Technische Universität München

Trustworthy AI: Federated and privacy-preserving ML

AI in Medicine I: Practical Exercise 5

Authors:

Name	ID
Mohammed Elbushnaq	03786474
Muhammed Elmasry	03786399
Evangelos Fragkiadakis	03786480
Nada Elsherbeny	03788337

Part 1: Differentially Private Model Training

a) In this part, we train a DP model to examine its general behavior. We trained an instance of the model ResNet18 for the dataset CIFAR10. We used the library Opacus to implement DP to our model. The dataset has 10 classes, the optimizer is RMSprop, and the loss function is the categorical cross-entropy. In table 1, we can see the parameters used and the model's output values. It is evident that the training time is substantially longer when we implement DP, compared to regular training and the model's accuracy suffers due to the introduction of noise in the data.

Dataset	CIFAR10
ε DP	4.0
δ	1.00E-05
Epochs	15
MAX_GRAD_NORM	1.0
ε	inf.
DP Train Time	20'26"
DP Test Accuracy (%)	51.64
Train Time	4'30"
Test Accuracy (%)	61.97

Table 1: Parameters and Output values for CIFAR10 dataset

b/c) Application of DP to Medical Data

Moving on to actual medical data, we apply DP on 3 different datasets: DermaMNIST, PneumoniaMNIST and BloodMNIST. We trained the model with and without DP. When applying DP, we examined 3 different "privacy levels": ε = 1.0 (high privacy), ε = 4.0 (medium privacy) and ε = 8.0 (low privacy). Additionally, for a given value of ε (ε=1.0), we used a different value for the maximum gradient normalization parameter (MGD = 2.0 instead of MGD = 1.0). We can observe all the data quantitatively in Table 2 and qualitatively in Figure 1. We observe that the model's actual performance depends on the dataset and its characteristics. DP can hinder the performance significantly overall, and the magnitude of the effect of each library is dependent on the dataset and the task.

In DermaMNIST, the model's predictive performance is not really affected, with relatively stable accuracy for all values of ε. In PneumoniaMNIST, all privacy levels have the same effect on accuracy. Lastly, in BloodMNIST, only the high level of privacy (ε= 1.0) significantly affected the model's accuracy. Medium and high level did not alter the performance. Timewise, all DP models required significantly more time, with higher levels of accuracy demanding more time per epoch.

Regarding the maximum gradient normalization parameter, our brief experiment leads us to hesitantly conclude that increasing the normalization cut-off does not improve the model's performance. To the contrary, accuracy remains unchanged and the training time rises.

Consequently, selecting the parameters for DP model is a process of trial and error, in which we have to experiment with different parameter set-ups in order to find the appropriate formulation for the specific dataset and machine learning task.

DermaMNIST				
	DP Test Acc. (%)			Test Acc. (%)
MAX_GRAD_NORM \ ε	1.0	4.0	8.0	inf
1.0	66.78	66.78	66.79	n/a
2.0	66.78			n/a
n/a	n/a	n/a	n/a	66.58
	DP Train Time			Train Time
MAX_GRAD_NORM \ ε	1.0	4.0	8.0	inf
1.0	47'36"	46'40"	42'53"	n/a
2.0	46'45"			n/a
n/a	n/a	n/a	n/a	18'32"
PneumoniaMNIST				
	DP Test Acc. (%)			Test Acc. (%)
MAX_GRAD_NORM \ ε	1.0	4.0	8.0	inf
1.0	62.60	62.60	62.60	n/a
2.0	62.60	n/a	n/a	n/a
n/a	n/a	n/a	n/a	78.96
	DP Train Time			Train Time
MAX_GRAD_NORM \ ε	1.0	4.0	8.0	inf
1.0	51'10"	50'56"	49'48"	n/a
2.0	51'55"	n/a	n/a	n/a
n/a	n/a	n/a	n/a	20"18"
BloodMNIST				
	DP Test Acc. (%)			Test Acc. (%)
MAX_GRAD_NORM \ ε	1.0	4.0	8.0	inf
1.0	30.21	53.16	54.76	n/a
2.0	20.57	n/a	n/a	n/a
n/a	n/a	n/a	n/a	39.82
	DP Train Time			Train Time
MAX_GRAD_NORM \ ε	1.0	4.0	8.0	inf
1.0	43'41"	42'59"	44'09"	n/a
2.0	46'12"	n/a	n/a	n/a
n/a	n/a	n/a	n/a	18'45"

Table 2: DP Parameters and Output Values for the datasets: DermaMNIST, PneumoniaMNIST, BloodMNIST

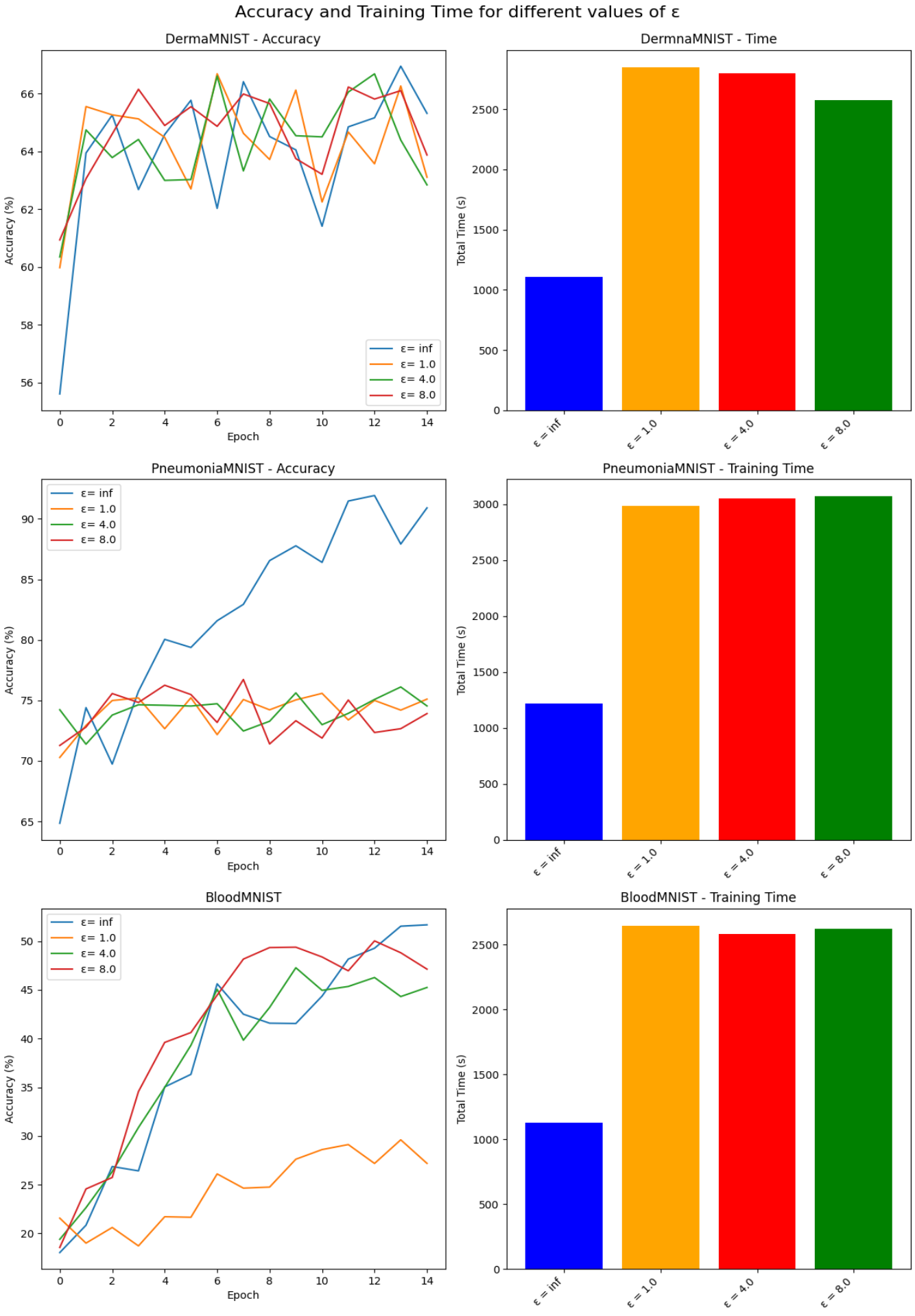

Figure 1: Accuracy per epoch and total training time for different values of ε

Part 2: Threats of federated learning

a) a) Given that the Data is IID and we have 5 clients, we can clearly see in Table 3 that the global model's accuracy is related to each client model's accuracy. However, it's also shown that if one client's accuracy is getter lower (as seen in client 0 round 20 to 30 in Table 3), doesn't mean that the accuracy of the global model will also be decreased, as it's dependent on all the clients, which might face increasing in the accuracy. In conclusion, we can see that the global model's accuracy was improving throughout the training despite the fact that some clients weren't. Same case with the loss.

Round number	Client 0 Accuracy	Client 1 Accuracy	Client 2 Accuracy	Client 3 Accuracy	Client 4 Accuracy	Global Accuracy
10	0.544	0.561	0.546	0.5615	0.549	0.5611
20	0.6125	0.5835	0.594	0.58	0.6015	0.5957
30	0.603	0.6075	0.609	0.615	0.6195	0.6108

Table 3: the accuracy of each client and the global model, every 10 epochs (rounds of federated learning)

b) adjustment of individual components

As shown in Table 4, we can see that changing the architecture can influence the accuracy and the loss of our prediction, as shown in the difference between ResNet-18 (with 76.3% accuracy) and our base model, LeNet (with 56.9% accuracy).

Model Architecture	Global loss (CE)	Global Accuracy
LeNet	0.03855	0.5691
Pretrained ResNet-18	0.03355	0.7633
ResNet-18	0.03896	0.7282

Table 4: the cross-entropy loss and accuracy of different model architectures, given the same hyperparameters and dataset.

For all the following experiences, the model was run for 10 epochs (rounds) on LeNet, with a batch size of 32 and Adam optimizer with its default momentum and learning rate (lr = 0.001, betas = (0.9, 0.999)). We can clearly see in Table 5 that having more clients tends to have less global accuracy.

Number of Clients	Global loss	Global Acc
1 (global model)	0.037395	0.6000
3	0.035669	0.6018
5	0.038587	0.5684
10	0.044497	0.4918

Table 5: the global loss and accuracy with different number of clients

As shown in Table 6, the higher the batch size, the lower the global accuracy. However, this is not the same case for accuracy here, as it keeps improving to a certain point but then decreases again, even while the loss decreases. Also, it's worth mentioning that the time of the very low batch size is much higher!

Batch size	Global loss	Global Accuracy	Simulation time Colab (T4 GPU)
1	1.34013	0.5246	25 mins
8	0.14413	0.5926	6 mins
16	0.07307	0.5851	4 mins
32	0.03915	0.5613	3 mins
64	0.02099	0.5293	3 mins
128	0.01149	0.4827	3 mins

Table 6: model evaluation with different batch sizes and the time it takes to run each.

Furthermore, after trying different Oprimizers, most of them had similar accuracies except for RMSprop, which showed significantly low accuracy and high loss, as shown in Table 7.

Moreover, when altering the learning rate (with the same Adam Optimizer), as shown in Table 8. The accuracy and loss of the global model showed much higher differences. With 0.001 being the best accuracy and loss by far

Optimizer	Global loss	Global Accuracy
Adagrad	0.03925	0.5541
RMSprop	0.06054	0.2695
AdamW	0.03817	0.5635
Adam	0.03965	0.5466

Table 7: model evaluation with different optimizers

Learning rate	Global loss	Global Accuracy
0.1	0.07256	0.1064
0.01	0.05096	0.4086
0.001	0.03965	0.5466
0.0001	0.05197	0.4072

Table 8: model evaluation with different learning rates

b) Adversarial samples in federated learning

In this experiment, after conducting 10 training rounds on our model, utilizing 10 different clients for each session, the model achieved an accuracy rate of 50.5%. However, when we introduced a data poisoning technique by manipulating the training data on one client, the accuracy dropped to 47.2%. Surprisingly, when we applied a slight noise to the data, it had the unexpected effect of acting as an augmentation technique, leading to an overall increase in accuracy of 52.1%, as shown in Table 9. This may be seen in this specific case as just data augmentation and helped generalize the model, but in the other scenarios (with a higher amount of noise), it can be considered as an adversarial attack.

Gaussian Noise STD	Global loss (CE)	Global Accuracy
0	0.0439	50.5%
2	0.0451	49.8%
3	0.0501	47.2%
0.5	0.0427	52.1%

Table 9: evaluation of the global model when introducing Gaussian noise of one client with different standard deviations

In conclusion, with a higher amount of noise, the scenario could be considered an adversarial attack. This implies that noise may start introducing harmful perturbations beyond a certain threshold, misleading the model and causing a decline in performance. This highlights the delicate trade-off between using noise for beneficial augmentation and avoiding noise that could be exploited as an adversarial strategy.