Deep Learning with Differential Privacy

Kaoutar BOULIF, Charlotte DURAND

Introduction

Article : « Deep Learning with Differential Privacy »

Authors : Martín Abadin and colleagues

Submission year : 2016

Contribution : combine machine learning algorithms with advanced privacy preserving mechanisms , training neural networks within a modest privacy budget

Implementation : Tensorflow

Datasets: MNIST / CIFAR-10

Basic concepts

DIFFERENTIAL PRIVACY:

Definition 1. A randomized mechanism $\mathcal{M}\colon \mathcal{D} \rightarrow \mathcal{R}$ 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

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

SENSITIVITY:

$$\Delta f = \max \| f(D_1) - f(D_2) \|_1$$

on datasets D_1, D_2 differing on at most one element

Differentially private SGD algorithm

Algorithm 1 Differentially private SGD (Outline)

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

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Hyperparameter tuning and results

MNIST:

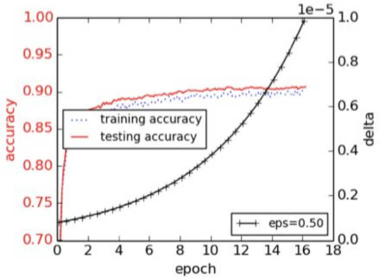
<u>Baseline model</u>: 98.3% in about 100 epochs

<u>Differentially private model:</u>

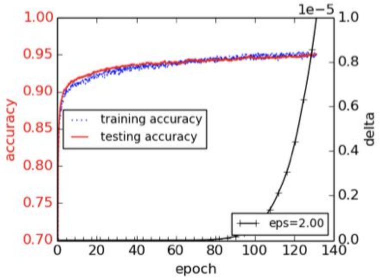
small noise scale ($\sigma = 2$, $\sigma p = 4$): 90% accuracy for (ε=0.5, $\delta=10^{-5}$)DP

medium ($\sigma = 4$, $\sigma p = 7$) : 95% accuracy for (ε=2, $\delta=10^{-5}$) DP

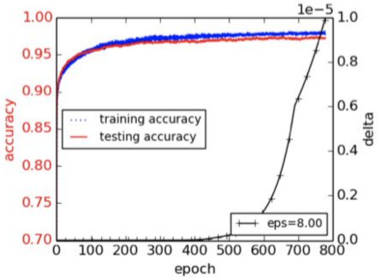
large ($\sigma = 8$, $\sigma p = 16$) : 97% accuracy for (ε=8, $\delta=10^{-5}$) DP



(1) Large noise



(2) Medium noise



(3) Small noise

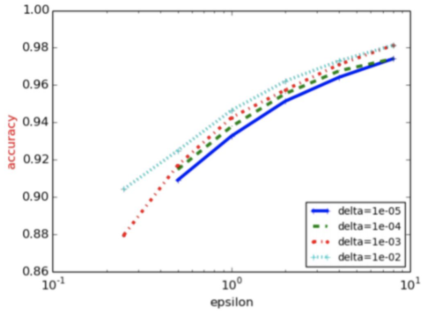
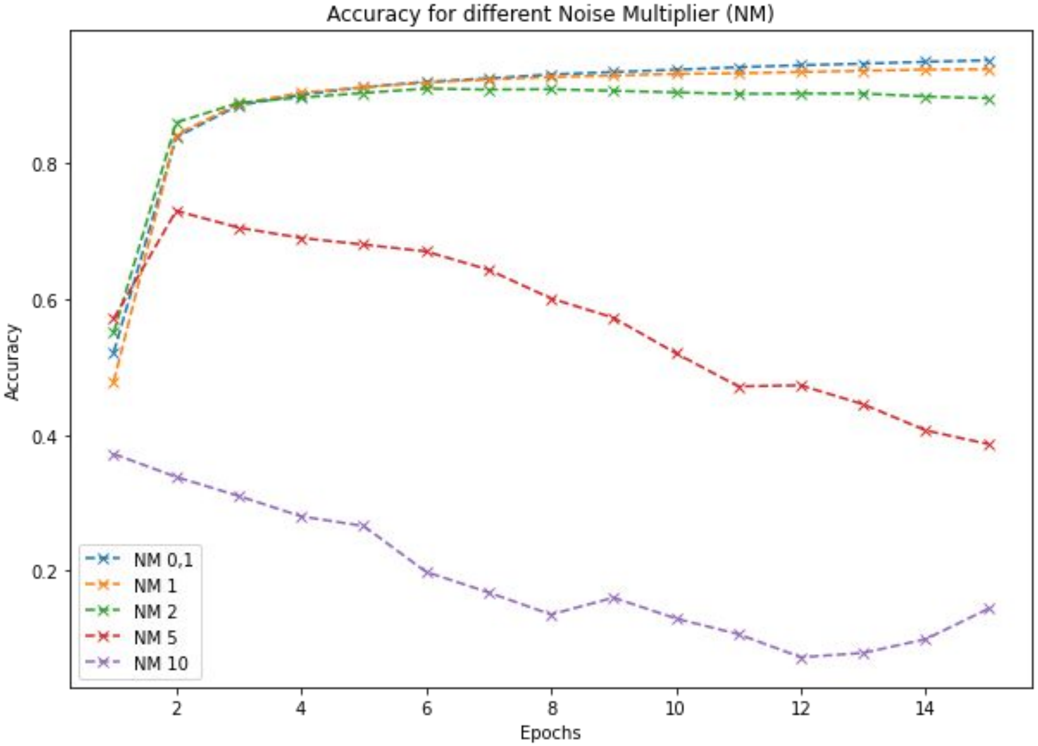
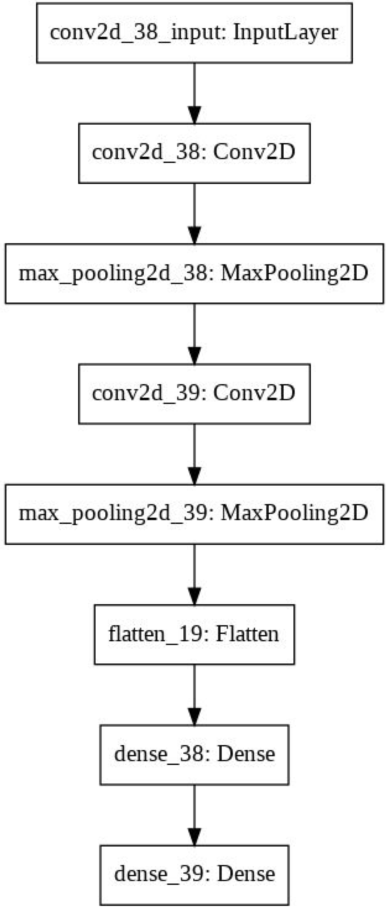


Figure 4: Accuracy of various (ε, δ) privacy values on the MNIST dataset. Each curve corresponds to a different δ value.

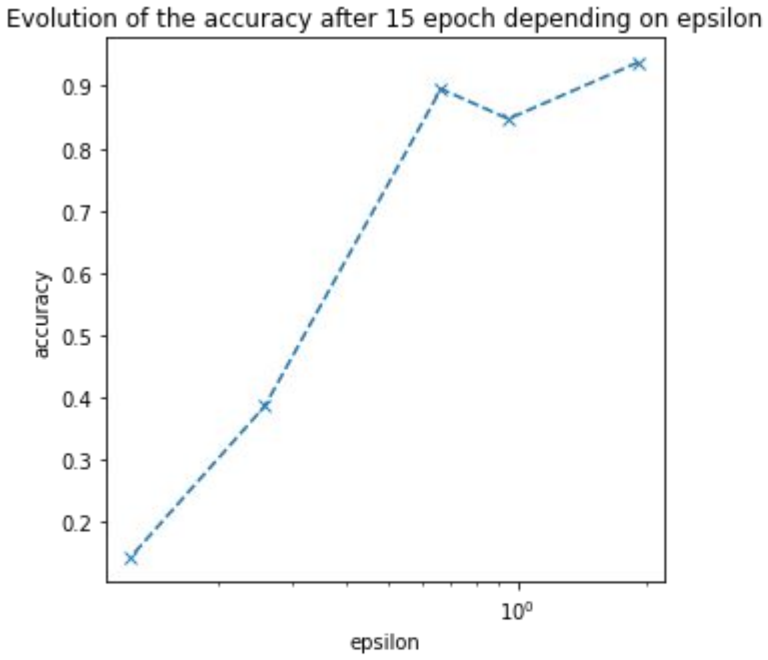
MNIST

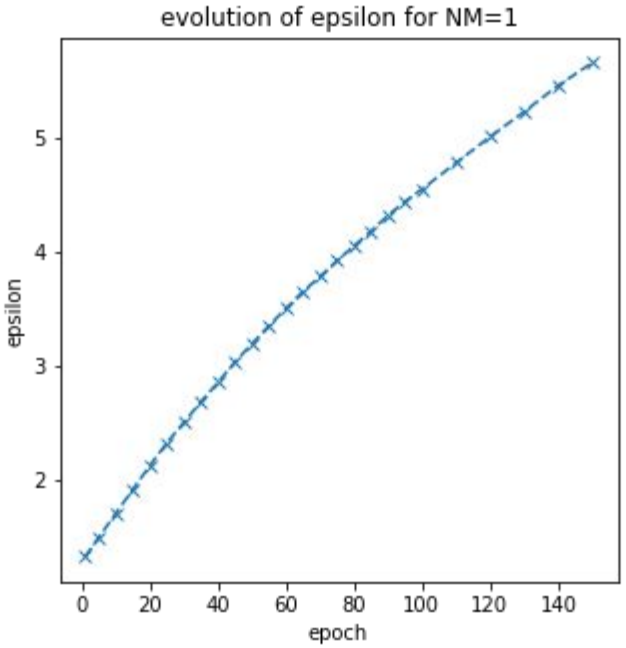




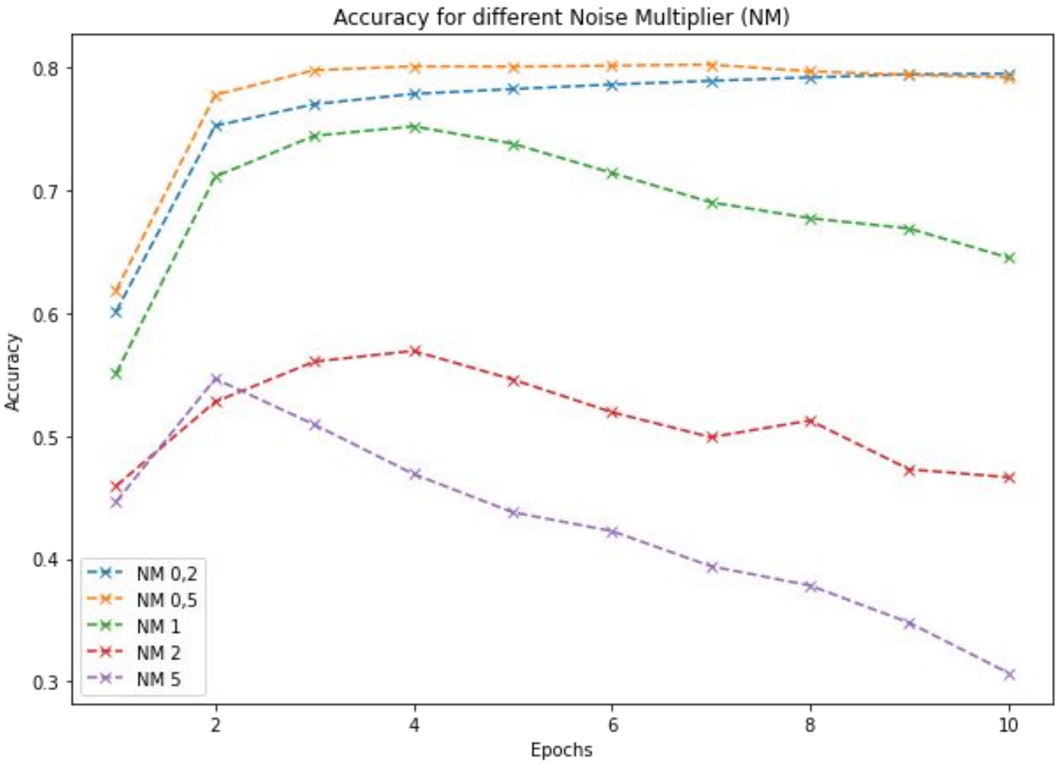
Architecture du modèle

MNIST


Evolution of the accuracy after 15 epoch depending on epsilon
accuracy
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
10^0
epsilon


evolution of epsilon for NM=1
epsilon
2
3
4
5
0
20
40
60
80
100
120
140
epoch

Fashion MNIST

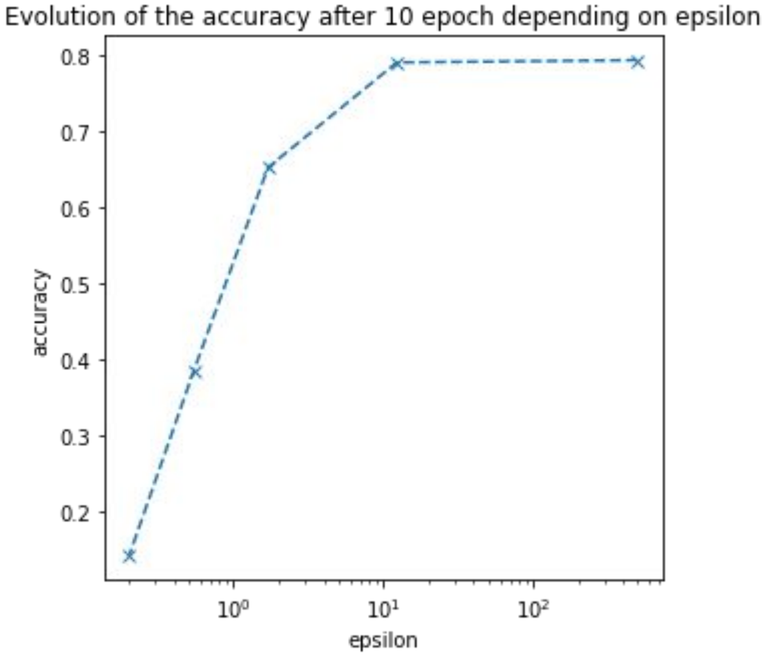


- lower accuracy
- fine tuning hyperparameters :
  l2_norm=5
  increase of the learning rate

- Longer computation

- Need to decrease the learning rate when Noise Multiplier is bigger

Fashion MNIST



Similar dependance than for MNIST

Find the good trade-off between accuracy and differential privacy

Conclusion

→ Fine-tuning hyperparameters isn’t easy

→ Yet, we obtain interesting results but computation is time consuming

Future possibilities :

- training with another optimizer (RMSProp for example)
- training on other dataset
- seeing the influence of the norm clipping

Thank you for your attention !

Any questions ?