Personalized Federated Learning through Local Memorization

Othmane Marfoq
Inria, Université Côte d'Azur,
Accenture Labs
Sophia Antipolis, France
othmane.marfoq@inria.fr

Giovanni Neglia
Inria, Université Côte d'Azur,
Sophia Antipolis, France
giovanni.neglia@inria.fr

Laetitia Kameni
Accenture Labs,
Sophia Antipolis, France
laetitia.kameni@accenture.com

Richard Vidal
Accenture Labs,
Sophia Antipolis, France
richard.vidal@accenture.com

Abstract

Federated learning allows clients to collaboratively learn statistical models while keeping their data local. Federated learning was originally used to train a unique global model to be served to all clients, but this approach might be sub-optimal when clients' local data distributions are heterogeneous. In order to tackle this limitation, recent *personalized federated learning* methods train a separate model for each client while still leveraging the knowledge available at other clients. In this work, we exploit the ability of deep neural networks to extract high quality vectorial representations (embeddings) from non-tabular data, e.g., images and text, to propose a personalization mechanism based on local memorization. Personalization is obtained by interpolating a collectively trained global model with a local k-nearest neighbors (kNN) model based on the shared representation provided by the global model. We provide generalization bounds for the proposed approach and we show on a suite of federated datasets that this approach achieves significantly higher accuracy and fairness than state-of-the-art methods.

1 Introduction

Heterogeneity is a core and fundamental challenge in Federated Learning (FL) [32, 23]. Indeed, clients highly differ both in size and distribution of their local datasets (*statistical heterogeneity*), and in their storage and computational capabilities (*system heterogeneity*). Those two aspects challenge the assumption that clients should train a common global model, as pursued in many federated learning papers [39, 28, 43, 24, 40]. Even when clients have similar hardware (e.g., they are all smartphones), statistical heterogeneity may cause a global model to be arbitrarily bad for some clients raising important fairness concerns [31]. In presence of system heterogeneity, the global model also forces all clients to select a common architecture on the basis of the minimum common capabilities.

Motivated by the recent success of memorization techniques based on nearest neighbours in the field of natural language processing, both for language modelling (`kNN-LM` [25]) and neural machine translation (`kNN-MT` [26]), we propose `kNN-Per`, a personalized FL algorithm based on local memorization. `kNN-Per` combines a global model trained collectively (e.g., via `FedAvg` [39]) with a kNN model on a client's local datastore. The global model also provides the shared representation used by the local kNN. Unlike `kNN-LM` and `kNN-MT`, where memorization only allows rare patterns to be memorized explicitly, rather than implicitly in model parameters, local memorization at each FL client can also capture the client's local distribution shift with respect to the global distribution. Indeed, our experiments show that memorization is more beneficial when the distribution shift is larger. The generalization bound in Sec. 3 contributes to justify our empirical findings as well as those in [25, 26].

kNN-Per offers a simple and effective way to address statistical heterogeneity even in a dynamic environment where client's data distributions change after training. It is indeed sufficient to update the local datastore with new data without the need to retrain the global model. Moreover, each client can independently tune the local kNN to its storage and computing capabilities, partially relieving the most powerful clients from the need to align their model to the weakest ones.

The paper is organized as follows. After an overview of related work in Sec. 2, we present kNN-Per in Sec. 3 and provide generalization bounds in Sec. 4. Experimental setup and results are described in Sec. 5 and Sec. 6, respectively.

2 Related Work

We discuss personalized FL approaches to address statistical heterogeneity and system heterogeneity.

2.1 Statistical Heterogeneity

This body of work considers that all clients have the same model architecture but potentially different parameters.

A simple approach for FL personalization is learning first a global model and then fine-tuning its parameters at each client through stochastic gradient descent for a few epochs [48] (we refer later to this approach as FedAvg+). The global model can then be considered as a meta-model to be used as initialization for a few-shot adaptation at each client. Later work [27, 12, 1] has formally established the connection with Model Agnostic Meta Learning (MAML) [20] and proposed different algorithms to train a more suitable meta-model for local personalization.

ClusteredFL [45, 13, 37] assumes that clients can be partitioned into several clusters, with clients in the same cluster sharing the same model, while models can be arbitrarily different across clusters. Clients jointly learn during training the cluster to which they belong as well as the cluster model. FedEM [38] can be considered as a soft clustering algorithm as clients learn personalized models as mixtures of a limited number of component models.

Multi-Task Learning (MTL) allows for more nuanced relations among clients' models by defining federated MTL as a penalized optimization problem, where the penalization term captures clients' dissimilarity. Seminal work [49, 52, 55] proposed algorithms able to deal with quite generic penalization terms, at the cost of learning only linear models or linear combinations of pre-trained models. Other MTL-based algorithms [16, 17, 50, 11, 31, 19, 31] are able to train more general models but consider simpler penalization terms (e.g., the distance to the average model).

An alternative approach is to interpolate a global model and one local model per client [9, 8, 37]. [57] extended this idea by letting each client interpolate the local models of other clients with opportune weights learned during training. Our algorithm, kNN-Per, also interpolates a global and a local model, but the global model plays a double role as it is also used to provide a useful representation for the local kNN.

Closer to our approach, FedRep [7], FedPer [3] and pFedGP [2] jointly learn a global latent representation and local models—linear models for FedRep and FedPer, Gaussian processes for pFedGP—that operate on top of this representation. In these algorithms, the progressive refinement of local models affects the shared representation. On the contrary, in kNN-Per only the global model (and then the shared representation) is the object of federated training, and the shared representation is not influenced by local models, which are learned separately by each client in a second moment. Our experiments suggest that kNN-Per's approach is more efficient. A possible explanation is that jointly learning the shared representation and the local models lead to potentially conflicting and interfering goals. A similar argument was provided by [31] to justify why Ditto replaces, as penalization term, the distance from the average of the local models—as proposed in [16, 17, 50]—with the distance from an independently learned global model. [34] proposed a someway opposite approach to FedRep, FedPer, and pFedGP by using local representations as input to a global model, but the representations and the global model are still jointly learned. An additional advantage of kNN-Per's clear separation between global and local model training is that, because each client does not share any information about its local model with the server, the risk of leaking private information is reduced. In particular, kNN-Per enjoys the same privacy guarantees as FedAvg, and can be easily combined with *differential privacy* techniques [54].

To the best of our knowledge, pFedGP and kNN-Per are the first attempts to learn semi-parametric models in a federated setting. pFedGP relies on Gaussian processes and then has higher computational cost than kNN-Per both at training and inference.

2.2 System Heterogeneity

Some FL application scenarios envision clients with highly heterogeneous hardware, like smartphones, IoT devices, edge computing severs, and the cloud. In this case each client should learn a potentially different model architecture, suited to its capabilities. Such system heterogeneity has been studied much less than statistical heterogeneity.

Some work [35, 30, 58, 56] proposed to address system heterogeneity by distilling the knowledge from a global teacher to clients' student models with different architectures. While early methods [30, 35] required the access to an extra (unlabeled) public dataset, more recent ones [58, 56] eliminated this requirement.

Some papers [10, 18, 41] propose that each client only trains a sub-model of a global model. The sub-model size is determined by the client's computational capabilities. The approach appears particularly advantageous for convolutional neural networks with clients selecting only a limited subset of channels.

[51] followed another approach where devices and server communicate prototypes, i.e., average representations for all samples in a given class, instead of communicating model's gradients or parameters, allowing each client to have a different model architecture and input space.

While in this paper we assume that `kNN-Per` relies on a shared global model, it is possible to replace it with heterogeneous models adapted to the clients' capabilities and jointly trained following one of the methods listed above. Then, `kNN-Per`'s interpolation with a kNN model extends these methods to address not only system heterogeneity, but also statistical heterogeneity.

To the best of our knowledge, the only existing method that takes into account both system and statistical heterogeneity is `pFedHN` [47]. `pFedHN` feeds local clients representations to a global (across clients) hypernetwork, which can output personalized heterogeneous models. Unfortunately, the hypernetwork has a large memory footprint already for small clients' models (e.g., the hypernetwork in the experiments in [47] has 100 more parameters than the output model): it is not clear if `pFedHN` can scale to complex models.

We observe that `kNN-Per`'s kNN model can itself be adapted to client's capabilities by tuning the size of the datastore and/or selecting an appropriate approximate kNN algorithm, like FAISS [21], HNSW [36], or ProtoNN [15] for IoT resource-scarce devices, e.g., based on Arduino.

3 `kNN-Per` Algorithm

In this work we consider M classification (or regression) clients (tasks). Each client $m \in [M]$ has a local dataset $\mathcal{S}_m = \left\{s_m^{(i)} = \left(\mathbf{x}_m^{(i)},\, y_m^{(i)}\right), 1 \leq i \leq n_m\right\}$ with n_m samples drawn i.i.d. from a distribution $\mathcal{D}_m$ over the domain $\mathcal{X}\times\mathcal{Y}$. Local data distributions $\{\mathcal{D}_m\}_{m\in[M]}$ are in general different, thus it is natural to fit a separate model (hypothesis) $h_m \in \mathcal{H}$ to each data distribution $\mathcal{D}_m$. We consider that each hypothesis $h \in H$ is a discriminative model mapping each input $\mathbf{x} \in \mathcal{X}$ to a probability distribution over the set $\mathcal{Y}$, i.e., $h : \mathcal{X} \mapsto \Delta^{|\mathcal{Y}|}$, where Δ^C denotes the unitary simplex of dimension C. A hypothesis then can be interpreted as (an estimation of) a conditional probability distribution $\mathcal{D}(y|\mathbf{x})$.

Personalized FL aims to solve (in parallel) the following optimization problems

$$\forall m \in [M], \quad h_m^* \in \operatorname*{arg\,min}_{h\in\mathcal{H}} \mathcal{L}_{\mathcal{D}_m}(h), \tag{1}$$

where $l : \Delta^{|\mathcal{Y}|} \times \mathcal{Y} \mapsto \mathbb{R}^+$ is the loss function,[1] and $\mathcal{L}_{\mathcal{D}_m}(h_m) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_m}\left[l(h_m(\mathbf{x}), y)\right]$ is the true risk of a model h_m under data distribution $\mathcal{D}_m$.

We suppose that all tasks have access to a global discriminative model $h_{\mathcal{S}}$ minimizing the empirical risk on the aggregated dataset $\mathcal{S} \triangleq \bigcup_{m=1}^{M} \mathcal{S}_m$, i.e.,

$$h_{\mathcal{S}} \in \operatorname*{arg\,min}_{h\in\mathcal{H}} \mathcal{L}_{\mathcal{S}}(h), \tag{2}$$

where $\mathcal{L}_{\mathcal{S}}(h) \triangleq \sum_{m=1}^{M} \frac{n_m}{n} \cdot \frac{1}{n_m} \sum_{i=1}^{n_m} l\left(h\left(\mathbf{x}_m^{(i)}\right), y_m^{(i)}\right)$, and $n = \sum_{m=1}^{M} n_m$. Typically $h_{\mathcal{S}}$ is a feed-forward neural network, jointly trained by the clients using a standard FL algorithm like `FedAvg`.

We also suppose that the global model can be used to compute a fixed-length representation for any input $\mathbf{x} \in \mathcal{X}$, and we use $\phi_{h_{\mathcal{S}}} : \mathcal{X} \mapsto \mathbb{R}^p$ to denote the function that maps the input $\mathbf{x} \in \mathcal{X}$ to its representation. The intermediate representation can be, for example, the output of the last convolutional layer in the case of CNNs, or the last hidden

[1] In the case of (multi-output) regression, we have $h_m : \mathcal{X} \mapsto \mathbb{R}^d$ for some $d \geq 1$ and $l : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^+$.

Algorithm 1 kNN-Per (Typical usage)

```
Learn global model using available clients with FedAvg.
for each client $m \in [M]$ (in parallel) do
    Build datastore using $\mathcal{S}_m$.
    At inference on $\mathbf{x} \in \mathcal{X}$, return $h_{m,\lambda_m}(\mathbf{x})$ given by (7)
end for
```

state in the case of recurrent networks or the output of an arbitrary self-attention layer in the case of transformers. Note that an alternative possible approach would be to separately learn an independent shared representation, e.g., using metric learning techniques [4].

Our method (see Algorithm 1) involves augmenting the global model with a local nearest neighbors' retrieval mechanism at each client. The proposed method does not need any additional training; it only requires a single forward pass over the local dataset $S_m,\ m \in [M]$: client m computes the intermediate representation $\phi_{h_\mathcal{S}}(\mathbf{x})$ for each sample $(\mathbf{x}, y) \in \mathcal{S}_m$. The corresponding representation-label pairs are stored in a local key-value datastore $(\mathcal{K}_m, \mathcal{V}_m)$ that is queried during inference. Formally,

$$(\mathcal{K}_m, \mathcal{V}_m) = \left\{ \left(\phi_{h_\mathcal{S}}\left(\mathbf{x}_m^{(i)}\right), y_m^{(i)} \right), \forall \left(\mathbf{x}_m^{(i)}, y_m^{(i)}\right) \in \mathcal{S}_m \right\} \tag{3}$$

At inference time, given input data $\mathbf{x} \in \mathcal{X}$, client $m \in [M]$ computes $h_\mathcal{S}(\mathbf{x})$ and the intermediate representation $\phi_{h_\mathcal{S}}(\mathbf{x})$. Then, it queries its local datastore $(\mathcal{K}_m, \mathcal{V}_m)$ with $\phi_{h_\mathcal{S}}(\mathbf{x})$ to retrieve its k-nearest neighbors $\mathcal{N}_m^{(k)}(\mathbf{x})$ according to a distance $d(\cdot,\cdot)$:

$$\mathcal{N}_m^{(k)}(\mathbf{x}) = \left(\phi_{h_\mathcal{S}}\left(\mathbf{x}_{\pi_m^{(i)}(\mathbf{x})}\right), y_{\pi_m^{(i)}(\mathbf{x})} \right)_{1 \leq i \leq k}, \tag{4}$$

where $\pi_m^{(1)}(\mathbf{x}), \dots, \pi_m^{(n_m)}(\mathbf{x})$ is a permutation of $[n_m]$ corresponding to the distance of the samples in $\mathcal{S}_m$ from $\mathbf{x}$, i.e., for $i \in [n_m - 1]$,

$$d\Big(\phi_{h_\mathcal{S}}(\mathbf{x}), \phi_{h_\mathcal{S}}\left(\mathbf{x}_{\pi_m^{(i)}(\mathbf{x})}\right)\Big) \leq d\Big(\phi_{h_\mathcal{S}}(\mathbf{x}), \phi_{h_\mathcal{S}}\left(\mathbf{x}_{\pi_m^{(i+1)}(\mathbf{x})}\right)\Big). \tag{5}$$

Then, the client computes a local hypothesis $h_{\mathcal{S}_m}^{(k)}$ which estimates the conditional probability $\mathcal{D}_m(y|\mathbf{x})$ using a kNN method, e.g., with a Gaussian kernel:

$$\left[h_{\mathcal{S}_m}^{(k)}(\mathbf{x})\right]_y \propto \sum_{i=1}^{k} \mathbb{1}\left\{y = y_{\pi_m^{(i)}(\mathbf{x})}\right\} \exp\left\{-d\left(\phi_{h_\mathcal{S}}(\mathbf{x}), \phi_{h_\mathcal{S}}\left(\mathbf{x}_{\pi_m^{(i)}(\mathbf{x})}\right)\right)\right\}. \tag{6}$$

The final decision rule (hypothesis) at client $m \in [M]$ (h_{m,λ_m}) is obtained interpolating the nearest neighbour distribution $h_{\mathcal{S}_m}^{(k)}$ with the distribution obtained from the global model $h_\mathcal{S}$ using a hyper-parameter $\lambda_m \in (0,1)$ to produce the final prediction, i.e.,

$$h_{m,\lambda_m}(\mathbf{x}) \triangleq \lambda_m \cdot h_{\mathcal{S}_m}^{(k)}(\mathbf{x}) + (1 - \lambda_m) \cdot h_\mathcal{S}(\mathbf{x}). \tag{7}$$

As h_{m,λ_m} may not belong to $\mathcal{H}$, we are considering an *improper learning* setting. The parameter λ_m is tuned at client m through a local validation dataset or cross-validation as in [8, 37, 57, 31]. Clients could also use different values k_m and different distance metrics $d_m(\cdot)$, but, in what follows, we consider them equal across clients. Also our experiments in Sec. 6 show that k and $d(\cdot)$ do not require careful tuning.

4 Generalization Bounds

In this section we provide a generalization bound associated with the proposed approach in the case of binary classification, namely $\mathcal{Y} = \{0, 1\}$, when only one neighbour is used for kNN estimation, i.e., $k = 1$, and $d(\cdot,\cdot)$ is the Euclidean distance. For client $m \in [M]$, we denote by $\eta_m : \mathcal{X} \mapsto \mathbb{R}$ the true conditional probability of label 1, that is

$$\eta_m(\mathbf{x}) = \mathcal{D}_m[y = 1|\mathbf{x}]. \tag{8}$$

Our result holds under the following assumptions:

Assumption 1 (Bounded representation). $\phi_{h_{\mathcal{S}}} : \mathcal{X} \mapsto [0,1]^p$.

Assumption 2 (Bounded loss). *$l : \Delta^{|\mathcal{Y}|} \times \mathcal{Y} \mapsto [0,1]$. Moreover, for $y, y' \in \{0,1\}$, $l(\mathbf{e}_y, y') = \mathbb{1}_{y \neq y'}$, where $\mathbf{e}_y \in \Delta^{|\mathcal{Y}|}$ is the vector having all entries equal to 0 except the entry on the y-th coordinate.*

Remark 1. *Loss boundedness is a common assumption (e.g., [37],[46, Ch. 4]). The second requirement is that the maximum loss is achieved when the model is fully confident about a wrong prediction. A simple transformation of common loss functions—e.g., exponentiating the logistic function—make them satisfy Assumption 2.*

Assumption 3 (Loss convexity). *The loss function is convex on the first variable*

$$\forall y_1, y_2 \in \Delta^{|\mathcal{Y}|}, \forall y \in \mathcal{Y}, \forall \quad \lambda_m \in [0,1], l(\lambda_m \cdot y_1 + (1-\lambda_m)\cdot y_2, y) \leq \lambda_m \cdot l(y_1, y) + (1-\lambda_m)\cdot l(y_2, y). \tag{9}$$

Remark 2. *Assumption 3 holds for most loss functions used in supervised machine learning, including the* mean squared error *loss, the* cross-entropy *loss, and the* hinge loss.

Assumption 4. *There exist constants $\gamma_1, \gamma_2 > 0$, such that for any dataset $\mathcal{S}$ drawn from $\mathcal{X} \times \mathcal{Y}$ and any data points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, we have*

$$\left|\eta_m(\mathbf{x}) - \eta_m(\mathbf{x}')\right| \leq d\left(\phi_{h_{\mathcal{S}}}(\mathbf{x}), \phi_{h_{\mathcal{S}}}(\mathbf{x}')\right) \times \left(\gamma_1 + \gamma_2\left(\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}}) - \mathcal{L}_{\mathcal{D}_m}(h_m^*)\right)\right), \tag{10}$$

where $h_m^* \in \arg\min_{h \in \mathcal{H}} \mathcal{L}_{D_m}(h)$.

This assumption means that if two samples $\mathbf{x}$ and $\mathbf{x}'$ have close representations $\phi_{h_{\mathcal{S}}}(\mathbf{x})$ and $\phi_{h_{\mathcal{S}}}(\mathbf{x}')$, then their labels are likely to be the same ($|\eta_m(\mathbf{x}) - \eta_m(\mathbf{x}')|$ is small). This is all the more so, the better $h_{\mathcal{S}}$ predictions are for distribution $\mathcal{D}_m$, $m \in [M]$ (the smaller $\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}}) - \mathcal{L}_{\mathcal{D}_m}(h_m^*)$ is). Experimental results support Assumption 4 (see Figure 3).

Our generalization bound depends, as usual, on the complexity of the hypothesis class $\mathcal{H}$ (expressed by its VC-dimension, $d_{\mathcal{H}}$) and on the size of the local and global datasets (n_m and n, respectively), but also on the distance between the local distribution $\mathcal{D}_m$ and the average distribution $\bar{\mathcal{D}} = \sum_{m=1}^{M} \frac{n_m}{n} \cdot \mathcal{D}_m$, which is the one the global model $h_{\mathcal{S}}$ is targeting (see (2)). The distance between two distributions $\mathcal{D}$ and $\mathcal{D}'$ associated to a hypothesis class $\mathcal{H}$ can be quantified by the *label discrepancy* [37]:

$$\operatorname{disc}_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = \max_{h \in \mathcal{H}} \left|\mathcal{L}_{\mathcal{D}}(h) - \mathcal{L}_{\mathcal{D}'}(h)\right|. \tag{11}$$

Theorem 4.1. *Suppose that Assumptions 1–4 hold, and consider $m \in [M]$ and $\lambda_m \in (0,1)$, then there exist constants c_1, c_2, c_3, c_4, and $c_5 \in \mathbb{R}$, such that*

$$\begin{aligned}\mathop{\mathbb{E}}_{\mathcal{S}\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m}}\left[\mathcal{L}_{\mathcal{D}_m}(h_{m,\lambda_m})\right] &\leq (1+\lambda_m)\cdot\mathcal{L}_{\mathcal{D}_m}(h_m^*) + c_1(1-\lambda_m)\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}}, \mathcal{D}_m\right)\\ &+ c_2\lambda_m\cdot\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\cdot\left(\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}}, \mathcal{D}_m\right)+1\right) + c_3(1-\lambda_m)\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{c_4+\log\left(\frac{n}{d_{\mathcal{H}}}\right)}\\ &+ c_5\lambda_m\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{c_4+\log\left(\frac{n}{d_{\mathcal{H}}}\right)}\cdot\frac{\sqrt{p}}{\sqrt[p+1]{n_m}},\end{aligned} \tag{12}$$

where $d_{\mathcal{H}}$ is the the VC dimension of the hypothesis class $\mathcal{H}$, $n = \sum_{m=1}^{M} n_m$, $\bar{\mathcal{D}} = \sum_{m=1}^{M} \frac{n_m}{n} \cdot \mathcal{D}_m$, p is the dimension of representations, and disc_H is the label discrepancy associated to the hypothesis class $\mathcal{H}$.

The proof of Theorem 4.1 is in Appendix A. We observe that, when clients only use the global model ($\lambda_m = 0$), our generalization bound is analogous to the probabilistic bound in [37, Eq. (2)]. In particular, if data is i.i.d. distributed across the clients ($\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}}, \mathcal{D}_m\right) = 0$), the expected loss decreases with rate $\tilde{\mathcal{O}}\left(\sqrt{\frac{d_{\mathcal{H}}}{n}}\right)$. Instead, when each client only uses the kNN model ($\lambda_m = 1$),[2] we recover the kNN generalization bound in [46, Thm 19.3].

The bound (12) leads to predict that client m should give a larger weight ($\lambda_m > 1/2$) to its kNN model, when n_m exceeds a given threshold, even when local distributions are identical. The bound contributes then to explain why adding a memorization mechanism on top of a pretrained model can improve performance, as observed in [25] and [26]. While it is difficult to quantify the threshold analytically (also because the constants involved depend on γ_1 and γ_2 in Assumption 4), our experiments in Sec. 6 show that even clients with a few tens of samples weigh more the kNN model than the global one.

[2]Note that the kNN model still relies on the representation provided by the global model.

Table 1: Datasets and models.

DATASET	TASK	CLIENTS	TOTAL SAMPLES	MODEL
FEMNIST	HANDWRITTEN CHARACTER RECOGNITION	$3,550$	$805,263$	MOBILENET-V2
CIFAR-10	IMAGE CLASSIFICATION	200	$60,000$	MOBILENET-V2
CIFAR-100	IMAGE CLASSIFICATION	200	$60,000$	MOBILENET-V2
SHAKESPEARE	NEXT-CHARACTER PREDICTION	778	$4,226,158$	STACKED-LSTM

Table 2: Test accuracy: average across clients / bottom decile.

DATASET	LOCAL	FEDAVG	FEDAVG+	CLUSTEREDFL	DITTO	FEDREP	APFL	PFEDGP	KNN-PER (OURS)
FEMNIST	71.0 / 57.5	83.4 / 68.9	84.3 / 69.4	83.7 / 69.4	84.3 / 71.3	85.3 / 72.7	84.1 / 69.4	– / –	**88.2 / 78.8**
CIFAR-10	57.6 / 41.1	72.8 / 59.6	75.2 / 62.3	73.3 / 61.5	80.0 / 66.5	77.7 / 65.2	78.9 / 68.1	– / –	**83.0 / 71.4**
CIFAR-10 (v2)	82.4 / 71.3	67.9 / 60.1	85.0 / 79.6	79.9 / 72.3	86.3 / 80.6	89.1 / 85.3	82.6 / 76.4	88.9 / 84.1	**93.8 / 88.2**
CIFAR-100	31.5 / 19.8	47.4 / 36.0	51.4 / 41.1	47.2 / 36.2	52.0 / 41.4	53.2 / 41.7	51.7 / 41.1	– / –	**55.0 / 43.6**
CIFAR-100 (v2)	45.7 / 38.2	42.3 / 34.8	48.1 / 41.9	43.5 / 37.2	48.7 / 40.3	70.1 / 65.2	48.3 / 42.1	61.1 / 50.0	**74.6 / 67.3**
SHAKESPEARE	32.0 / 16.0	48.1 / 43.1	47.0 / 42.2	46.7 / 41.4	47.9 / 42.6	47.2 / 42.3	45.9 / 42.4	– / –	**51.4 / 45.4**

5 Experimental Setup

We evaluate kNN-Per on five federated datasets spanning a wide range of machine learning tasks: language modeling (Shakespeare [6, 39]), image classification (CIFAR-10 and CIFAR-100 [29]), handwritten character recognition (FEMNIST [6]). Unless otherwise said, kNN-Per's global model $h_{\mathcal{S}}$ is trained by all clients through FedAvg.

Datasets. For Shakespeare and FEMNIST datasets there is a natural way to partition data through clients (by character and by writer, respectively). We relied on common approaches in the literature to sample heterogeneous local datasets from CIFAR-10 and CIFAR-100. We created a federated version of CIFAR-10 by randomly partitioning the dataset among clients using a symmetric Dirichlet distribution, as done in [53]. In particular, for each label y we sampled a vector p_y from a Dirichlet distribution of order $M = 200$ and parameter $\alpha = 0.3$ (unless otherwise specified) and allocated to client m a $p_{y,m}$ fraction of all training instances of class y. The approach ensures that the number of data points and label distributions are unbalanced across clients. For CIFAR-100, we exploit the availability of "coarse" and "fine" label structure, to partition the dataset using Pachinko allocation method [33] as in [42]. The method generates local datasets with heterogeneous distributions by combining a per-client Dirichlet distribution with parameter $\alpha = 0.3$ (unless otherwise specified) over the coarse labels and a per-coarse-label Dirichlet distribution with parameter $\beta = 10$ over the corresponding fine labels. We also partitioned CIFAR-10 and CIFAR-100 in a different way following [2]: each client has only samples from two and ten classes for CIFAR-10 and CIFAR-100, respectively. We refer to the resulting datasets as CIFAR-10 (v2) and CIFAR-100 (v2). For FEMNIST and Shakespeare, we randomly split each local dataset into training (60%), validation (20%), and test (20%) sets. For CIFAR-10 and CIFAR-100, we maintained the original training/test data split and used 20% of the training dataset as validation dataset. Table 1 summarizes datasets, models and number of clients.

Models and representations. For CIFAR-100, CIFAR-10, and FEMNIST, we used MobileNet-v2 [44] as a base model with the output of the last hidden layer—a 1280-dimensional vector—as representation. For Shakespeare, the base model was a stacked LSTM model with two layers, each of them with 256 units; a 1024-dimensional representation was obtained by concatenating the hidden states and the cell states.

Benchmarks. We compared kNN-Per with locally trained models (with no collaboration across clients) and FedAvg [39], as well as with one method for each of the personalization approaches described in Sec. 2, namely, federated averaging with local tuning (FedAvg+) [20],[3] ClusteredFL [45], Ditto [31], FedRep [7], APFL [9] and pFedGP [2].[4] For each method, and each dataset, we tuned the learning rate via grid search on the values $\{10^{-0.5}, 10^{-1}, 10^{-1.5}, 10^{-2}, 10^{-2.5}\}$. FedPer's learning rate for network heads' training was separately tuned on the same grid. Ditto's penalization parameter λ_m was selected among the values $\{10^1, 10^0, 10^{-1}, 10^{-2}\}$ on a per-client basis. For ClusteredFL, we used the same values of tolerance specified in its official implementation [45]. We found tuning tol1 and tol2 particularly hard: no empirical rule is provided in [45], and the few random settings we tried did

[3] We also implemented the more sophisticated first-order MAML approach from [12], but had worse performance than FedAvg+.

[4] We were able to run the official pFedGP's code (`https://github.com/IdanAchituve/pFedGP`) only on datasets partitioned as in [2].

Table 3: Average test accuracy across clients unseen at training (train accuracy between parentheses).

Dataset	FedAvg	FedAvg+	ClusteredFL	Ditto	FedRep	APFL	pFedGP	kNN-Per (Ours)
FEMNIST	83.1 (83.3)	84.2 (88.5)	83.2 (86.0)	83.9 (86.9)	85.4 (88.9)	84.2 (85.5)	–	**88.1** (90.5)
CIFAR-10	72.9 (72.8)	75.3 (78.2)	73.9 (76.2)	79.7 (84.3)	76.4 (79.5)	79.2 (80.6)	–	**82.4** (87.1)
CIFAR-10 (v2)	67.5 (68.1)	85.1 (85.0)	79.6 (79.9)	85.9 (86.0)	89.0 (89.1)	82.3 (82.5)	89.0 (88.8)	**93.0** (93.1)
CIFAR-100	47.1 (47.5)	50.8 (53.4)	47.1 (48.2)	52.1 (57.3)	53.5 (58.2)	49.1 (52.7)	–	**56.1** (59.3)
CIFAR-100 (v2)	42.1 (42.2)	47.9 (48.1)	43.2 (43.4)	48.8 (48.5)	69.8 (70.0)	48.2 (48.4)	61.3 (61.0)	**74.3** (74.5)
Shakespeare	49.0 (48.3)	49.3 (48.1)	49.4 (46.7)	48.1 (49.2)	48.7 (47.8)	46.1 (52.7)	–	**50.7** (64.2)

not show any improvement in comparison to the default ones. For APFL, the mixing parameter α was tuned via grid search on the grid $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. For pFedGP, we used the same hyperparameters as in [2]. The parameter λ_m of kNN-Per was tuned for each client via grid search on the grid $\{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$, and the number of neighbours was set to $k = 10$. Once the optimal hyperparameters' values were selected, models were retrained on the concatenation of training and the validation sets.

Training details. In all experiments with CIFAR-10 and CIFAR-100, training spanned 200 rounds with full clients' participation at each round for all methods. The learning rate was reduced by a factor 10 at round 100 and then again at round 150. For Shakespeare, 10% of clients were sampled uniformly at random without replacement at each round, and we trained for 300 rounds with a constant learning rate. For FEMNIST, 5% of the clients participated at each round for a total 1000 rounds, with the learning rate dropping by a factor 10 at round 500 and 750. In all our experiments we employed the following aggregation scheme

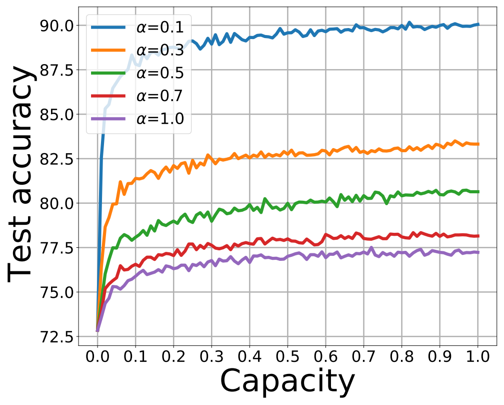
$$\mathbf{w}_{t+1} = \sum_{m \notin \mathbb{S}_t} \frac{n_m}{n} \mathbf{w}_t + \sum_{m \in \mathbb{S}_t} \frac{n_m}{n} \mathbf{w}_t^m, \tag{13}$$

where $\mathbf{w}_t$, $\mathbf{w}_t^m$, and $\mathbb{S}_t$ denote, respectively, the global model, the updated model at client m, and the set of clients participating to training at round t.

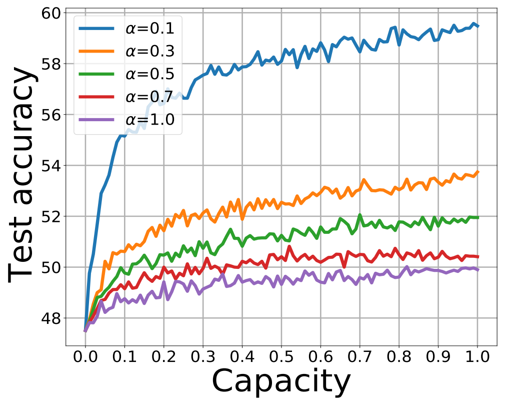
kNN retrieval relied on FAISS library [21].

6 Experiments

Average performance of personalized models. The performance of each personalized model (which coincides with the global one in the case of FedAvg) is evaluated on the local test dataset (unseen at training). Table 2 shows the average weighted accuracy with weights proportional to local dataset sizes. kNN-Per consistently achieves the highest accuracy across all datasets.

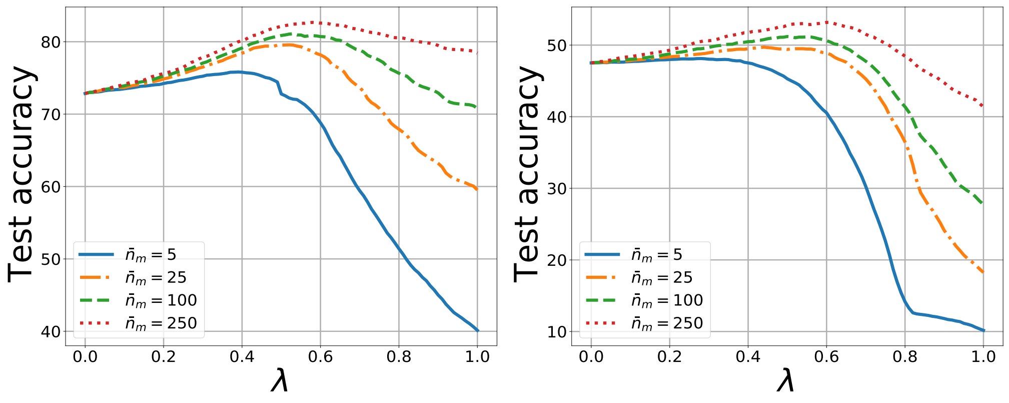


(a) CIFAR-10



(b) CIFAR-100

Figure 1: Accuracy vs capacity (local datastore size). The capacity is normalized with respect to the initial size of the client's dataset partition. Smaller values of α correspond to more heterogeneous data distributions across clients.



(a) CIFAR-10 (b) CIFAR-100

Figure 2: Accuracy vs the interpolation parameter λ (shared across clients) for different average local dataset sizes. For $\lambda = 1$ (resp. $\lambda = 0$) the client uses only the kNN model (resp. the global model).

Fairness across clients. Table 2 also shows the bottom decile of the accuracy of personalized models, i.e., the $(M/10)$-th worst accuracy (the minimum accuracy is particularly noisy, notably because some local test datasets are very small). We observe that even clients with the worst personalized models are still better off when `kNN-Per` is used for training.

Generalization to unseen clients. An advantage of `kNN-Per` is that a "new" client arriving after training may easily learn a personalized model: it may simply retrieve the global model (whose training it did not participate to) from the orchestrator and use it to build the local datastore for kNN. Even if this scenario was not explicitly considered in their original papers, other personalized FL methods can also be adapted to new clients as follows. `FedAvg`+ personalizes the global model through stochastic gradient updates on the new client's local dataset. `Ditto` operates similarly, but maintains a penalization term proportional to the distance between the personalized model and the global model. `FedRep` trains the network head using the local dataset, while freezing the body as in the global model. For `pFedGP` new clients inherit the previously trained shared network and compute their local kernel. `ClusteredFL` assigns the new client to one learned cluster model using a held-out validation set. In the case of `FedAvg`, there is no personalization and the new client uses directly the global model. We performed an experiment where only 80% of the clients participated to the training and the remaining 20% joined later. Results in Table 3 show that, despite its simplicity in dealing with new clients, `kNN-Per` still outperforms all other methods.

Effect of local dataset's size. Beside its relevance for some practical scenarios, the distinction between old and new clients also helps us to evaluate how different factors contribute to the final performance of `kNN-Per`. For example, to understand how the size of the local dataset affects performance, we reduced proportionally the size of new clients' local datasets, while maintaining unchanged the global model, which was trained on old clients. Figure 1 shows that new clients still reap most of `kNN-Per`'s benefits even if their local datastore is reduced by a factor 3. Note that if we had changed the local dataset sizes also for old clients, the global model (and then the representation) would have changed too, making it difficult to isolate the effect of the local datastore size.

Effect of data heterogeneity. Figure 1 also shows that, as expected, the benefit of the memorization mechanism is larger when data distributions are more heterogeneous (smaller α). `kNN-Per` appears then to be a very effective approach to address statistical heterogeneity.

Hyperparameters. `kNN-Per`'s performance is not highly sensitive to the value k which can be selected between 7 and 14 for CIFAR-10 and between 5 and 12 for CIFAR-100 with less than 0.2 percentage points of accuracy variation (see Figure 4 in Appendix B). Similarly, scaling the Euclidean distance by a constant factor σ has almost no effect for values of σ between 0.1 and 1000 (see Figure 5 in Appendix B). The interpolation parameter λ_m plays a more important role. Experiments in Appendix B (Figure 6) show that, as expected, the larger the local dataset, the more clients rely on the local kNN model. Interestingly, clients give a larger weight to the kNN model than to the global one ($\lambda > 1/2$) for datasets with just one hundred samples (Figure 2).

Effect of global model's quality. Assumption 4 stipulates that the smaller the expected loss of the global model, the better representations' distances capture the variability of $\mathbf{x} \mapsto \mathcal{D}_m(\cdot|\mathbf{x})$ and then the more accurate the kNN model. This effect is quantified by Lemma A.2, where the loss of the local memorization mechanism is upper bounded by a

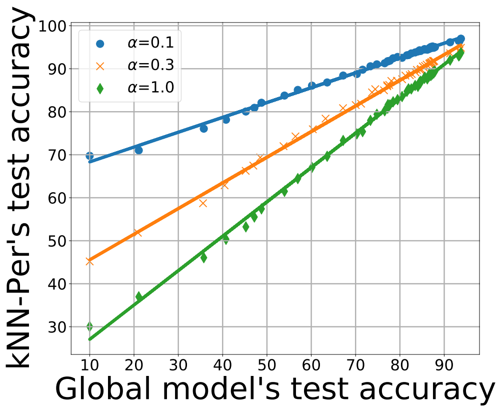


Figure 3: Effect of the global model quality on the test accuracy of `kNN-Per` with $\lambda = 1$. CIFAR-10.

term that depends linearly on the loss of the global model. In order to validate this assumption, we study the relation between the test accuracies of the global model and `kNN-Per`. In particular, we trained a global model for CIFAR-10, in a centralized way, and we save the weights at different stages of the training, leading to global models with different accuracy. Figure 3 shows the test accuracy of `kNN-Per` with $\lambda = 1$ (i.e., when only the kNN predictor is used) as a function of the global model's test accuracy for different levels of heterogeneity. We observe that, quite unexpectedly, the relation between the two accuracies is almost linear. Additional experiments (also including CIFAR-100) in Appendix B confirm these findings.

Appendix B also includes experiments to evaluate the effect of system heterogeneity, the possibility to use aggressive nearest neighbours compression techniques like `ProtoNN` [15], and the robustness to distribution shifts over time.

7 Conclusion

In this paper, we showed that local memorization at each client is a simple and effective way to address statistical heterogeneity in federated learning. In particular, while a global model trained with classic FL techniques, like `FedAvg`, may not deliver accurate predictions at each client, it may still provide a good representation of the input, which can be advantageously used by a local kNN model. This finding suggests that combining memorization techniques with neural networks has additional benefits other than those highlighted in the seminal papers [14, 22] and the recent applications to natural language processing [25, 26].

The better performance of `kNN-Per` in comparison to `FedRep` and `pFedGP` show that jointly learning the shared representation and the local models (as `FedRep` and `pFedGP` do) may lead to potentially conflicting and interfering goals, but further study is required to understand this interaction. Semi-parametric learning [5] could be the right framework to formalize this problem, but its extension to a federated setting is still unexplored.

References

[1] Durmus Alp Emre Acar et al. "Debiasing Model Updates for Improving Personalized Federated Training". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 21–31. URL: `https://proceedings.mlr.press/v139/acar21a.html`.

[2] Idan Achituve et al. "Personalized Federated Learning with Gaussian Processes". In: *Advances in Neural Information Processing Systems* 34 (2021).

[3] Manoj Ghuhan Arivazhagan et al. "Federated Learning with Personalization Layers". In: *ArXiv* abs/1912.00818 (2019).

[4] Aurélien Bellet, Amaury Habrard, and Marc Sebban. *Metric Learning*. Vol. 9. Synthesis Lectures on Artificial Intelligence and Machine Learning 1. Morgan & Claypool Publishers (USA), Synthesis Lectures on Artificial Intelligence and Machine Learning, pp 1-151, Jan. 2015, pp. 1–151. DOI: `10.2200/S00626ED1V01Y201501AIM030`. URL: `https://hal.archives-ouvertes.fr/hal-01121733`.

[5] Peter J Bickel et al. *Efficient and adaptive estimation for semiparametric models*. Vol. 4. Johns Hopkins University Press Baltimore, 1993.

[6] Sebastian Caldas et al. "Leaf: A benchmark for federated settings". In: *arXiv preprint arXiv:1812.01097* (2018).

[7] Liam Collins et al. "Exploiting Shared Representations for Personalized Federated Learning". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 2089–2099. URL: `https://proceedings.mlr.press/v139/collins21a.html`.

[8] Luca Corinzia and Joachim M. Buhmann. *Variational Federated Multi-Task Learning*. 2019. arXiv: `1906.06268 [cs.LG]`.

[9] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. "Adaptive Personalized Federated Learning". In: *arXiv preprint arXiv:2003.13461* (2020).

[10] Enmao Diao, Jie Ding, and Vahid Tarokh. "HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients". In: *International Conference on Learning Representations*. 2020.

[11] Canh T Dinh et al. "FedU: A Unified Framework for Federated Multi-Task Learning with Laplacian Regularization". In: *arXiv preprint arXiv:2102.07148* (2021).

[12] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 3557–3568. URL: `https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf`.

[13] Avishek Ghosh et al. "An Efficient Framework for Clustered Federated Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19586–19597. URL: `https://proceedings.neurips.cc/paper/2020/file/e32cc80bf07915058ce90722ee17bb71-Paper.pdf`.

[14] Edward Grefenstette et al. "Learning to Transduce with Unbounded Memory". In: *NIPS*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1828–1836. URL: `http://dl.acm.org/citation.cfm?id=2969442.2969444`.

[15] Chirag Gupta et al. "ProtoNN: Compressed and Accurate kNN for Resource-scarce Devices". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 1331–1340. URL: `https://proceedings.mlr.press/v70/gupta17a.html`.

[16] Filip Hanzely and Peter Richtárik. *Federated Learning of a Mixture of Global and Local Models*. 2020. arXiv: `2002.05516 [cs.LG]`.

[17] Filip Hanzely et al. "Lower Bounds and Optimal Algorithms for Personalized Federated Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 2304–2315. URL: `https://proceedings.neurips.cc/paper/2020/file/187acf7982f3c169b3075132380986e4-Paper.pdf`.

[18] Samuel Horváth et al. "FjORD: Fair and Accurate Federated Learning under heterogeneous targets with Ordered Dropout". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: `https://openreview.net/forum?id=4fLr7H5D_eT`.

[19] Yutao Huang et al. "Personalized cross-silo federated learning on non-iid data". In: 2021.

[20] Yihan Jiang et al. "Improving federated learning personalization via model agnostic meta learning". In: *arXiv preprint arXiv:1909.12488* (2019).

[21] Jeff Johnson, Matthijs Douze, and Herve Jegou. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* (2019), pp. 1–1.

[22] Armand Joulin and Tomas Mikolov. "Inferring Algorithmic Patterns with Stack-augmented Recurrent Nets". In: *NIPS*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 190–198. URL: `http://dl.acm.org/citation.cfm?id=2969239.2969261`.

[23] Peter Kairouz et al. "Advances and open problems in federated learning". In: *arXiv preprint arXiv:1912.04977* (2019).

[24] Sai Praneeth Karimireddy et al. "Scaffold: Stochastic controlled averaging for federated learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5132–5143.

[25] Urvashi Khandelwal et al. "Generalization through Memorization: Nearest Neighbor Language Models". In: *International Conference on Learning Representations*. 2019.

[26] Urvashi Khandelwal et al. "Nearest neighbor machine translation". In: *arXiv preprint arXiv:2010.00710* (2020).

[27] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. "Adaptive gradient-based meta-learning methods". In: *Advances in Neural Information Processing Systems*. 2019, pp. 5917–5928.

[28] Jakub Konečny et al. "Federated learning: Strategies for improving communication efficiency". In: *arXiv preprint arXiv:1610.05492* (2016).

[29] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[30] Daliang Li and Junpu Wang. "Fedmd: Heterogenous federated learning via model distillation". In: *arXiv preprint arXiv:1910.03581* (2019).

[31] Tian Li et al. "Ditto: Fair and robust federated learning through personalization". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6357–6368.

[32] Tian Li et al. "Federated learning: Challenges, methods, and future directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60.

[33] Wei Li and Andrew McCallum. "Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 577–584. ISBN: 1595933832. DOI: `10.1145/1143844.1143917`. URL: `https://doi.org/10.1145/1143844.1143917`.

[34] Paul Pu Liang et al. "Think locally, act globally: Federated learning with local and global representations". In: *arXiv preprint arXiv:2001.01523* (2020).

[35] Tao Lin et al. "Ensemble Distillation for Robust Model Fusion in Federated Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 2351–2363. URL: `https://proceedings.neurips.cc/paper/2020/file/18df51b97ccd68128e994804f3eccc87-Paper.pdf`.

[36] Yu A Malkov and DA Yashunin. "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4 (2020), pp. 824–836.

[37] Yishay Mansour et al. "Three approaches for personalization with applications to federated learning". In: *arXiv preprint arXiv:2002.10619* (2020).

[38] Othmane Marfoq et al. "Federated Multi-Task Learning under a Mixture of Distributions". In: *arXiv preprint arXiv:2108.10252* (2021).

[39] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 1273–1282.

[40] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. "Agnostic federated learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4615–4625.

[41] Amaury Bouchra Pilet, Davide Frey, and François Taïani. "Simple, Efficient and Convenient Decentralized Multi-task Learning for Neural Networks." In: *IDA*. 2021, pp. 37–49.

[42] Sashank J. Reddi et al. "Adaptive Federated Optimization". In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=LkFG3lB13U5`.

[43] Anit Kumar Sahu et al. "On the Convergence of Federated Optimization in Heterogeneous Networks". In: *ArXiv* abs/1812.06127 (2018).

[44] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[45] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. "Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[46] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[47] Aviv Shamsian et al. "Personalized Federated Learning using Hypernetworks". In: *ICML*. 2021.

[48] Khe Chai Sim, Petr Zadrazil, and Françoise Beaufays. "An investigation into on-device personalization of end-to-end automatic speech recognition models". In: *arXiv preprint arXiv:1909.06678* (2019).

[49] Virginia Smith et al. "Federated Multi-Task Learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4427–4437. ISBN: 9781510860964.

[50] Canh T. Dinh, Nguyen Tran, and Josh Nguyen. "Personalized Federated Learning with Moreau Envelopes". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 21394–21405. URL: `https://proceedings.neurips.cc/paper/2020/file/f4f1f13c8289ac1b1ee0ff176b56fc60-Paper.pdf`.

[51] Yue Tan et al. "FedProto: Federated Prototype Learning across Heterogeneous Clients". In: *AAAI Conference on Artificial Intelligence*. 2022.

[52] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. "Decentralized Collaborative Learning of Personalized Models over Networks". In: *AISTATS*. 2017.

[53] Hongyi Wang et al. "Federated Learning with Matched Averaging". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BkluqlSFDS.

[54] Kang Wei et al. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3454–3469. DOI: 10.1109/TIFS.2020.2988575.

[55] Valentina Zantedeschi, Aurélien Bellet, and Marc Tommasi. "Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs". In: ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. Online: PMLR, Aug. 2020, pp. 864–874. URL: http://proceedings.mlr.press/v108/zantedeschi20a.html.

[56] Lan Zhang and Xiaoyong Yuan. "FedZKT: Zero-Shot Knowledge Transfer towards Heterogeneous On-Device Models in Federated Learning". In: *arXiv preprint arXiv:2109.03775* (2021).

[57] Michael Zhang et al. "Personalized Federated Learning with First Order Model Optimization". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=ehJqJQk9cw.

[58] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. "Data-Free Knowledge Distillation for Heterogeneous Federated Learning". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 12878–12889. URL: https://proceedings.mlr.press/v139/zhu21b.html.

A Proofs

In the general description of kNN-Per, and in our experiments, we considered that each client $m \in [M]$ uses its whole dataset $\mathcal{S}_m$ both to train the base shared model $h_{\mathcal{S}}$—and the corresponding representation function $\phi_{h_{\mathcal{S}}}$—and to populate the local datastore.

In the analysis, for simplicity, we deviate by this operation and consider that each local dataset $\mathcal{S}_m$ is split in two disjoint parts ($\mathcal{S}_m = \mathcal{S}'_m \dot{\bigcup} \mathcal{S}''_m$), with $\mathcal{S}'_m$ used to train the base model and $\mathcal{S}''_m$ used to populate the local datastore. Moreover, we assume that the two parts have the same size, i.e., $n'_m = n''_m = n_m/2$ for all $m \in [M]$, where n'_m and n''_m denote the size of $\mathcal{S}'_m$ and $\mathcal{S}''_m$, respectively. In general, the result holds if the two parts have a fixed relative size across clients (i.e., $n'_{m_1}/n_{m_1} = n'_{m_2}/n_{m_2}$ for all m_1 and m_2 in $[m]$).

Let $\mathcal{S}'$ denote the whole data used to train the base model, i.e., $\mathcal{S}' = \bigcup_{m\in[M]} \mathcal{S}'_m$. We observe that the base model $h_{\mathcal{S}}$ is only function of $\mathcal{S}'$, and then we can write $h_{\mathcal{S}'}$. Instead, the local model $h^{(1)}_{\mathcal{S}_m}$ is both a function of $\mathcal{S}'$ (used to learn the shared representation $\phi'_{\mathcal{S}}$) and of $\mathcal{S}''_m$ (used to populate the datastore). In order to stress such dependence, we then write $h^{(1)}_{\mathcal{S}''_m,\mathcal{S}'}$.

A.1 Proof of Theorem 4.1

Theorem 4.1. *Suppose that Assumptions 1–4 hold, and consider $m \in [M]$ and $\lambda_m \in (0,1)$, then there exist constants c_1, c_2, c_3, c_4, and $c_5 \in \mathbb{R}$, such that*

$$\begin{aligned}
\underset{\mathcal{S}\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m}}{\mathbb{E}} \left[\mathcal{L}_{\mathcal{D}_m}\left(h_{m,\lambda_m}\right)\right] \leq (1+\lambda_m)\cdot\mathcal{L}_{\mathcal{D}_m}\left(h^*_m\right) + c_1\left(1-\lambda_m\right)\cdot\left(\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right)+1\right) \\
+ c_2\lambda_m\cdot\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right) + c_3\left(1-\lambda_m\right)\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{c_4+\log\left(\frac{n}{d_{\mathcal{H}}}\right)} \\
+ c_5\lambda_m\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{c_4+\log\left(\frac{n}{d_{\mathcal{H}}}\right)}\cdot\frac{\sqrt{p}}{\sqrt[p+1]{n_m}},
\end{aligned} \tag{14}$$

where $d_{\mathcal{H}}$ is the the VC dimension of the hypothesis class $\mathcal{H}$, $n = \sum_{m=1}^{M} n_m$, $\bar{\mathcal{D}} = \sum_{m=1}^{M}\frac{n_m}{n}\cdot\mathcal{D}_m$, p is the dimension of representations, and disc_H is the label discrepancy associated to the hypothesis class $\mathcal{H}$.

Proof. The idea of the proof is to bound both the expected error of the *shared* base model (Lemma A.1) and the error of the local kNN retrieval mechanism (Lemma A.2) before using the convexity of the loss function to bound the error of h_{m,λ_m}.

Consider $\mathcal{S} \sim \otimes_{m=1}^{M}\mathcal{D}_m^{n_m}$ or, equivalently, $\mathcal{S} = \mathcal{S}' \cup \mathcal{S}''$, where $\mathcal{S}' \sim \otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}$, and $\mathcal{S}'' = \cup_{m\in[M]}\mathcal{S}''_m$ and $\mathcal{S}''_m \sim \mathcal{D}_m^{n_m/2}$.

For $m \in [M]$, and $\lambda_m \in (0,1)$, we have

$$h_{m,\lambda_m} = \lambda_m \cdot h^{(1)}_{\mathcal{S}''_m,\mathcal{S}'} + (1-\lambda_m)\cdot h_{\mathcal{S}'}. \tag{15}$$

From Assumption 3 and the linearity of the expectation, it follows

$$\mathcal{L}_{\mathcal{D}_m}\left(h_{m,\lambda_m}\right) \leq \lambda_m\cdot\mathcal{L}_{\mathcal{D}_m}\left(h^{(1)}_{\mathcal{S}''_m,\mathcal{S}'}\right) + (1-\lambda_m)\cdot\mathcal{L}_{\mathcal{D}_m}\left(h_{\mathcal{S}'}\right). \tag{16}$$

Using Lemma A.2 and Lemma A.1, and applying expectation over samples $\mathcal{S} \sim \otimes_{m=1}^{M} \mathcal{D}_m^{n_m}$, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathcal{S}\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m}}\left[\mathcal{L}_{\mathcal{D}_m}\left(h_{m,\lambda_m}\right)\right] &\leq \lambda_m \cdot \mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathop{\mathbb{E}}_{\mathcal{S}''\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathcal{L}_{\mathcal{D}_m}\left(h^{(1)}_{\mathcal{S}''_m,\mathcal{S}'}\right)\right]\right] \\
&\quad + (1-\lambda_m)\cdot \mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathop{\mathbb{E}}_{\mathcal{S}''\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathcal{L}_{\mathcal{D}_m}\left(h_{\mathcal{S}'}\right)\right]\right] \qquad (17)\\
&\leq 2\lambda_m\mathcal{L}_{\mathcal{D}_m}\left(h_m^*\right) + 6\lambda_m\gamma_1\frac{\sqrt{p}}{\sqrt[p+1]{n_m}} \\
&\quad + 6\lambda_m\gamma_2\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\cdot\left(\mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathcal{L}_{\mathcal{D}_m}\left(h_{\mathcal{S}'}\right)\right] - \mathcal{L}_{\mathcal{D}_m}\left(h_m^*\right)\right) \\
&\quad + (1-\lambda_m)\cdot\mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathcal{L}_{\mathcal{D}_m}\left(h_{\mathcal{S}'}\right)\right] \qquad (18)\\
&\leq 2\lambda_m\mathcal{L}_{\mathcal{D}_m}\left(h_m^*\right) + 6\lambda_m\gamma_1\frac{\sqrt{p}}{\sqrt[p+1]{n_m}} \\
&\quad + 6\lambda_m\gamma_2\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\cdot\left(\delta_1\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2+\log\left(\frac{n}{d_{\mathcal{H}}}\right)} + 2\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right)\right) \\
&\quad + (1-\lambda_m)\cdot\left(\mathcal{L}_{\mathcal{D}_m}\left(h_m^*\right) + \delta_1\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2+\log\left(\frac{n}{d_{\mathcal{H}}}\right)} + 2\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right)\right) \qquad (19)\\
&= (1+\lambda_m)\mathcal{L}_{\mathcal{D}_m}\left(h_m^*\right) + 6\lambda_m\gamma_1\frac{\sqrt{p}}{\sqrt[p+1]{n_m}} \\
&\quad + 6\lambda_m\gamma_2\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\delta_1\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2+\log\left(\frac{n}{d_{\mathcal{H}}}\right)} + 12\lambda_m\gamma_2\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right) \\
&\quad + \delta_1(1-\lambda_m)\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2+\log\left(\frac{n}{d_{\mathcal{H}}}\right)} + 2\cdot(1-\lambda_m)\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right). \qquad (20)
\end{aligned}$$

Rearranging the terms and taking $c_1 \triangleq 2$, $c_2 \triangleq \max\{12\gamma_2, 6\gamma_1\}$, $c_3 \triangleq \delta_1$, $c_4 \triangleq \delta_2$ and $c_5 \triangleq 6\gamma_2\delta_1$, the final result follows. □

A.2 Intermediate Lemmas

Lemma A.1. *Consider $m \in [M]$, then there exists constants $\delta_1, \delta_2 \in \mathbb{R}$ such that*

$$\mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}}\left[\mathcal{L}_{\mathcal{D}_m}\left(h_{\mathcal{S}'}\right)\right] \leq \mathcal{L}_{\mathcal{D}_m}\left(h_m^*\right) + \delta_1\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2+\log\left(\frac{n}{d_{\mathcal{H}}}\right)} + 2\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}},\mathcal{D}_m\right), \qquad (21)$$

where d is the VC dimension of the hypothesis class $\mathcal{H}$, $\bar{\mathcal{D}} = \sum_{m=1}^{M}\frac{n_m}{n}\cdot\mathcal{D}_m$ and disc_H is the label discrepancy associated to the hypothesis class $\mathcal{H}$.

Proof. We remind that the label discrepancy associated to the hypothesis class $\mathcal{H}$ for two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over features and labels is defined as [37]:

$$\operatorname{disc}_{\mathcal{H}}\left(\mathcal{D}_1,\mathcal{D}_2\right) = \max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_1}\left(h\right) - \mathcal{L}_{\mathcal{D}_2}\left(h\right)\right|. \qquad (22)$$

Consider $m \in [M]$ and $h^* \in \arg\min_{h\in\mathcal{H}} \mathcal{L}_{\bar{\mathcal{D}}}(h)$. For $\mathcal{S}' \sim \otimes_{m=1}^{M} \mathcal{D}_m^{n_m/2}$, we have

$$\begin{aligned}
&\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}'}) - \mathcal{L}_{\mathcal{D}_m}(h_m^*) \\
&\quad = \mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}'}) - \mathcal{L}_{\bar{\mathcal{D}}}(h_{\mathcal{S}'}) + \mathcal{L}_{\bar{\mathcal{D}}}(h_{\mathcal{S}'}) - \mathcal{L}_{\bar{\mathcal{D}}}(h_m^*) + \mathcal{L}_{\bar{\mathcal{D}}}(h_m^*) - \mathcal{L}_{\bar{\mathcal{D}}}(h^*) + \mathcal{L}_{\bar{\mathcal{D}}}(h^*) - \mathcal{L}_{\mathcal{D}_m}(h_m^*) && (23)\\
&\quad = \underbrace{\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}'}) - \mathcal{L}_{\bar{\mathcal{D}}}(h_{\mathcal{S}'})}_{\leq \operatorname{disc}_{\mathcal{H}}(\mathcal{D}_m,\bar{\mathcal{D}})} + \underbrace{\mathcal{L}_{\bar{\mathcal{D}}}(h_m^*) - \mathcal{L}_{\mathcal{D}_m}(h_m^*)}_{\leq \operatorname{disc}_{\mathcal{H}}(\mathcal{D}_m,\bar{\mathcal{D}})} + \underbrace{\mathcal{L}_{\bar{\mathcal{D}}}(h^*) - \mathcal{L}_{\bar{\mathcal{D}}}(h_m^*)}_{\leq 0} + \mathcal{L}_{\bar{\mathcal{D}}}(h_{\mathcal{S}'}) - \mathcal{L}_{\bar{\mathcal{D}}}(h^*) && (24)\\
&\quad \leq 2\cdot \operatorname{disc}_{\mathcal{H}}\left(\mathcal{D}_m, \bar{\mathcal{D}}\right) + \mathcal{L}_{\bar{\mathcal{D}}}(h_{\mathcal{S}'}) - \mathcal{L}_{\bar{\mathcal{D}}}(h^*) && (25)\\
&\quad = 2\cdot \operatorname{disc}_{\mathcal{H}}\left(\mathcal{D}_m, \bar{\mathcal{D}}\right) + \mathcal{L}_{\bar{\mathcal{D}}}(h_{\mathcal{S}'}) - \mathcal{L}_{\mathcal{S}'}(h_{\mathcal{S}'}) + \underbrace{\mathcal{L}_{\mathcal{S}'}(h_{\mathcal{S}'}) - \mathcal{L}_{\mathcal{S}'}(h^*)}_{\leq 0} + \mathcal{L}_{\mathcal{S}'}(h^*) - \mathcal{L}_{\bar{\mathcal{D}}}(h^*) && (26)\\
&\quad \leq 2\operatorname{disc}_{\mathcal{H}}\left(\mathcal{D}_m, \bar{\mathcal{D}}\right) + 2\sup_{h\in\mathcal{H}} \left|\mathcal{L}_{\bar{\mathcal{D}}}(h) - \mathcal{L}_{\mathcal{S}'}(h)\right|. && (27)
\end{aligned}$$

We now bound $\mathbb{E}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}} \sup_{h\in\mathcal{H}} |\mathcal{L}_{\bar{\mathcal{D}}}(h) - \mathcal{L}_{\mathcal{S}'}(h)|$. We first observe that for every $h \in \mathcal{H}$, we can write $\mathcal{L}_{\bar{\mathcal{D}}}(h) = \mathbb{E}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}} \mathcal{L}_{\mathcal{S}'}(h)$. Therefore, despite the fact that the samples in $\mathcal{S}'$ are not i.i.d., we can follow the same steps as in the proof of [46, Theorem 6.11], and conclude

$$\mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}} \sup_{h\in\mathcal{H}} \left|\mathcal{L}_{\bar{\mathcal{D}}}(h) - \mathcal{L}_{\mathcal{S}'}(h)\right| \leq \frac{4 + \sqrt{\log\left(\tau_{\mathcal{H}}(n)\right)}}{\sqrt{n}}, \tag{28}$$

where $\tau_{\mathcal{H}}$ is the growth function of class $\mathcal{H}$.

Let d denote the VC dimension of $\mathcal{H}$. From Sauer's lemma [46, Lemma 6.10], we have that for $n > d+1$, $\tau_{\mathcal{H}}(n) \leq (en/d)^{d_{\mathcal{H}}}$. Therefore, there exist constants $\delta_1, \delta_2 \in \mathbb{R}$ (e.g., $\delta_1 = 4$, $\delta_2 = \max\{4/\sqrt{d_{\mathcal{H}}}, 1\}$), such that

$$\mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}} \sup_{h\in\mathcal{H}} \left|\mathcal{L}_{\bar{\mathcal{D}}}(h) - \mathcal{L}_{\mathcal{S}'}(h)\right| \leq \frac{\delta_1}{2}\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2 + \log\left(\frac{n}{d_{\mathcal{H}}}\right)}. \tag{29}$$

Taking the expectation in Eq. (27) and using this inequality, we have

$$\mathop{\mathbb{E}}_{\mathcal{S}'\sim\otimes_{m=1}^{M}\mathcal{D}_m^{n_m/2}} \left[\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}'})\right] \leq \mathcal{L}_{\mathcal{D}_m}(h_m^*) + \delta_1\cdot\sqrt{\frac{d_{\mathcal{H}}}{n}}\cdot\sqrt{\delta_2 + \log\left(\frac{n}{d_{\mathcal{H}}}\right)} + 2\cdot\operatorname{disc}_{\mathcal{H}}\left(\bar{\mathcal{D}}, \mathcal{D}_m\right). \tag{30}$$

□

The following Lemma proves an upper bound on the expected error of the 1-NN learning rule.

Lemma A.2 (Adapted from [46, Thm 19.3])**.** *Under Assumptions 1, 2, and 4 for all $m \in [M]$, it holds*

$$\mathop{\mathbb{E}}_{\mathcal{S}_m''\sim\mathcal{D}_m^{n_m/2}} \left[\mathcal{L}_{\mathcal{D}_m}\left(h^{(1)}_{\mathcal{S}_m'',\mathcal{S}'}\right)\right] \leq 2\mathcal{L}_{\mathcal{D}_m}(h_m^*) + 6\left\{\gamma_1 + \gamma_2\cdot\left[\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}'}) - \mathcal{L}_{\mathcal{D}_m}(h_m^*)\right]\right\}\cdot\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}. \tag{31}$$

Proof. Recall that for $m \in [M]$, the Bayes optimal rule, i.e., the hypothesis that minimizes $\mathcal{L}_{\mathcal{D}_m}(h)$ over all functions, is

$$h_m^*(\mathbf{x}) = \mathbb{1}_{\{\eta_m(\mathbf{x})>1/2\}}. \tag{32}$$

We note that the 1-NN rule can be expressed as follows:

$$\left[h^{(1)}_{\mathcal{S}_m'',\mathcal{S}'}(\mathbf{x})\right]_y = \mathbb{1}_{\left\{y=\pi^{(1)}_{\mathcal{S}_m''}(\mathbf{x})\right\}}, \tag{33}$$

where we are putting in evidence that the permutation π_m depends on the dataset $\mathcal{S}_m''$. Then, under Assumption 2, the loss function $l(\cdot)$ reduces to the 0-1 loss.

Consider samples $\mathcal{S} \sim \otimes_{m=1}^{M}\mathcal{D}_m^{n_m}$. Using Assumptions 1, 2 and 4, and following the same steps as in [46, Lemma 19.1], we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\mathcal{S}_m''\sim\mathcal{D}_m^{n_m/2}} \left[\mathcal{L}_{\mathcal{D}_m}\left(h^{(1)}_{\mathcal{S}_m'',\mathcal{S}'}\right)\right] - 2\mathcal{L}_{\mathcal{D}_m}(h_m^*) \leq \\
&\qquad \left\{\gamma_1 + \gamma_2\cdot\left[\mathcal{L}_{\mathcal{D}_m}(h_{\mathcal{S}'}) - \mathcal{L}_{\mathcal{D}_m}(h_m^*)\right]\right\} \times \underbrace{\mathop{\mathbb{E}}_{\mathcal{S}_{m,\mathcal{X}}''\sim\mathcal{D}_{m,\mathcal{X}}^{n_m/2},\ \mathbf{x}\sim\mathcal{D}_{m,\mathcal{X}}} \left[d\left(\phi_{h_{\mathcal{S}'}}(\mathbf{x}), \phi_{h_{\mathcal{S}'}}\left(\pi^{(1)}_{\mathcal{S}_m''}(\mathbf{x})\right)\right)\right]}_{\triangleq \mathcal{T}_{\mathcal{S}'}},
\end{aligned} \tag{34}$$

where $\mathcal{S}''_{m,\mathcal{X}}$ denotes the set of input features in the dataset $\mathcal{S}''_m$ and $\mathcal{D}_{m,\mathcal{X}}$ the marginal distribution of $\mathcal{D}_m$ over $\mathcal{X}$. Note that $\mathcal{S}''_m$ is independent from $\mathcal{S}'$.

As in the proof of [46, Theorem 19.3], let T be an integer to be precised later on. We consider $r = T^p$ and $C_1, \dots, C_r$ to be the cover of the set $[0,1]^p$ using boxes with side $1/T$. We bound the term $\mathcal{T}_{\mathcal{S}'}$ independently from $\mathcal{S}'$ as follows

$$\mathop{\mathbb{E}}_{\mathcal{S}''_m \sim \mathcal{D}^{n_m/2}_{m,\mathcal{X}},\ \mathbf{x} \sim \mathcal{D}_{m,\mathcal{X}}} \left[d\left(\phi_{h_{\mathcal{S}'}}\left(\mathbf{x}\right), \phi_{h_{\mathcal{S}'}}\left(\pi^{(1)}_{\mathcal{S}''_m}\left(\mathbf{x}\right)\right)\right)\right] \leq \sqrt{p}\left(\frac{2T^p}{n_m e} + \frac{1}{T}\right). \tag{35}$$

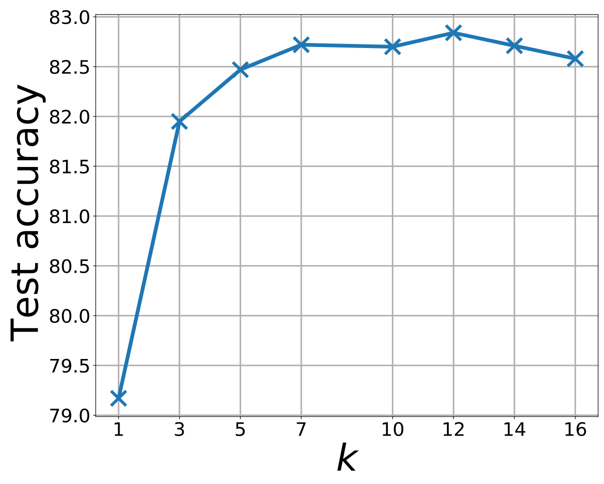
If we set $\epsilon = 2\left(\frac{2}{n_m}\right)^{\frac{1}{p+1}}$ and $T = \lceil 1/\epsilon \rceil$, it follows $1/\epsilon \leq T < 2/\epsilon$ and then

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathcal{S}''_m \sim \mathcal{D}^{n_m/2}_{m,\mathcal{X}},\ \mathbf{x} \sim \mathcal{D}_{m,\mathcal{X}}} \left[d\left(\phi_{h_{\mathcal{S}'}}\left(\mathbf{x}\right), \phi_{h_{\mathcal{S}'}}\left(\pi^{(1)}_{\mathcal{S}''_m}\left(\mathbf{x}\right)\right)\right)\right] &\leq \sqrt{p}\left(\frac{2(2/\epsilon)^p}{n_m e} + \epsilon\right) && (36)\\
&= \sqrt{p}\left(\frac{1}{e} + 2\right)\left(\frac{2}{n_m}\right)^{\frac{1}{p+1}} && (37)\\
&\leq 6\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}. && (38)
\end{aligned}$$

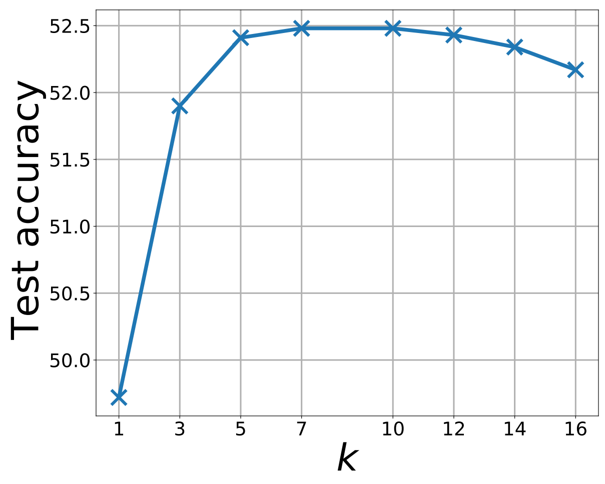
Thus,

$$\mathop{\mathbb{E}}_{\mathcal{S}'_m \sim \mathcal{D}^{n_m}_m} \left[\mathcal{L}_{\mathcal{D}_m}\left(h^{(1)}_{\mathcal{S}''_m, \mathcal{S}'}\right)\right] \leq 2\mathcal{L}_{\mathcal{D}_m}\left(h^*_m\right) + 6\frac{\sqrt{p}}{\sqrt[p+1]{n_m}}\left\{\gamma_1 + \gamma_2 \cdot \left[\mathcal{L}_{\mathcal{D}_m}\left(h_{\mathcal{S}}\right) - \mathcal{L}_{\mathcal{D}_m}\left(h^*_m\right)\right]\right\}. \tag{39}$$

□

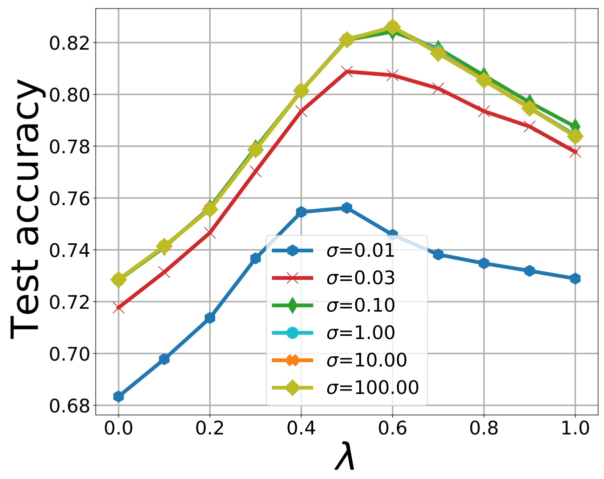


(a) CIFAR-10.

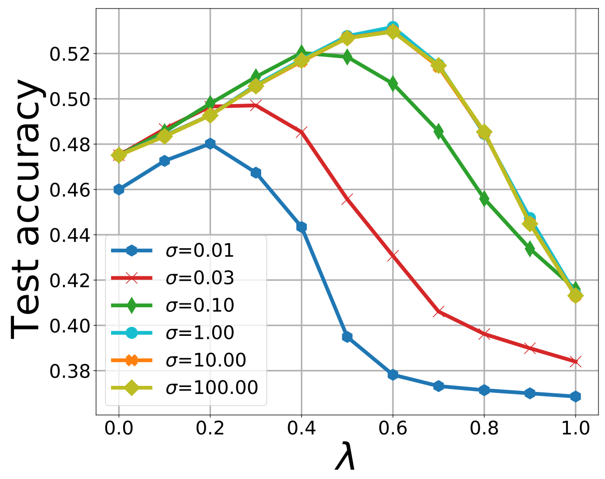


(b) CIFAR-100.

Figure 4: Test accuracy vs number of neighbors k.



(a) CIFAR-10.



(b) CIFAR-100.

Figure 5: Test accuracy vs the interpolation parameter λ for different values of the scale length of the kernel σ.

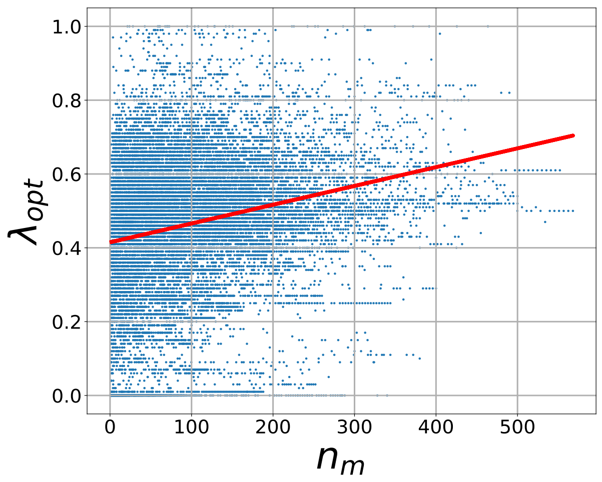
B Additional Experiments

Effect of length scale of the kernel σ. We consider distance metrics of the form

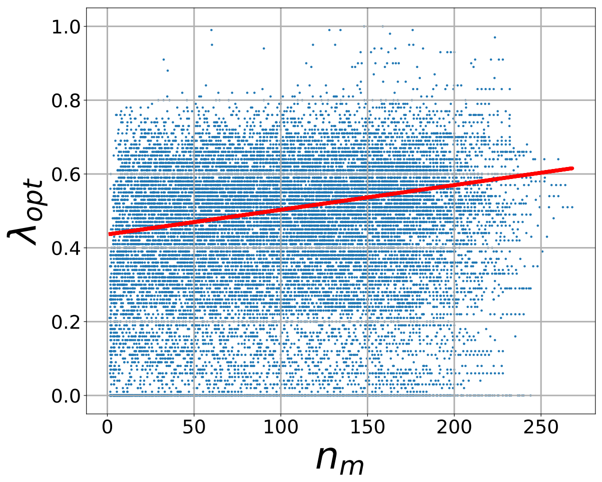
$$\forall \mathbf{z}, \mathbf{z}' \in \mathbb{R}^p;\ d_\sigma\left(\mathbf{z}, \mathbf{z}'\right) = \frac{\left\|\mathbf{z} - \mathbf{z}'\right\|_2}{\sigma}, \tag{40}$$

where $\sigma \in \mathbb{R}^+$ is a length scale parameter. Figure 5 shows that kNN-Per's performance is not highly sensitive to the selection of the length scale parameter, as scaling the Euclidean distance by a constant factor σ has almost no effect for values of σ between 0.1 and 1000.

Effect of datastore's size on the optimal λ. Figure 6 shows the effect of the local number of samples n_m on the optimal mixing parameter λ_{opt} (evaluated on the client's test dataset). The number of samples changes across clients and, for the same client, with different values of the capacity. The figure shows a positive correlation between the local number of samples and the optimal mixing parameter and then validates the intuition that clients with more samples tend to rely more on the memorization mechanism than on the base model, as captured by the generalization bound from Theorem 4.1.



(a) CIFAR-10.



(b) CIFAR-100.

Figure 6: λ_{opt} vs local number of samples n_m.

Effect of hardware heterogeneity. In our experiments above, clients' local datasets had different size, which can also be due to different memory capabilities. In order to investigate more in depth the effect of system heterogeneity, we split the new clients in two groups: "weak" clients with normalized capacity $1/2 - \Delta C$ and "strong" clients with normalized capacity $1/2 + \Delta C$, where $\Delta C \in (0, 1/2)$ is a parameter controlling the hardware heterogeneity of the system. Note that the total amount of memory in the system is constant, but varying ΔC changes its distribution across clients from a homogeneous scenario ($\Delta C = 0$) to an extremely heterogeneous one ($\Delta C = 0.5$). Figure 7 shows the effect of the hardware heterogeneity, as captured by ΔC. As the marginal improvement from additional memory is decreasing (see, e.g., Fig. 1) the gain for strong clients does not compensate the loss for weak ones. The overall effect is then that the average test accuracy decreases as system heterogeneity increases.

Adding compression techniques. `kNN-Per` can be combined with nearest neighbours compression techniques as `ProtoNN` [15]. `ProtoNN` reduces the amount of memory required by jointly learning 1) a small number of prototypes to represent the entire training set and 2) a data projection into a low dimensional space. We combined `kNN-Per` and `ProtoNN` and explored both the effect of the number of prototypes and the projection dimension used in `ProtoNN`. For each client, the number of prototypes is set to a given fraction of the total number of available samples. We refer to this quantity also as capacity. We varied the capacity in the grid $\{i \times 10^{-1}, i \in [10]\}$, and the projection dimension in the grid $\{i \times 100, i \in [12]\} \cup \{1280\}$. Note that smaller projection dimension and less prototypes correspond to a smaller memory footprint, suited for more restricted hardware. Our implementation is based on `ProtoNN`'s official.[5] Figure 8a shows that, on CIFAR-10, `ProtoNN` allows to reduce the `kNN-Per`'s memory footprint by a factor four (using $n_m/3$ prototypes and projection dimension 1000) at the cost of a limited reduction in test accuracy (82.3% versus 83.0% in Table 2). Note that `kNN-Per` with `ProtoNN` still outperforms all other methods. On CIFAR-100, `ProtoNN`'s compression techniques appear less advantageous: the approach loses about 3 percentage points (52.1% versus 55.0% in Table 2) while only reducing memory requirement by 20%.

Effect of global model's quality. Assumption 4 stipulates that the smaller the expected loss of the global model, the more accurate the corresponding representation. As representation quality improves, we can expect that kNN accuracy improves too. This effect is quantified by Lemma A.2, where the loss of the local memorization mechanism is upper bounded by a term that depends linearly on the loss of the global model. In order to validate this assumption, we study the relation between the test accuracies of the global model and `kNN-Per`. In particular, we train two global models, one for CIFAR-10 and the other for CIFAR-100, in a centralized way, and we save the weights at different stages of the training, leading to global models with different qualities. Figure 9 shows the test accuracy of `kNN-Per` with $\lambda = 1$ (i.e., when only the knn predictor is used) as a function of the global model's test accuracy for different levels of heterogeneity on CIFAR-10 and CIFAR-100 datasets. We observe that, quite unexpectedly, the relation between the two accuracies is almost linear. The experiments also confirm what observed in Fig. 1: `kNN-Per`performs better when local distributions are more heterogeneous (smaller α). Similar plots with λ optimized locally at every client are shown in Fig. 10.

[5]`https://github.com/Microsoft/EdgeML`.

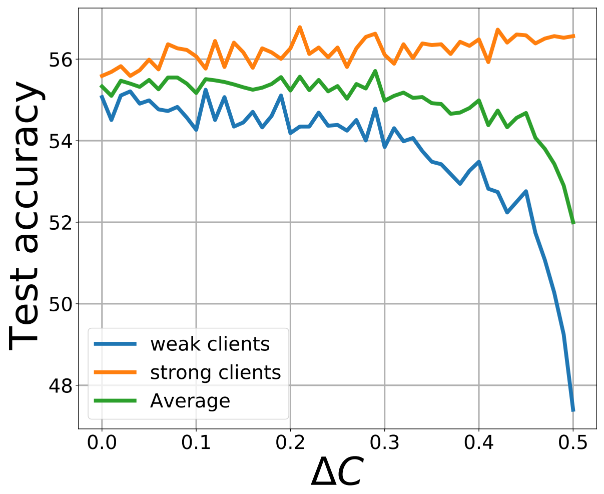
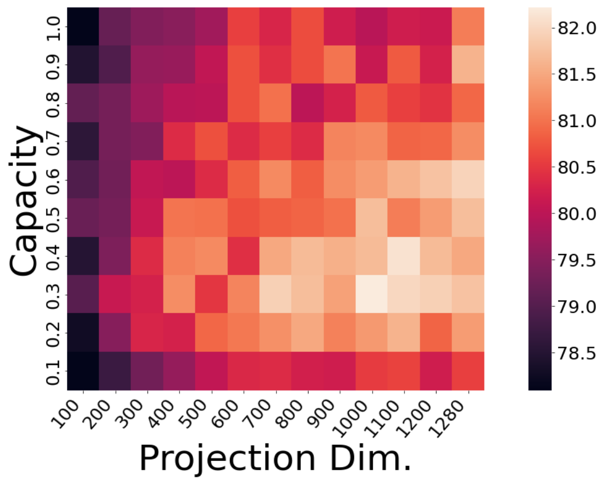


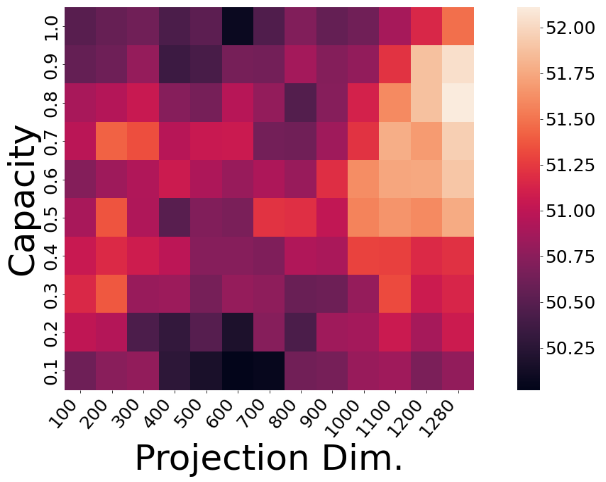
Figure 7: Effect of system heterogeneity across clients on CIFAR-100 dataset. The size of the local datastore increases (resp. decreases) with ΔC for strong (resp. weak) clients.

Robustness to distribution shift. As previously mentioned, `kNN-Per` offers a simple and effective way to address statistical heterogeneity in a dynamic environment where client's data distributions change after training. We simulate such a dynamic environment as follows. Client m initially has a datastore built using instances sampled from a data distribution $\mathcal{D}_m$. For time step $t < t_0$, client m receives a batch of $n_m^{(t)}$ instances drawn from $\mathcal{D}_m$. At time step t_0, we suppose that a data distribution shift takes place, i.e., for $t_0 \leq t \leq T$, client m receives $n_m^{(t)}$ instances drawn from a data distribution $\mathcal{D}'_m \neq \mathcal{D}_m$. Upon receiving new instances, client m may use those instances to update its datastore. We consider 3 different strategies: (1) *first-in-first-out* (FIFO) where, at time step t, the $n_m^{(t)}$ oldest samples are replaced by the newly obtained samples. (2) *concatenate*, where the new samples are simply added to the datastore. (3) *fixed datastore*, where the datastore is not updated at all. In our simulations, we consider CIFAR-10/100 datasets with $M = 100$ clients. Once again, we used a symmetric Dirichlet distribution to generate two datasets for every client. In particular, for each label y we sampled two vectors p_y and p'_y from a Dirichlet distribution of order $M = 100$ and parameter $\alpha = 0.3$. Then, for client m, we generated two datasets $\mathcal{S}_m$ and $\mathcal{S}'_m$ by allocating $p_{y,m}$ and $p'_{y,m}$ fraction of all training instances of class y.[6] Both $\mathcal{S}_m$ and $\mathcal{S}'_m$ are partitioned into training and test sets following the original CIFAR training/test data split. Half of the training set obtained from $\mathcal{S}_m$ is stored in the datastore, while the rest is further partitioned into t_0 batches $\mathcal{S}_m^{(0)}, \ldots, \mathcal{S}_m^{(t_0-1)}$. These batches are the new samples arriving at client m. Similarly, $\mathcal{S}'_m$ is partitioned into $T - t_0$ equally sized batches. Figure 11 shows the evaluation of the test accuracy across time. If clients do not update their datastores, there is a significant drop in accuracy as soon as the distribution changes at $t_0 = 50$. If datastore are updated using FIFO, we observe some random fluctuations for the accuracy for $t < t_0$, as repository changes affect the kNN predictions. While accuracy inevitably drops for $t = t_0$, it then increases as datastores are progressively populated by instances from the new distributions. Once all samples from the previous distributions are evicted, the accuracy settles around a new value (higher or lower than the one for $t < t_0$ depending on the difference between the new and the old distributions). If clients keep adding new samples to their datastores (the "concatenate" strategy), results are similar, but 1) accuracy increases for $t < t_0$ as the quality of kNN predictors improves for larger datastores, 2) accuracy increases also for $t > t_0$, but at a slower pace than what observed under FIFO, as samples from the old distribution are never evicted.

[6]We always make sure that $|\mathcal{S}_m| \leq |\mathcal{S}'_m|$.

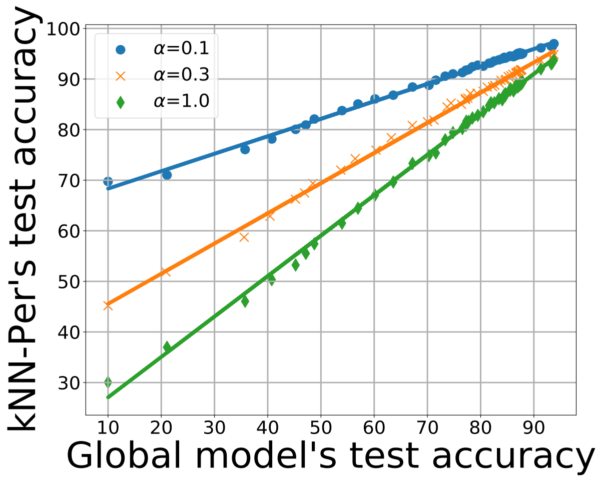


(a) CIFAR-10.

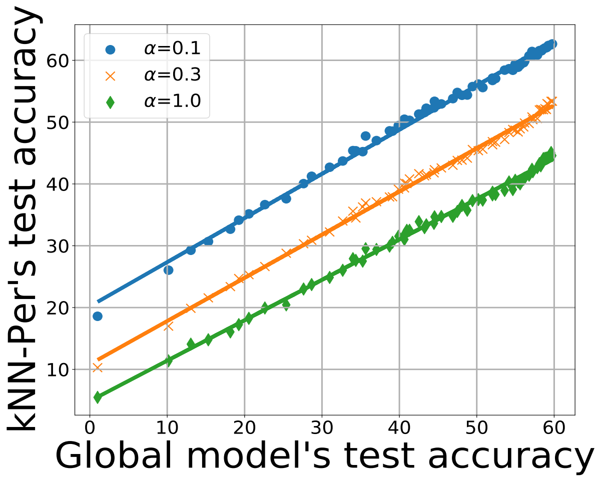


(b) CIFAR-100.

Figure 8: Test accuracy when the kNN mechanism is implemented through `ProtoNN` for different values of projection dimension and number of prototypes (expressed as a fraction of the local dataset). CIFAR-10 (left) and CIFAR-100 (right) datasets.

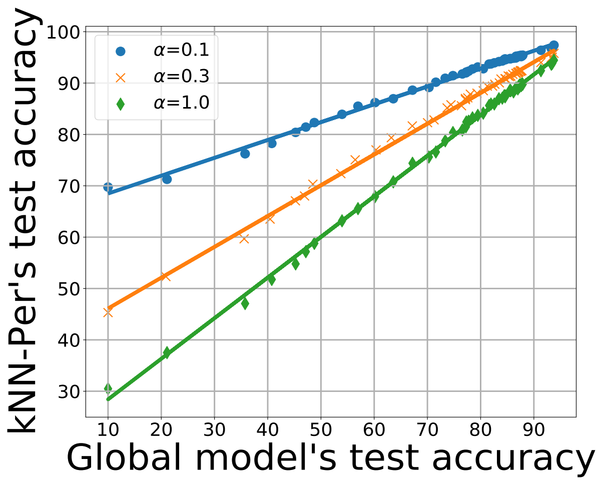


(a) CIFAR-10.

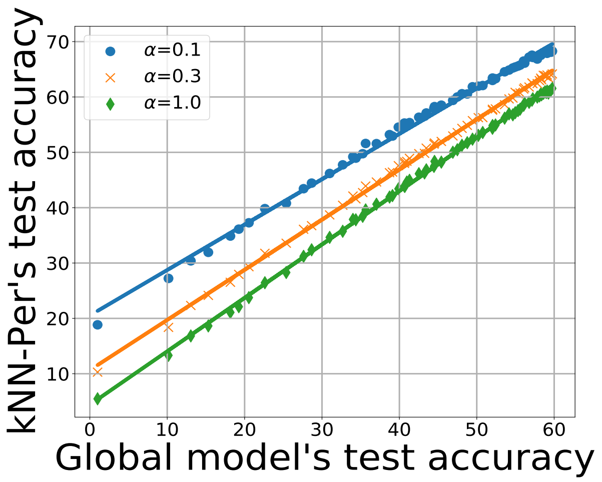


(b) CIFAR-100.

Figure 9: Effect of the global model quality on the test accuracy of `kNN-Per` with $\lambda_m = 1$ for each $m \in [M]$.

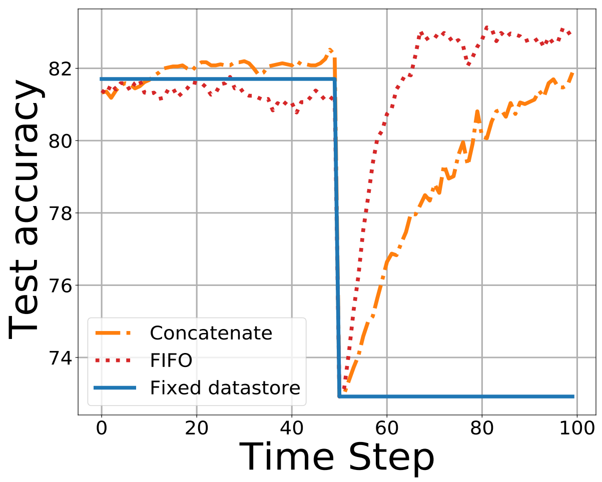


(a) CIFAR-10.

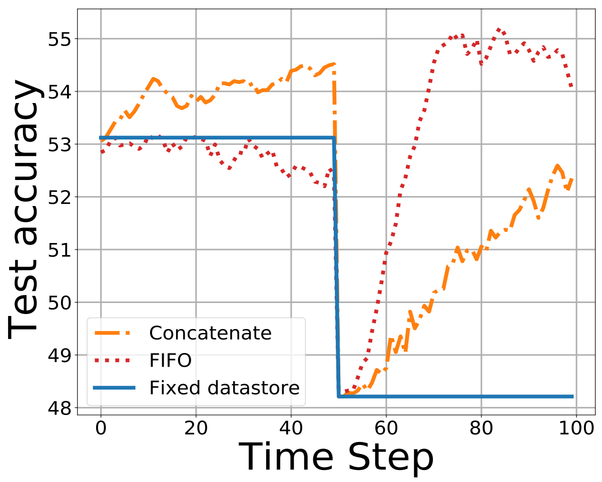


(b) CIFAR-100.

Figure 10: Effect of the global model quality on the test accuracy of `kNN-Per` with λ_m tuned per client.



(a) CIFAR-10.



(b) CIFAR-100.

Figure 11: Test accuracy accuracy when a distribution shift happens at time step $t_0 = 50$ for different datastore management strategies.