|  |  |  |  |
| --- | --- | --- | --- |
|  |  | DP | No DP |
| **CIFAR-10** | Acc Test | 58.06 | 72.08 |
| Acc Train | 61.11 | 97.64 |
| ε | 49.99 | inf |
| Time | 28201.90 s | 385.38 s |
| **DermaMNIST** | Acc Test | 71.86 | 68.46 |
| Acc Train | 72.47 | 90.89 |
| ε | 49.99 | inf |
| Time | 2883.29 s | 52.59 s |
| **PneumoniaMNIST** | Acc Test | 86.77 | 83.93 |
| Acc Train | 97.96 | 99.96 |
| ε | 49.99 | inf |
| Time | 2047.42 s | 35.00 s |
| **BloodMNIST** | Acc Test | 84.34 | 58.09 |
| Acc Train | 87.07 | 94.08 |
| ε | 49.99 | inf |
| Time | 5395.57 s | 90.73 s |
| **OrganCMNIST** | Acc Test | 79.36 | 85.49 |
| Acc Train | 90.14 | 99.05 |
| ε | 49.99 | inf |
| Time | 5989.60 s | 98.28 s |

**Task 1 a)**

The code provided implements a robust training pipeline for a neural network model using PyTorch, with the added ability to apply Differential Privacy (DP) for improved privacy preserving mechanisms. The process starts with the definition of data transformations, such as tensor conversion and normalization. Key parameters related to differential privacy, including `MAX\_GRAD\_NORM`, `EPSILON` and `DELTA`, are set to control the level of privacy during training.

The `make\_private` function introduces differential privacy into the model, optimizer and data loader. This function uses a privacy engine (`privacy\_engine`) to ensure that the training process complies with privacy constraints.

The training function (`train`) iterates through the specified number of epochs, updates the model parameters using backpropagation and monitors the training progress, including the average loss, accuracy and privacy budget (ε). The training time is also tracked.

A test function (`test`) evaluates the model on a separate test data set and provides insights into the generalization performance.

The entire training pipeline is orchestrated by the "pipeline" function, which initializes the model and optimizer, corrects the model with ModuleValidator and trains two models (one with and one without DP).

The CIFAR-10 dataset is used for training and testing, with data loading configured via PyTorch's DataLoader. The ResNet18 model architecture is used for image classification with 10 output classes, and the RMSprop optimizer with a given learning rate (LR) is used for updating the model parameters during training.

Table 1: Accuracies, Privacy Budgets and Training Times for Task 1a) and b)

**Task 1 b)**

For the datasets ‘PneumoniaMNIST’ and ‘OrganCMNIST’ I had to change the number of input channels of the first convolutional layer to one.

The introduction of noise into the training process in DP, designed to safeguard individual data points, heightens the complexity of optimization. This is noticeable in the training time, which is significantly longer with DP than without.

After 20 epochs the privacy budget is 49.99 in all four cases. It seems, that in all four cases the same amount of privacy is provided.

Furthermore, the training accuracy without DP is higher in all four cases than with DP. The noise introduced for privacy protection in DP may hinder the model's ability to fit the training data perfectly, leading to lower training accuracy. DP's focus on protecting individual data points can potentially compromise the model's ability to memorize the training set.

However, the training with DP apparently generalizes more than the training without DP, since in 3 of four datasets (DermaMNIST, PneumoniaMNIST, BloodMNIST) the accuracy of the test set with DP is higher than that without DP. That could be since DP encourages models to learn more robust features that are less likely to overfit to the training data. This suggests that the models trained with DP are more effective at making predictions on new, unseen examples.

**Task 1 c)**

The deliverables for this section include figures that illustrate how various parameters affect the model's utility and potentially its privacy level, along with a supporting table of results and a written report that elucidates the insights gained. We selected the PneumoniaMNIST dataset, as it exhibited the shortest training time in Task 1b. Our methodology employed a Resnet18 architecture with a batch size of 256, a learning rate of 1e-03, RMSProp as the optimizer, a delta of 1e-05, and a gradient clipping of 1.0. We also considered three different levels of privacy, defined by epsilon values of 0.5, 4.0, and 9.0, representing high, medium, and low privacy for medical data, respectively. In total, we trained 10 different parameter combinations for each privacy level. Memory constraints led to the failure of training models such as VGG16, AlexNet, and any Resnet variants deeper than Resnet34.

Figure 1 displays the variance in test accuracy and achieved epsilon after 10 epochs of training for each level. All models confirmed that the target epsilon value was met after the training period. Notably, there was a greater variance in test accuracy for the level 1 and 2 models, suggesting that a lower epsilon, indicating stricter privacy constraints, makes parameter selection more critical for model performance.

Contrary to our initial hypothesis that a small epsilon value would lead to reduced test accuracy, our experiments indicated that the chosen hyperparameters and model architectures greatly influence model utility. On average, level 2 models performed comparably or even better than level 3 models in terms of test accuracy.

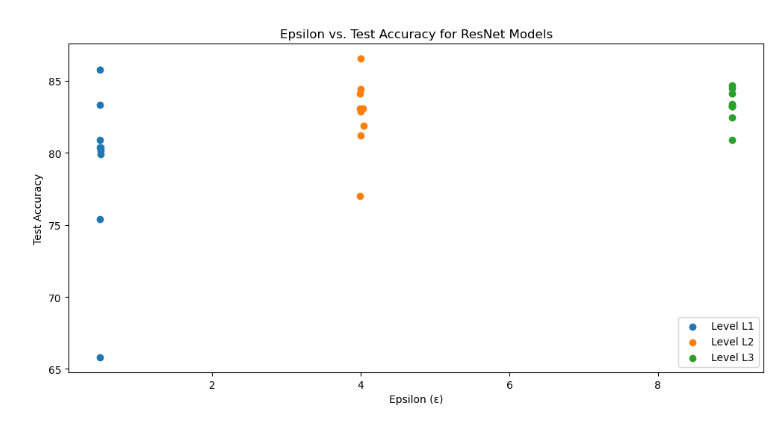
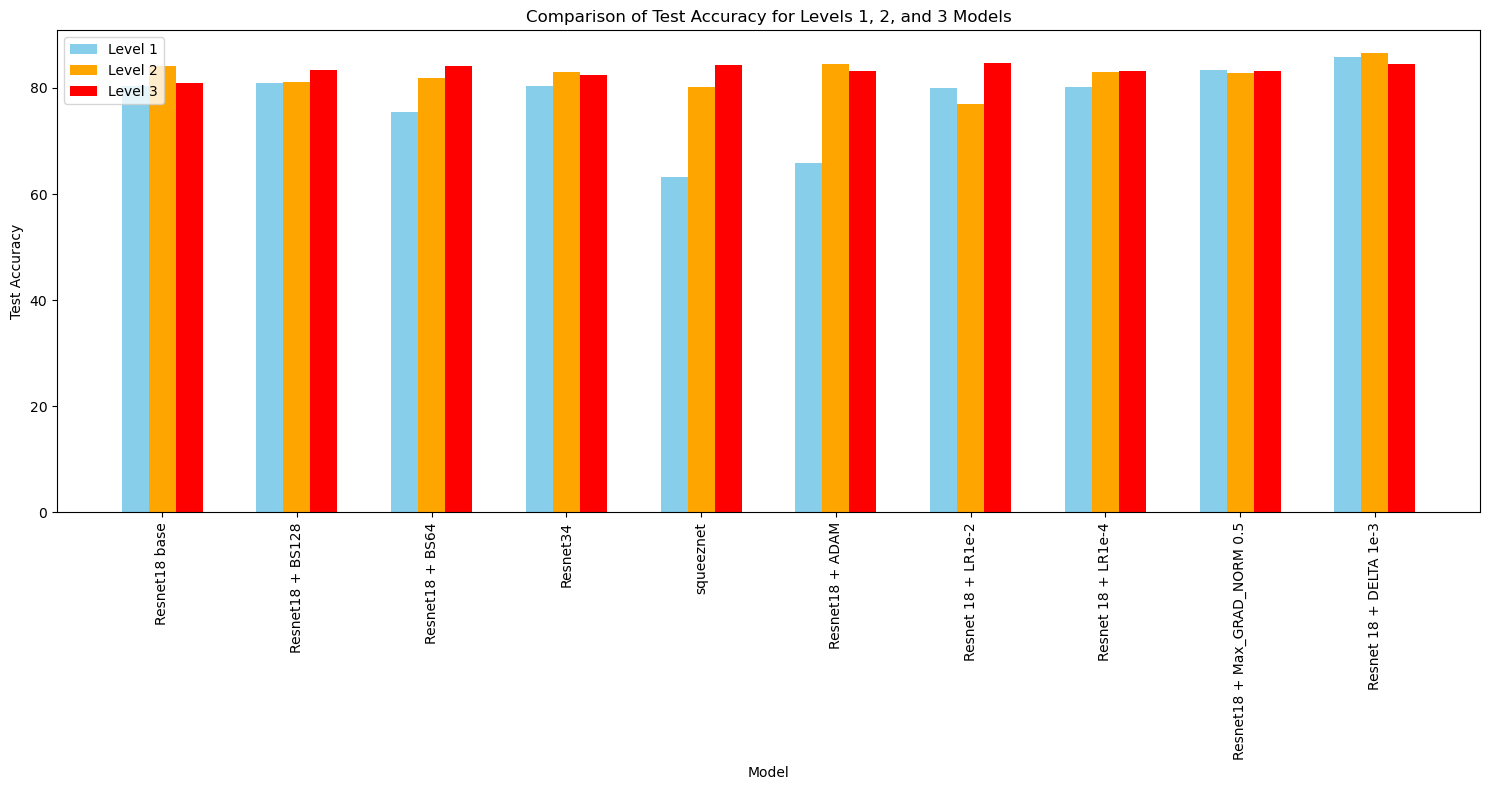
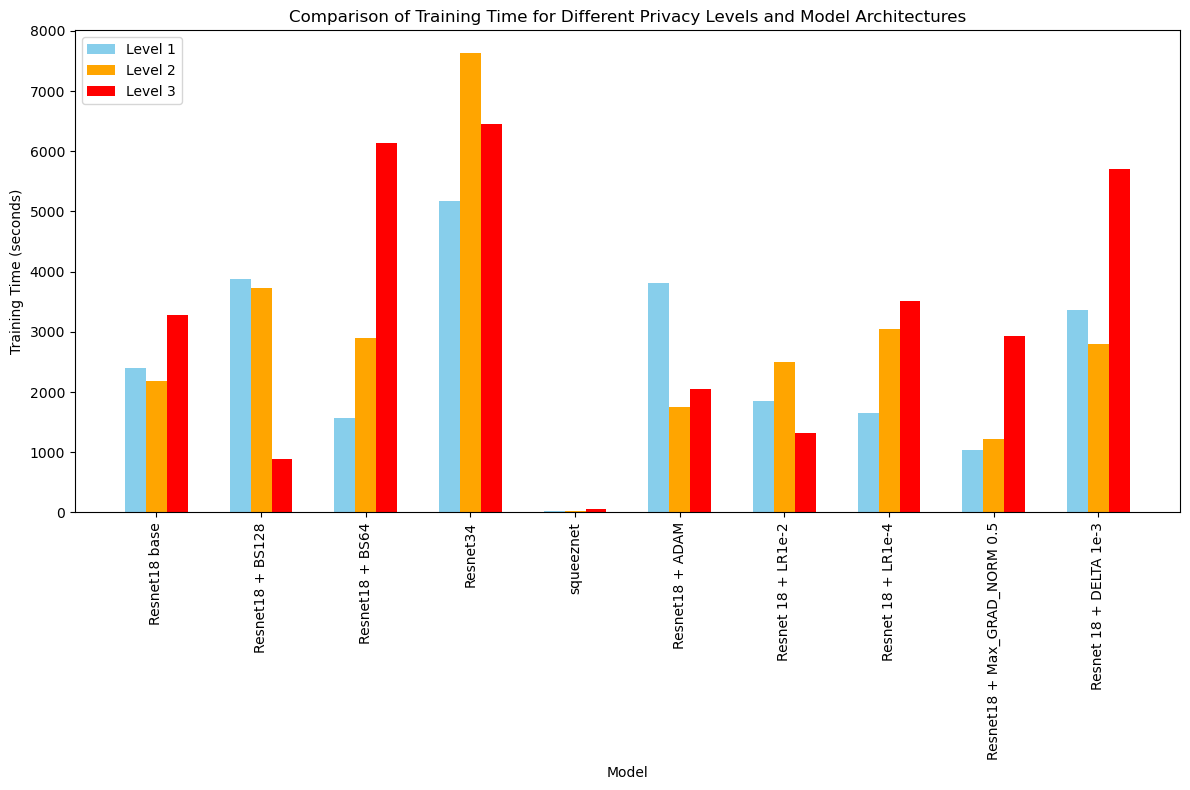
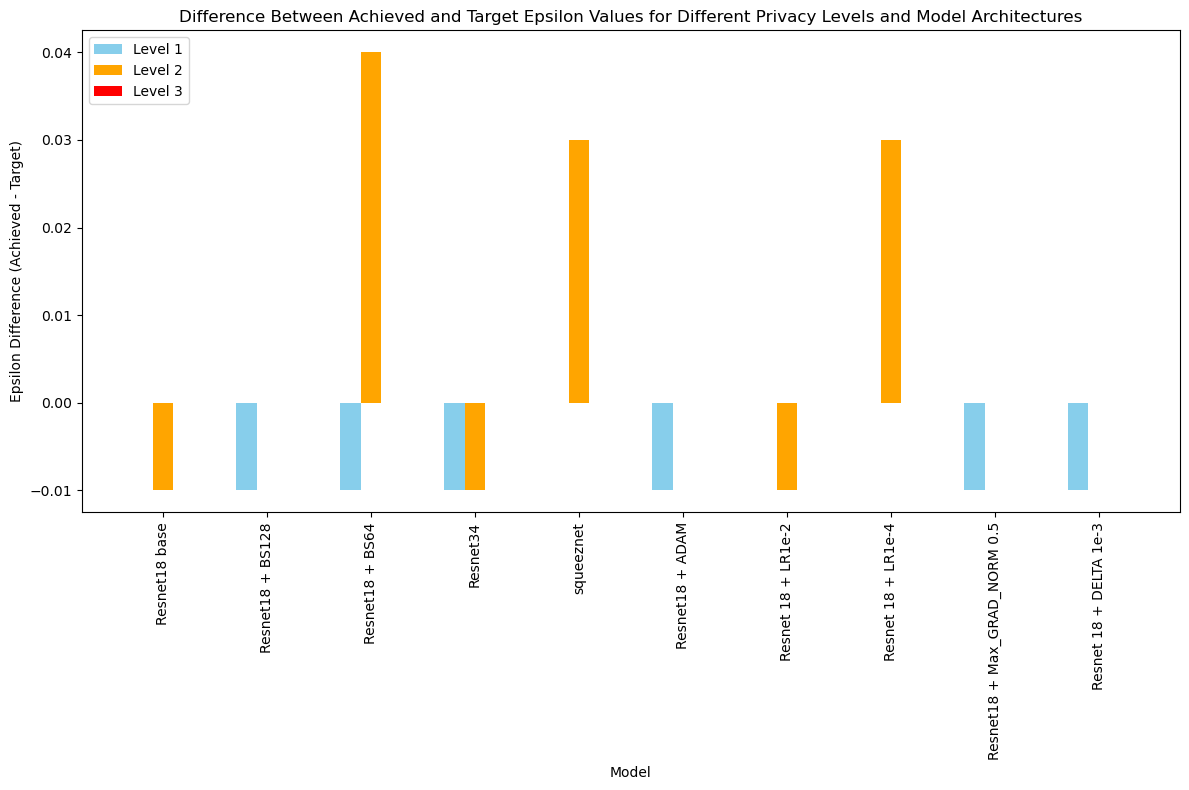
Additionally, the highest test accuracy was observed within the level 2 models. When this is viewed alongside the data in Table 3, which indicates that level 3 models have higher training accuracy compared to levels 2 and 1, it suggests that the noise added for privacy also aids in model generalization. However, for level 1 models, the more pronounced noise could impede model training. Thus, the first takeaway is that model utility can be enhanced with the addition of privacy, provided that extensive fine-tuning is performed to identify the optimal model parameters, and the added privacy is not excessive.

Figure 2 further compares the test accuracy across all trained models, reinforcing the observation that level 2 models often perform similarly or better than level 3 models. A sharper decrease in test accuracy is noted for level 1 models. When comparing levels 1 and 3 directly, we observe a greater impact of added privacy on overall model performance. Hence, the second takeaway is that a high privacy budget necessitates intensive fine-tuning, while models with a lower privacy budget tend to perform better.

Figure 3 plots the training times for various models across different privacy levels. Contrary to our expectations, higher epsilon values, less added noise, did not correspond to shorter training times, which might be attributed to the specific implementation of privacy mechanisms or, more likely, the underlying hardware. The subsequent training of level 3 models, after levels 1 and 2, could have taxed the GPU memory and performance. Nevertheless, Squeezenet consistently showed the best performance in terms of training time without significant loss in test accuracy, while still achieving the target epsilon value, as evidenced in Table 2.

Figure 4 examines how the achieved epsilon values deviate from the target. Level 3 consistently met the target epsilon, whereas level 2 displayed the most deviations, albeit minor. This leads to the third takeaway: hyperparameter tuning does not significantly impact the privacy budget when the method make\_private\_with\_epsilon() is employed. 





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resnet18 + ADAM | Acc Test | 65.81 | 84.43 | 83.26 |
| Acc Train | 76.74 | 94.53 | 96.10 |
| ε | 0.49 | 4.00 | 9.00 |
| Time | 3808.09 s | 1748.01 s | 2057.86 s |
| Resnet 18 + LR 1e-2 | Acc Test | 79.91 | 76.99 | 84.71 |
| Acc Train | 92.08 | 93.49 | 95.17 |
| ε | 0.5 | 3.99 | 9.00 |
| Time | 1858.59 s | 2493.27 s | 1322.70 s |
| Resnet 18 + LR 1e-4 | Acc Test | 80.17 | 83.07 | 83.24 |
| Acc Train | 92.60 | 95.57 | 96.54 |
| ε | 0.5 | 4.03 | 9.00 |
| Time | 1652.56 | 3042.72 s | 3516.22 s |
| Resnet18 + Max\_GRAD\_NORM 0.5 | Acc Test | 83.35 | 82.86 | 83.46 |
| Acc Train | 93.48 | 95.41 | 95.75 |
| ε | 0.49 | 4.00 | 9.00 |
| Time | 1037.10 s | 1223.82 s | 2923.50 |
| Resnet 18 + DELTA 1e-3 | Acc Test | 85.80 | 86.54 | 84.49 |
| Acc Train | 93.63 | 95.69 | 95.79 |
| ε | 0.49 | 4.00 | 9.00 |
| Time | 3362.42s | 2801.20 s | 5708.05 s |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Level 1 | Level 2 | Level 3 |
| **Resnet18 base model** | Acc Test | 80.36 | 84.11 | 80.88 |
| Acc Train | 92.58 | 95.91 | 96.13 |
| ε | 0.5 | 3.99 | 9.00 |
| Time | 2392.24 s | 2188.93 s | 3279.22 s |
| **Resnet 18 + batch size 128** | Acc Test | 80.89 | 81.21 | 83.37 |
| Acc Train | 91.78 | 94.78 | 96.17 |
| ε | 0.49 | 4.00 | 9.00 |
| Time | 3882.28 s | 3722.81 s | 890.55s |
| **Resnet 18 + batch size 64** | Acc Test | 75.41 | 81.88 | 84.11 |
| Acc Train | 91.81 | 95.17 | 95.73 |
| ε | 0.49 | 4.04 | 9.00 |
| Time | 1560.89 s | 2897.25 s | 6138.37 |
| **Resnet34** | Acc Test | 80.38 | 83.07 | 82.48 |
| Acc Train | 90.19 | 94.96 | 95.54 |
| ε | 0.49 | 3.99 | 9.00 |
| Time | 5176.87s | 7629.04 s | 6447.23 s |
| **squeeznet** | Acc Test | 63.17 | 80.13 | 84.38 |
| Acc Train | 73.90 | 86.68 | 85.22 |
| ε | 0.5 | 4.03 | 9.84 |
| Time | 22.28 s | 22.73 s | 57.54 s |

**Learning Rate:**

The learning rate did not change the test accuracy within the level 1-3 models, indicating that it is not much influenced by different privacy level. However, across the different learning rate a substantial change is noted, making it a crucial parameter in differential privacy tuning.

**Batch Size:**

Batch size specifies the number of samples processed simultaneously by the network. Generally, larger batch sizes result in fewer noise gradient updates and potentially better generalization. Figure 2 indicates that smaller batch sizes correspond to lower test accuracy for levels 1 and 2 models, suggesting that the privacy budget more significantly affects model utility when training with smaller batches

**Model Architecture / Complexity:**

We analyzed three different architectures: Resnet18, Resnet34, and Squeezenet. Squeezenet's utility was highly sensitive to smaller epsilon values, which introduce more noise. The Resnet34 model, however, showed similar test accuracy across all privacy levels compared to the base model, despite longer training times. This suggests that model utility is directly correlated with privacy level for less complex models, and not directly correlated with privacy level for more complex architectures.

**Gradient Clipping:**

We assessed the impact of gradient clipping, hypothesizing it would directly influence privacy since it limits the influence of individual data points. Surprisingly, all models yielded similar test accuracy, with the level 1 model slightly outperforming others. This suggests that lower gradient clipping values can improve model utility by preventing overfitting and balancing performance across privacy levels.

**Target Delta:**

We experimented with a lower target delta, which affects privacy as it is integral to the privacy budget. It quantifies the likelihood of the privacy guarantee failing. Our findings indicate consistent test accuracy and average training time compared to the base model, demonstrating that a lower delta value can enhance privacy without compromising utility.

**Optimizer Strategy:**

We evaluated the ADAM optimizer against the RMSProp used in the base model. The level 1 model trained with ADAM exhibited poorer test accuracy, and both level 1 and level 3 models experienced longer training times. Interestingly, the level 3 model attained higher test accuracy with ADAM than the base model, implying that the choice of optimizer influences model utility based on the selected privacy level.

The two main takeaways regarding the impact of parameter tuning on model utility and privacy level are:

Model Utility Enhancement: Proper fine-tuning can enhance model utility even with added privacy constraints, provided the privacy level is not excessively high.

Fine-Tuning Necessity: High privacy models require more intensive fine-tuning to maintain performance, while those with lower privacy budgets are less sensitive and generally perform better.

**Task 2 a)**

We ran a simulation for federated learning with 3 clients, we used a simple CNN implementation consisting of 5 convolution layers and one full connected layer.

We created a load dataset function to load the ChestMnist dataset in a way to create a list of *trainloaders* and *valloaders* with *length = number of clients*, the function also returns the testloader which is used at the server-side to evaluate the training

The metrics were aggregated using a simple weighted average function.

The evaluation was done at the client side with a small portion of the validation dataset (referred to as distributed in the table) and on the server-side with the test dataset (referred to as centralized in the table)

We tried this setup and documented the results while varying the batch size and learning rate as follows

* Batch size = 32 and learning rate = 1e-3
* Batch size = 8 and learning rate = 1e-3
* Batch size = 64 and learning rate = 1e-3
* Batch size = 32 and learning rate = 1e-1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Round | Batch\_size=32  L\_R = 1e-3 | Batch\_size=8  L\_R = 1e-3 | Batch\_size=64  L\_R = 1e-3 | Batch\_size=32  L\_R = 1e-1 |
| Loss ↓ Distributed | 1 | 0.0216 | 0.0866 | **0.0107** | 0.0223 |
| 2 | 0.0195 | 0.0789 | **0.0097** | 0.0216 |
| 3 | 0.0195 | 0.0777 | **0.0097** | 0.0216 |
| Loss ↓ Centralized | 1 | 0.0216 | 0.0865 | **0.0106** | 0.0225 |
| 2 | 0.0196 | 0.0792 | **0.0097** | 0.0216 |
| 3 | 0.0194 | 0.0777 | **0.0096** | 0.0216 |
| Accuracy ↑ Distributed | 1 | 50.414 | 50.271 | **65.282** | 54.184 |
| 2 | 66.076 | 65.799 | **66.734** | 54.184 |
| 3 | 66.432 | 66.45 | **66.717** | 54.184 |
| Accuracy ↑ Centralized | 1 | 51.05 | 50.31 | **64.396** | 53.171 |
| 2 | 65.354 | 65.288 | **66.598** | 53.171 |
| 3 | 66.719 | 66.424 | **66.896** | 53.171 |

Results, as expected, are better with a larger batch size; the model is capable of converging faster, Comparing a batch size of 64 with batch size of 8, the accuracy with the first one was capable of jumping to ~65% from the first round, while with batch size 8 accuracy needed one more round to achieve similar results.

The same is true for the learning rate with a high learning rate (last column); the model didn’t converge at all, and as shown, loss and accuracy do not change round after the other.

**Task 2 b)**

In our federated learning study, we experimented with small image changes (single pixel perturbations) and intentionally incorrect labels (label poisoning) on one client's data, referred to as Client 1.

We trained the model using batches = 32 and a learning rate of 0.001, without altering the model itself throughout the process.

**Labels Poisoning**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Round | Labels Poisoning | Labels Poisoning | Labels Poisoning |
| Rate = 0 | Rate = 0.5 | Rate = 1 |
|  |  |  |
| Loss ¯ Distributed | 1 | 0.022 | 0.022 | 0.0213 |
| 2 | 0.019 | 0.0197 | 0.0197 |
| 3 | 0.019 | 0.0194 | 0.0193 |
| Loss ¯ Centralized | 1 | 0.0215 | 0.0214 | 0.0213 |
| 2 | 0.0195 | 0.0199 | 0.0198 |
| 3 | 0.0193 | 0.0199 | 0.0194 |
| Accuracy ­ ↑ Distributed | 1 | 58.222 | 54.327 | 55.060 |
| 2 | 67.064 | 65.692 | 65.638 |
| 3 | 66.975 | 66.645 | 66.628 |
| Accuracy ­ ↑ Centralized | 1 | 57.803 | 53.372 | 53.884 |
| 2 | 66.460 | 64.788 | 64.891 |
| 3 | 66.656 | 65.760 | 66.232 |

In both distributed and centralized scenarios, as the label poisoning rate increases from 0 to 0.5, there is generally an increase in loss and a decrease in accuracy, suggesting that the poisoning negatively impacts the our model's performance.

However, when the poisoning rate is at its maximum 1, the impact is not as straightforward. The loss is sometimes lower than at a poisoning rate of 0.5, and the accuracy is higher in some cases. This could be due to overfitting on the poisoned labels or other factors specific to the training algorithm or data.

Across rounds, you can see the training process is likely converging as the loss decreases and accuracy increases in a typical learning scenario.

**Single Pixel Perturbations (SPP)**

Our images are 28x28 in size. We changed the pixels in x=10 and y=10 coordinates.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Round | SPP | SPP | SPP |
| Rate = 0 | Rate = 0.5 | Rate = 1 |
|  | X= 10  Y =10 | X= 10  Y =10 |
| Loss ¯ Distributed | 1 | 0.022 | 0.0216 | 0.0214 |
| 2 | 0.019 | 0.0197 | 0.0196 |
| 3 | 0.019 | 0.0194 | 0.0193 |
| Loss ¯ Centralized | 1 | 0.0215 | 0.0214 | 0.0214 |
| 2 | 0.0195 | 0.0199 | 0.0196 |
| 3 | 0.0193 | 0.0195 | 0.0194 |
| Accuracy ­ Distributed | 1 | 58.222 | 54.327 | 56.422 |
| 2 | 67.064 | 65.692 | 66.209 |
| 3 | 66.975 | 66.645 | 66.816 |
| Accuracy ­ Centralized | 1 | 57.803 | 53.372 | 55.984 |
| 2 | 66.460 | 64.788 | 65.577 |
| 3 | 66.656 | 65.760 | 66.433 |

We can infer that the model is somewhat sensitive to single pixel perturbations, but it still retains a degree of learning capability even with the highest perturbation rate.

It's also notable that the distributed learning system consistently shows a slight advantage in accuracy over the centralized system under perturbed conditions.