Differential Privacy in Image Classification using ResNet-20 and DP-SGD, DP-Adam, and DP-RMSProp Optimization techniques

Praveen Rangavajhula
Department of Computer Science
University of Georgia
Athens, GA, 30602
praveen.rangavajhula@uga.edu

Alexander Darwiche
Department of Computer Science
University of Georgia
Athens, GA, 30605
alexander.darwiche@uga.edu

Deven Allen
Department of Computer Science
University of Georgia
Athens, GA, 30605
dca09692@uga.edu

1 Introduction

Stochastic Gradient Descent (SGD) and its differentially private (DP) relative DP-SGD are frequently used optimizers for image classification tasks. DP-SGD introduces noise and gradient clipping to the standard SGD algorithm, to ensure that the underlying data remains private upon release of model parameters/hyperparameters. While DP-SGD is perhaps the most popular optimizer for tasks concerned with differential privacy, we also explore 2 additional optimization techniques. In this interim report, we looked to explore the impact of changing the optimizer on the test accuracy achieveable on the CIFAR-10 dataset.

The first additional optimizers tested in this paper are Differentially Private Root Mean Square Propogation (DP-RMSProp) and Adaptive Moment Estimation (DP-Adam). The motivation to try these optimizers is to properly bound the benefit of employing different optimization techniques from DP-SGD. RMSProp improves on SGD in that it includes a moving average of squared gradients. This additional term attempts to lessen the possibility of the vanishing/exploding gradient phenomenon. Adam, similarly, includes an estimation of first and second moment of gradients. Adam attempts to build on and improve on both RMSProp and AdaGrad [5].

While implementing these additional optimizers is not novel unto itself, we believe it provides a solid groundwork for additional novelty and testing going forward. This interim report will highlight the current results of our testing and the anticipated path forward. We will briefly describe our proposed transition to a new optimization technique called Lion. To ensure Lion satisfies differential privacy, we will need to make adjustments, specifically by adding noise and clipping.

2 Formal Description of Models Tried

2.1 Motivation and Rationale

As mentioned, the motivation for trying these optimizers was to appropriately bound the strength of the 3 optimization techniques for CIFAR-10. This lays a strong groundwork for future additions to the algorithms and allows us to have a wide range of options in making these adjustments.

Preprint. Under review.

DP-SGD is likely the most common optimization algorithm used in differentially private image classification. The motivation for using DP-SGD is simply to have a strong baseline for comparison. Most literature regarding differential privacy begins by using DP-SGD, so it would make sense as the first algorithm implemented and tested.

RMSProp and Adam can be viewed as successors to DP-SGD. Both introduce approaches that allow for adaptivity of the step size taken at each iteration of the algorithm. These algorithms were first proposed to deal with the vanishing/exlpoding gradient phenomenon and have now been used frequently in the differentially private image classification literature. We felt that implementing these algorithms would provide us with a strong bound on the performance benefits associated with changing optimizer, essentially treating optimizers as another hyperparameter that we could tune.

2.2 Description of Optimization Algorithm (Psuedocode)

Below are pseudocodes for the DP-SGD, DP-RMSprop, and DP-Adam algorithms that we plan to privatize, adapted or referenced from [2]:

Algorithm 1 DP-SGD

```
Input: A step size $\alpha$, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and access to a (possibly noisy) oracle for gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
Initialize: $\mathbf{v}_0 = \mathbf{0}$
for $t = 1, 2, \ldots$ do
    Compute gradient $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    Clip gradient $\mathbf{g}_t = \mathbf{g}_t / \max(1, \frac{\|\mathbf{g}_t\|_2}{C})$
    Add noise: $\hat{\mathbf{g}}_t = \mathbf{g}_t + \mathcal{N}(0, \sigma^2 C^2 I)$
    Descent: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \hat{\mathbf{g}}_t$
end for
```

Algorithm 2 DP-RMSProp

```
Input: A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d \geq 0$, parameter $\beta_2 \in [0, 1)$, step size $\alpha$, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and access to a (possibly noisy) oracle for gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
Initialize: $\mathbf{v}_0 = \mathbf{0}$
for $t = 1, 2, \ldots$ do
    Compute gradient $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    Clip gradient $\mathbf{g}_t = \mathbf{g}_t / \max(1, \frac{\|\mathbf{g}_t\|_2}{C})$
    Add noise: $\hat{\mathbf{g}}_t = \mathbf{g}_t + \mathcal{N}(0, \sigma^2 C^2 I)$
    Calculate moving average of squared gradient $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)(\hat{\mathbf{g}}_t^2 + \xi \mathbf{1}_d)$
    Descent: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \mathbf{V}_t^{-1/2} \hat{\mathbf{g}}_t$
end for
```

Algorithm 3 DP-Adam

```
Input: A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d > 0$, parameters $\beta_1, \beta_2 \in [0, 1)$, a sequence of step sizes $\{\alpha_t\}_{t=1,2,\ldots}$, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and oracle access to gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
Initialize: $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = \mathbf{0}$
for $t = 1, 2, \ldots$ do
    Compute gradient $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    Clip gradient: $\mathbf{g}_t = \mathbf{g}_t / \max(1, \frac{\|\mathbf{g}_t\|_2}{C})$
    Add noise: $\hat{\mathbf{g}}_t = \mathbf{g}_t + \mathcal{N}(0, \sigma^2 C^2 I)$
    Calculate moving average of squared gradient: $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)(\hat{\mathbf{g}}_t^2 + \xi \mathbf{1}_d)$
    Calculate moving average of gradient: $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \hat{\mathbf{g}}_t$
    Descent: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \frac{\mathbf{m}_t}{\sqrt{\mathbf{v}_t} + \epsilon}$
end for
```

2.3 Proposed Modifications and Variations

Most of our current work has focused on implementing and testing the existing DP-SGD, DP-RMSProp, and DP-Adam optimization techniques. The main axes of testing involved changing hyperparameters like epsilon, batch size, or noise level. This has allowed us to establish a base with 3 optimization techniques that can each be used to test our proposed modifications and variations going forward.

Our next proposed approach will be privatizing Evolved Sign Momentum (Lion) optimizer. This optimization technique builds on variants of SGD and Adam and having differentially private implementations, of those, available should provide a good basis of comparison for our newly privatized version of Lion, DP-Lion. In the private world, DP-SGD is still the most popular optimization technique, however in the non-private world, versions of Adam (AdamW) or Adafactor are the usual choices for training a deep neural netwrok. [3] We feel that transitioning from the primary private optimizer to a newly privatized Lion optimizer will be an interesting approach to examine.

To privatize Lion, we will need to add gaussian noise that is relative to the number of iterations, epsilon, and sensitivity. This will take a very similar form to the privatization done for DP-SGD, DP-RMSProp, and DP-Adam. Next, we also plan to implement a clipping constant C, so as to properly bound the sensitivity associated with neighboring datasets. The following psuedocode introduces our proposed adjustments to the Lion Optimizer from [3].

Algorithm 4 DP-Lion

```
Input: A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d > 0$, parameters $\beta_1, \beta_2 \in [0, 1)$, a sequence of step sizes $\{\alpha_t\}_{t=1,2,...}$, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and oracle access to gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
Initialize: $\mathbf{m}_0 = \mathbf{0}$
for $t = 1, 2, \ldots$ do
    Compute gradient $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    Clip gradient $\mathbf{g}_t = \mathbf{g}_t / \max(1, \frac{\|\mathbf{g}_t\|_2}{C})$
    Add noise: $\hat{\mathbf{g}}_t = \mathbf{g}_t + \mathcal{N}(0, \sigma^2 C^2 I)$
    Compute $c_t = \beta_1 m_{t-1} + (1 - \beta_1)\hat{\mathbf{g}}_t$
    Descent: $\mathbf{x}_t = \mathbf{x}_{t-1} - \alpha_t(\text{sign}(c_t) + \lambda \mathbf{x}_{t-1})$
    Calculate moving average of gradient $\mathbf{m}_t = \beta_2 \mathbf{m}_{t-1} + (1 - \beta_2)\hat{\mathbf{g}}_t$
end for
```

2.4 Privacy Proof

Below is a privacy proof from Zhou et al. (2020) [7] that provides generalization bounds for the DP algorithms that we have implemented. This is taken directly from that paper:

In DPAGD, set σ to be as in (6), and for any $\mu > 0$, ε, δ and sample size n satisfying $\varepsilon \leq \frac{\sigma}{13}$, $\delta \leq \frac{\sigma \exp(-\mu^2/2)}{13 \ln(26/\sigma)}$ and $n \geq \frac{2\ln(8/\delta)}{\varepsilon^2}$, the noisy gradients $\tilde{g}_1, \ldots, \tilde{g}_T$ produced in Algorithm 1 satisfy

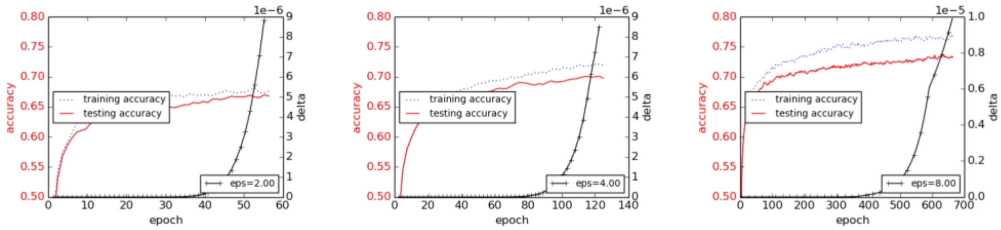
$$P\left\{\|\tilde{g}_t - g_t\| \geq \sqrt{p}\sigma(1 + \mu)\right\} \leq 4p\exp(-\mu^2/2)$$

for all $t \in [T]$.

To obtain a generalization bound (i.e., the upper bound on the L_2-norm of the population gradient) of Algorithm 1 with the guarantee of being (ε, δ)-differentially private, we set the parameter σ and the iteration T in Algorithm 1 to satisfy the conditions in Theorem 2. This also imposes a requirement on the sample size n.

2.5 Related Work: DP-SGD

Abadi et al. [1] introduced DP-SGD, which has become a foundational technique for privacy-preserving model training. When specifically working on the CIFAR-10 dataset, this 2016 paper was able to achieve accuracies of 65-75% while keeping epsilon at or below 8. The biggest difference in their analysis and ours, is that they pretrain their convolutional layers on CIFAR-100 before migrating to CIFAR-10, assuming that CIFAR-100 is a public dataset. Their results are seen below:

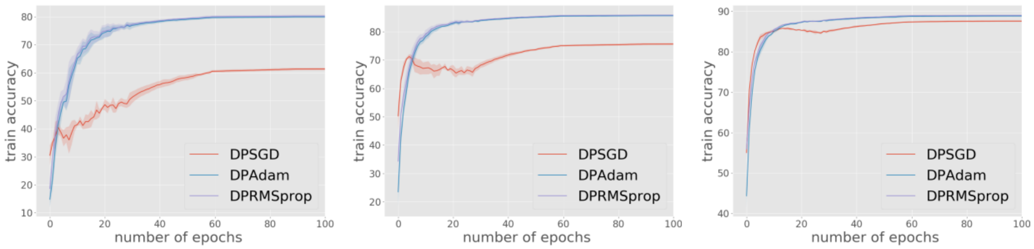


(1) $\varepsilon = 2$ (2) $\varepsilon = 4$ (3) $\varepsilon = 8$

Figure 1: DP-SGD Results on CIFAR-10 [1]

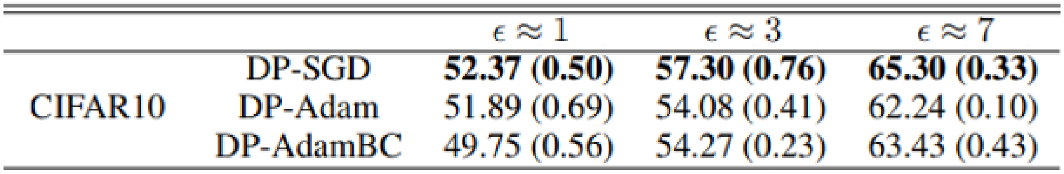
2.6 Related Work: DP-RMSProp and DP-Adam

For comparison of DP-RMSProp and DP-Adam, we turn to the results from Zhou et al. (2020) [7] and Tang et al. (2023) [6]. These 2 papers illustrate some situations where DP-Adam and DP-RMSProp greatly outperform DP-SGD. They also show that these algorithms perform very similarly on other tasks, specifically on CIFAR-10 in many cases.



(a) Training accuracy, $\epsilon = 0.28$ (b) Training accuracy, $\epsilon = 0.57$ (c) Training accuracy, $\epsilon = 1.22$

Figure 2: DP-Adam Results on CIFAR-10 [6]



		$\epsilon \approx 1$	$\epsilon \approx 3$	$\epsilon \approx 7$
CIFAR10	DP-SGD	**52.37 (0.50)**	**57.30 (0.76)**	**65.30 (0.33)**
	DP-Adam	51.89 (0.69)	54.08 (0.41)	62.24 (0.10)
	DP-AdamBC	49.75 (0.56)	54.27 (0.23)	63.43 (0.43)

Figure 3: DP-Adam Results on CIFAR-10 [6]

3 Preliminary Results

3.1 Optimizer Details (with Hyperparameter values)

In the course of testing these 3 optimizers, we have looked to vary the following hyperparameters: Batch Size, Learning Rate, and Epsilon. We used a basic grid search approach to testing the hyperparameters for each optimizer. We varied Batch Size from 64 to 1024, we varied learning rate from 0.1 to 0.5, and we varied epsilon from 3 to 50. The table below shows the entire grid of possible combinations used for each optimizer. The grid search approach allowed us to effectively test the hyperparameter space and appropriately isolate the effects of these changes. The results of these tests will be provided in the following section.

Table 1: Hyperparameter Grid Search

Hyperparameter	Values
Epsilon	[3, 10, 50]
Learning Rate	[0.1, 0.2, 0.3, 0.4, 0.5]
Batch Size	[64, 128, 256, 512, 1024]
Noise Multiplier	1.1
Epochs	30
Clipping Constant	1.0

3.2 Results varying Learning Rate

The following results illustrate our current accuracies when varying learning rate for DP-SGD and DP-Adam. We do not include the current results for DP-RMSProp because we were unable to produce consistent results with while varying learning rate. We've seen some evidence, in our testing, that DP-RMSProp should use learning rates that are lower than those currently tested for the other two optimzers. Results generally show that larger learning rates are bad for these DP algorithms.

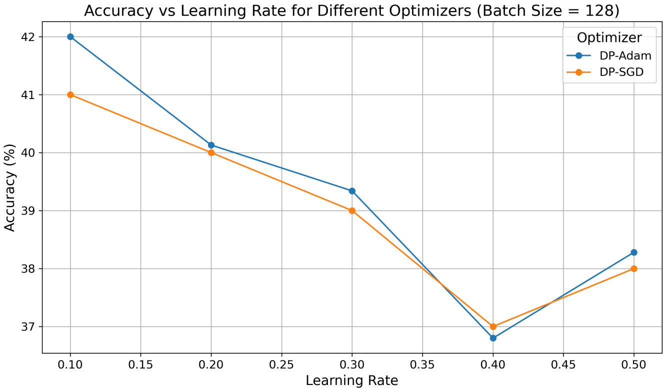


Figure 4: Varying Learning Rate for CIFAR-10

3.3 Results varying Batch Size

The following results illustrate our current accuracies when varying batch size for DP-SGD, DP-RMSProp, and DP-Adam. Accuracy appears to "cap out" around mid sized batches. Specifically, 256 and 512 appear to provide the greatest utility for our optimizers.

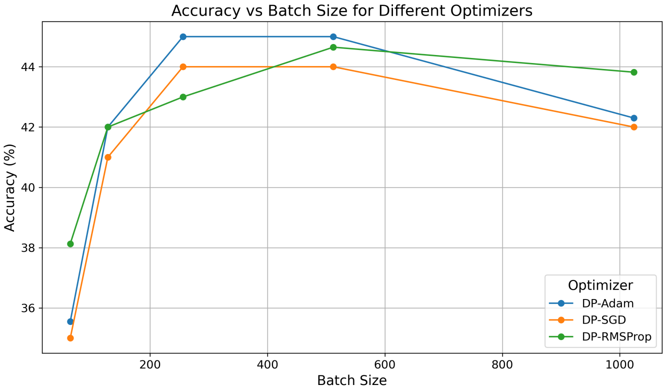


Figure 5: Varying Batch Size for CIFAR-10

3.4 Results varying Privacy Cost

The following results illustrate our current accuracies when varying Privacy Budget for DP-SGD, DP-RMSProp, and DP-Adam. For the most part, accuracies increase as the privacy budget increases. This makes sense as the amount of noise added to the gradients is inversely related to the size of the privacy budget (larger ϵ means lower $\mathcal{N}$). We have an outlier for DP-Adam where the accuracy is much higher at lower privacy budgets. This will be a point of testing going forward, to determine if this was a statistical anamoly or something else entirely.

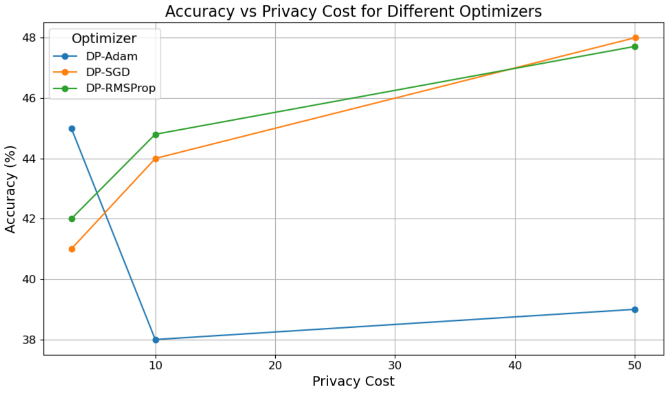


Figure 6: Varying Privacy Cost for CIFAR-10

4 Discussion of Results

4.1 Train Accuracy Discussion

Currently, our best models are able to achieve accuracies in the high 40%'s. We have not been able to break that 50% threshold consistently yet. Given the results above, we find that the "best" hyperparameters for the 3 optimization techniques are relatively consistent. For each, a batch size around 256 and 512 performs the best. We are aware of recommended batch sizes from Soham De et al. (2022) [4] recommending batch sizes that are much larger for CIFAR-10. We are planning to continue testing on Google Colab in the future, which should provide the additional computing power needed to test larger batch sizes. This will also allow us to run models for more epochs.

The results also indicate that the best learning rates are small (compared to our current grid of testing). Currently the best performing learning rate for DP-SGD and DP-Adam is 0.1. For DP-RMSProp, we found that a smaller learning rate, nearer to 0.01 was effective in getting higher test accuracies.

4.2 Moving Forward with Lion Hyperparameters

We now have a good basis to start testing a privatized DP-Lion. Once we're able to get that algorithm functioning with opacus and pytorch, we will use the hyperparameters that showed the best accuracy as our starting points. Obviously, there is no guarantee that these hyperparameters will be "best" for DP-Lion, but it at least guides our hyperparameter search for our newly proposed algorithm.

4.3 Lessons Learned and Plans to Improve

We've come to find that our algorithms struggle to break 50% accuracy on a consistent basis. The varying of the optimizers, while potentially valuable in a non-private setting, has not driven much (if any) improvement in private test accuracies. DP-SGD output our best test accuracy to date. We believe that migrating to a new optimizer, Lion, might help us break into higher test accuracies, but we realize that this might not be as true in a private setting, as it is in a non-private setting.

In our "related work review", Zhou et al. (2020) [7] showed that DP-Adam and DP-RMSProp could outperform DP-SGD in certain sitations. With that in mind, we believe there is additional optimizations we can make to these algorithms to improve their test accuracies. We plan to consult this paper heavily in future testing to improve the outputs of these 2 models.

Another area of improvement could be data augmentation. Currently, our data augmentation implementation only flips and crops the images in CIFAR-10, while training. This can be improved by introducing rotation and some additional translations. We believe that adding more augmentation will correlate with improved model results.

Lastly, the hyperparameter grid search conducted in this first half of the project gave us keen insight into testing going forward. We've now built pipelines to succinctly run multiple model runs in succession and to easily output their results to graphs and charts. We plan to continue developing this pipeline going forward, to allow for faster and cleaner post-run analysis.

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*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Amitabh Basu, Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and ADAM in non-convex optimization and their comparison to nesterov acceleration on autoencoders. *CoRR*, abs/1807.06766, 2018.

[3] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023.

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

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[6] Qiaoyue Tang, Frederick Shpilevskiy, and Mathias Lécuyer. Dp-adambc: Your dp-adam is actually dp-sgd (unless you apply bias correction), 2023.

[7] Yingxue Zhou, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Arindam Banerjee. Private stochastic non-convex optimization: Adaptive algorithms and tighter generalization bounds, 2020.

GitHub Contributions

The code and related materials for this project are available at our GitHub repository: `https://github.com/CS8960-Privacy-Preserving-Data-Analysis/final-project`. Contributions, issues, and discussions are welcome.

Appendix A: Full Model Results

Table 2: Experimental Results
All experiments were conducted with a constant privacy budget $\delta = 10^{-5}$, momentum $\beta = 0.9$, weight decay $\lambda = 10^{-4}$ and a maximum gradient norm of $C = 1.0$.

Experiment ID	**Optimizer**	**Epochs**	**Accuracy**	**Training Time (s)**	**Privacy Cost**	**Learning Rate**	**Batch Size**	**Noise Multiplier**
1	SGD	100	87%	-	-	0.1	128	-
2	SGD	200	94%	-	-	0.1	-	-
3	DP-SGD	30	41%	481.49	3	0.1	128	1.1
4	DP-SGD	30	40%	598.68	3	0.2	128	1.1
5	DP-SGD	30	39%	527.37	3	0.3	128	1.1
6	DP-SGD	30	37%	584.55	3	0.4	128	1.1
7	DP-SGD	30	38%	597.60	3	0.5	128	1.1
8	DP-SGD	30	35%	995.68	3	0.1	64	1.1
9	DP-SGD	30	44%	473.86	3	0.1	256	1.1
10	DP-SGD	30	44%	597.29	2.99	0.1	512	1.1
11	DP-SGD	30	42%	677.04	3	0.1	1024	1.1
12	DP-SGD	30	43%	519.55	8.01	0.1	128	1.1
13	DP-SGD	30	44%	627.49	10.01	0.1	128	1.1
14	DP-SGD	30	48%	553.12	50.04	0.1	128	0.1
15	DP-SGD	30	50%	375.37	50.04	0.1	256	0.1
16	DP-Adam	30	42%	852.44	3	0.1	128	1.1
17	DP-Adam	30	35.55%	1426.00	3	0.1	64	1.1
18	DP-Adam	30	45%	686.00	3	0.1	256	1.1
19	DP-Adam	30	45%	688.00	3	0.1	512	1.1
20	DP-Adam	30	42.30%	740.00	3	0.1	1024	1.1
21	DP-Adam	30	40.13%	713.50	3	0.2	128	1.1
22	DP-Adam	30	39.34%	725.00	3	0.3	128	1.1
23	DP-Adam	30	36.80%	717.70	3	0.4	128	1.1
24	DP-Adam	30	38.28%	733.40	3	0.5	128	1.1
25	DP-Adam	30	30%	642.00	3	0.2	256	1.1
26	DP-Adam	30	38%	690.00	10	0.1	256	1.1
27	DP-Adam	30	39%	743.00	50	0.1	256	1.1
28	DP-Adam	100	36%	2247.00	3	0.1	256	1.1
32	RMSProp	30	42%	459.73	3	0.01	128	1.1
33	RMSProp	30	43%	673.73	3	0.01	256	1.1
34	RMSProp	30	48%	684.61	7.99	0.01	256	1.1
35	RMSProp	30	45%	684.67	10	0.01	256	1.1
36	RMSProp	30	44%	667.19	3	0.01	256	0.1
37	RMSProp	30	39%	669.75	3	0.1	256	1.1
38	RMSProp	30	44%	664.00	3	0.005	256	1.1
39	RMSProp	30	46%	674.38	7.99	0.005	256	0.1
40	RMSProp	30	45%	672.98	8	0.01	256	0.1
41	RMSProp	30	47%	772.55	8.01	0.01	128	1.1
42	RMSProp	30	39%	772.55	3	0.01	128	0.1
43	RMSProp	30	42%	759.14	3	0.005	128	1.1
44	RMSProp	30	45%	775.69	8	0.005	128	1.1
45	RMSProp	30	41%	1283.99	8	0.001	64	1.1
46	RMSProp	30	44.65%	544.07	3	0.01	512	1.1
47	RMSProp	30	43.82%	756.23	3	0.01	1024	1.1