WAFFLE: Weighted Averaging for Personalized Federated Learning

Martin Beaussart
EPFL
martin.beaussart@epfl.ch

Felix Grimberg
EPFL
felix.grimberg@epfl.ch

Mary-Anne Hartley
EPFL
mary-anne.hartley@epfl.ch

Martin Jaggi
EPFL
martin.jaggi@epfl.ch

Abstract

In federated learning, model personalization can be a very effective strategy to deal with heterogeneous training data across clients. We introduce WAFFLE (Weighted Averaging For Federated LEarning), a personalized collaborative machine learning algorithm that leverages stochastic control variates for faster convergence. WAFFLE uses the Euclidean distance between clients' updates to weigh their individual contributions and thus minimize the personalized model loss on the specific agent of interest. Through a series of experiments, we compare our new approach to two recent personalized federated learning methods—Weight Erosion and APFL—as well as two general FL methods—Federated Averaging and SCAFFOLD. Performance is evaluated using two categories of non-identical client data distributions—concept shift and label skew—on two image data sets (MNIST and CIFAR10). Our experiments demonstrate the comparative effectiveness of WAFFLE, as it achieves or improves accuracy with faster convergence.

1 Introduction

Federated learning (FL) is a collaborative learning technique to build machine learning (ML) models from data distributed among several participants ("agents"). The objective is to generate a common, robust model not by exchanging the data between agents, but rather through exchanging parameter updates for the common model that all agents share at a certain frequency. This technique addresses some major problems of centralized data-sharing approaches, can enable data privacy and security and is therefore attractive to many applications with sensitive data.

We consider centralized FL, in which a single server orchestrates the execution of the algorithms. Each iteration of learning can thus be divided into four main steps. Firstly, the central server sends the current model to all agents. Secondly, each agent trains the model with their data and, thirdly, sends the updated model parameters back to the central server. Finally, the central server aggregates these results and generates a new global model. WAFFLE intervenes in this last step: instead of a global one-size-fits-all model, the aggregation produces a personalized model for each specific agent.

1.1 Control Variates for Heterogeneous Data

Federated learning develops a common model for all agents, typically under the assumption that data is *independent and identically distributed (IID)* across agents. In real-world applications, this assumption rarely holds, and thus the convergence rate and final performance of FL algorithms like Federated Averaging (FedAvg) can vary significantly [McMahan et al., 2017, Karimireddy et al.,

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

2020]. Karimireddy et al. [2020] introduce SCAFFOLD, which uses stochastic control variates (SCV) to tackle the *client-drift* that results from non-IID-ness across agents. Each agent maintains an SCV estimating the sum of all other agents' gradients, and uses it to "correct" each step of stochastic gradient descent (SGD). Compared to FedAvg, SCAFFOLD converges faster and towards a better model. WAFFLE extends SCAFFOLD to collaboratively train a personalized ML model.

1.2 Model Personalization

A complementary solution for dealing with non-IID data is to train a unique (personalized) model for each agent, rather than a single global model. Recently, there have been several key works on personalization in federated learning, see e.g. Kulkarni et al. [2020] for an overview. Some methods of adapting global models for individual agents are summarized in Kairouz et al. [2019], such as local fine-tuning [Deng et al., 2020, Kairouz et al., 2019], multi-task learning [Smith et al., 2017], and model-agnostic meta-learning (MAML) [Finn et al., 2017, Fallah et al., 2020].

The model personalization method most similar to WAFFLE is Weight Erosion (WE) [Grimberg et al., 2020]. It is conceptually related to local fine-tuning, which consists of training a common global model and subsequently personalizing it for each agent by performing a small number of SGD steps on the agent's local data (resulting in one model per agent) [Kairouz et al., 2019]. WE achieves a smoother and more differentiated transition from global to local training by using a *weighted average* of the agents' gradients to update the server model at each communication round. Thus, the server model after running WE once is personalized for one specific agent—called *user* or *Alice*.[1] Like WE, WAFFLE uses aggregation weights derived from the Euclidean distance between gradients to update the global model, but it combines this approach with the use of SCVs (Section 1.1). These different approaches for transitioning from global to local training are illustrated in Figure 3 (Section 3).

Instead of interpolating between gradients at each round, a slightly different approach consists of interpolating between a local and global model [Grimberg et al., 2021, Donahue and Kleinberg, 2020, Deng et al., 2020, Zhang et al., 2021]. For instance, Deng et al. [2020] propose the adaptive FL algorithm APFL that achieves personalization by learning a global model and a set of local corrective models with which to interpolate. Another related approach is personalized MAML, where a global model is optimized with respect to (w.r.t.) the *loss after local fine-tuning* on each agent's local data [Finn et al., 2017, Fallah et al., 2020]. Finally, regularization can be used (instead of weighted averaging) to mix global and local training [Smith et al., 2017, Li et al., 2021]. For instance, Li et al. [2021] recently showed very promising results with Ditto, where the local model's training is split across all communication rounds and regularized by its distance from the current global model.

1.3 Benchmark

While several categories of non-IID data have been identified by Kairouz et al. [2019], we focus only on label skew and concept shift:

- Label skew: The distribution of labels varies between agents, but the "true" classification function is the same for all agents. This is, so far, the type of non-IID-ness most commonly used in personalized FL benchmarks [Li et al., 2021, Fallah et al., 2020, Deng et al., 2020].
- Concept shift: The mapping from the features to the label varies across agents. We include concept shift to model naturally partitioned data sets like those used by Smith et al. [2017].

1.4 Contributions

- We present a new personalized collaborative ML algorithm named WAFFLE (Weighted Averaging For Federated LEarning), which builds on the existing methods SCAFFOLD and Weight Erosion (WE) by using stochastic control (SC) and weighted gradient aggregation. To the best of our knowledge, this is the first method that uses SC in model personalization.
- We build an image classification benchmark with label skew and concept shift. We then use it to compare WAFFLE against baselines (Federated Averaging and Local), its parent

[1] The authors motivate this setting, in which all other agents contribute selflessly to Alice's objective, by constructing a fictive scenario of cross-silo FL across hospitals. Alternatively, for cross-silo FL, WE (or WAFFLE) can be run several times in parallel to obtain personalized models for all agents.

methods (SCAFFOLD and WE), and against other state-of-the-art personalized FL methods (APFL), thus contributing to further empirical evaluation of these methods.

- Finally, we show that in most cases, WAFFLE converges faster than its competitors WE and APFL and matches or improves their accuracy, while requiring less hyperparameter tuning.

2 The WAFFLE Algorithm

To introduce WAFFLE, we will first summarize the known SCAFFOLD FL algorithm on which it is based, and then detail how personalization can be achieved by weighted averaging between agents, for suitable weight choices. SCAFFOLD maintains an estimate of what all agents are learning, called a *control variate*, so that agents can simulate having all the data and estimate the direction in which they should update their gradient. Thus, it is more efficient and addresses the problem of *client drift*. Indeed Karimireddy et al. [2020] prove that SCAFFOLD converges to the globally optimal model instead of converging to a weighted average of local models and does so much faster and more accurately than FedAvg in the presence of inter-agent heterogeneity.

2.1 Personalization Using Weighted Update Aggregation

WAFFLE is essentially a personalized version of SCAFFOLD, where the goal is no longer to get a global model for all agents but a personalized model for one particular agent, (*Alice*). The idea is to start from global training (SCAFFOLD) and to gradually move to local training. SCAFFOLD computes the gradient for the server model ($\Delta \boldsymbol{x}$) and the *control variate* $\Delta \boldsymbol{c}$ (the assumption of what other agents are learning) by averaging each agent's update $\Delta \boldsymbol{y}_i$ and local *control variates* $\boldsymbol{c}_i$ at each round [Karimireddy et al., 2020, Algorithm 1, Line 16]. It is at this step that WAFFLE intervenes: Instead of averaging all agents at each round, WAFFLE assigns a weight α_i to each agent and computes $\Delta \boldsymbol{x}$ and $\Delta \boldsymbol{c}$ by a weighted combination as per Equation (1). Changes w.r.t. SCAFFOLD are highlighted in red:

$$(\Delta \boldsymbol{x}, \Delta \boldsymbol{c}) \leftarrow \sum_{i \in \{1,\dots,N\}} \alpha_i \, (\Delta \boldsymbol{y}_i, \Delta \boldsymbol{c}_i) \tag{1}$$

For N agents, we recover SCAFFOLD (global training) by setting all weights equal to $\frac{1}{N}$. In contrast, we recover local training by setting all weights to 0, except the weight of Alice.

WAFFLE is based on a smooth transition from global to local training. To do this, the weight of each agent is updated according to the degree of personalization desired for the round. Just as SCAFFOLD uses the control variates to converge to the global model for the union of all agents' datasets, WAFFLE uses them to converge to a personalized model based on a weighted subset of agents (Figure 1). Each agent is included and weighted in this subset based on the current degree of personalization and whether their dataset is sufficiently similar to Alice's, as measured by the distance between their gradients at each round.

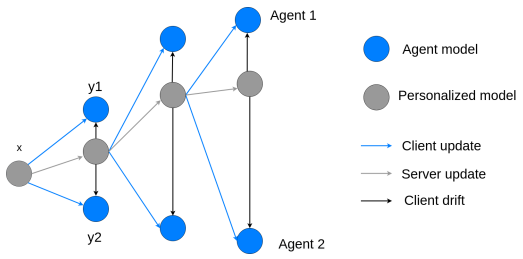

Figure 1: Evolution of the personalized model and agent models on WAFFLE in a three-rounds scenario where agent 1 has a higher weight than agent 2.

The weight of each agent at a given round depends on the distance between its and Alice's gradient. As the method uses SGD (i.e., stochastic gradients), the agent weights are also subject to stochastic noise. Indeed, at some rounds, some agents may have an atypical gradient resulting in a bad weight, even if their contribution is still needed for the global model. Therefore, we average the current weight with the last two weights to smooth out strong random effects (line 10 of Algorithm 1).

To compare each agent with Alice, we use the Euclidean distance between gradients. Simply, we want an agent with a small distance to have a higher weight than an agent with a larger distance. Like SCAFFOLD, we have a central server that coordinates all the agents. At each starting round it gives the current personalized model and control variate to all agents for training. When the agents finish training the model with their data, they send the updated model and their local control variate back to the server. The server then calculates the Euclidean distance of each update from the selected agent and modifies the weight assigned to each agent accordingly. Finally, it calculates the new personalized model and the new control variate (Equation (1)). The implementation of WAFFLE can be seen in the modified version of SCAFFOLD (Algorithm 2, Appendix B) where changes w.r.t. to the original code are written in red ink. Agent weights are obtained by CalcWeight (Algorithm 1), which is explained in the next subsection.

2.2 Computing Agent Weights for WAFFLE

WAFFLE uses a new data-dependent approach to select the contribution weight α_i^r of agent i at each round r. It relies on hyperparameters Ω and Ψ: two functions defining the degree of personalization for each round r (where a value of 1 corresponds to global training and 0 is personalized (local) training). As explained later, we reduce the functions Ω and Ψ to a single, positive real-valued hyperparameter for ease of tuning.

The weight is calculated according to Equations (2) and (3), where

- agent $i^\star$ is Alice
- d_i is the distance between the gradients of agents i and $i^\star$
- dM is the largest such distance: $dM \leftarrow \max_i d_i$
- Analogously, dm is the smallest such distance (excluding $d_{i^\star}$)
- $\Omega(r)$ and $\Psi(r)$ denote the degree of personalization for round r

The weight of an agent is based on the distance between its gradient and the gradient of agent $i^\star$, but also on the distribution of the distances of the agents. The distance of $i^\star$ would always be 0, we want to change it to a similar value compared to the distances of other agents. The further away Alice's distance is from the others, the less weight we assume the agents have compared to Alice. We need a "distance" $d_{i^\star}$ for Alice in Equation (3). Thus, Equation (2) serves to assign her such a distance based on the hyperparameter Ω. When $\Omega \approx 1$ (i.e., more global training), Alice's assigned "distance" ($d_{i^\star}$) is close to the next smallest distance dm, resulting in similar weights for these two agents. The more Ω decreases towards 0, the smaller ($d_{i^\star}$) becomes compared to all other agents' distances, resulting in a large weight for Alice (i.e., more local training).

$$d_{i^\star} \leftarrow dm \cdot \left(1 - \frac{dM - dm}{dM}(1 - \Omega(r))\right) \tag{2}$$

$$\alpha_i^r \leftarrow \max\left\{\Psi(r) - \frac{d_i - d_{i^\star}}{dM - d_{i^\star}},\, 0\right\} \tag{3}$$

Equation (3) serves as a threshold of inter-agent utility. To pass the threshold, the gradient of agent i must be closer than the other agents' gradients (based on the range of distances). With Ψ close to 0 (i.e., more local training), only a small fraction of the range between the minimum and maximum distance will map to a non-zero weight. Thus, learning is concentrated in a subset of agents with the most similar gradients.

Using the two functions Ω and Ψ, we can define different strategies for WAFFLE. We could, as an example, shift rapidly toward zero and thus drastically limit larger steps in global training in the initial phase of personalization.

As Algorithm 1 would otherwise have $2R$ positive real-valued parameters to tune (values of Ω and Ψ for each round), we restrict ourselves to $\Omega(r) = \Psi(r) =$ a single-parameter function of r. However, more methods should be investigated to see the full potential of WAFFLE in different scenarios.

In this paper, we use a sigmoid function to give the general training a gradual slope towards smaller values Equation (4) and Figure 8. The function has a parameter $\Delta\Omega$ that controls the slope, a higher value will steepen the slope and thus move faster towards the local training. According to our

experiments, a good value for $\Delta\Omega$ is around 3.2, irrespective of the model used. With this value, `WAFFLE` transitions to mostly local SGD after $70-90\%$ of the total number of rounds, depending on the similarity of the agents' gradient to that of Alice. A simple modification to delay or accelerate the local learning time is to move the Ω function horizontally by adding a value to r.

$$\Omega(r) = \Psi(r) = \frac{1}{1 + e^{\Delta\Omega \cdot (r/(R/2)-1))}} \tag{4}$$

Algorithm 1. CalcWeights-Ω

input : N, R, $i^\star$, r, $\{\Delta \boldsymbol{y}_i\}$, α^{r-1}, α^{r-2}
hyperparameter : Functions Ω and Ψ from $\{1, \ldots, R\}$ to $\mathbb{R}_{>0}$
output : Weight vectors $\bar{\alpha}^r, \alpha^r$

foreach agent $i \in \{1, \ldots, N\}$ **do**
  `// Compute the Euclidean distance between updates of agents` i `and` $i^\star$`.`
  $d_i \leftarrow \|\Delta \boldsymbol{y}_i - \Delta \boldsymbol{y}_{i^\star}\|_2$ `//` $d_{i^\star} = 0$
$dM \leftarrow \max_{i \neq i^\star} d_i, \qquad dm \leftarrow \min_{i \neq i^\star} d_i$
$d_{i^\star} \leftarrow dm \cdot \left(1 - \frac{dM - dm}{dM}(1 - \Omega(r))\right)$ `//` $0 \leq d_{i^\star} \leq dm$ `if` $\Omega(r) \in [0, 1]$
foreach agent $i \in \{1, \ldots, N\}$ **do**
  $\alpha_i^0 \leftarrow \max\left\{\Psi(r) - \frac{d_i - d_{i^\star}}{dM - \min_i d_i}, 0\right\}$
  if $r \geq 0.95R$ **then**
    $\alpha_i^0 \leftarrow 0$ if $i \neq i^\star$ else 1
$\alpha^0 \leftarrow \frac{1}{\sum \alpha_i^0}(\alpha_1^0, \ldots, \alpha_N^0)^\top$
$\bar{\alpha}^r \leftarrow \frac{1}{3}(\alpha^{-2} + \alpha^{-1} + \alpha^0)$

3 Benchmarking Personalized Collaborative Learning Methods

To compare the performance of `WAFFLE` with the state of the art, we evaluate it against two recent personalized FL methods—`APFL` [Deng et al., 2020] and `Weight Erosion (WE)` [Grimberg et al., 2020]—as well as two global methods—`Federated Averaging` [McMahan et al., 2017] and `SCAFFOLD` [Karimireddy et al., 2020]—and finally, local training using only Alice's dataset (`Local`). `SCAFFOLD` and `WE` were selected to validate whether `WAFFLE` outperforms the methods it builds upon. We included `APFL` in the benchmark as an unrelated personalized FL method which has an open-source implementation. The benchmark consists of training standard image classification networks on standard IID and non-IID image datasets described below.

Two 10-class image classification tasks are used based on either the MNIST or CIFAR10 datasets. To generate confidence estimates, we run the MNIST benchmark with five different seeds. However, using five seeds for CIFAR10 would be prohibitive due to computational limitations.

Distributions and Models. We test five distributions (A,B,C,A*,B*), the first one (A) is IID and serves as a baseline. A classic label skew distribution (B) was selected for consistency with several other benchmarks [Collins et al., 2021, Deng et al., 2020, McMahan et al., 2016], where only a small number of agents have samples from any given class (uniformly distributed amongst these agents). Distribution C exhibits more nuanced label skew, where the labels are not uniformly distributed among the agents. Instead, each agent will have 40% of one label, 20% of two labels, and 10% of two other labels. We also present two other distributions—concept shift on the IID distribution (A*), and concept shift on top of classic label shift (B*). Concept shift is achieved by randomly swapping the class labels of all agents, except for Alice. We specify each distribution as a list of ten numbers that sum to one (see Figure 2)). For each agent i, we move the list to the right i times to get non-IID data, e.g. with distribution B, agent 0 will only have labels 0 to 3, agent 1 labels 1 to 4, etc.

For MNIST we use a LeNet-5 model [LeCun et al., 1989] and for CIFAR10 we use DLA [Yu et al., 2017].

Hyperparameters. We use an SGD optimizer with a learning rate of 0.1 for MNIST and 0.01 for CIFAR10. Following [Yu et al., 2017] we add a momentum 0.9 and a weight decay 5e-4 for CIFAR10 . Concerning the batch size, it is set to 32 for MNIST and 128 for CIFAR10. Except for the APFL method, we define the hyperparameters in a principled manner (based on the expected behaviour of the algorithm). As WE and WAFFLE use a weight system, we attempt to set the parameters in such a way that both algorithms reach fully local learning at about the same time. In practice, this was difficult to achieve as some hyperparameters are very sensitive. Fully local learning is obtained when all weights reach zero except for Alice (Agent 0), who has a weight of one. Figure 3 shows an example of the evolution of the weights. Alice (agent 0) is placed in the centre. For the hyperparameters of APFL, we use the same method mention in the paper by [Deng et al., 2020], and adaptively update the hyperparameter to guarantee the best generalization performance.

Class		0	1	2	3	4	5	6	7	8	9
Distribution	A	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
	B	0,25	0,25	0,25	0,25	0	0	0	0	0	0
	C	0	0	0	0,1	0,2	0,4	0,2	0,1	0	0

Figure 2: Repartition of labels between all agents for each distribution benchmarked

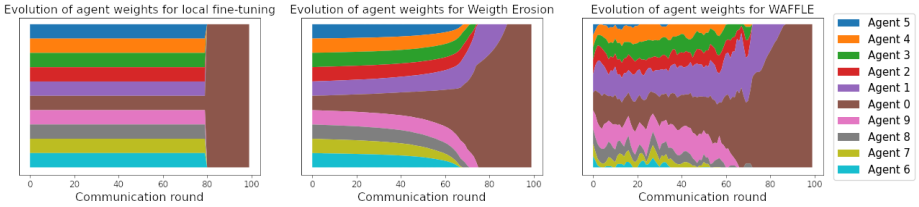

Figure 3: Evolution of weight for personalized FL with the distribution C on MNIST (seed = 1)

The tuned hyperparameters are listed in Table 3 (Appendix B). Out of the two methods, WAFFLE is the easiest to parametrize, as the hyperparameter does not change in contrast to the weighting system used by WE. This is because WAFFLE can work very well without tuning, as opposed to WE which requires hyperparameter tuning to perform well. As explained in Section 2.2, we purposefully restrict WAFFLE to a single hyperparameter, $\Delta\Omega$ (Equation (4)) and we use the same value for all experiments. Nevertheless, we show in the result section that WAFFLE is able to obtain a very good accuracy even without tuning.

4 Results

We present the results of our benchmarking in Table 1 below. For each method and each distribution, it shows the best accuracy reached at any of the 100 epochs. In the MNIST section, results are averaged across five random seeds and listed along with the standard deviation as $avg \pm std$.

To see the evolution of the accuracy, we provide an example in Figure 4. The evolution for each experiment is shown in the appendix (see Figures 6 and 7).

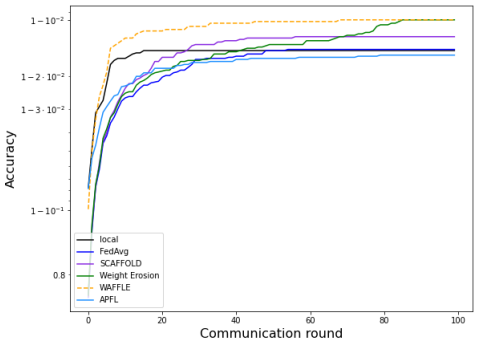

Figure 4: Evolution of accuracy with distribution C on MNIST, average over five seeds of the best accuracy obtained up to each turn.

MNIST. On MNIST, WAFFLE and WE outperform the other methods for the label skews (B, C), but not for IID distribution (A). The difference between WAFFLE and WE lies in the speed of convergence: As expected WAFFLE converges faster thanks to the use of SCVs. For the concept shift distributions (A*, B*), all personalized FL methods improve accuracy compared to global FL methods. Among all personalized FL methods, WE performs slightly better in this setting. Here, WAFFLE is not more efficient than WE (time taken to reach convergence). This may be related to the fact that only small fractions of agents are useful in this distribution, so until we reach this critical point, both algorithms are bound to the same slow increase until a threshold. Regarding APFL, the method also performs better than global FL methods except for the IID distribution (A), but it struggles to achieve a better result than the other two personalized FL methods and Local.

CIFAR10. Concerning CIFAR10, the lack of other seeds makes us less confident about the interpretation of our results, however, the results are coherent with our expectations. For example, like with MNIST, global FL methods outperform personalized methods on the IID distribution and present a significant improvement compared to `Local`. Distribution B and C (label skew) produce results close to those of MNIST, where `WAFFLE` and `WE` outperform others. Concerning concept shift, personalized FL methods are still a great improvement compared to global FL or `Local`. The accuracy of `WAFFLE` and `WE` is approximately the same but as for MNIST we don't have a noticeable difference between the speed of convergence to best accuracy. For `APFL`, we get similar results as MNIST, `APFL` performs better than the global FL methods, but this time it is better than `Local`. Unfortunately, compared to the other two customized methods, it performs much worse.

Table 1: Comparison of average local test accuracy and standard deviation of different algorithms given different data distributions.

	Algorithms					
	`Local`	`FedAvg`	`SCAFFOLD`	`WE`	`WAFFLE`	`APFL`
MNIST						
IID (A)	96.88% ±0.47	98.99% ±0.08	**99.03%** ±0.08	99.01% ±0.16	98.93% ±0.1	97.55% ±0.13
Distr. B	99.38% ±0.09	98.97% ±0.18	99.24% ±0.17	**99.69%** ±0.07	**99.69%** ±0.03	99.36% ±0.09
Distr. C	98.54% ±0.1	98.56% ±0.1	98.77% ±0.1	**99.0%** ±0.08	**99.0%** ±0.08	98.45% ±0.08
Distr. A*	96.56% ±0.27	13.37% ±3.21	19.93% ±5.95	**98.09%** ±0.26	97.99% ±0.26	96.24% ±0.22
Distr. B*	99.47% ±0.07	45.26% ±4.1	57.51% ±4.02	**99.67%** ±0.09	99.63% ±0.11	99.44% ±0.07
CIFAR10						
IID (A)	0.771%	0.925%	**0.927%**	0.912%	0.907%	0.776%
Distr. B	0.888%	0.884%	0.85%	**0.935%**	0.931%	0.895
Distr. C	0.832%	0.857%	0.868%	0.907%	**0.914%**	0.809%
Distr. A*	0.763%	0.248%	0.259%	0.877%	**0.9%**	0.78%
Distr. B*	0.876%	0.782%	0.476%	**0.937%**	0.926%	0.878%

5 Limitations and Future Work

The benchmark has some limitations that should be addressed in future work and are discussed in detail in Appendix A. Firstly, the theoretical properties of `WAFFLE` should be investigated, in particular whether the guarantees of `SCAFFOLD` are maintained. Secondly, our benchmark should be expanded to cover more types of non-IID distributions and naturally partitioned datasets, and to include more personalized FL methods like the recently published `Ditto`. Finally, the versatility of `WAFFLE` could also be further investigated by optimizing the functions Ω and Ψ for different contexts.

6 Conclusion

In this work, we propose `WAFFLE`, a personalized FL algorithm based on `SCAFFOLD`, that can overcome client drift and accelerate convergence to an optimal personalized model. `WAFFLE` uses the Euclidean distance between agent updates to weigh their contributions and thus transition gradually from global to more local training. We explored the performance of `WAFFLE` in two cases of non-identical data—concept shift and label skew—on two standard image datasets—MNIST and CIFAR10. We demonstrate that in most cases `WAFFLE` shows a faster convergence and a possible improvement in accuracy compared with other personalized FL methods. We also demonstrate that `WAFFLE` is easier to handle than most personalized learning methods. Indeed, a fixed default value for its single hyperparameter yields competitive results in all test cases. However, by design, `WAFFLE` can be adapted to a task by modifying Ω and Ψ functions and thus potentially reach even better accuracy.

Acknowledgements

We thank David Roschewitz for sharing with us his implementation of APFL, and Freya Behrens for inspiring us on how to create synthetic concept shift. We are grateful to Celiane De Luca for her suggestions and comments on the writing of this paper. We thank Vincent Yuan and Damien Gengler for their collaboration with Martin Beaussart on adapting Weight Erosion to a neural network for the first time, which lead directly to the present work [Gengler et al., 2020].

References

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS - Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282. PMLR, 2017.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. Survey of personalization techniques for federated learning, 2020.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *Advances in Neural Information Processing Systems*, 2017-December:4425–4435, 5 2017. URL http://arxiv.org/abs/1705.10467.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *34th International Conference on Machine Learning, ICML 2017*, 3:1856–1868, 3 2017. URL http://arxiv.org/abs/1703.03400.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. 2020. URL http://arxiv.org/abs/2002.07948.

Felix Grimberg, Mary-Anne Hartley, Martin Jaggi, and Sai Praneeth Karimireddy. Weight erosion: An update aggregation scheme for personalized collaborative machine learning. In *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning*, pages 160–169. Springer, 2020.

Felix Grimberg, Mary-Anne Hartley, Sai P Karimireddy, and Martin Jaggi. Optimal model averaging: Towards personalized collaborative learning. In *FL ICML workshop*, 2021.

Kate Donahue and Jon Kleinberg. Model-sharing games: Analyzing federated learning under voluntary participation. *arXiv preprint arXiv:2010.00753*, 2020. URL http://arxiv.org/abs/2010.00753.

Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *ICLR - International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ehJqJQk9cw.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. 2021. URL http://arxiv.org/abs/2012.04221.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. 2 2021. URL http://arxiv.org/abs/2102.07078.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2 2016. URL http://arxiv.org/abs/1602.05629.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4): 541–551, 1989. ISSN 0899-7667. doi: 10.1162/NECO.1989.1.4.541.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, 7 2017. URL http://arxiv.org/abs/1707.06484.

Damien Gengler, Martin Beaussart, and Vincent Yuan. Personalized federated image classification using weight erosion. 2020. URL https://www.epfl.ch/labs/mlo/wp-content/uploads/2021/05/crpmlcourse-paper854.pdf.