

A Report on

# **Privacy Preserving Machine learning via the DP-SGD algorithm**

---

( IUDX program unit )



## Table Of Contents

1. Introduction to Differential Privacy	2
2. Extension of DP to ML models	7
3. Experimental Setup	12
4. Code	14
5. Results	16
6. References	26




## **What is Differential Privacy ?**

Differential privacy is a concept in data privacy that aims to protect the privacy of individuals while allowing for the analysis and utilization of their data. It involves adding controlled noise or randomness to data before sharing it, in order to prevent the identification of specific individuals or sensitive information.

The basic idea of differential privacy is to provide statistical guarantees that the presence or absence of an individual's data in a dataset will not significantly impact the results of any queries or analyses performed on that dataset. This means that even if an attacker has access to the released data, they should not be able to determine with high certainty whether a particular individual's data is included in the dataset or not.

Differential privacy is typically achieved by adding carefully calibrated noise to the data during the data aggregation process, such as when computing statistical aggregates like counts, sums, or averages. This noise helps to mask the true values of individual data points, while still allowing for meaningful aggregate analyses to be performed.

Differential privacy has become an important concept in the field of data privacy, particularly in the context of data sharing for



research, analysis, and policy-making purposes. It provides a rigorous and mathematically sound approach to protecting individuals' privacy in data-driven applications, while still enabling valuable insights to be drawn from the data.

### **Mathematical Definition :**

Let's assume we have two datasets,  $D$  and  $D'$ , that differ in only one data point. In other words,  $D'$  can be obtained from  $D$  by adding or removing a single data point. These datasets are said to be "neighboring datasets".

A randomized algorithm  $A$  is said to satisfy epsilon-differential privacy if, for any pair of neighboring datasets  $D$  and  $D'$ :

$$P(A(D) \in S) \leq e^\epsilon P(A(D') \in S) + \delta$$

where:

- $P(A(D) \in S)$  is the probability that the randomized algorithm  $A$  outputs a result in the set  $S$  when given dataset  $D$ .
- $P(A(D') \in S)$  is the probability that the randomized algorithm  $A$  outputs a result in the set  $S$  when given dataset  $D'$ .
- $\epsilon$  (epsilon) is a privacy parameter that controls the strength of privacy guarantees. Smaller epsilon values provide stronger privacy guarantees.
- $\delta$  (delta) is a small constant that represents an additional privacy parameter called "delta-differential privacy" that

allows for a small amount of probability mass to be distributed arbitrarily. It is typically set to a small value, such as  $1e-5$ , to ensure that the privacy guarantee holds with high probability.

In essence, privacy is achieved by adding noise to the results thereby compromising on the accuracy which in return provides plausible deniability for a particular entity for which we are trying to protect the privacy of.

Plausible deniability:

- Flip a coin (the coin's bias is the probability that its outcome is head and it will be denoted as  $p_{head}$ ).
- If heads, return the answer in the entry.
- If tails, then flip a second coin and return “yes” if heads and “no” if tails.

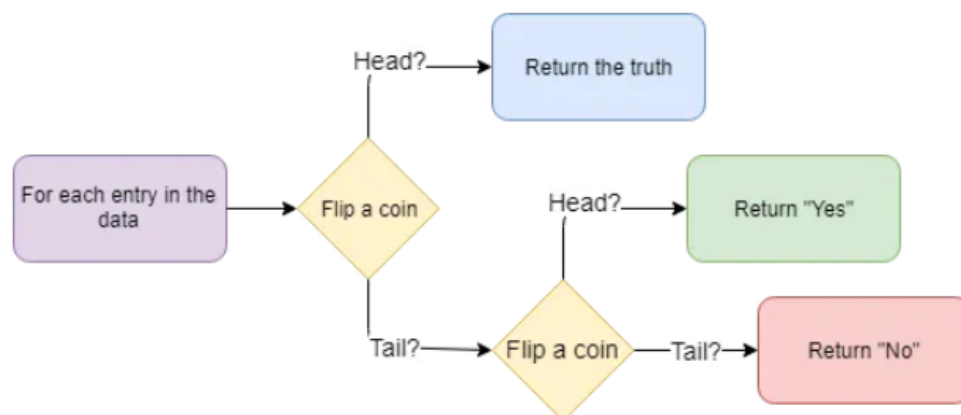



Figure 3. Flow diagram of the Differential privacy algorithm.

Now, each person is protected with “plausible deniability”, because a person is plausible to deny the answer by the randomness of flipping a coin.



Some of the common noise mechanisms are laplacian and gaussian mechanisms.

Laplacian mechanism proposed by Cynthia Dwork has an upside of ensuring  $(\epsilon, \delta=0)$  differential privacy.


We will be using a gaussian mechanism for the work related to DP-ML.


For a simple and fun overview of DP :

( <https://desfontain.es/privacy/friendly-intro-to-differential-privacy.html> )

### **What does DP guarantee ?**

- Privacy Protection: Differential privacy guarantees that the presence or absence of any individual's data in a dataset will not significantly impact the privacy of that individual. It prevents re-identification of specific individuals in the dataset, protecting their privacy and confidentiality.
- Robustness to Linkage Attacks: Differential privacy helps protect against linkage attacks, where an attacker combines the released data with external information to identify individuals. By adding controlled noise to the data, differential privacy makes it harder for an attacker to deduce sensitive information through linking with other data sources.

- 
- Quantifiable Privacy: Differential privacy provides a quantitative measure of privacy through the privacy parameter epsilon. Smaller epsilon values indicate stronger privacy guarantees, allowing for fine-grained control over the trade-off between privacy and data utility.
  - Flexibility: Differential privacy is a flexible concept that can be applied to various types of data and analysis tasks, including aggregate queries, machine learning algorithms, and other data analysis techniques. It can be used in different data sharing scenarios, such as centralized, distributed, and federated settings.
  - Probabilistic Bound: Differential privacy provides a probabilistic bound on the privacy guarantee, allowing for a rigorous mathematical foundation. The delta parameter controls the probability of any additional privacy loss beyond epsilon, providing an additional level of privacy assurance.
  - Reproducibility: Differential privacy allows for reproducible results, as the same input dataset will produce the same output with high probability, even with the added noise. This



ensures that analyses and results can be replicated, verified, and audited.

Overall, differential privacy provides a strong and quantifiable privacy guarantee that enables the sharing and analysis of data while protecting the privacy of individuals. It has become an important tool in the field of data privacy, with applications in various domains such as healthcare, finance, social sciences, and more.

To learn more on DP :

(Calibrating Noise to Sensitivity in Private Data Analysis)


<https://journalprivacyconfidentiality.org/index.php/jpc/article/view/405>

### **Extension of the concept of Differential privacy to Machine learning models :**

Consider a case where we would like to publish a machine learning model which was trained on a sensitive/private dataset which we do not want to disclose, but we want to publish a model that makes some predictions based on that dataset.

People have discovered adversarial attacks that violate exactly this requirement.






Differential privacy can provide protection against several types of adversarial attacks in machine learning models, including:

- Membership Inference Attacks: Membership inference attacks involve an adversary trying to determine if a particular data point was used in the training data of a machine learning model. By exploiting the model's outputs, an adversary may be able to infer whether a data point was part of the training data, thereby violating privacy. Differential privacy can protect against membership inference attacks by adding controlled noise to the model's output, making it difficult for the adversary to determine if a specific data point was used in the training data or not.

To know more : Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017, May). Membership inference attacks against machine learning models. In 2017 IEEE symposium on security and privacy (SP) (pp. 3-18). IEEE.

- Reconstruction Attacks: Reconstruction attacks involve an adversary trying to reconstruct the sensitive data points used in the training data of a machine learning model based on the model's outputs. This can be done by exploiting the outputs of the model, such as the predicted probabilities or model parameters, to reconstruct the original data points.



Differential privacy can thwart reconstruction attacks by adding noise to the model's outputs, making it challenging for the adversary to accurately reconstruct the original data points.


To know more : Salem, A. M. G., Bhattacharyya, A., Backes, M., Fritz, M., & Zhang, Y. (2020). Updates-leak: Data set inference and reconstruction attacks in online learning. In 29th USENIX Security Symposium (pp. 1291-1308). USENIX.

- Model Poisoning Attacks: Model poisoning attacks involve an adversary trying to manipulate the training data to inject malicious data points with the aim of compromising the integrity or performance of the trained model. Differential privacy can provide protection against model poisoning attacks by perturbing the training data with controlled noise, making it harder for the adversary to maliciously manipulate the data points to poison the model.

To know more :

<https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db>

- Query-based Attacks: Query-based attacks involve an adversary making repeated queries to a machine learning model to gain insight into the training data or infer sensitive



information about the individuals in the training data.

Differential privacy can protect against query-based attacks by adding noise to the model's outputs or aggregating the query results in a privacy-preserving manner, preventing the adversary from extracting sensitive information through repeated queries.

To know more : Ilyas, A., Engstrom, L., Athalye, A., & Lin, J. (2018, July). Black-box adversarial attacks with limited queries and information. In International conference on machine learning (pp. 2137-2146). PMLR.

By incorporating differential privacy into machine learning models, organizations can mitigate the risk of these and other adversarial attacks, ensuring that the privacy of the individuals whose data is used for model training or inference is protected. However, it is important to note that the level of protection provided by differential privacy depends on the specific design and implementation of the differential privacy mechanism, and careful consideration should be given to the choice of privacy parameters and noise levels to achieve the desired privacy-utility trade-off.

In order to achieve differential privacy on Machine learning/ Deep learning models we have used the algorithm proposed by Abadi et.al.

While training the ML model, in the backpropagating step when the gradients are computed, the gradients are clipped to a particular threshold value and then the noise is added based on the required noise distribution (here the authors have picked gaussian distribution ).

---

**Algorithm 1** Differentially private SGD (Outline)

---

**Input:** Examples  $\{x_1, \dots, x_N\}$ , loss function  $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$ . Parameters: learning rate  $\eta_t$ , noise scale  $\sigma$ , group size  $L$ , gradient norm bound  $C$ .  
**Initialize**  $\theta_0$  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**  
     $\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C})$   
    **Add noise**  
     $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L} (\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$   
    **Descent**  
     $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$   
**Output**  $\theta_T$  and compute the overall privacy cost  $(\epsilon, \delta)$  using a privacy accounting method.

---

To learn more on differential privacy for machine learning/deep learning :

(Deep learning with Differential Privacy)(Abadi et.al)

( <https://arxiv.org/abs/1607.00133> )



## **Experimental Setup for replication of Concept and Results mentioned in Abadi et.al :**

The authors have trained 2 models :

- a) Without DP (baseline)
- b) With DP

in order to quantify the privacy results and the impact of epsilon on accuracy.


There is no mention of the scaling mechanism for the input parameters in the paper. So in order to test its impact we have performed Min-Max and Standard Scaling before and after PCA (as mentioned in the paper)

The authors use a Differentially Private PCA, but due to time constraints we use a generic PCA.

In addition to this we have also tested the impact of momentum in the training process. We have tested with the momentum values of 0 and 0.9.

With the mechanism and parameters mentioned in the paper and the above mentioned scaling and momentum in mind, we have performed experiments using 2 different Frameworks :

- i) Pytorch and ii) tensorflow.



On top of these frameworks, in order to apply differential privacy, we have used libraries such as: i) Opacus (for DP on pytorch) and ii) TF-privacy (for DP on tensorflow) .

Better results and faster training is achieved using Opacus and hence Pytorch-Opacus combo is our choice for future experiments.

The machine learning model coded for the experiment is basically a hand-written digit classifier for which the dataset it is trained on is the famous MNIST dataset.

(MNIST : 70,000 28x28 black-and-white images of handwritten digits (0-9) extracted from two NIST databases)

To briefly categorize the experiments performed :

- 1) Baseline model in
  - a) pytorch (linear layers)
  - b) tensorflow (linear layers)
- 2) Baseline model integrated with a DP-SGD privacy engine in
  - a) pytorch-opacus (linear, batchnorm and CNN layers)
  - b) tensorflow-tf privacy (linear layers)

One important issue that had to be addressed was the incompatibility of batch normalization for differential privacy.

( <https://arxiv.org/pdf/2006.10919.pdf> )

**Code implemented for the experiments :**

<https://github.com/datakaveri/differential-privacy-ml>

Example code snippet :

```
privacy_engine = PrivacyEngine()
model, optimizer, train_loader = privacy_engine.make_private_with_epsilon(
    module=model,
    optimizer=optimizer,
    data_loader=train_loader,
    epochs=epochs,
    target_epsilon=epsilon,
    target_delta=delta,
    max_grad_norm=max_grad_norm
)
```

To install opacus on our runtime :

```
!pip install opacus
```

To import the privacy engine that we require from opacus and the module validator that check for parts of code that are incompatible with opacus (ie DP incompatible) :

```
from opacus import PrivacyEngine
from opacus.validators import ModuleValidator
```

To track and print the epsilon throughout the epochs :

```
epsilon = privacy_engine.get_epsilon(delta)
print('Trained Epoch: {} with loss= {:.6f}, accuracy= {:.2f}, and epsilon= {:.2f}'.format(
    epoch,
    epoch_loss / count,
    100. * correct / len(train_loader.dataset),
    epsilon
))
```

To check for incompatibility and change the type of layers (for example batch norm in this case ) :

```
model = Net()
model = model.to(device)
errors = ModuleValidator.validate(model, strict=False)
print("ERRORS: ", errors[-5:])
model = ModuleValidator.fix(model)
errors = ModuleValidator.validate(model, strict=False)
print("NEW ERRORS: ", errors[-5:])
optimizer = optim.SGD(model.parameters(), lr=lr, momentum=momentum)
```

All these above code snippets can be better understood when we go through the below documentation of Opacus and the code in the github from the beginning.

Since moving forward Opacus will be preferred, here is the documentation for its usage :

( <https://opacus.ai/api/> )

## Results :

Here are the results of the experiments performed :



	Sigma = 1 gradclipnorm = 4 epochs = 100 learningrate = 0.05 delta = 1e-5	No Scaling	Min-Max Scaling before PCA	Min-Max scaling after PCA	Standard Scaling before PCA	Standard Scaling after PCA
Momentum = 0.9	Pytorch baseline	97.87	97.56	97.4	97.51	97.53
	Pytorch DP	93.73	96.45	92.17	94.68	95.19
	TF baseline	-	-	-	-	-
	TF-Privacy	-	-	-	-	-
Momentum = 0	Pytorch baseline	97.62	97.76	97.61	97.58	97.52
	Pytorch DP	94.6	95.82	90.56	95.62	95.79
	TF baseline	98.21	98.09	98.16	98.26	98.01
	TF-Privacy	96.14	96.01	90.39	95.76	95.89
Average Accuracy		96.36166667	96.94833333	94.38166667	96.56833333	96.655

DPML with Opacus										
Sigma	No Scaling		MinMax Before		MinMax After		Standard Before		Standard After	
	m=0	m=0.9	m=0	m=0.9	m=0	m=0.9	m=0	m=0.9	m=0	m=0.9
3.86718	89.8	89.29	94.47	89.25	89.74	87.14	93.61	89.47	93.72	87.14
2.13867	91.56	90.84	95.65	92.41	90.18	90.18	95.18	91.17	95.07	90.18
1.12548	93.87	92.9	95.72	95.23	90.42	93.07	95.79	93.04	95.5	93.07
0.8023	94.34	93.59	95.94	96.31	90.43	95.07	95.68	94.84	95.68	95.07
0.6269	94.97	94.24	96.05	97.03	90.56	96.18	95.79	95.78	95.74	96.18

Here are some research questions and its answers we are able to infer from our experiments :

Question	Resolution
What is the relationship between momentum and accuracy in DP-SGDM?	For differentially private model training, momentum shows an increasing accuracy trend but is outperformed by a model that is trained without momentum
Momentum should intuitively affect the rate of convergence of the model, so why does it have an affect on the overall accuracy of the model? Is the 'saddle hypothesis' worth exploring?	Not worth exploring
Can we safely assume the usage of momentum in Abadi et al. and can we take the results obtained in their paper at face value?	In Abadi: momentum is not used Tensor Flow is used with a custom implementation of DP
Is our implementation of DP-SGD directly comparable to Abadi et al.?	Yes, in both TensorFlow and Pytorch models. We can progress with Pytorch as it is comparable to tensorflow with caveats noted in D2

Does setting sigma (noise variance) to 0 give the same results as non-DP trained models?	Not possible to set sigma to 0 in DP models
What is the relationship between scaling and PCA in machine learning? Is there a particular scaling method that works better for PCA?	Scaling should be done before PCA, scaling method depends on application

---

## References and Resources :

1. Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2016). Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3), 17-51.
2. Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308-318).
3. <https://github.com/datakaveri/differential-privacy-ml> (Github repo of experiments performed)
4. <https://desfontain.es/privacy/friendly-intro-to-differential-privacy.html> (Simple and interactive explanation of DP)
5. <https://discuss.pytorch.org/t/why-batchnorm-layer-is-not-compatible-with-dp-sgd/154208> ( Effect of Batch Normalization in DP-SGD )
6. <https://opacus.ai/api/> ( Opacus documentation )

- 
7. Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017, May). Membership inference attacks against machine learning models. In 2017 IEEE symposium on security and privacy (SP) (pp. 3-18). IEEE.
  8. Salem, A. M. G., Bhattacharyya, A., Backes, M., Fritz, M., & Zhang, Y. (2020). Updates-leak: Data set inference and reconstruction attacks in online learning. In 29th USENIX Security Symposium (pp. 1291-1308). USENIX.
  9. <https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db>
  10. Ilyas, A., Engstrom, L., Athalye, A., & Lin, J. (2018, July). Black-box adversarial attacks with limited queries and information. In International conference on machine learning (pp. 2137-2146). PMLR.