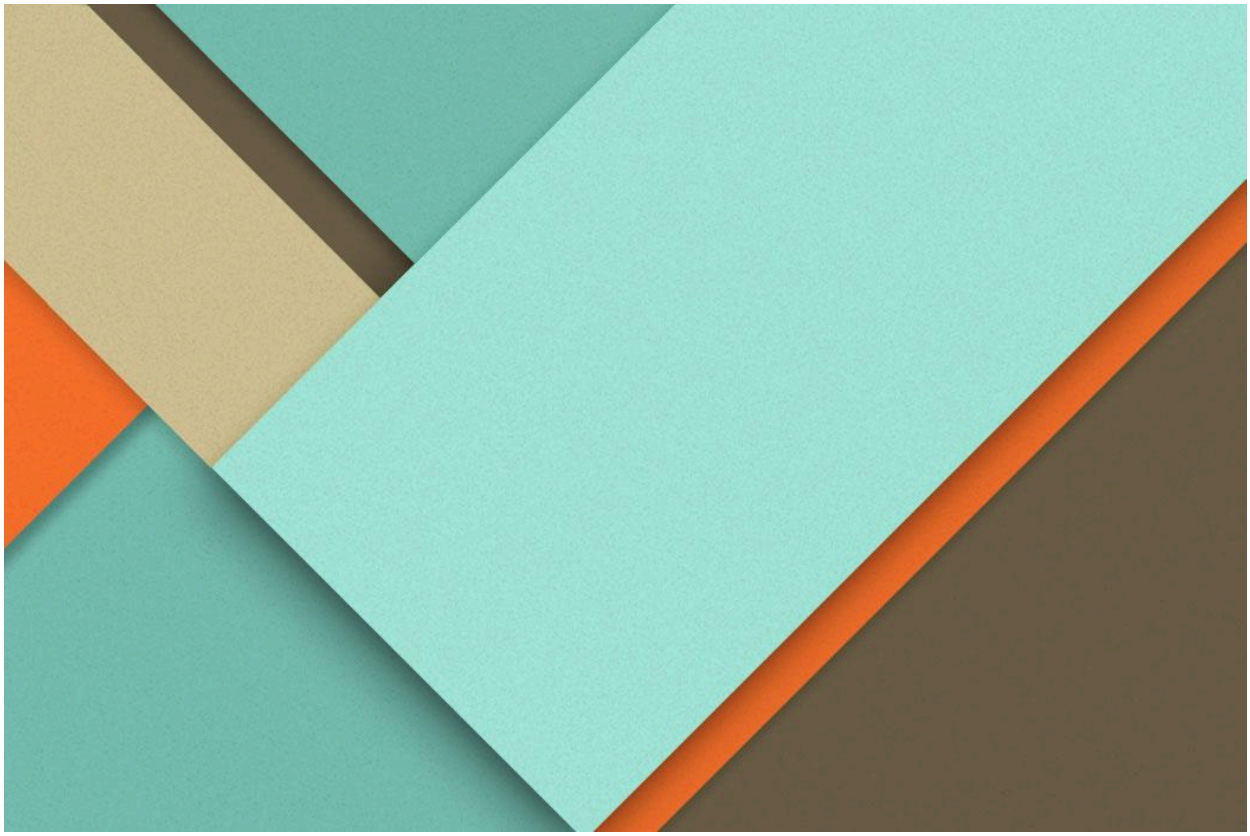


A Brief Treatise: **Deep Learning with Differential Privacy**

SCRIBING ASSIGNMENT

Contributors (Team: **GloVe n Caffe**)**:**

- Shrinivas Prasad Kulkarni, 180070057
- Yash Rajkumar Nehra, 180070066

Based on the paper: 'Deep learning with differential privacy' [1]

Introduction to the problem being solved

Techniques based on neural networks are achieving remarkable results in a wide variety of domains, for which often, the training of models requires large, representative datasets, which may be crowdsourced and contain sensitive information. The models should not expose private information in these datasets. Addressing this goal, the paper develops new algorithmic techniques for learning and a refined analysis of privacy costs within the framework of differential privacy.

The paper treats models with non-convex objectives, several layers, and tens of thousands to millions of parameters, while previous works obtains strong results on convex models with smaller numbers of parameters, or treats complex neural networks but with a large privacy loss. For this purpose, the authors developed new algorithmic strategies, and in the paper they show the following:

1. By tracking detailed information of the privacy loss, we can obtain much tighter estimates on the overall privacy loss, both asymptotically and empirically.
2. Improve the computational efficiency of differentially private training by introducing new techniques.
3. The results obtained using the described approach on two standard image classification tasks, MNIST and CIFAR-10

The approach offers protection against a strong adversary with full knowledge of the training mechanism and access to the model's parameters. This is very relevant, in particular, for applications of machine learning on mobile phones, tablets, and other devices. Storing models on-device enables power-efficient, low-latency inference, and may contribute to privacy since inference does not require communicating user data to a central server; on the other hand, we must assume that the model parameters themselves may be exposed to hostile inspection.

The paper revolves around differential privacy, so we need to understand it first. This section briefs about what is DP, and the Methodologies section would give details about the same, extending the idea further.

Differential privacy is a system for publicly sharing information about a dataset by describing the patterns of groups within the dataset while withholding information about individuals in the dataset. The idea behind differential privacy is that if the effect of making an arbitrary single substitution in the database is small enough, the query result cannot be used to infer much about any single individual, and therefore provides privacy. Another way to describe differential privacy is as a constraint on the algorithms used to publish aggregate information about a statistical database which limits the disclosure of private information of records whose information is in the database.

Definition 1. A randomized mechanism $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ε, δ)-differential privacy if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it holds that

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(d') \in S] + \delta.$$

Where d and d' are adjacent datasets (datasets differing with only one entry), *D* can be thought of as a collection of such datasets, and *R* as some important characterisation of the model, for instance, its parameters.

Intuition Behind the definition:

If we ignore the term δ for time being (also known as $\varepsilon - differential\ privacy$), we can see that a good model, which performs better in private data protection would have a smaller value of epsilon, and that inclusion or exclusion of one datapoint in *d* should affect the probability of the model changing as less as possible, the change being

denoted using e^{ε}. Consequently, a more lucid interpretation of $\varepsilon - DP$ comes from a restatement of the definition as follows:

$$e^{-|\varepsilon|} \leq \frac{Pr[M(d) \in S]}{Pr[M(d') \in S]} \leq e^{|\varepsilon|}$$

The δ can be viewed as just an additional offset to allow for a weaker bound, with additional benefits whatsoever.

The whole idea of *Differential Privacy* comes from the fact that, if some datapoint negligibly affects the model's parameters, it would obviously be difficult to infer whether it was present while training.

Methodology and the Approach

The three main components of the modified learning approach stated in the paper are: *a differentially private SGD algorithm, the moments accountant and the hyperparameter tuning*.

The **Differentially Private Stochastic Gradient Descent Algorithm** is based upon the technique of approximating a function ($f: D \rightarrow R$) with a differentially private mechanism via additive noise calibrated to the f's sensitivity towards adjacent datasets S_f. An intelligent consideration being the inclusion of the additive noise at a rather sophisticatedly strategized step in the algorithm as opposed to the naive implementation which would add the noise directly to the learning/learned parameters and destroy the utility of the model.

The pseudocode for the complete algorithm (The Algorithm 1):

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.
Initialize θ_0 randomly
for $t \in [T]$ **do**
  Take a random sample L_t with sampling probability L/N
  Compute gradient
  For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$
  Clip gradient
  $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$
  Add noise
  $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
  Descent

The *Norm Clipping* is done to ensure a bound over the sensitivity S_f which in turn bounds the privacy loss thus enabling the privacy accounting.

In a more generic setting, the parameters can be grouped layer-wise (in a deeper NN) for the application of the above mechanism thus allowing a different setting (*C* and σ) for each layer.

Furthermore, the algorithm undertakes the noise addition step over groups of the *minibatches* (as in the traditional SGD), which are used for gradient computation, by partitioning the set of all minibatches, these groups are termed *lots.* The group size denoted by L in the above pseudocode is actually the lot size.

Privacy Accounting using the **Moments Accountant** involves the book-keeping operation to be done at each step of the algorithm in order to keep track of the privacy cost incurred hitherto. The composability property of the $(\varepsilon, \delta) - DP$ is leveraged in this part which guarantees and states the bound over the privacy cost of a composition of differentially private mechanisms.

The *Moments Accountant* proposed is a major improvement over the accountant based on the *strong composability theorem* and can be seen as a key contribution of this paper. The path breaking improvements ensued by the above method are evident from the two results, stated as theorems, below, which we see in the following discussions.

THEOREM 1. *There exist constants c_1 and c_2 so that given the sampling probability $q = L/N$ and the number of steps T, for any $\varepsilon < c_1 q^2 T$, Algorithm 1 is (ε, δ)-differentially private for any $\delta > 0$ if we choose*

$$\sigma \geq c_2 \frac{q\sqrt{T \log(1/\delta)}}{\varepsilon} .$$

This theorem can be interpreted in the following ways:

- For given σ, q and T, we have a bound over the privacy cost, in the sense that (ε, δ) can be determined. Also, ε and δ both cannot be decreased indefinitely as both share an inverse kind of relationship in such a setting (assuming equality to ensure optimality).
- For a given (ε, δ), the target privacy cost, the value of σ rapidly increases with the number of epochs T. This is quite intuitive for the more we train, the more our output will be fitted to the input, so more will be the noise we need to add.
- σ increases linearly with the lot size L $(= qN)$, keeping the other parameters constant. This also follows from the fact that we add noise only once per lot.

The second theorem is a bit more involved and provides the necessary tools (procedure) to implement the accountant as opposed to Theorem 1, which provides the ground for it.

THEOREM 2. *Let $\alpha_{\mathcal{M}}(\lambda)$ defined as above. Then*

1. ***[Composability]*** *Suppose that a mechanism $\mathcal{M}$ consists of a sequence of adaptive mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$ where $\mathcal{M}_i : \prod_{j=1}^{i-1} \mathcal{R}_j \times \mathcal{D} \to \mathcal{R}_i$. Then, for any λ*

$$\alpha_{\mathcal{M}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{M}_i}(\lambda) .$$

2. ***[Tail bound]*** *For any $\varepsilon > 0$, the mechanism $\mathcal{M}$ is (ε, δ)-differentially private for*

$$\delta = \min_{\lambda} \exp(\alpha_{\mathcal{M}}(\lambda) - \lambda\varepsilon) .$$

Here, we skip the definition of $\alpha_M(\lambda)$ as it is a bit involved. We just state that it is the λ^{th} moment of a random variable, which can be regarded as a direct measure of privacy cost. Thus, the first clause gives us a composability guarantee, the much sought criterion for a modular design, and the second one gives a handle on the best possible (least) privacy cost. Note that this is an instance of *adaptive composition* which involves sequential

application of mechanisms ($M_1, \ldots, M_k$), and which we model by letting the auxiliary input of the i^{th} mechanism M_i be the output of all the previous mechanisms.

Here, mostly we use the Gaussian mechanism, for which the above computation is much more simplified and the Moments accountant is thus implemented.

The **Hyperparameter Tuning** is relatively empirical and not much insight was gained on it. In particular, we observe that model accuracy is more sensitive to training parameters such as batch size and noise level than to the structure of a neural network. Using theoretical insights, the paper tries to reduce the number of hyperparameter settings that need to be tried. While differentially private optimization of convex objective functions is best achieved using batch sizes as small as 1, non-convex learning, which is inherently less stable, benefits from aggregation into larger batches. At the same time, making batches too large increases the privacy cost (Theorem 1), thus a reasonable tradeoff is to let the number of batches per epoch to be of the same order as the no. of epochs ($O(L/N) = O(T)$).

Another modification concerns the *learning rate* adjustment process, here we never need to decrease the learning rate to a very small value, because differentially private training never reaches a regime where it would be justified, as opposed to ordinary non-private training.

The Implementation

The implementation closely follows the methodology described above, so we just briefly describe the major parts and any empirical deviations/additions from/to the above methodology.

The python class `DPSGD_Optimizer` incorporates as attributes:

1. `sanitizer:` This sanitizes the gradient before using it to update the parameters i.e. periodically adds random noise

2. `privacy_accountant`: This keeps track of the privacy spending over the course of training

(Not including the code here, for brevity)

A major addition here is the preprocessing of the input by projecting it on the principal directions (*PCA*) and/or feeding it through a pre-trained convolutional layer (learned on public data). This is done through the realization of a differentially private version of the ordinary PCA. This ensues an overall development in the model quality.

A minor deviation was done in the experiments with regard to the modification of the learning rate across epochs. A small benefit was observed in starting with a relatively larger rate and then linearly decaying it to a smaller value over a few epochs and then keeping it constant.

Comparison with previous works

Initial works on privacy-preserving learning were done in the framework of secure function evaluation (SFE) and secure multi-party computations (MPC), where the input is split between two or more parties, and the focus is on minimizing information leaked during the joint computation of some agreed-to functionality. In contrast, the paper has assumed that data is held centrally, and the focus is on leakage from the functionality's output (i.e., the model).

A different method, namely k-anonymity and related notions seek to offer a degree of protection to underlying data by generalizing and suppressing certain identifying attributes. According to the paper, this approach has strong theoretical and empirical limitations that make it all but inapplicable to deanonymization of high-dimensional, diverse input datasets. The authors have not pursued input sanitization, keeping the underlying raw records intact and they have perturbed the derived data instead.

The authors have also invented a stronger **privacy accounting,** called as *the moments accountant* method, than which produced the best results previously (the strong

composition theorem). While doing so, they have exploited the fact that the strong composition theorem does not take into account the particular noise distribution under consideration, and is loose at times.

The following figure summarizes the same, and the difference between the plots of **ε** vs epochs for the moments accountant method gives much better results (tighter bound). It has been discussed in the introduction section why a smaller **ε** is better.

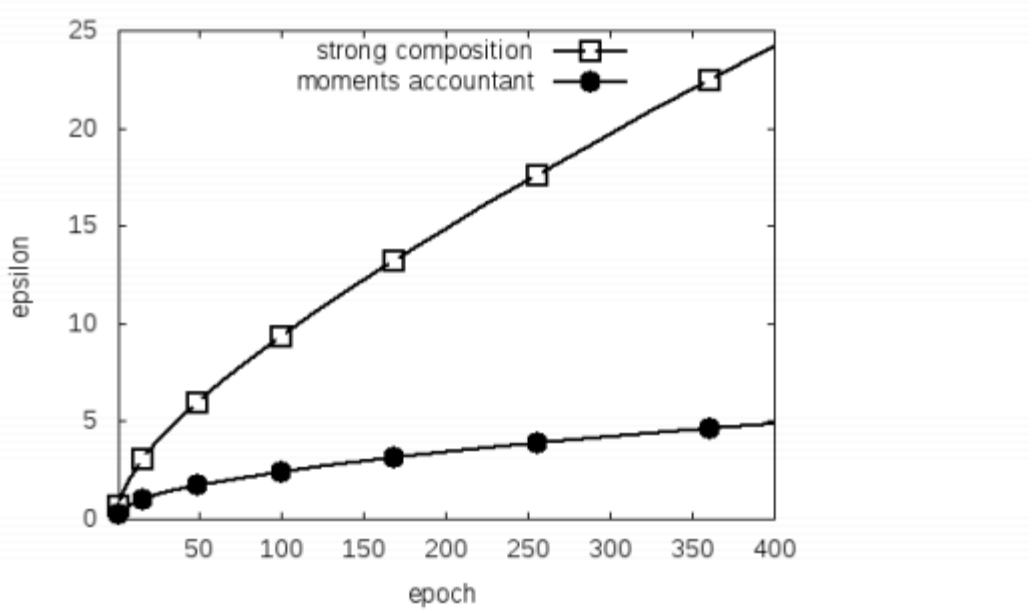
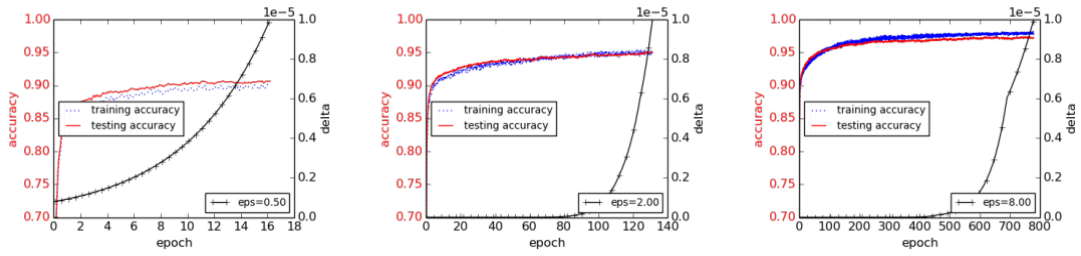


Figure 2: The ε value as a function of epoch E for $q = 0.01$, $\sigma = 4$, $\delta = 10^{-5}$, using the strong composition theorem and the moments accountant respectively.

Significant Experimental results

The two popular image datasets: MNIST and CIFAR-10 are used to test the proposed framework, as these datasets are very common, and hence can give a good idea of performance.

MNIST digits dataset :-



(1) Large noise (2) Medium noise (3) Small noise

Figure 3: Results on the accuracy for different noise levels on the MNIST dataset. In all the experiments, the network uses 60 dimension PCA projection, 1,000 hidden units, and is trained using lot size 600 and clipping threshold 4. The noise levels (σ, σ_p) for training the neural network and for PCA projection are set at (8, 16), (4, 7), and (2, 4), respectively, for the three experiments.

The above figure shows the results for different noise levels. In each plot, we show the evolution of the training and testing accuracy as a function of the number of epochs as well as the corresponding δ value, keeping ε fixed. The paper achieved 90%, 95%, and 97% test set accuracy for $(0.5, 10^{-5})$, $(2, 10^{-5})$, and $(8, 10^{-5})$-differential privacy respectively.

Evaluating the moments accountant

The accuracy for different (ε, δ) pairs is plotted in the figure below. In the figure, each curve corresponds to the best accuracy achieved for a fixed δ, as it varies between 10^{-5} and 10^{-2}. For example, we can achieve 90% accuracy for ε = 0.25 and δ = 0.01. As can be observed from the figure, for a fixed δ, varying the value of ε can have a large impact on accuracy, but for any fixed ε, there is less difference with different δ values.

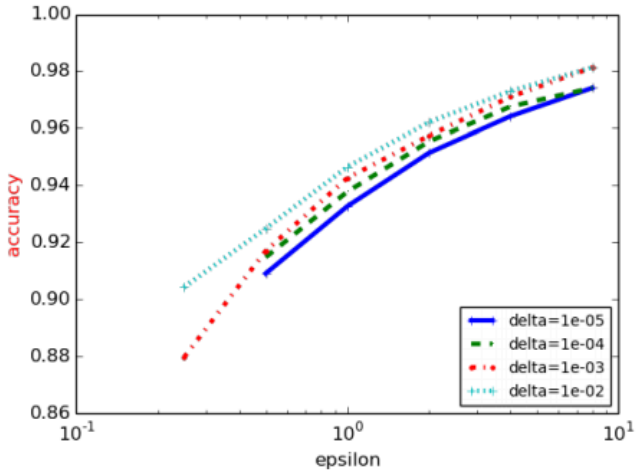
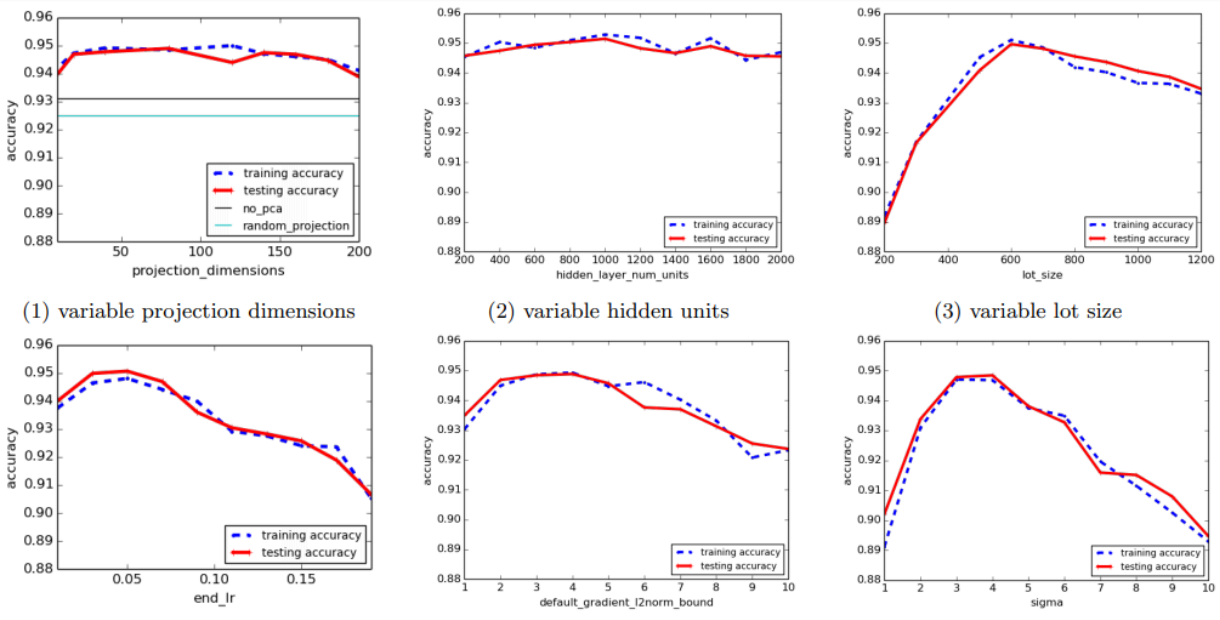


Figure 4: Accuracy of various (ε, δ) privacy values on the MNIST dataset. Each curve corresponds to a different δ value.

Effect of different parameters on the model accuracy

The following set of plots show the effect of varying parameters, such as lot size, learning rate etc, keeping others constant, so that we get an idea of how the model behaves w.r.t each of these parameters.



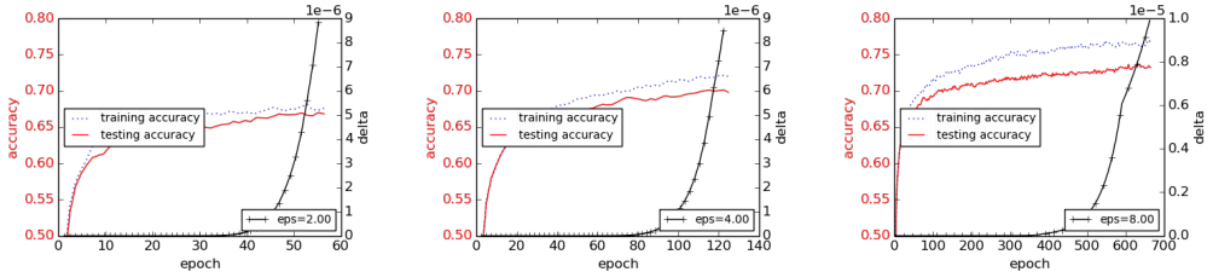
(1) variable projection dimensions (2) variable hidden units (3) variable lot size

(4) variable learning rate (5) variable gradient clipping norm (6) variable noise level

Figure 5: MNIST accuracy when one parameter varies, and the others are fixed at reference values.

CIFAR-10 dataset (colour images)

In the following figure, the paper shows the evolution of the accuracy and the privacy cost, as a function of the number of epochs, for a few different parameter settings. The various parameters influence the accuracy one gets, in ways not too different from that in the MNIST experiments. A lot size of 600 leads to poor results on this dataset and we need to increase it to 2,000 or more for results reported.



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

Figure 6: Results on accuracy for different noise levels on CIFAR-10. With δ set to 10^{-5}, we achieve accuracy 67%, 70%, and 73%, with ε being 2, 4, and 8, respectively. The first graph uses a lot size of 2,000, (2) and (3) use a lot size of 4,000. In all cases, σ is set to 6, and clipping is set to 3.

In a nutshell

The paper demonstrated the training of deep neural networks with differential privacy, incurring a modest total privacy loss, computed over entire models with many parameters. In the experiments for MNIST, they achieve 97% training accuracy and for CIFAR-10 they achieve 73% accuracy, both with $(8, 10^{-5})$-differential privacy. The algorithms used are based on a differentially private version of stochastic gradient descent.

Relation with our project

The paper we have chosen for our project is *"The Secret Revealer: generative model-inversion attacks against deep neural networks".* The main aim of this paper is to make an accurate model to invert NNs. Model Inversion (MI) attacks are attacks where access to a model is abused to infer information about the training data.

The paper claims that Differential Privacy, a standard technique for preserving privacy of individual data points in ML/DL, in its canonical form cannot protect data from GMI.
But what GMI does is reconstruction of an average representation of each class, while DP secures individual data points so that model trained from excluding one datapoint is indistinguishable (to a certain degree) from the model trained including that datapoint, hence preserving the privacy of that datapoint. So, learning an average representation of the classes is not equivalent to membership inference (We can't say that a particular data point belongs to that class of the training data of a model, given an average representation of the class!)
So, it is not wholly correct that DP is not useful against GMI, as GMI is not even an attack from the DP perspective! However, we thought that this is a very interesting topic of exploration: **Can we use similar approaches to break DP?**

Glossary of the terms and abbreviations used:

- **DP** - Differential Privacy
- **Privacy cost/ privacy loss** - characterized by the tuple (ε, δ)
- **GMI attack** - the Generative Model Inversion attack (ref: [2])
- **Gaussian mechanism** - the randomising mechanism using the additive Gaussian noise. Viz.

  $$M(d) = f(d) + N(0, S_f^2 \sigma^2)$$

- **GMI** - Generative Model Inversion
- **PCA** - Principal Components 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, pages 308–318. ACM, 2016.

[2] The Secret Revealer: Generative Model-Inversion Attacks Against Deep Neural Networks. Published in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)