site Learning and Beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2020.

Andrew Brock, Theo Lim, JM Ritchie, and Nick Weston. SMASH: One-Shot Model Architecture Search through HyperNetworks. In *International Conference on Learning Representations*, 2018.

Fei Chen, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated Meta-Learning with Fast Convergence and Efficient Communication. *arXiv preprint arXiv:1802.07876*, 2018.

Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang Jiang. Personalized Federated Learning With a Graph. In *International Joint Conferences on Artifical Intelligence (IJCAI)*, 2022.

Kyunghyun Cho, Bart van Merrienboer, **c**Caglar Gül**c**cehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 2014.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting Shared Representations for Personalized Federated Learning. In *International conference on machine learning*, pp. 2089–2099. PMLR, 2021.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020a.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized Federated Learning: A Meta-Learning Approach. *Advances in Neural Information Processing Systems*, 2020b.

Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pp. 301–316, San Sebastian, 2020. USENIX Association.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *Proc. International Conference on Learning Representations (ICLR)*, 2017.

Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Yu Rong, Peilin Zhao, Junzhou Huang, Murali Annavaram, and Salman Avestimehr. FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks. In *Workshop on Graph Neural Networks and Systems*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *ProC. the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Yihan Jiang, Jakub Kone**v**cnỳ, Keith Rush, and Sreeram Kannan. Improving Federated Learning Personalization via Model Agnostic Meta Learning. *arXiv preprint arXiv:1909.12488*, 2019.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic Controlled Averaging for Federated Learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive Gradient-Based Meta-Learning Methods. *Advances in Neural Information Processing Systems*, 32, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. International Conference for Learning Representations (ICLR)*, 2015.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and Robust Federated Learning Through Personalization. In *International Conference on Machine Learning*, pp. 6357–6368. PMLR, 2021.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the Convergence of FedAvg on Non-IID Data. In *International Conference on Learning Representations*, 2019.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*, 2018.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized News Recommendation Based on Click Behavior. In *Proc. the 15th International Conference on Intelligent User Interfaces*, pp. 31–40, 2010.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael Jordan, Peter Richtárik, and Martin Takác. Adding vs. Averaging in Distributed Primal-Dual Optimization. In *International Conference on Machine Learning*, pp. 1973–1982. PMLR, 2015.

TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library. `https://github.com/pytorch/vision`, 2016.

Othmane Marfoq, Giovanni Neglia, Aurélien Bellet, Laetitia Kameni, and Richard Vidal. Federated Multi-Task Learning under A Mixture of Distributions. *Advances in Neural Information Processing Systems*, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. Cross-Node Federated Graph Neural Network for Spatio-Temporal Data Modeling. In *Proc. the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2021.

Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A Survey on Security and Privacy of Federated Learning. *Future Generation Computer Systems*, 115:619–640, 2021.

Eliya Nachmani and Lior Wolf. Molecule Property Prediction and Classification with Graph Hypernetworks. *arXiv preprint arXiv:2002.00240*, 2020.

Xingchao Peng, Yichen Li, and Kate Saenko. Domain2vec: Domain Embedding for Unsupervised Domain Adaptation. In *European Conference on Computer Vision*, pp. 756–774. Springer, 2020.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized Federated Learning using Hypernetworks. In *International Conference on Machine Learning*, pp. 9489–9502. PMLR, 2021.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated Multi-Task Learning. *Advances in Neural Information Processing Systems*, 2017.

Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized Federated Learning with Moreau Envelopes. *Advances in Neural Information Processing Systems*, 2020.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Paul Voigt and Axel Von dem Bussche. The EU General Data Protection Regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

R Vose, S Applequist, M Squires, I Durre, MJ Menne, CN Williams Jr, C Fenimore, K Gleason, and D Arndt. Gridded 5KM GHCN-Daily Temperature and Precipitation Dataset (nCLIMGRID) Version 1. *Maximum Temperature, Minimum Temperature, Average Temperature, and Precipitation*, 2014.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated Learning with Matched Averaging. In *International Conference on Learning Representations*, 2020.

Zihao Xu, Hao He, Guang-He Lee, Bernie Wang, and Hao Wang. Graph-Relational Domain Adaptation. In *International Conference on Learning Representations*, 2021.

Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A Large-Scale Car Dataset for Fine-Grained Categorization and Verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph HyperNetworks for Neural Architecture Search. In *International Conference on Learning Representations*, 2019.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-IID Data. *arXiv preprint arXiv:1806.00582*, 2018.