Supplementary Material:
Learning Rate Adaptation for Differentially Private Learning

1 Comparison to the Private Selection algorithm by Liu and Talwar

	Mean acc.	Std acc.	Mean evals
ADADP	0.6349	0.0033	1
Priv. $\gamma = 2^{-9}$	0.6386	0.0011	601.04
Priv. $\gamma = 2^{-8}$	0.6368	0.0113	224.30
Priv. $\gamma = 2^{-7}$	0.6317	0.0399	131.76
Priv. $\gamma = 2^{-6}$	0.6266	0.0516	49.77
Priv. $\gamma = 2^{-5}$	0.6215	0.0552	32.25
Priv. $\gamma = 2^{-4}$	0.6053	0.0662	14.79
Priv. $\gamma = 2^{-3}$	0.5576	0.1280	6.72
Priv. $\gamma = 2^{-2}$	0.5467	0.1278	3.42
Priv. $\gamma = 2^{-1}$	0.4728	0.1688	2.04

Table 1: Comparison of ADADP and the private selection algorithm [1, Alg. 2] for different values of the parameter γ. 'Mean evals' denotes the mean of the number of training runs (100 epochs each) needed for one evaluation of each algorithm. 'Mean acc.' and 'Std acc.' denote the mean and standard deviation of the test accuracy of the resulting model, respectively. Both methods have the same (ε, δ)-privacy for the training dataset. The private selection algorithm needs also a validation set which it also exposes with (ε, δ)-DP.

2 Additional Figure to Section 5.5

Figure 1 shows the likelihoods of the test data as the learning progresses for the Gaussian mixture model of Section 5.5. The values of ε are for $\delta = 10^{-6}$.

3 Application to Federated Learning

Federated learning [2] presents another setting where classical hyperparameter adaptation with a validation set may be impractical. One obvious pain point is skewed distribution of data on different clients, which may lead to different clients requiring very different learning rates that would be very difficult to tune without an adaptive algorithm.

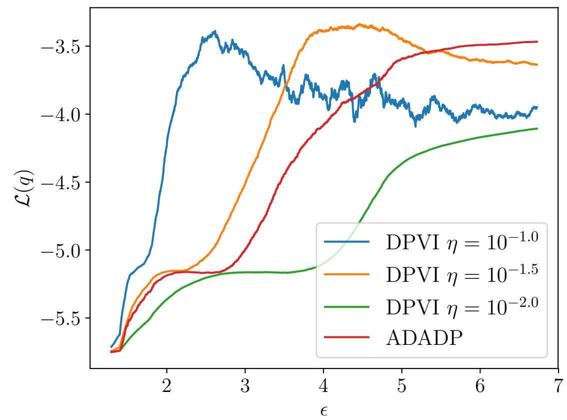


Figure 1: Log-likelihood $\mathcal{L}(q_\xi)$ for ADADP and for DP-SGD for different learning rates η, when $\sigma = 1.2$.

3.1 Adaptive federated averaging algorithm

We consider the federated averaging algorithm described in [2]. The idea is such that the same model is first distributed to several clients. The clients update their models based on their local data, and these models are then aggregated after a given interval by a server which then averages the models to obtain a global model. This global model is then again distributed to the clients.

In the algorithm described in [2, Algorithm 1], a random subset of clients is considered at each aggregation. We consider the case $C = 1$ where each client participates in every aggregation, and replace the gradient step in client update with a non-private variant of Algorithm 2 given in the main text.

In [2], SGD with a constant learning rate is used for the updates of the clients. The motivation for using the learning rate adaptation comes from the fact that after averaging and distributing, the model at each client may be very far from the optimum for the local data and thus small steps are needed in the beginning of each sub training. Moreover, the data may vary considerably between the clients, leading to varying optimal learning rates.

For the learning rate adaptation, we use the same pro-

Algorithm 1 Learning rate adaptive client update

ClientUpdate(k, θ)
Split $\mathcal{P}_k$ into batches of size $|B|$.
for each local step $\ell = 1, \ldots, E$ **do**
Draw a batch B_1 and evaluate at θ_ℓ:

$$G_1 \leftarrow \frac{1}{|B|} \sum_{i \in B_1} \nabla f_{\theta_\ell}(x_i).$$

Take a step size $\frac{\eta_\ell}{2}$:

$$\theta_{\ell+1/2} \leftarrow \theta_\ell - \frac{\eta_\ell}{2} G_1,$$

Take a step of size η_ℓ:

$$\widehat{\theta}_{\ell+1} \leftarrow \theta_\ell - \eta_\ell G_1,$$

Draw a batch B_2, and evaluate at $\theta_{\ell+1/2}$:

$$G_2 \leftarrow \frac{1}{|B|} \sum_{i \in B_2} \nabla f_{\theta_{\ell+1/2}}(x_i).$$

Take a step of size $\frac{\eta_\ell}{2}$:

$$\widetilde{\theta}_{\ell+1} \leftarrow \theta_{\ell+1/2} - \frac{\eta_\ell}{2} G_2$$

Evaluate: $\mathrm{err}_\ell \leftarrow \|\mathrm{err}(\theta_{\ell+1}, \widehat{\theta}_{\ell+1})\|_2$
if $\mathrm{err}_\ell > \tau$: **then**
$\theta_{\ell+1} \leftarrow \theta_\ell$ (Discard step)
end if
update: $\eta_{\ell+1} = \min(\max(\frac{\tau}{\mathrm{err}_\ell}, \alpha_{\min}), \alpha_{\max}) \cdot \eta_\ell$.
end for

cedure as in ADADP, but without the additive noise and clipping of the gradients. We also use the condition $\mathrm{err}_i < \tau$ for the model update as it makes the algorithm considerably more stable.

3.2 Experiments

We compare the federated averaging algorithm with adaptive learning rates to the constant learning rate SGD. All experiments are implemented using PyTorch.

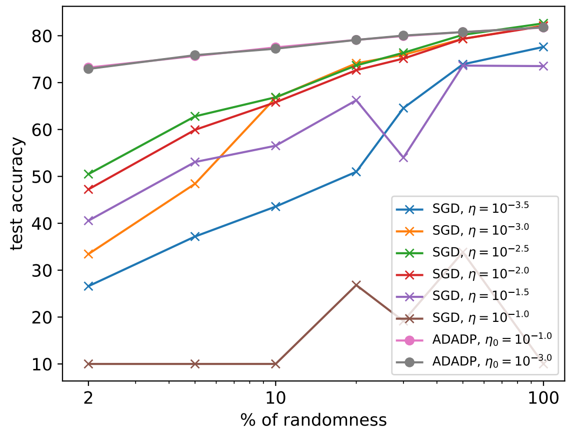
3.2.1 Dataset and test architectures

CIFAR-10 consists of colour images classified into 10 classes. The training set contains 50000 and the test set 10000 examples. Each example is a 32×32 image with three RGB channels. We use a simple neural network, which consists of two convolutional layers followed by three fully connected layers. The convolutional layers use 3×3 convolutions with stride 1, followed by ReLU and max pools, with 64 channels each. The output of the second convolutional layer is flattened into a vector of dimension 1600. The fully connected layers have 500 hidden units. Last layer is passed to softmax of 10 classes with cross-entropy loss. The total number of parameters for this network is about 10^6.

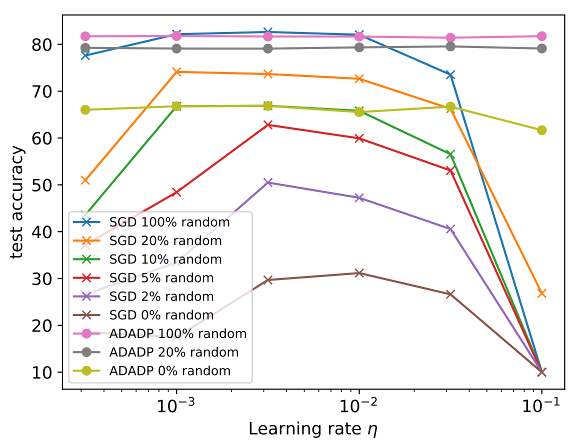
3.2.2 Experimental Results

We consider a pathological case, where the CIFAR-10 training data is divided to five clients such that client 1 has cars and trucks, client 2 planes and ships, client 3 cats and dogs, client 4 birds and frogs and client 5 deers and horses. Then, each client has 10000 images. We interpolate between this pathological case and a uniformly random distribution of data between the five clients. Figure 3a depicts the test accuracies for the learning rate adaptive algorithm and SGD. The learning rate of SGD is tuned in the grid $\{\ldots, 10^{-2.5}, 10^{-2.0}, 10^{-1.5}, \ldots\}$. We use in all alternatives $|B| = 10$. We see that as the distribution of data becomes more pathological (33% of the data chosen randomly), the learning rate adaptive method is able to maintain the overall performance much better than SGD. Figure 3b corresponds here to the fully pathological case. For a given minibatch size $|B|$, each client carries out E number of sub steps between each aggregation such that $|B| \cdot E = 10000$ (one epoch of data for each client).

Figure 2 illustrates further how ADADP is able to adapt even for highly pathological distribution of data whereas the performance of (even an optimally tuned) SGD reduces drastically when the data becomes less uniformly distributed.



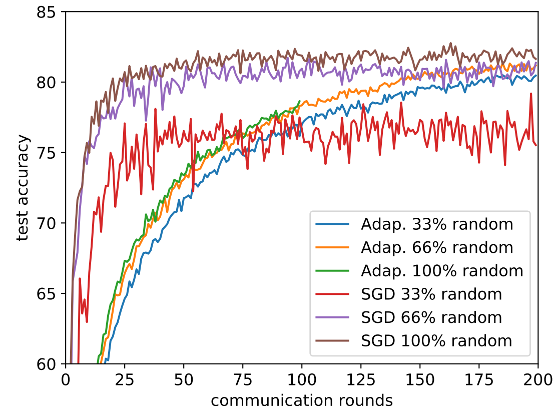
(a) Test accuracy vs. % of randomness.



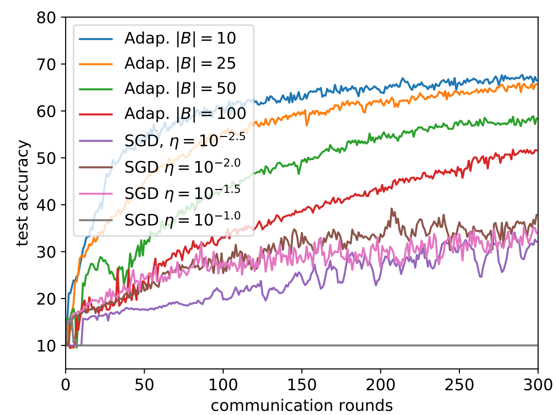
(b) Test accuracy vs. learning rate η for different levels of randomness. For ADADP η means here the initial η_0.

Figure 2: CIFAR-10 test accuracies for the federated averaging algorithm after 200 communication rounds using ADADP for different initial learning rates η_0 and constant learning rate SGD for different η. The training data is interpolated between the pathological case and the uniformly random distribution of data. All points are averages of three runs.

Adam gave poor results in this example. Figure 4 shows the test accuracies in the interpolated case, where 33% of the data is chosen randomly for each client, for the best initial learning rates found from the grid $\{\ldots, 10^{-5.5}, 10^{-5.0}, 10^{-4.5}, \ldots\}$. We use here $|B| = 10$. Notice here the different scale of y-axis as in Figure 3a.



(a) Interpolated case.



(b) 0% random case.

Figure 3: CIFAR-10 test accuracies for the federated averaging algorithm using ADADP and a learning rate tuned SGD, when the training data is interpolated between the pathological case and the uniformly random distribution of data.

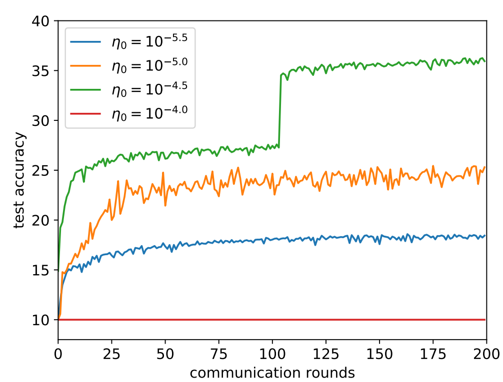


Figure 4: Test accuracies for the federated averaging algorithm, when the client updates are done using Adam using different initial learning rates η_0.

References

[1] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 298–309. ACM, 2019.

[2] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, 2017.