



**HAL**  
open science

# Federated Learning with Nonvacuous Generalisation Bounds

Pierre Jobic, Maxime Haddouche, Benjamin Guedj

► **To cite this version:**

Pierre Jobic, Maxime Haddouche, Benjamin Guedj. Federated Learning with Nonvacuous Generalisation Bounds. 2023. hal-04260486

**HAL Id: hal-04260486**

**<https://inria.hal.science/hal-04260486>**

Preprint submitted on 26 Oct 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

---

# Federated Learning with Nonvacuous Generalisation Bounds

---

**Pierre Jobic**  
CEA, List  
Université Paris-Saclay

**Maxime Haddouche**  
Inria  
University College London  
Université de Lille

**Benjamin Guedj**  
Inria  
University College London

## Abstract

We introduce a novel strategy to train randomised predictors in federated learning, where each node of the network aims at preserving its privacy by releasing a local predictor but keeping secret its training dataset with respect to the other nodes. We then build a global randomised predictor which inherits the properties of the local private predictors in the sense of a PAC-Bayesian generalisation bound. We consider the synchronous case where all nodes share the same training objective (derived from a generalisation bound), and the asynchronous case where each node may have its own personalised training objective. We show through a series of numerical experiments that our approach achieves a comparable predictive performance to that of the batch approach where all datasets are shared across nodes. Moreover the predictors are supported by numerically nonvacuous generalisation bounds while preserving privacy for each node. We explicitly compute the increment on predictive performance and generalisation bounds between batch and federated settings, highlighting the price to pay to preserve privacy.

## 1 INTRODUCTION

In the federated learning (FL) paradigm, a group of *users* (or nodes) is learning in parallel, and typically aims at preserving their personal datasets while sharing a common predictor. While maintaining the privacy of their own data, users mutually share informa-

tion through a central *server*. There has been a significant surge of interest in federated learning in the past decade (Konečný et al., 2016b), with clear applications in healthcare, transportation and retail, where it is typically of the utmost interest to avoid the leak of private information to other organisations or devices, for ethical or business motivations. The existing literature essentially categorises *horizontal* and *vertical* FL, depending on whether users’ datasets share many features or individuals. These two streams have generated various contributions such as the design of efficient communication strategies (Konečný et al., 2016a,b; Suresh et al., 2017), the preservation of privacy through differentially-private distributed optimization methods (Agarwal et al., 2018), the enforcement of fairness (as in many cases, post-training learning models may be biased or unfair and may discriminate against some protected groups – Hardt et al., 2016; Mohri et al., 2019). We refer to Zhang et al. (2021); Mammen (2021); Kairouz et al. (2021) for recent surveys on FL.

Consider a simple federated learning framework (Bonawitz et al., 2017; McMahan et al., 2017b). In each round, the server first provides the initial model to each user, then each user updates the initial model with its personal data. Finally, the server aggregates the collected local models into a single global model, which is used as next round’s initialisation if needed. Hereafter we will refer to this learning problem as *FL-SOB* (Federated Learning with Synchronous Objectives). This is especially relevant when all users share a common learning goal (*e.g.*, hospitals learning from different datasets to identify or predict a specific single pathology). Deep neural networks have been used to develop powerful federated algorithms (McMahan et al., 2017a). A more complex scenario consists in *personalised FL* (*PFL*, Tan et al., 2022) where users may have their own distinct learning goal but still want to share joint information as these goals share some level of similarity. This corresponds, for instance, to *transfer learning* (see *e.g.*, Zhuang et al., 2021) situations where one wants to extract some information of

a learning problem (*e.g.*, detecting tigers in images) to perform better on another one, sharing some similarities (*e.g.*, detecting cats).

**Towards a unified framework.** The recent PAC-FL framework of Zhang et al. (2023b) proposes a unified framework to formalise FL, intrincating the notion of generalisation ability (designed as utility) alongside privacy, and quantifying how much data are protected (*i.e.* impossible to retrieve) while transmitting partial information to the server. This framework builds from the work of Zhang et al. (2019) investigating the tradeoffs between privacy, utility and efficiency. The question of an optimal trade-off is crucial to deploy the FL framework in practice (Tsipras et al., 2019).

**On the place of generalisation in FL.** Using their PAC-FL framework, Zhang et al. (2023b) proposed generalisation bounds involving the dimension of the predictor space. The question of generalisation in FL is central: Mohri et al. (2019); Zhang et al. (2023a) established Rademacher-based generalisation bounds, Yagli et al. (2020) provided bounds based on mutual information to explain both generalisation ability and privacy leakage per user. Bayesian methods have also been considered in FL-SOB (Yurochkin et al., 2019; Chen and Chao, 2021; Zhang et al., 2022) as well as in PFL (Kotelevskii et al., 2022).

**PAC-Bayes learning in FL.** Beyond Bayesian methods, PAC-Bayes learning (see the seminal works of Shawe-Taylor and Williamson, 1997; McAllester, 1998, 2003; Maurer, 2004 – we refer to the surveys of Guedj, 2019; Alquier, 2021, and to the recent monograph of Hellström et al., 2023) has recently re-emerged as a powerful framework in batch learning to explain the generalisation ability of neural nets by providing non-vacuous generalisation bounds (Dziugaite and Roy, 2017; Letarte et al., 2019; Pérez-Ortiz et al., 2021; Biggs and Guedj, 2021, 2022). PAC-Bayes combines information-theoretic tools with the Bayesian paradigm of generating a data-dependent *posterior* distribution over a predictor space from a *prior* distribution (or reference measure), usually data-independent. The flexibility of the PAC-Bayes framework makes it useful to explain generalisation in many learning settings. In particular, theoretical results and practical algorithms have been derived for various learning problems such as reinforcement learning (Fard and Pineau, 2010), online learning (Li et al., 2018; Haddouche and Guedj, 2022), constrative learning (Nozawa et al., 2020), generative models (Chérif-Abdellatif et al., 2022), multi-armed bandits (Seldin et al., 2011, 2012; Sakhi et al., 2022), meta-learning (Amit and Meir, 2018; Farid and Majumdar, 2021; Rothfuss et al., 2021, 2022; Ding et al., 2021), majority votes (Zantedeschi et al., 2021; Biggs et al., 2022)

to name but a few.

Recently, some works have used the PAC-Bayes framework in FL: Reiszadeh et al. (2020) and Achituve et al. (2021) have evaluated the post-training predictor shared by all users through a PAC-Bayes bound. Rather than exploiting existing bounds, new PAC-Bayes results, tailored for personalised FL, recently emerged with the aim to explain the efficiency of learning procedures (Scott et al., 2023; Sefidgaran et al., 2023), although the PAC-Bayes bound is not minimised by the algorithm. Finally, recent works showed that the Bayesian procedure ELBO, adapted to FL, is exactly the minimisation of a PAC-Bayes upper bound (Kim and Hospedales, 2023; Vedadi et al., 2023). Thus, they show that those methods are well incorporated in a theoretical framework explaining their good generalisation ability.

**Our contributions.** Beyond being a safety check for generalisation, PAC-Bayes theory provides state-of-the-art learning algorithms with tight generalisation guarantees in the batch setting. We adapt those algorithms to the FL-SOB and PFL settings. We propose GENFL (standing for Generalisation-driven Federated Learning), an algorithm in which users optimise local PAC-Bayes objectives (bounds from Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021). We show a global generalisation bound for all users in FL-SOB, and local ones in PFL. Finally, we show in numerical experiments that our procedure is competitive with the state-of-the-art and we bring nonvacuous generalisation guarantees to practitioners of federated learning.

**Outline.** We describe our notation in Section 2 and introduce in Section 3 a novel algorithm called GENFL, alongside two instantiations to FL-SOB and PFL. We present numerical experiments to support our methods. in Section 4. Our algorithms and the code used to generate figures in this paper is available at <https://anonymous.4open.science/r/GenFL-010C/README.md>. The paper closes with a discussion in Section 5. In Appendix A, we comment on strategies to compute PAC-Bayesian bounds, Appendix B contains a comprehensive description of our procedure in the PFL setting, and Appendix C provides additional experiments.

## 2 BACKGROUND

**Federated learning.** We consider a predictor set  $\mathcal{H}$ , a data space  $\mathcal{Z}$  and denote the space of distributions over  $\mathcal{H}$ ,  $\mathcal{M}(\mathcal{H})$ . We let  $\ell: \mathcal{H} \times \mathcal{Z} \rightarrow [0, 1]$  denote a loss function. In FL, we consider an ensemble of  $K \in \mathbb{N}^*$  users, and for each user  $1 \leq i \leq K$ , we denote by  $\mathcal{S}_i = (\mathbf{z}_{i,j})_{j=1 \dots m_i}$  its associated dataset of size  $m_i$ .

We define  $\mathcal{S}$ , of size  $m = \sum_{i=1}^K m_i$ , to be the union of all  $\mathcal{S}_i$ . We assume that each  $\mathcal{S}_i$  is *i.i.d.* with associated distribution  $\mathcal{D}_i$ . Each user  $1 \leq i \leq K$  aims to jointly learn a predictor  $h \in \mathcal{H}$  while keeping private their training dataset  $\mathcal{S}_i$ .

**Learning theory.** In PAC-Bayes learning, instead of directly crafting a predictor  $h \in \mathcal{H}$ , we design a data-driven posterior distribution  $Q \in \mathcal{M}(\mathcal{H})$  with respect to a prior distribution  $P$ . To assess the generalisation ability of a predictor  $h \in \mathcal{H}$ , we define for each user  $k$  the *risk* to be  $R_{\mathcal{D}_i} := \mathbb{E}_{\mathbf{z} \sim \mu}[\ell(h, \mathbf{z})]$  and its empirical counterpart  $\hat{R}_{\mathcal{S}_i} := \frac{1}{m} \sum_{j=1}^{m_i} \ell(h, \mathbf{z}_{i,j})$ . As PAC-Bayes focuses on elements of  $\mathcal{M}(\mathcal{H})$ , we also define the expected risk and empirical risks for  $Q \in \mathcal{M}(\mathcal{H})$  as  $R_{\mathcal{D}_i}(Q) := \mathbb{E}_{h \sim Q}[R_{\mathcal{D}_i}(h)]$  and  $\hat{R}_{\mathcal{S}_i}(Q) := \mathbb{E}_{h \sim Q}[\hat{R}_{\mathcal{S}_i}(h)]$ .

**Background on PAC-Bayes learning.** In a batch setting, we only consider the dataset  $\mathcal{S}$  (this can be seen as the case where there is only one user) and we assume that all data are *i.i.d.* with distribution  $\mathcal{D}$ . For two probability measures  $P, Q$  we define the *Kullback-Leibler divergence* to be  $\text{KL}(Q, P) = \mathbb{E}_{h \sim P} \left[ \frac{dQ}{dP}(h) \right]$  where  $\frac{dQ}{dP}$  is the Radon-Nikodym derivative. We also denote by  $\text{kl}$  the KL divergence between two Bernoulli distributions.

**Generalisation bounds.** We recall the following bound, due to McAllester (2003); Maurer (2004), which holds for bounded losses.

**Theorem 1** (McAllester’s bound). *For any data-free prior distribution  $P \in \mathcal{M}(\mathcal{H})$ , any  $\delta \in [0, 1]$ , with probability at least  $1 - \delta$ , for any posterior distribution  $Q \in \mathcal{M}(\mathcal{H})$ ,*

$$\text{kl}(R_{\mathcal{D}}(Q), R_{\mathcal{S}}(Q)) \leq \frac{\text{KL}(Q||P) + \ln \frac{2\sqrt{m}}{\delta}}{m}, \quad (1)$$

which leads to the following upper bound on the risk

$$R_{\mathcal{D}}(Q) \leq \text{kl}^{-1} \left( \hat{R}_{\mathcal{S}}(Q) \left\| \frac{\text{KL}(Q||P) + \ln \frac{2\sqrt{m}}{\delta}}{m} \right\| \right), \quad (2)$$

where  $\text{kl}^{-1}(x, b) = \sup \{y \in [x, 1] \mid \text{kl}(x, y) \leq b\}$ .

Note that by the definition of  $\text{kl}^{-1}$ , (2) is the tightest upper bound on  $R_{\mathcal{D}}(Q)$  that we can obtain starting from (1). While  $\text{kl}^{-1}$  has no closed form, it is possible to approximate it efficiently via root-finding techniques (see, *e.g.*, Dziugaite and Roy, 2017, Appendix A). However, this function is hard to evaluate, and even harder to optimise. We need to rely on looser relaxations of (1) to design tractable optimisation procedures.

**Relaxations of McAllester’s bound.** The most classical relaxation of (1) relies on Pinsker’s inequality  $\text{kl}(q||p) \geq \frac{(p-q)^2}{2}$  and leads to the following high-probability bound, valid under the same assumptions as Theorem 1:

$$R_{\mathcal{D}}(Q) \leq \hat{R}_{\mathcal{S}}(Q) + \sqrt{\frac{\text{KL}(Q||P) + \ln \frac{2\sqrt{m}}{\delta}}{2m}}. \quad (3)$$

While (3) is well known and already appears in McAllester (2003), novel relaxations, exploiting refined Pinsker’s inequality (see, *e.g.*, Boucheron et al., 2013, Lemma 8.4) have been exploited to obtain PAC-Bayes Bernstein bounds (Tolstikhin and Seldin, 2013; Mhammedi et al., 2019). Building on this inequality, Rivasplata et al. (2019); Pérez-Ortiz et al. (2021) proposed a *PAC-Bayes quadratic bound* recalled below, valid under the assumptions of Theorem 1

$$R_{\mathcal{D}}(Q) \leq \left( \sqrt{\hat{R}_{\mathcal{S}}(Q) + \frac{\text{KL}(Q||P) + \log \left( \frac{2\sqrt{m}}{\delta} \right)}{2m}} \right) \quad (4)$$

$$+ \sqrt{\frac{\text{KL}(Q||P) + \log \left( \frac{2\sqrt{m}}{\delta} \right)}{2m}} \quad (5)$$

Note that (3) and (4) are easier to optimise than (2), making them more relevant for practical learning algorithms.

**Generalisation-driven learning algorithms.** Most of the PAC-Bayesian bounds in the literature are fully empirical. This paves the way to use the bound as a training objectives and leads to generalisation-driven learning algorithms. A classical PAC-Bayesian algorithm is derived from Catoni’s bound (see, *e.g.*, Catoni, 2007, Alquier et al., 2016, Theorem 4.1):

$$\underset{Q \in \mathcal{M}(\mathcal{H})}{\text{argmin}} R_{\mathcal{S}}(Q) + \frac{\text{KL}(Q, P)}{\lambda}. \quad (6)$$

In (6) an *inverse temperature*  $\lambda > 0$  appears and acts as a learning rate in gradient descent. Similarly, it is possible to derive batch learning algorithms from (3) (4) (Dziugaite and Roy, 2017; Pérez-Ortiz et al., 2021). Then, we have access to a theoretical upper bound which requires to approximate expectations over  $Q$ . We next discuss how to mitigate this.

**Computing generalisation guarantees.** In practice, the tightest McAllester’s bound is computed, *i.e.*, (1). First, note that the KL divergence is easy to compute in the Gaussian case as it has a closed form (see,

e.g., Duchi, 2007, Section 9). Then, it remains to estimate the expected empirical risk over  $Q$ , which is costly in practice as it involves Monte Carlo approximations. To alleviate this issue, we leverage the trick from Dziugaite and Roy (2017, Section 3.3), which exploit a high probability upper bound of  $\hat{R}_S(Q)$  with confidence level  $\delta'$

$$\hat{R}_S(Q) \leq \text{kl}^{-1} \left( \hat{R}_S(\hat{Q}_n) \left\| \frac{1}{n} \ln \left( \frac{2}{\delta'} \right) \right. \right),$$

where  $\hat{R}_S(\hat{Q}_n) = \frac{1}{n} \sum_{i=1}^n \ell(W_i, z_i), \forall i, W_i \sim Q$ . Then incorporating this upper bound in (1) gives the final bound we use, with probability at least  $1 - \delta - \delta'$ :

$$R_{\mathcal{D}}(Q) \leq \text{kl}^{-1} \left( \hat{R}_S^n(Q) \left\| \frac{\text{KL}(Q\|P) + \ln \frac{2\sqrt{m}}{\delta}}{m} \right. \right), \quad (7)$$

where  $\hat{R}_S^n(Q) = \text{kl}^{-1} \left( \hat{R}_S(\hat{Q}_n) \left\| \frac{1}{n} \ln \left( \frac{2}{\delta'} \right) \right. \right)$ .

To compute  $\text{kl}^{-1}(p, c)$  for any  $p, c$ , we leverage Dziugaite and Roy (2017, Appendix A) described in appendix A.

### 3 GENERALISATION-DRIVEN FEDERATED LEARNING

In this section we introduce our algorithm GENFL.

**From batch to federated PAC-Bayes algorithms.** When training stochastic neural nets (SNNs) with PAC-Bayes objectives, it is common to assume that each weight follows a Gaussian distribution. For conciseness, we identify a SNN to the Gaussian distribution of all its weights  $\mathcal{N}(\mu, \text{Diag}(\sigma_i))$ . The works of Dziugaite and Roy (2017); Rivasplata et al. (2019); Biggs and Guedj (2021); Pérez-Ortiz et al. (2021); Perez-Ortiz et al. (2021); Biggs and Guedj (2022) proposed successful PAC-Bayesian training algorithms for SNNs which ensure generalisation guarantees. All these methods operate in a batch setting, *i.e.*, the optimiser has access to all data simultaneously. Thus, building on the work of Rivasplata et al. (2019); Pérez-Ortiz et al. (2021), we propose Alg. 1, a new learning algorithm called GENFL (Generalisation-driven Federated Learning), casting PAC-Bayes into FL. We stress that GENFL benefits from nonvacuous generalisation guarantees (Section 4).

**A generalisation-driven FL algorithm.** GENFL combines a federated learning protocol (*i.e.*, FedSGD, FedAvg, McMahan et al. (2017a)) with a PAC-Bayes objective  $f$ . It starts from the vector  $w_1 = (\mu_{\text{prior}}, \sigma_{\text{prior}})$  corresponding to the initial distribution  $P = \mathcal{N}(\mu_{\text{prior}}, \sigma_{\text{prior}})$  and outputs after  $T$  rounds

---

**Algorithm 1** GENFL. users are indexed by  $k$ ;  $B$  is the local minibatch size,  $E$  is the number of local epochs,  $\eta$  is the learning rate,  $f$  the PAC-Bayes objective. The prior is  $\mathcal{N}(\mu_{\text{prior}}, \sigma_{\text{prior}})$  with parameter  $\delta$ .

---

```

1: Server executes:
2:  $m \leftarrow \sum_{k=1}^K m_k$  ▷ Total dataset size
3:  $w_1 \leftarrow (\mu_{\text{prior}}, \sigma_{\text{prior}})$ 
4: for each round  $t$  do
5:    $S_t \leftarrow$  random set of  $\max(C \cdot K, 1)$  users
6:   for each user  $k \in S_t$  in parallel do
7:      $w_{t+1}^k \leftarrow$  userUpdate( $k, w_t, m$ )
8:   end for
9:    $w_{t+1} \leftarrow \sum_{k=1}^K \frac{m_k}{m} w_{t+1}^k$ 
10: end for
11:
12: userUpdate(k, w, m):
13:  $\mathcal{B} \leftarrow$  (split  $\mathcal{S}_k$  into batches of size  $B$ )
14: for each local epoch  $e = 1, 2, \dots, E$  do
15:   for each local minibatch  $b \in B$  do
16:      $w_s^k \leftarrow \mu^k + \sigma^k \odot \mathcal{N}(0, 1)$  ▷ Reparam. trick
17:      $w^k \leftarrow w^k - \eta \nabla_w f_{m, \delta, \mu_{\text{prior}}, \sigma_{\text{prior}}}(w_s^k; b)$ 
18:   end for
19: end for
20: return  $w^k$ 
Ensure: Global model distribution  $w_T$  mapped to  $\mathcal{N}(\mu_T, \sigma_T)$ 

```

---

$w_T = (\mu_T, \sigma_T)$  corresponding to the posterior  $Q = \mathcal{N}(\mu_T, \sigma_T)$ . Hence the learning procedure is divided in rounds, where subsets of users are sampled. Sampled users are requested to perform local updates following a PAC-Bayes training procedure designed to ensure a good generalisation of the posterior distribution. Because users train SNNs, they sample weights from the posterior. In order to learn the variance parameter of the posterior (Gaussian), we use the well-known reparameterisation trick (Kingma and Welling, 2014): instead of directly sampling from the distribution, we sample from a standard Gaussian distribution and then apply a transformation to obtain the sampled weights:  $W = \mu + \sigma V$  where  $V \sim \mathcal{N}(0, \text{Id})$ , this allows to compute with respect to  $\sigma$ . When the round ends, the global model is computed from the local updates, thanks to the aggregation function from the FL protocol (weighted mean, median).

Next we show that with a PAC-Bayes objective  $f$ , we adapt GENFL to FL-SOB, where  $\mathcal{S}$  is fully *i.i.d.*, *i.e.*, all user datasets have the same distribution, and to PFL where each dataset  $\mathcal{S}_i$  is *i.i.d.* but any two dataset can have distinct distributions.

### 3.1 GenFL for FL-SOB

**PAC-Bayesian objectives.** We assume that for any  $i$ ,  $\mathcal{D}_i = \mathcal{D}$ . Thus,  $\mathcal{S}$  is a *i.i.d.* dataset of  $m$  points. We then consider the true and empirical risks on all  $\mathcal{S}$   $R_{\mathcal{D}}, R_{\mathcal{S}}$ . In a batch setting, it would be natural to optimise the bounds (3), (4). However, the user  $i$  only has access to its personal dataset  $\mathcal{S}_i$  (of size  $m_i$ ) to optimise its model *while knowing other datasets are involved*. We then derive accordingly from (3), (4) two PAC-Bayesian learning algorithms, valid for any user, namely  $f_1, f_2$ . Note that  $f_1$  is adapted from the  $f_{classic}$  objective and  $f_2$  is adapted from  $f_{quad}$  of Pérez-Ortiz et al. (2021):

$$f_1(\mathcal{S}_i) = \hat{R}_{\mathcal{S}_i}(Q) + \sqrt{\frac{\text{KL}(Q\|P) + \ln \frac{2\sqrt{m}}{\delta}}{2m}}. \quad (8)$$

$$f_2(\mathcal{S}_i) = \left( \sqrt{\hat{R}_{\mathcal{S}_i}(Q) + \frac{\text{KL}(Q\|P) + \log\left(\frac{2\sqrt{m}}{\delta}\right)}{2m}} + \sqrt{\frac{\text{KL}(Q\|P) + \log\left(\frac{2\sqrt{m}}{\delta}\right)}{2m}} \right)^2. \quad (9)$$

Note that (8), (9) can be seen as proxys of (3), (4). Indeed, every user has access to the total number of data points  $m$  (as long as it is transmitted to the server), so the regularisation term (containing the KL divergence) is fully available, contrary to the empirical risk  $\hat{R}_{\mathcal{S}}$  which is then replaced by  $\hat{R}_{\mathcal{S}_i}$ . Note that in this case, the KL divergence is divided by  $m$  instead of  $m_i$  (which would be natural if we were optimising (3) (4) for  $\mathcal{S}_i$  instead of  $\mathcal{S}$ ). This suggests that each user has to give more weight, during the optimisation phase, to its data than to the regularisation. The reason behind this is that the server, by aggregating predictors, performs a regularisation step on a global level, hence the need to prioritise data on a local one.

**A global generalisation guarantee.** A major interest of the *i.i.d.* assumption on  $\mathcal{S}$  is that, as long as users all exploit the same posterior distribution  $Q$ , and they transmit their empirical scores  $\hat{R}_{\mathcal{S}_i}(Q)$ , every user is able to compute the global generalisation guarantee of (2). This allows to maintain the bound of the batch setting (involving the total number of data points  $m$ ), despite being in FL. This is empirically shown in Section 4. We present in Algorithm 2 FEDBOUND, the algorithm we use to compute the global bound (7), valid for all users simultaneously.

---

**Algorithm 2** FEDBOUND. The  $K$  users are indexed by  $k$ ;  $f$  PAC-Bayes objective, prior  $\mathcal{N}(\mu_{\text{prior}}, \sigma_{\text{prior}})$ ; posterior  $\mathcal{N}(\mu_T, \sigma_T)$ ,  $\delta, \delta'$  parameters,  $n$  number of Monte Carlo sampling

---

```

1: Server executes:
2:  $m \leftarrow \sum_{k=1}^K m_k$   $\triangleright$  Total dataset size
3:  $P = \mathcal{N}(\mu_{\text{prior}}, \sigma_{\text{prior}})$   $\triangleright$  Prior (learned or random)
4:  $Q = \mathcal{N}(\mu_T, \sigma_T)$   $\triangleright$  Posterior (learned)
5: for each user  $k \in K$  in parallel do
6:    $error^k \leftarrow \text{userMCSampling}(k, w_t, m)$ 
7: end for
8:  $error \leftarrow \sum_{k=1}^K \frac{m_k}{m} error^k$ 
9:  $KL\_inv \leftarrow \text{kl}^{-1}\left(error \mid \frac{1}{n} \ln\left(\frac{2}{\delta'}\right)\right)$ 
10:  $\text{Up-bound} \leftarrow \text{kl}^{-1}\left(KL\_inv \mid \frac{\text{KL}(Q\|P) + \ln \frac{2\sqrt{m}}{\delta}}{m}\right)$ 
11:
12: userMCSampling(k, w, m):
13: for each MC sampling  $i = 1, 2, \dots, n$  do
14:    $W_i^k \sim Q$   $\triangleright$  Sample weights from the posterior
15:    $error_i^k \leftarrow \hat{R}_{\mathcal{S}_k}(W_i^k)$   $\triangleright$  local empirical risk
16: end for
17:  $error^k \leftarrow \frac{1}{n} \sum_{i=1}^n error_i^k$ 
18: return  $error^k$ 

```

---

**Ensure:** Up-bound holding with probability  $1 - \delta - \delta'$

---

### 3.2 GenFL for personalised federated learning

**A general training for the prior distribution.** In PFL, the learning objective of each user may differ, while sharing some similarities that can be learned and transferred from one user to another. This framework requires adjustments of our learning objectives. Indeed, contrary to Section 3.1, there is no clear global generalisation guarantee, so each user has then to optimise its own personal learning objective from a commonly shared prior. Using either a random prior or a learnt one on a fraction of users data, we run GENFL similarly to Section 3.1 with our PAC-Bayesian objective of interest  $f$ . The output distribution of GENFL is then considered as a common prior for all users which then needs to be personalised.

**A personalisation step.** Once a common prior distribution has been obtained from the federated training, it is necessary for each user to personalise it to its own problem. To do so, we apply PAC-Bayesian objectives similar to those in Section 3.1, namely  $f_1$  (8) and  $f_2$  (9), where the batch size is modified from  $m$  to  $m_i$  for the  $i$ -th user. This reflect that each user now optimises its local goal instead of the global one. Each user ends up with its own personalised posterior distribution. The way personalised bounds are implemented is similar to Algorithm 2, but without the aggregation

step. We refer to Appendix B for additional details.

## 4 NUMERICAL EXPERIMENTS

### 4.1 Experimental framework

We follow the experimental framework of Pérez-Ortiz et al. (2021) combined with the FedAvg protocol McMahan et al. (2017a). We use the following libraries: Pytorch (Paszke et al., 2019) for deep learning, Flower (Beutel et al., 2020) for federated learning, Slurm (Yoo et al., 2003) for cluster experiments, and Hydra (Yadan, 2019) for overall experiment management. The cluster nodes we use have 48 SKYLAKE 3GHz CPUs. We do not use GPUs.

**Prior distribution over weights.** We propose two types of priors: data-free (random) prior chosen randomly around  $\mathcal{N}(0, \text{Id})$  (as in Dziugaite and Roy, 2017) and a data-dependent (learnt) prior. The latter is powerful to attenuate the KL divergence term, leading to sharper generalisation bounds and better accuracy. Such a data-driven prior implies to use a fraction of the dataset from the training data to optimise the prior. The bound computation is then realised with a reduced dataset size (divided by 2 in practice). However, the prior has gained efficiency (lower empirical risk) and the PAC-Bayes optimisation starts from a relevant point.

We use Gaussian distribution for both prior and posterior over the weights of a neural network. When data-free, the prior is  $P = \bigotimes_{l \in \text{layers}} \mathcal{N}(\text{truncated}(\mu_{\text{rand}}^l), \text{Diag}(\sigma_{\text{prior}}))$  with  $\mu_{\text{rand}}^l \sim \mathcal{N}(0, \frac{1}{\sqrt{n_{\text{in}}^l}})$ , and  $\sigma_{\text{prior}} \in \mathbb{R}^{+*}$ . The truncature is done at  $\pm \frac{2}{\sqrt{n_{\text{in}}^l}}$  where  $n_{\text{in}}^l$  is the dimension of the inputs of the layer  $l$ . In the case of data-dependent prior, we have  $P = \bigotimes_{l \in \text{layers}} \mathcal{N}(\mu_{\text{learnt}}^l, \text{Diag}(\sigma_{\text{prior}}))$ , where  $\mu_{\text{learnt}}^l$  is obtained via ERM on the prior set on half of the training set, the other half being used for bound computation and posterior optimisation.

**Dataset partition.** To build a *i.i.d.* FL setup, we consider the case where each user has exactly the same number of samples per class. We partition MNIST as follows: we fix the number of users to 100. Then, each user receives a dataset size of 540, each class having 54 images. In the case of the learnt prior, we split the training set of each user in half, the first one being used to train the prior, the second exploited by our learning algorithms.

In the case of non-*i.i.d.* FL setup, we follow McMahan et al. (2017a). First we sort MNIST by label, then we partition the dataset into chunks of 300 contiguous samples each (thus containing at most 2 labels, be-

cause it is sorted). Again we split each user dataset in several parts. When the prior is random, we save 10% of the dataset to create a validation set. Remaining data is exploited for optimisation. When considering learnt priors, we save again 10% of the dataset as a validation set, we exploit 40% of the dataset to train the prior (respecting the proportion of each class), the remaining 50% being used by our learning algorithms.

**Bound parameters.** We used  $\delta = 0.05$ ,  $\delta' = 0.01$ ,  $n = 150000$  Monte Carlo samples.

**Optimisation hyperparameters.** The prior distribution scale  $\sigma_{\text{prior}}$  is set to  $2,5 \times 10^{-2}$ , the learning rate is  $5 \times 10^{-3}$  for 100 users. In order to compare with the batch learning setting, we compute our algorithms with 1 user. In this case, we use a learning rate of  $5 \times 10^{-4}$  to reach better performances. The momentum is 0.95 for posterior optimization and 0.99 in prior optimisation. During prior optimisation, we used a dropout rate of 0.2 to avoid overfitting. As theoretical results of Section 2 require a loss function in  $[0, 1]$ , we use the bounded cross-entropy as in Pérez-Ortiz et al. (2021), *i.e.*,  $\ell(x, y) = \frac{1}{\ln(p_{\text{min}})} \cdot \ln(\tilde{\sigma}(x)_y)$  with  $\tilde{\sigma}(x)_y = \max(\text{sigmoid}(x)_y, p_{\text{min}})$ . We took  $p_{\text{min}} = 10^{-4}$ .

**KL penalty.** For stability reasons, we penalise the KL term during posterior optimisation (similarly to Pérez-Ortiz et al., 2021), thus we give more impact to the empirical risk during optimisation. Such a penalty helps performance and stability during training when random priors are involved. In this case, we use a penalty of 0.1.

**Federated Learning hyperparameters.** Starting from a random prior, we perform our algorithm during 200 rounds to make the SNNs converge. When learnt priors are considered, they are trained with a run of 100 rounds with 5 local epochs (convergence around 50 epochs). We then perform 10 additional rounds with 5 local epochs to train the posterior. In both cases, we select 10% of the users each round to participate in the training. As the dataset size of each user is small, we use a batch size of 25 (compared to 250 in the work of Pérez-Ortiz et al. (2021)).

**Neural Architecture.** We consider a stochastic 2 hidden layer MLP with 600 units each, resulting in 1,198,210 number of parameters for the prior (with fixed covariance matrix) and doubled for the SNN (as we consider diagonal covariance matrices).

**Positive variance prior.** To have a constrained positive standard deviation  $\sigma$  when sampling weights, we use the following transformation:  $\sigma = \ln(1 + \exp(\rho))$ . It makes  $\sigma$  always positive, and  $\rho$  can be any real number that is optimised during training procedure.

Table 1: Results for the FL-SOB scenario.  $\ell^{0-1}$  corresponds to the 0-1 loss. The test error column is made on the test set of MNIST. The Bound column corresponds to the generalisation bounds, computed with Alg. 2. The  $KL/m$  column corresponds to the KL divergence term in the bound divided by  $m = 60000$  in data-free prior or  $m = 30000$  data-dependent prior.

Setup		Bound	Test Err.	KL div
Prior	Obj.	$\ell^{0-1}$	$\ell^{0-1}$	$KL/m$
Random (1 client)	$f_1$	0.330	0,141	0,081
	$f_2$	0.316	0,092	0,138
Learnt (1 client)	$f_1$	0.028	0,023	<0,001
	$f_2$	0.028	0,020	0,001
100 users - GenFL - KL Penalty=0.1				
Random (us)	$f_1$	0.333	0,123	0,107
	$f_2$	0.342	0,090	0,163
Learnt (us)	$f_1$	0.061	0,030	<0,001
	$f_2$	0.088	0,029	0,002
100 users - GenFL - No KL Penalty				
Random (us)	$f_1$	0.415	0,256	0,039
	$f_2$	0.408	0,251	0,041
Learnt (us)	$f_1$	0.039	0,030	<0,001
	$f_2$	0.040	0,030	<0,001

## 4.2 Results

Note that in a classification problem, the generalisation error translates a positive influence of the learning phase as long as it is smaller than 1 (which is what we refer to with the term nonvacuous). Indeed, a bound below this threshold shows that the posterior will not fail at each try. However, we focus on posteriors with generalisation bounds or test error smaller than 50%. The reason is that, for a classification task, this threshold is the generalisation error of a randomised predictor with associated distribution Bernoulli(0.5). Thus, having results below this threshold provably show we generalise better than a naive strategy.

### FL-SOB setting.

Table 1 gathers our results for GENFL applied with  $f_1, f_2$  alongside FEDBOUND. We compared our results with 100 clients with the output of our algorithms for 1 client, corresponding to the batch learning case. Our FL algorithms benefit from nonvacuous theoretical guarantees and test errors. In the case of data-dependent priors, test errors of GENFL nearly reach for both  $f_1$  and  $f_2$ , the precision of their batch counterpart (3% in FL and 2% in batch). In the case of random prior, the KL penalty has a strong positive influence on the test errors. The generalisation bounds of our algorithms are uniformly deteriorated compared to the batch setting, *while being of the same magni-*

Table 2: Results for the PFL scenario.  $\ell^{0-1}$  corresponds to the 0-1 loss. The test error column is made on the test set of each user (10% of local dataset). The Gen. Bound column gathers generalisation bounds. Each user bound is computed locally with  $m_i = 300$  for learnt prior, while  $m_i = 540$  for random prior.

Setup		Gen. Bound $\ell^{0-1}$		
Prior	Obj.	min	mean	max
Random (us)	$f_1$	0,063	0,680	0,847
	$f_2$	0,075	0,713	0,893
Learnt (us)	$f_1$	0,054	0,112	0,222
	$f_2$	0,052	0,111	0,220
Test Error $\ell^{0-1}$				
Random (us)	$f_1$	0	0,552	0,767
	$f_2$	0	0,588	0,833
Learnt (us)	$f_1$	0	0,050	0,183
	$f_2$	0	0,044	0,150

*tude*. This is important to notice as this is the price to pay to adapt batch bounds to a federated setting. Indeed, as each user only optimised a proxy of the common generalisation bound, it is legitimate to retrieve in our results a short discrepancy comparing to the batch case.

While  $f_2$  is consistently achieving better generalisation upper bounds in Pérez-Ortiz et al. (2021), it is outperformed by  $f_1$  in the FL setting. However, notice that  $f_2$  provides uniformly better test errors than  $f_1$ , similarly to Pérez-Ortiz et al. (2021). Note that better results are achieved if one considers the KL penalty trick with data-free prior and no KL penalty trick with data-dependent prior. We interpret this fact as follows: given that the data-dependent prior is already performing well on training data, allowing the posterior optimisation to be unconstrained is not an issue as we found an area close from a local minimiser. However, as the random prior is not necessarily efficient, we need to move far from it to reach good empirical performances. However, moving freely from the prior distribution leads to a large KL divergence, hence the need to constrain the posterior optimisation to obtain both better bounds and test errors. A take-home message is that adapting PAC-Bayes algorithms to FL is effective: it gives nonvacuous results close to the batch setting.

### PFL setting.

Table 2 provides an overview of our results in the non-*i.i.d.* case. It gathers, for both generalisation bounds and test error, the minimum, mean and maximum performance of all 100 users. The averaged performances are deteriorated compared to Table 1 for all settings as we consider a harder problem. It is worth noticing



that our bounds are nonvacuous and that our algorithms with learnt priors benefit from bounds and test errors lower than 50%. Indeed, if we do not learn the prior, we see from the distribution of errors in Figure 1 that most users have a deteriorated bound and test errors. For learnt priors, the error distribution shows that all users enjoy a meaningful bound as well as sound performance. An interesting point is that the common prior distribution does not support all users uniformly as we can see in Figure 1. Indeed, our algorithms with random priors suffer from deteriorated bounds and test errors on average and the worst case, but approximately 15% enjoy good test errors and 9% benefit from theoretical guarantees lower than 40%. Also, our algorithms with learnt priors enjoy test errors and generalisation guarantees lower than 20% for all users. Furthermore, approximately half of the users benefit from test errors lower than 5%. This highlights the importance of the prior in this non-*i.i.d.* setting. As the test set of each user is small (60 images), some users achieve a 0% test error.

## 5 DISCUSSION

In this work, we propose a novel algorithm for FL in two different settings: FL-SOB, which allows to exploit a global generalisation guarantee while keeping data separated; as well as PFL, which only involves an *i.i.d.* assumption for each user’s dataset. Our work raises two questions: **(a) is it possible to remove the *i.i.d.* assumption?** **(b) is it possible to maintain a global generalisation guarantee, even in the personalised setting?** To answer (a), a line of work first initiated by Kuzborskij and Szepesvári (2019) (for *i.i.d.* data) and continued by Haddouche and Guedj (2023); Chugg et al. (2023); Jang et al. (2023) (for non-*i.i.d.* ones) focuses on PAC-Bayes bounds valid for data distribution with bounded variances. In the PFL setting, this could provide novel generalisation bounds without assuming each user possesses an *i.i.d.* dataset. About (b), the recent work of Sefidgaran et al. (2023) provides elements of answer: they derive a general PAC-Bayesian bound holding for the classical FL setting *for all users simultaneously* involving explicitly the number of users and rounds. This allows fruitful theoretical interpretations (especially on the number of rounds involved during the FL training), but leads to vacuous generalisation guarantees for classification task with a federated SVM. This contrasts with our nonvacuous guarantees, even for the personalised setting, at the cost of considering generic PAC-Bayesian and leading to a non-vacuous generalisation guarantees remains an open challenge that we aim to adress in a

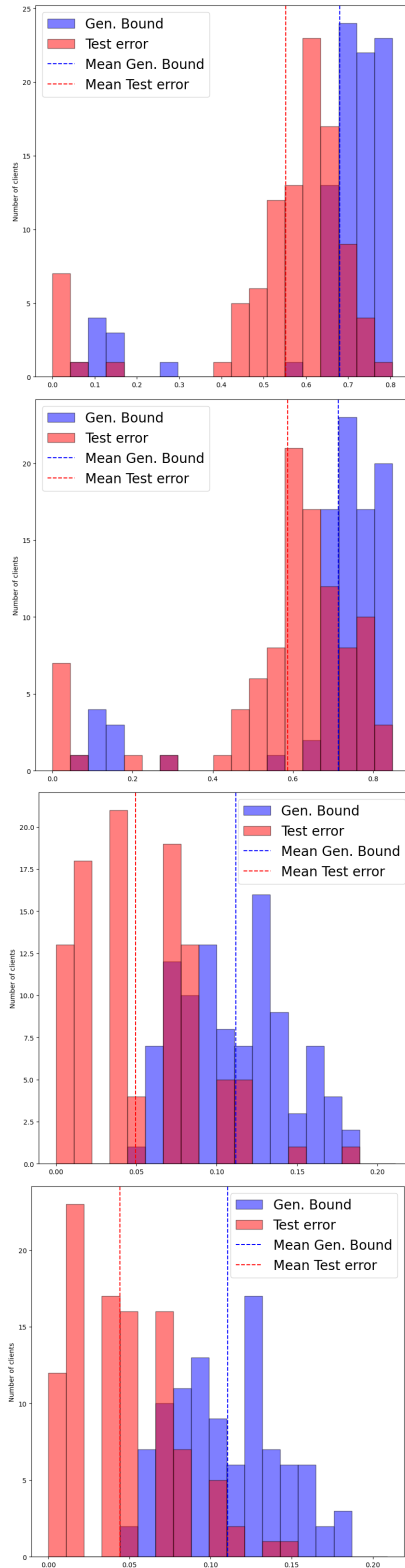


Figure 1: Histograms gathering test errors (red) and bound (blue) of all 100 users of the PFL setting. In order from top to bottom: Random prior- $f_1$ , Random prior- $f_2$ , Learnt prior- $f_1$ , Learnt prior- $f_2$ .

future work.

## References

- Achituve, I., Shamsian, A., Navon, A., Chechik, G., and Fetaya, E. (2021). Personalized Federated Learning With Gaussian Processes. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8392–8406.
- Agarwal, N., Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, B. (2018). cpSGD: Communication-efficient and differentially-private distributed SGD. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7575–7586.
- Alquier, P. (2021). User-friendly introduction to PAC-Bayes bounds. *arXiv*, abs/2110.11216.
- Alquier, P., Ridgway, J., and Chopin, N. (2016). On the properties of variational approximations of Gibbs posteriors. *Journal of Machine Learning Research*.
- Amit, R. and Meir, R. (2018). Meta-Learning by Adjusting Priors Based on Extended PAC-Bayes Theory. In *International Conference on Machine Learning (ICML)*.
- Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Kwing, H. L., Parcollet, T., Gusmão, P. P. d., and Lane, N. D. (2020). Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*.
- Biggs, F. and Guedj, B. (2021). Differentiable PAC-Bayes objectives with partially aggregated neural networks. *Entropy*, 23(10).
- Biggs, F. and Guedj, B. (2022). Non-vacuous generalisation bounds for shallow neural networks. In *Proceedings of the 39th International Conference on Machine Learning [ICML]*, volume 162 of *Proceedings of Machine Learning Research*, pages 1963–1981. PMLR.
- Biggs, F., Zantedeschi, V., and Guedj, B. (2022). On margins and generalisation for voting classifiers. In *NeurIPS*.
- Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191.
- Boucheron, S., Lugosi, G., and Massart, P. (2013). *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press.
- Catoni, O. (2007). *PAC-Bayesian supervised classification: the thermodynamics of statistical learning*. Institute of Mathematical Statistics.
- Chen, H. and Chao, W. (2021). FedBE: Making Bayesian Model Ensemble Applicable to Federated Learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- Chérif-Abdellatif, B.-E., Shi, Y., Doucet, A., and Guedj, B. (2022). On PAC-Bayesian reconstruction guarantees for VAEs. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics [AISTATS]*, volume 151 of *Proceedings of Machine Learning Research*, pages 3066–3079. PMLR.
- Chugg, B., Wang, H., and Ramdas, A. (2023). A unified recipe for deriving (time-uniform) PAC-Bayes bounds. *arXiv*, abs/2302.03421.
- Ding, N., Chen, X., Levinboim, T., Goodman, S., and Soricut, R. (2021). Bridging the Gap Between Practice and PAC-Bayes Theory in Few-Shot Meta-Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Duchi, J. (2007). Derivations for linear algebra and optimization. *Berkeley, California*, 3(1):2325–5870.
- Dziugaite, G. K. and Roy, D. (2017). Computing Non-vacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.
- Fard, M. M. and Pineau, J. (2010). PAC-Bayesian model selection for reinforcement learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Farid, A. and Majumdar, A. (2021). Generalization Bounds for Meta-Learning via PAC-Bayes and Uniform Stability. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Guedj, B. (2019). A Primer on PAC-Bayesian Learning. In *Proceedings of the second congress of the French Mathematical Society*, volume 33.
- Haddouche, M. and Guedj, B. (2022). Online PAC-Bayes Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Haddouche, M. and Guedj, B. (2023). PAC-Bayes Generalisation Bounds for Heavy-Tailed Losses

- through Supermartingales. *Transactions on Machine Learning Research*.
- Hardt, M., Price, E., and Srebro, N. (2016). Equality of Opportunity in Supervised Learning. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3315–3323.
- Hellström, F., Durisi, G., Guedj, B., and Raginsky, M. (2023). Generalization Bounds: Perspectives from Information Theory and PAC-Bayes. *arXiv*.
- Jang, K., Jun, K.-S., Kuzborskij, I., and Orabona, F. (2023). Tighter PAC-Bayes Bounds Through Coin-Betting. *arXiv*, abs/2302.05829.
- Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D’Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021). Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210.
- Kim, M. and Hospedales, T. M. (2023). FedHB: Hierarchical Bayesian Federated Learning. *arXiv*, abs/2305.04979.
- Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.
- Konečný, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016a). Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv*, abs/1610.02527.
- Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016b). Federated Learning: Strategies for Improving Communication Efficiency. *arXiv*, abs/1610.05492.
- Kotelevskii, N., Vono, M., Durmus, A., and Moulines, E. (2022). FedPop: A Bayesian Approach for Personalised Federated Learning. In *NeurIPS*.
- Kuzborskij, I. and Szepesvári, C. (2019). Efron-Stein PAC-Bayesian Inequalities. *arXiv:1909.01931*.
- Letarte, G., Germain, P., Guedj, B., and Laviolette, F. (2019). Dichotomize and generalize: PAC-Bayesian binary activated deep neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems [NeurIPS] 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 6869–6879.
- Li, L., Guedj, B., and Loustau, S. (2018). A quasi-Bayesian perspective to online clustering. *Electron. J. Statist.*, 12(2):3071–3113.
- Mammen, P. M. (2021). Federated Learning: Opportunities and Challenges.
- Maurer, A. (2004). A note on the PAC-Bayesian theorem. *arXiv*, cs/0411099.
- McAllester, D. A. (1998). Some PAC-Bayesian theorems. In *Proceedings of the eleventh annual conference on Computational Learning Theory*, pages 230–234. ACM.
- McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1):5–21.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017a). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.
- McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017b). Communication-Efficient Learning of Deep Networks from Decentralized Data.
- Mhammedi, Z., Grünwald, P., and Guedj, B. (2019). PAC-Bayes un-expected Bernstein inequality. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems [NeurIPS] 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 12180–12191.
- Mohri, M., Sivek, G., and Suresh, A. T. (2019). Agnostic Federated Learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4615–4625. PMLR.
- Nozawa, K., Germain, P., and Guedj, B. (2020). PAC-Bayesian contrastive unsupervised representation learning. In *Conference on Uncertainty in Artificial Intelligence [UAI]*.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein,

- N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Perez-Ortiz, M., Rivasplata, O., Parrado-Hernandez, E., Guedj, B., and Shawe-Taylor, J. (2021). Progress in Self-Certified Neural Networks. In *NeurIPS 2021 Workshop on Bayesian Deep Learning*.
- Pérez-Ortiz, M., Rivasplata, O., Shawe-Taylor, J., and Szepesvári, C. (2021). Tighter Risk Certificates for Neural Networks. *Journal of Machine Learning Research*, 22(227):1–40.
- Reisizadeh, A., Farnia, F., Pedarsani, R., and Jadbabaie, A. (2020). Robust Federated Learning: The Case of Affine Distribution Shifts. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.
- Rivasplata, O., Tankasali, V. M., and Szepesvári, C. (2019). PAC-Bayes with Backprop. *arXiv*, abs/1908.07380.
- Rothfuss, J., Fortuin, V., Josifoski, M., and Krause, A. (2021). PACOH: Bayes-optimal meta-learning with PAC-guarantees. In *International Conference on Machine Learning (ICML)*.
- Rothfuss, J., Josifoski, M., Fortuin, V., and Krause, A. (2022). PAC-Bayesian Meta-Learning: From Theory to Practice. *arXiv*, abs/2211.07206.
- Sakhi, O., Chopin, N., and Alquier, P. (2022). PAC-Bayesian Offline Contextual Bandits With Guarantees. *arXiv*, abs/2210.13132. To appear in ICML 2023.
- Scott, J., Zakerinia, H., and Lampert, C. H. (2023). PeFLL: A Lifelong Learning Approach to Personalized Federated Learning. *arXiv*, abs/2306.05515.
- Sefidgaran, M., Chor, R., Zaidi, A., and Wan, Y. (2023). Federated Learning You May Communicate Less Often! *arXiv*, abs/2306.05862.
- Seldin, Y., Laviolette, F., Cesa-Bianchi, N., Shawe-Taylor, J., and Auer, P. (2012). PAC-Bayesian Inequalities for Martingales. *IEEE Transactions on Information Theory*.
- Seldin, Y., Laviolette, F., Shawe-Taylor, J., Peters, J., and Auer, P. (2011). PAC-Bayesian Analysis of Martingales and Multiarmed Bandits. *arXiv*, abs/1105.2416.
- Shawe-Taylor, J. and Williamson, R. C. (1997). A PAC analysis of a Bayes estimator. In *Proceedings of the 10th annual conference on Computational Learning Theory*, pages 2–9. ACM.
- Suresh, A. T., Yu, F. X., Kumar, S., and McMahan, H. B. (2017). Distributed Mean Estimation with Limited Communication. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3329–3337. PMLR.
- Tan, A. Z., Yu, H., Cui, L., and Yang, Q. (2022). Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–17.
- Tolstikhin, I. O. and Seldin, Y. (2013). PAC-Bayes-Empirical-Bernstein Inequality. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States*, pages 109–117.
- Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2019). Robustness May Be at Odds with Accuracy.
- Vedadi, E., Dillon, J. V., Mansfield, P. A., Singhal, K., Afkanpour, A., and Morningstar, W. R. (2023). Federated Variational Inference: Towards Improved Personalization and Generalization. *arXiv*, abs/2305.13672.
- Yadan, O. (2019). Hydra - A framework for elegantly configuring complex applications. Github.
- Yagli, S., Dytso, A., and Vincent Poor, H. (2020). Information-Theoretic Bounds on the Generalization Error and Privacy Leakage in Federated Learning. In *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5.
- Yoo, A. B., Jette, M. A., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In Feitelson, D., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, pages 44–60, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. (2019). Bayesian Nonparametric Federated Learning of Neural Networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Pro-*

*ceedings of Machine Learning Research*, pages 7252–7261. PMLR.

Zantedeschi, V., Viillard, P., Morvant, E., Emonet, R., Habrard, A., Germain, P., and Guedj, B. (2021). Learning Stochastic Majority Votes by Minimizing a PAC-Bayes Generalization Bound. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. (2021). A survey on federated learning. *Knowledge-Based Systems*, 216:106775.

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L. E., and Jordan, M. (2019). Theoretically Principled Trade-off between Robustness and Accuracy. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR.

Zhang, L., Lei, X., Shi, Y., Huang, H., and Chen, C. (2023a). Federated Learning for IoT Devices With Domain Generalization. *IEEE Internet Things J.*, 10(11):9622–9633.

Zhang, X., Huang, A., Fan, L., Chen, K., and Yang, Q. (2023b). Probably Approximately Correct Federated Learning.

Zhang, X., Li, Y., Li, W., Guo, K., and Shao, Y. (2022). Personalized Federated Learning via Variational Bayesian Inference. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26293–26310. PMLR.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2021). A Comprehensive Survey on Transfer Learning. *Proc. IEEE*, 109(1):43–76.

## Checklist

1. For all models and algorithms presented, check if you include:
  - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
  - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
  - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
  - (a) Statements of the full set of assumptions of all theoretical results. [Yes]
  - (b) Complete proofs of all theoretical results. [Not Applicable]
  - (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
  - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
  - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
  - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No]
  - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
  - (a) Citations of the creator If your work uses existing assets. [Yes]
  - (b) The license information of the assets, if applicable. [Not Applicable]
  - (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
  - (d) Information about consent from data providers/curators. [Not Applicable]
  - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
  - (a) The full text of instructions given to participants and screenshots. [Not Applicable]
  - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
  - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

## Federated Learning with Nonvacuous Generalisation Bounds Supplementary Material

### A Approximating the inverted $kl$

To approximate the inverted kl divergence, we re-use the technique presented in Dziugaite and Roy (2017, Appendix A). As there is no closed form formula for  $\text{kl}^{-1}(q | c)$ , we approximate it via root-finding techniques. For all  $q \in (0, 1)$  and  $c \geq 0$ , define  $h_{q,c}(p) = \text{KL}(q||p) - c$ . Then  $h'_{q,c}(p) = \frac{1-q}{1-p} - \frac{q}{p}$ . Assuming we possess a good enough initial estimate  $p_0$  of a root of  $h_{q,c}(\cdot)$ , we can obtain improved estimates of a root via Newton's method:

$$p_{n+1} = \text{N}(p_n; q, c) \text{ where } \text{N}(p; q, c) = p - \frac{h_{q,c}(p)}{h'_{q,c}(p)}.$$

This suggests the following approximation to  $\text{kl}^{-1}(q | c)$ : 1. Let  $\tilde{b} = q + \sqrt{\frac{c}{2}}$ . 2. If  $\tilde{b} \geq 1$ , then return 1. 3. Otherwise, return  $\text{N}^k(\tilde{b})$ , for some integer  $k > 0$ .

### B Additional details for Section 3.2

We provide here more details about the procedures of section 3.2.

**PAC-Bayes learning objectives** As stated in the main documents, our learning objectives here are mainly similar to those in Equations (8) and (9). At the variation that now, the KL divergence is regularised by  $m_i$  for the user  $i$  instead of the common  $m$ . This comes from the fact that now, each user does not try to optimise proxys of a common global generalisation goal but only its personal McAllester bound, depending only on its data.

$$f_1(\mathcal{S}_i) = \hat{\text{R}}_{\mathcal{S}_i}(Q) + \sqrt{\frac{\text{KL}(Q||P) + \ln \frac{2\sqrt{m_i}}{\delta}}{2m_i}}$$

$$f_2(\mathcal{S}_i) = \left( \sqrt{\hat{\text{R}}_{\mathcal{S}_i}(Q) + \frac{\text{KL}(Q||P) + \log\left(\frac{2\sqrt{m_i}}{\delta}\right)}{2m_i}} + \sqrt{\frac{\text{KL}(Q||P) + \log\left(\frac{2\sqrt{m_i}}{\delta}\right)}{2m_i}} \right)^2$$

More precisely we state explicitly the personalised algorithm we use in Algorithm 3. This algorithm shows that each user sacrifices half of its data in phase 1 to learn jointly a prior through GENFL. In phase 2, we re-use the ClientUpdate procedure of Alg. 1 to personalise the prior to each client through the PAC-Bayesian learning goal  $f$  (being  $f_1$  or  $f_2$  in practice).

For the sake of completeness, we also precise in Algorithm 4 how we calculate the personalised generalisation bounds once the PFL training of Algorithm 3 is performed.

Algorithm 4 simply computes each PAC-Bayesian associated fort each client. Note that here  $\mathcal{S}_k^2$  denotes the halve of  $\mathcal{S}_k$  which has not been used to train the prior distribution.

**Algorithm 3** PFL algorithm with PAC-Bayesian personalisation step. The  $K$  clients are indexed by  $k$ ;  $B$  is the local minibatch size,  $E$  is the number of local epochs,  $\eta$  is the learning rate  $f$  PAC-Bayes objective,  $\mu_{\text{prior}}$  prior center parameters,  $\sigma_{\text{prior}}$  prior scale hyper-parameter,  $\delta$  confidence parameter

---

```

1:  $w_0 \leftarrow (\mu_{\text{prior}}, \sigma_{\text{prior}})$ 
2:  $\forall k, \mathcal{S}_k = \mathcal{S}_k^1 \cup \mathcal{S}_k^2$  ▷ splitting datasets in half
3: Step 1: learning the prior.
4:  $\mathcal{S}_P \leftarrow \cup_{k=1}^K \frac{1}{2} \mathcal{S}_k^1$  ▷ each user involves half of its dataset.
5:  $\mu_P \leftarrow \text{GENFL}(\mu_{\text{prior}}, \mathcal{S}_P)$ 
6:  $w_1 \leftarrow (\mu_P, \sigma_{\text{prior}})$  ▷ Prior is constructed.
7: Step 2: Personalisation step.
8: for each client  $k = 1, \dots, K$  do
9:    $w_Q^k \leftarrow \text{ClientUpdate}(k, w_1, \frac{m_k}{2})$ 
10: end for
11: Return  $(w_Q^k)_{k=1 \dots K}$ 
12:
13: ClientUpdate(k, w, m):
14:  $\mathcal{B} \leftarrow$  (split  $\mathcal{S}_k^2$  into batches of size  $B$ )
15: for each local epoch  $e = 1, 2, \dots, E$  do
16:   for each local minibatch  $b \in B$  do
17:      $w_s^k \leftarrow \mu^k + \sigma^k \odot \mathcal{N}(0, 1)$  ▷ Reparam. trick
18:      $w^k \leftarrow w^k - \eta \nabla_w f_{m, \delta, \mu_{\text{prior}}, \sigma_{\text{prior}}}(w_s^k; b)$ 
19:   end for
20: end for
21: return  $w^k$ 
Ensure: Personalised distributions  $w_Q^k \Leftrightarrow \mathcal{N}(\mu_Q^k, \sigma_Q^k)$ 

```

---

## C Additional experiments

### C.1 Additional experiments to compare with Dziugaite and Roy (2017)

We also provide a proof of concept of GENFL by comparing with former method from Dziugaite and Roy (2017).

**Computation time** The training procedures (prior, posterior) take separately several hours. The bound computation is slowed due Monte Carlo sampling.

#### C.1.1 Setup

**Dataset Partition** We used the exact same dataset partition as previously, the exact *i.i.d.* partition. Each Client will have the exact same number of samples per class and same number of class. Resulting in a dataset size of  $540 * 100 = 54000$

**Random initialization of the prior** The prior is  $P = \mathcal{N}(w_0, \lambda Id)$ , and  $\lambda$  is a hyperparameter that is learn during optimization. In order to have a union bound argument, This  $\lambda$  must be discretize using the following formula:  $\lambda = c \exp(-j/b)$  with  $c = 0.1, b = 100$  two fixed constant and  $j \in \mathbb{N}$ . The  $w_0$  is sampled throught the distribution  $\mathcal{N}(0, \sigma)$  and then truncated to  $[-2\sigma, 2\sigma]$ . We used  $\sigma = 0.04$ .

**SGD on centered parameters of the posterior** For stability reason, Dziugaite and Roy (2017) proposed to learn the centered parameters of the posterior  $w$  via SGD on train set. Doing this step, provide better result than learning the posterior directly. But is a deceiving step, because in this case, PAC-Bayes from scratch is not self-sufficient.

**Learn posterior from the SGD** Then the posterior is initialized to  $Q = \mathcal{N}(w, \text{Diag}(|w|))$  ( $w$  coming from the previous SGD). This posterior is optimized on the classic Mc Allester loss:  $\hat{R}_S(Q) + \sqrt{\frac{\text{KL}(Q \| P) + \ln \frac{2\sqrt{m}}{\delta}}{2m}}$

**Algorithm 4** FEDBOUND. The  $K$  clients are indexed by  $k$ ;  $f$  PAC-Bayes objective, prior distribution  $\mu_P, \sigma_{\text{prior}}$ ; posterior  $\mathcal{N}(\mu_T, \sigma_T)$  output of Algorithm 3,  $\delta, \delta'$  confidence parameters,  $n$  number of MC sampling

```

1: Server executes:
2:  $P = \mathcal{N}(\mu_P, \sigma_{\text{prior}})$  ▷ Learned prior from Alg. 3
3:  $Q = \mathcal{N}(\mu_T, \sigma_T)$  ▷ Posterior (learned)
4: for each client  $k \in K$  in parallel do
5:    $error^k \leftarrow ClientMCSampling(k, w_t, m_k/2)$ 
6:    $KL\_inv \leftarrow kt^{-1} \left( error^k \mid \frac{1}{n} \ln \left( \frac{2}{\delta'} \right) \right)$ 
7:    $Up\_bound_k \leftarrow kt^{-1} \left( KL\_inv \mid \frac{KL(Q||P) + \ln \frac{2\sqrt{m}}{\delta}}{m} \right)$ 
8: end for
9:
10: ClientMCSampling(k, w, m):
11: for each MC sampling  $i = 1, 2, \dots, n$  do
12:    $W_i^k \sim Q$  ▷ Sample weights from the posterior
13:    $error_i^k \leftarrow \hat{R}_{S_k^2}(W_i^k)$  ▷ local empirical risk
14: end for
15:  $error^k \leftarrow \frac{1}{n} \sum_{i=1}^n error_i^k$ 
16: return  $error^k$ 
Ensure:  $(Up\_bound_k)_{k=1 \dots K}$  each valid with probability  $1 - \delta - \delta'$ 

```

**Bounds parameters** Bounds were computed with confidence parameters  $\delta = 0.05, \delta' = 0.01$  and with  $n = 150000$  monte carlo samples.

### C.1.2 Results

Metrics	GenFL	GenFL	Dziugaite
Clients number	100	1	"1"
Trainset Size	54000	54000	60000
Train Error	0.013	~0	~0
Test Error	0.019	0.018	0.016
SNN Train Error	0.051	0.039	0.028
SNN Test Error	0.051	0.043	0.033
PAC-Bayes Bound	0.241	0.175	0.186
KL divergence	7039	4629	6534

Table 3: Train and Test Error are the ones from the optimized (deterministic) NN learnt by the SGD.