EXPLORING THE HYPERPARAMETERS OF DIFFERENTIALLY-PRIVATE SGD

SOPHIE THOREL [STHOREL@SEAS], ANGELINA HEYLER [AHEYLER@SEAS], RACHAEL TAMAKLOE [RACHEALT@SEAS],

ABSTRACT. Differentially private SGD (DP-SGD) provides privacy guarantees for data, yet its performance is considerably worse than regular SGD. In this work, we seek to explore the effect of hyperparameters on performance in DP-SGD, since hyperparameter tuning may be particularly important for learning under DP-SGD. In particular, we run experiments in which we change the batch size and learning rates during training and evaluate how the batch size, learning rate, and amount of the privacy budget used at each epoch can affect learning. We find that our fastest learning rate (0.001) works best; model performance does not exhibit clear trends with respect to batch size; in particular, we note that for slower learning rates there is a decrease in model performance with an increase in batch size, whereas for fast performing learning rates, there's very little variation in performance as batch size varies. Lastly we note that the model tends to learn more in epochs with more privacy budget spent.

1. INTRODUCTION

1.1. **Motivation.** As machine learning applications proliferate, one important question is how to ensure that individuals' data remain private. Opt-in choices for data collection and abiding by data access rights are important in preserving privacy, but even allowed data uses can pose risks to user privacy. In particular, repeated observations of model predictions can allow adversaries to learn facts about private data—these are called "inference attacks." Differential privacy (DP) has grown popular for its balance between providing statistical guarantees on privacy versus still providing useful data for statistical analyses. At a high level, DP adds noise to model outputs to ensure that an outsider has difficulty discerning whether a particular individual's data was used in training the model. Prior research has implemented a differentially-private version of SGD (DP-SGD), but one major drawback is that its performance is considerably worse than non-private SGD.

Because it is more difficult to train accurate models with DP-SGD, hyperparameter tuning is especially important. Our initial motivation was to understand the tradeoff between batch size and accuracy, as larger batch sizes lead to more accurate gradient estimates and fewer model queries/larger privacy budget per weight update. We also investigated the effect of learning rates on DP-SGD, as we may need smaller learning rates for convergence to compensate for noisy gradients. Finally, we explored how the amount of privacy budget spent per epoch affects the amount learned in that epoch.

1.2. **Contributions.** We contribute an analysis of how DP-SGD performs with respect to three variables: batch size, learning rate, and the privacy budget used per epoch. While the tradeoff between accuracy and the privacy budget is well-known [4], the effect of these hyperparameters are less well-studied. We analyze three experiments to understand how batch size, learning rate, and the budget used per epoch affect training on CIFAR-10. In summary, we find that:

(1) For slower learning rates, an increase in batch size results in a decrease in model performance, and for faster learning rates batch size does not have much impact on model performance.
(2) Faster learning rates demonstrate better performance over slower learning rates.
(3) As the privacy budget used per epoch increases, so does the amount learned during that epoch, as measured by training/test accuracy and losses.

2. BACKGROUND

2.1. **Definitions.** A randomized algorithm $\mathcal{A}$ is said to be ϵ, δ-differentially private if for all D and D',

$$P(\mathcal{A}(D) \in S) \leq e^{\epsilon} \cdot P(\mathcal{A}(D') \in S) + \delta \tag{1}$$

where D and D' are neighboring datasets (i.e., datasets that differ by one example), ϵ and δ are positive real numbers, and S is the training dataset [3]. Setting $\epsilon = 0$ would guarantee perfect privacy, since D and D' are indistinguishable, while infinitely large ϵ would give no privacy. Using ϵ as a knob that controls the amount of privacy guaranteed, differential privacy adds Gaussian or Laplacian noise to model outputs, proportional to the function's sensitivity, to satisfy the definition of differential privacy.

Algorithm 1 DP-SGD [1]

```
Inputs: training dataset $\{x_1, ..., x_N\}$, loss function $\mathcal{L}$, learning rate $\eta_t$, noise scale $\sigma$, minibatch size $L$, gradient clipping norm bound $C$.
1: procedure DP-SGD($\{x_1, ..., x_N\}, \mathcal{L}, \eta_t, \sigma, L, C$)
2:     Initialize $w_0$ randomly
3:     for each epoch t = 1, 2, ..., T do
4:         Take a random sample $L_t$ with probability $L/N$
5:         for each $x_i$ sampled do
6:             $g_t(x_i) \leftarrow \nabla_{w_t}\mathcal{L}(w_t, x_i)$        ▷ Compute per-sample gradient with respect to weights
7:             $\bar{g}_t(x_i) \leftarrow g_t(x_i)/\max(1, \frac{||g_t(x_i)||_2}{C})$        ▷ Clip per-sample gradient
8:         end for
9:         $\tilde{g_t(x_i)} \leftarrow \sum_i g_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I)$        ▷ Add noise
10:        $w_{t+1} \leftarrow w_t - \eta_t \tilde{g}_t$        ▷ Gradient descent
11:        $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$        ▷ Weighted average of local updates
12:    end for
13: end procedure
```

Note that the gradient clipping intuitively ensures that the model does not learn more information than the set quantity from any given training sample, no matter how different it is from the rest.

2.2. **Opacus Library.** Opacus is an open-source PyTorch library developed by Meta AI, optimized for simplicity, flexibility and speed, supporting a wide range of neural network architectures. DP-SGD requires clipping per-sample gradients, making it much more computationally intensive than non-private training. Opacus optimizes these calculations.

In TensorFlow or PyTorch, only batch-averaged gradients are typically available to users. It is possible to get per-sample gradients through "microbatching," or using a batch size of 1, but this is very slow since this approach does not take advantage of GPUs' and TPUs' batched, parallel computation optimizations. Opacus implements a vectorized computation instead of micro-batching to increase efficiency (10x for small MNIST, 50x for Transformers) [6].

Opacus allows us to keep track of the privacy budget spent so far (of the total budget ϵ), accessed via the PrivacyEngine class. This class takes in the model, optimizer and data loader, along with privacy parameters (noise multiplier and maximum norm of the gradients). It outputs the differentially-private analogues of these, namely: 1) the model, wrapped with GradSampleModule - this adds the ability for calculating per-sample gradients; 2) The optimizer wrapped with additional code for gradient clipping and injecting noise; 3) the data loader transformed into one using Poisson sampling, as required by DP-SGD.

3. Related Work

Song, Chaudhuri, and Sarwate first propose differentially private SGD updates [5]. While it is not the focus of their work, they perform initial experiments investigating how batch size affects performance. They find that increasing the batch size improves performance up to a limit: beyond a step size $\frac{1}{\sqrt{t}}$, larger batch sizes degrade performance in their work. That said, in their implementation, increasing the batch size decreases the number of iterations. They further find that larger batches do not degrade performance as much when the learning rate is larger. They conclude that choosing a learning rate and step size is "often a matter of art, and differentially private noise complicates this choice."

DeepMind [2] more recently published an investigation of how to close the accuracy gap of DP-SGD via hyperparameter tuning and other tricks. While other work has postulated that DP-SGD is inherently poor for large models since the norm of noise injected is proportional to the dimensionality of the model, the authors find that overparameterized models can be trained to 81% accuracy on CIFAR-10, which is currently SOTA. With fine-tuning, they find that accuracy increases with batch size and ReNet model size.

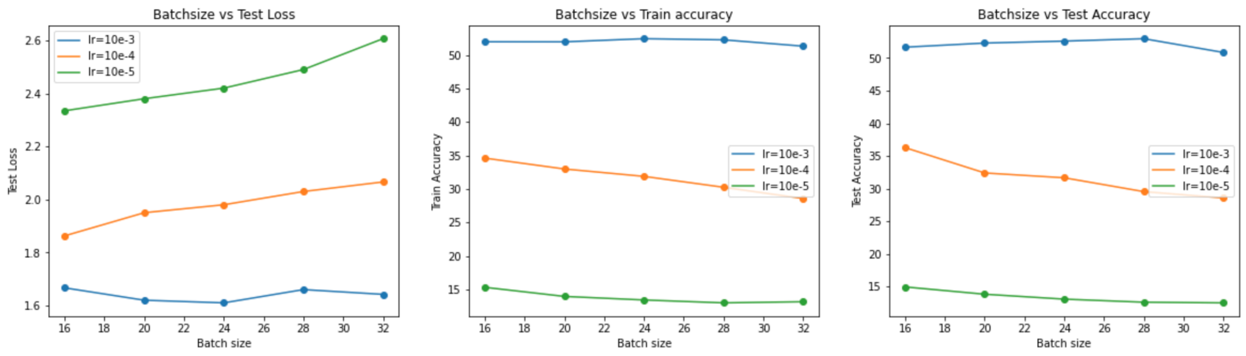
Our work investigates the original batch size question posed in [5], but we also extend our investigation to how the learning rate and privacy budget spent affects learning. Our work is similar to [2], but without extensive fine-tuning. This may be relevant to individuals using DP-SGD at places other than research institutions, for whom the fine-tuning process may be very computationally and time intensive. We find that these trends are not as clean-cut without this extensive fine-tuning process, which may make use and tuning of DP-SGD more challenging in applied settings.

4. Approach

In order to understand the effect of batch size on differentially private SGD, we experiment on different batch sizes with a fixed privacy budget. Specifically, we use a Wide Resnet 50-2 to test its performance on the CIFAR-10 dataset. Note that Opacus runs a model "fixer" that changes non-DP layers, such as changing BatchNorm to LayerNorms. In our experiment set-up, we fixed $\epsilon = 10$ and $\delta = 10^{-5}$. These values were picked with reason to be within the typical range of acceptable differentially private privacy budgets for the cifar-10 dataset. Most privacy budgets lie in the range from 4-8. In our training, we found that the accuracy was still low with $\epsilon = 8$, so we increased it to 10 for the purposes of our experiments. We explored the performance of our ResNet on 5 different batch sizes and three different learning rates during training. The batch sizes were 16, 20, 24, 28, and 32 and the learning rates used were 1e-3, 1e-4, and 1e-5. For each of the three different learning rates, all 5 batch sizes were trained and tested. So in total, 15 different models were trained for 30 epochs each. This was done to further explore how other hyper parameters such as learning rate for example could be more suitable to use in combination with particular batch sizes. In answering our question regarding how batch size affects performance of differentially private SGD models, this approach allows us to explore this.

5. Experimental Results

5.1. **Learning Rate and Batch Size Experiments.** Our first experiment investigates how model performance changes as a function of learning rate and batch size. We trained the ResNet 50-2 for 30 epochs while varying batch sizes from 16,20, 24, 28, and 32 and learning rates from 10^{-3}, 10^{-4}, and 10^{-5}.

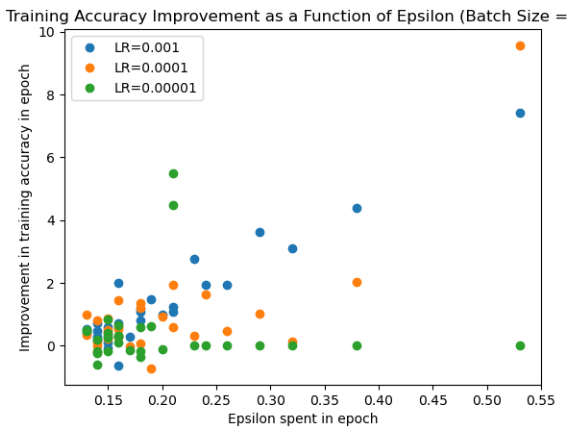


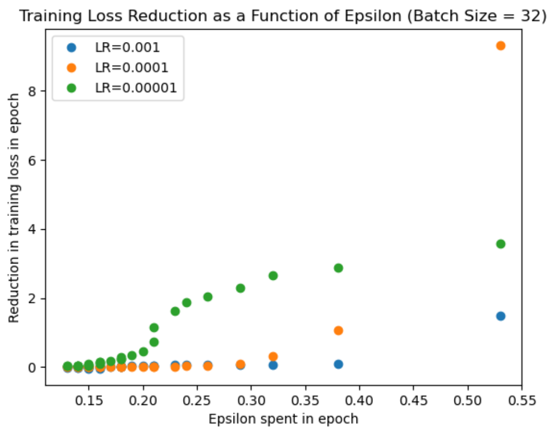
Learning Rate: First, we find that learning rates of $\eta = 10^{-3}$ grant us the lowest test loss, highest training accuracy, and highest test accuracy when compared to batch sizes with different learning rates. Overall, we find that models trained with faster learning rates provide better performance in our settings, as we see performance dramatically increase with faster learning rates. We see the largest difference with a batch size of 28, where our test accuracy with $\eta = 10^{-3}$ is 52% but just 13% with $\eta = 10^{-5}$.

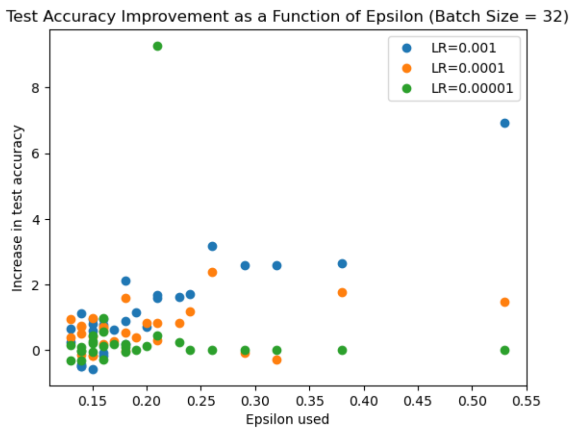
Batch Size: We also analyze how batch sizes affect performance by comparing the training/test accuracy and test loss for fixed learning rates. For $\eta = 10^{-4}$ and $\eta = 10^{-5}$, which generally exhibited more poor performance, we see a trend of a decrease in performance with the increase in batch size. This trend is persistent across all three metrics we used to measure model performance. This shows that for particular learning rates, the batch size that is used plays a moderate role in the performance of the model.

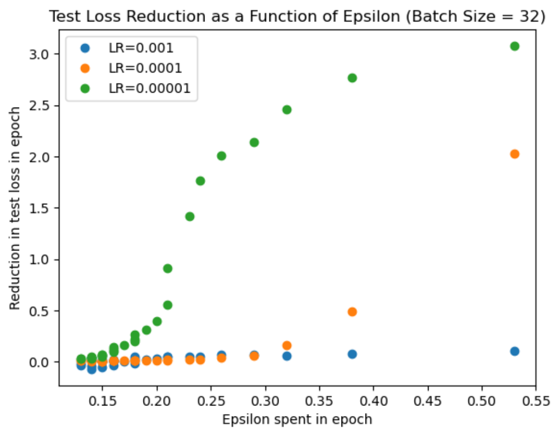
However, in the case of $\eta = 10^{-3}$, this negative relationship between batch size and model performance that we saw in the other two learning rates is no longer present. When examining the accuracy for both training and test data sets, there is less variation in performance as batch size varies for $\eta = 10^{-3}$.

5.2. **Privacy Budget Per Epoch Experiments.** We next investigate how the privacy budget spent during an epoch of training affects the amount of learning that occurs in that epoch. We ran experiments with $\epsilon = 10$ and $\delta = 10^{-5}$, 30 epochs, a Wide ResNet-50-2, a batch size of 32, and learning rates of $10^{-3}, 10^{-4}, 10^{-5}$. We calculate the increase in training/test accuracy from the previous to current epoch, as well as the reduction in training/test loss as a measure of learning. As expected, learning appears to increase with the amount of the privacy budget spent during that epoch, with a clearer relationship with accuracy than cross entropy loss values.









6. Discussion

6.1. **Importance of Hyperparameters.** Our results demonstrate the difficulty of implementing DP-SGD for training a deep network to reasonable accuracy and the importance of hyperparameter tuning. Our most poorly performing model reached an acuracy of just 12%, which is barely better than random guessing on CIFAR-10, whereas our best model reached an accuracy of 52%. Our accuracies varied greatly according to the learning rate and batch size selected. Learning rate appeared to be most important, as our test accuracy reached 52% with $\eta = 10^{-3}$ but only 13% with $\eta = 10^{-5}$ with all other hyperparameters unchanged. Batch size tuning can also provide modest improvements—we found a 3% (percentage point) increase among experiments with $\eta = 10^{-3}$, a 8% increase among experiments with $\eta = 10^{-4}$, and a 3% increase among experiments with $\eta = 10^{-5}$.

6.2. **Learning Rate Findings.** We find that our models reach better performance with faster learning rates. This is different from what the Opacus documentation suggests, which is that because differentially private gradients have more noise, we may need a smaller learning rate for the model performance to converge. In practice, we find that this is not the case in our set of hyperparameters, though given more time, we could have liked to experiment with even faster learning rates to test where this threshold lies.

6.3. **Batch Size Findings.** We find that the batch size yields only modest improvements on test performance when all other hyperparameters are held equal. Furthermore, we find mixed results with respect to how performance varies as a function of batch size. With slower learning rates, the performance appears to decrease with larger batch sizes, while with our fastest learning rate, there is not much of a trend. This is a much less clear trend than [2] found, as the authors found that accuracy increases with batch size. This may be expected given that gradient estimates are less noisy and fewer model queries (average gradients) can allow for a greater privacy budget per query. However, our results suggest that this trend may only be observable once fine-tuning has been performed, or that batch size has a smaller impact for well-performing or faster learning rates. Of course this statement is made with caution, acknowledging that the sample space of batch sizes that we explored in our experiment was quite restricted. A more conclusive judgement could be made if performance of larger batch sizes were also explored.

6.4. **Privacy Budget Spent per Epoch Findings.** Finally, we observe that the greatest learning occurs with greater spending on the privacy budget, as measured by accuracies and cross entropy losses. This makes sense as more of the privacy budget being spent means that less noise is added to the batch gradients, allowing for more accurate and useful gradient updates. It may be interesting to investigate how different schedules of budget allocation compare in performance in the future.

6.5. **Challenges.** One of the largest blocks we encountered was the memory size required for running DP-SGD. Although we originally planned to run batch sizes from 16 to 512, in practice, we were not able to train models with batch sizes larger than 64. As a result, we trained models with sizes from 12-36 with smaller increments instead. The original paper introducing Opacus shows why this is the case mathematically [6]. Let L denote the number of trainable parameters in a given model (each of size 1); C denote the size of the features, label and model output for a single data point; and M denote the total memory usage for one forward and backward pass on a batch of size b. Ignoring the intermediate computations and constant additive overhead, we have:

$$M_{nonDP} = bC + 2LM_{DP} = bC + (1-b)L \tag{2}$$

Without DP, we expect the gradient to occupy memory of size L, but with DP, this increases by a factor of b. For $b \gg 1$, the memory overhead can be approximated as:

$$\frac{M_{DP}}{M_{non-DP}} = \begin{cases} \frac{bC+(1-b)L}{bC} \approx 1 + \frac{L}{C} \text{ ,if } \frac{L}{C} \ll b \\ \frac{2+b}{3} \approx \frac{b}{3} \text{ , if } \frac{L}{C} \approx b \\ \frac{1+b}{2} \approx \frac{b}{2} \text{ , if } \frac{L}{C} \gg b \end{cases} \tag{3}$$

We thus see that peak allocated memory overhead significantly increases with the batch sizes, as we encountered.

6.6. **Future Directions.** There are a handful of future directions of our work that could be interesting. Firstly, given the tradeoff between the privacy budget spent per epoch and the amount learned in that epoch, one direction is to explore how different privacy accountant schedules and allocations of the budget over training affect the final performance of the model. We could also have tried changing the learning rate as a function of the number of epochs. Under Abadi's implementation of DP-SGD finds small improvement in starting with a relatively larger learning rate, then linearly decaying to a smaller value after a few epochs, then keeping it constant until the end of training [1]. Putting these experiments together, it might also be interesting to explore if there are relationships between learning rate scheduling and the privacy accountant scheduling—for example, a faster learning rate could be used when more of the privacy budget is used to take advantage of a more accurate gradient estimate at that step. That said, this is contingent on how the variance of the noise is calculated as a function of the learning rate.

A further direction we could have taken this project is to have tried different frameworks implementing DP other than Opacus. For example, Objax, a framework for JAX, implements DP-SGD slightly more efficiently for smaller batch sizes and implements a different gradient clipping function [6]. DeepMind also has publicly available code on GitHub from their work in [2]—this would allow us to investigate how fine-tuning can boost performance as well.

7. Conclusion

We find that training models via DP-SGD to comparable accuracy as non-private training is difficult, though hyperparameter tuning can help considerably and may be particularly important in these settings. By exploring the relationship of learning rate, batch size, and privacy budget spent on learning, we contribute an exploratory work of how hyperparameters affect DP-SGD without extensive fine-tuning. This code is publicly available at `https://github.com/aheyler/Differential-Privacy-Batch-Exp`. We find that learning rate brought the biggest performance boosts, though optimizing batch size alos led to modest improvements. We also find that faster learning rates trained better performing models, and that the learning that occurs at each epoch is positively related to the privacy budget spent. Looking forward, further optimizing DP-SGD may include investigating different privacy budget allocations during training, how this works with learning rate scheduling, and fine-tuning of hyperparameters.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2016.

[2] Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale, 2022.

[3] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014.

[4] Xin Qian and Diego Klabjan. The impact of the mini-batch size on the variance of gradients in stochastic gradient descent, 2020.

[5] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248, 2013.

[6] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Gosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. Opacus: User-friendly differential privacy library in pytorch. *CoRR*, abs/2109.12298, 2021.