Differential Privacy and Neural Networks: A Preliminary Analysis

Giuseppe Manco and Giuseppe Pirrò

ICAR-CNR, Italy
{manco,pirro}@icar.cnr.it

Abstract. The soaring amount of data coming from a variety of sources including social networks and mobile devices opens up new perspectives while at the same time posing new challenges. On one hand, AI-systems like Neural Networks paved the way toward new applications ranging from self-driving cars to text understanding. On the other hand, the management and analysis of data that fed these applications raise concerns about the privacy of data contributors. One robust (from the mathematical point of view) privacy definition is that of Differential Privacy (DP). The peculiarity of DP-based algorithms is that they do not work on anonymized versions of the data; they add a calibrated amount of noise before releasing the results, instead. The goals of this paper are: to give an overview on recent research results marrying DP and neural networks; to present a blueprint for differentially private neural networks; and, to discuss our findings and point out new research challenges.

1 Introduction

Neural networks (NNs) have recently found many applications in several areas of Artificial Intelligence, ranging from self-driving cars to language understanding. This is both due to significant progresses in terms of techniques [3, 10] to (pre)train complex networks including hundreds of layers (the so called Deep Network) involving millions of units, and the availability of large and representative datasets. However, information about individuals maintained by third-parties raises concerns about the protection of user-privacy when analyzing such data. Therefore, it emerges the need for techniques able to offer both utility to the applications and rigorous privacy guarantees.

This problem has been attacked by combining competences from different fields, viz. data mining, cryptography, information hiding, to cite a few. In 2006, C. Dwork introduced the notion of *differential privacy* [7]. The idea behind differential privacy (DP) can be summarized as follows: *the removal or addition of a single database item does not (substantially) affect the outcome of any analysis.* This can be mathematically guaranteed and thus any data contributor would be more willing to take part into the database as her risk (in terms of privacy violation) does not *substantially* increase. The main challenge is to release aggregate information about the databases while protecting the privacy of individual contributors. While DP has been explored in a variety of many machine learning

and data mining tasks ranging from regression analysis [21] to classification [2], its usage in neural networks is still in its infancy [1]. The goals of this paper are as follows: (i) providing an overview of techniques that combine DP and Neural Networks; (ii) reporting on our own experience in this field; (iii) discussing pros and cons of our approach and pointing out open problems.

The remainder of the paper is organized as follows. In Section 2 we introduce some background. We give an overview on related research in Section 3. We describe our approach in Section 4. We conclude in Section 5.

2 Background

We consider a dataset $\mathcal{D}$ of N tuples $\mathbf{x_1}, ...\mathbf{x_N}$ where each tuple $\mathbf{x}_i \in \mathcal{D}$ has $d+q$ attributes $\mathbf{x}_i = \langle x_i^1, ..., x_i^d, t_i^1, ...t_i^q \rangle$. We assume without loss of generality that $\sqrt{\sum_{i=1}^{d} x_{id}^2} \leq 1$. We model the general data analysis problem by considering a function f, which given a tuple $\mathbf{x} \in \mathcal{D}$ takes as input the tuple 's attributes $x_i^1, ...x_i^d$ and returns an output $y_i^1, ...y_i^k$ as accurate as possible. The function $f(\mathcal{D}, \mathbf{w})$ requires some model parameter $\mathbf{w}$, and its accuracy is evaluated by using some function $E(\mathcal{D}, \mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{D}} E(\mathbf{x}, \mathbf{w})$ that compares the output of f with the reference values. The goal is to find the optimal parameter models $\mathbf{w}^*$ by solving the optimization problem $\mathbf{w}^* = \underset{\mathbf{w}}{\text{minimize}}\ E(\mathcal{D}, \mathbf{w})$.

2.1 Neural Networks

In terms of the general learning problem introduced before, the idea is to model the function $f(\mathcal{D}, \mathbf{w})$ via Neural Networks. We will focus on the Multi Layer Perceptron (MLP) [4] model. The MLP is a feed forward neural network, that is, a network where connections neither are allowed between units in the same layer nor backward. Strictly speaking, a neural network models a nonlinear function from a set of input variables $\mathbf{x_i}, i \in [1, d]$ to a set of output variables $\mathbf{t_l}, l \in [1, q]$ controlled by a vector $\mathbf{w}$ of adjustable parameters.

The topology of a $\langle d, m, q \rangle$-layer network is given as follows. We assume that each layer $j \in \{1, \ldots, m\}$ is characterized by a size s_j, two vectors $\mathbf{z}^{(j)}$ and $\mathbf{a}^{(j)}$ and a matrix $\mathbf{w}^{(j)}$ of weights. We assume that $s_0 = d$ is the size of the input data $\mathbf{x} \in \mathbb{R}^d$ (i.e., number of units in the input layer), and $s_m = q$ is the size of the output (i.e., number of units in the output layer). We refer to ϕ as the activation function for intermediate units and ψ for output units, respectively. The relationships between these components are recursively define by the following general formulas:

$$
\begin{aligned}
z_i^{(0)}(\mathbf{x}, \mathbf{w}) &= x_i \\
z_i^{(j)}(\mathbf{x}, \mathbf{w}) &= \phi\left(a_i^{(j)}(\mathbf{x}, \mathbf{w})\right) \\
a_i^{(j)}(\mathbf{x}, \mathbf{w}) &= \sum_{k=1}^{n-1} w_{i,k}^{(j)} \cdot z_k^{(j-1)}(\mathbf{x}, \mathbf{w}) \\
y_q(\mathbf{x}, \mathbf{w}) &= \psi\left(a_q^{(m)}(\mathbf{x}, \mathbf{w})\right)
\end{aligned}
$$

To keep notation uncluttered, we leave out bias terms and assume that $S{=}\sum_{j=1}^{m} s_j$ and the vectors $\mathbf{z}$ and $\mathbf{a}$ span over $\mathbb{R}^S$ where $a_i{\equiv}\, a_v^{(h)}$ for some encoding $i = \langle h, v\rangle$. This induces a partial order $j{\prec}\, i$ which holds when $i = \langle h+1, v\rangle$ and $j = \langle h, u\rangle$. Thus, the weight matrices can be represented by a unique vector $\mathbf{w}$, where $w_{i,j}$ corresponds to the weight of the connection between nodes j and i such that $j \prec i$. This allows us to simplify the previous equations as:

$$z_{\mathbf{x},i} = \phi_i\left(a_{\mathbf{x},i}\right)$$
$$a_{\mathbf{x},i} = \sum_{j \prec i} w_{i,j} \cdot z_{\mathbf{x},j}$$

where $z_{\mathbf{x},i}$ (resp. $a_{\mathbf{x},i}$) represents the application of a_i to $\mathbf{x}$, ϕ_i represents the activation function relative to unit i.

Training Neural Networks. After defining the topology, the problem that arises is that of *training the network*, that is, learning the model parameters (weights and biases) that minimize the error function $E(\mathcal{D}, \mathbf{w})$. This is typically done by using some variant of the back-propagation algorithm [13] coupled with pre-training techniques [10, 3] that have been successful for deep networks (ie., networks including many layers).

2.2 Differential Privacy

Differential privacy captures the increased risk to one's privacy that incurs by participating into a database. It does so by offering to individual participants the guarantees that the output of a query would also have been the same, with sufficiently high probability, even if the record(s) of a participant were not present in the database. Being differentially private is a property of the data access mechanism, and is unrelated to the presence or absence of auxiliary information available to the adversary. Differential privacy works by introducing randomness [8].

Proposition 1 (**(ϵ,δ)-Differential Privacy**). *A randomized function f guarantees (ϵ-δ)-differential privacy if for all data sets $\mathcal{D}$ and $\mathcal{D}'$ differing on at most one element, and all $O \subseteq Range(f)$, it holds that:* $\frac{Pr[f(\mathcal{D})\in O]}{Pr[f(\mathcal{D}')\in O]} \leq e^{\epsilon} + \delta$

In the definition, $\epsilon{>}0$ is a known privacy parameter that controls the strength of the differential privacy guarantee: larger (resp., smaller) values of ϵ yields weaker (resp., stronger) privacy. δ models the probability that the condition on ϵ fails to hold. The definition of (ϵ-δ)-DP becomes more stringent as ϵ and δ approach 0. Moreover, when ϵ is small, the $e^{\epsilon} \sim 1 + \epsilon$. Since differential privacy works in an interactive setting, the randomized computation f is the algorithm applied by the data curator when releasing information. When $\delta{=}0$ we talk about ϵ-DP; however, (ϵ, δ)-DP behaves better when one needs to *compose* (ϵ, δ)-DP mechanisms e.g., when multiple accesses to the database are required [11].

2.3 Achieving Differential Privacy

We shall now outline how to achieve DP in practice. For real-valued functions, DP can be achieved by adding a calibrated amount of noise to the answer of a query before releasing its results. The amount of noise depends on the *sensitivity* of the function f, that is, how much the answer would change when an arbitrary tuple in the input data is modified.

Definition 2 (Sensitivity). *Given a function $f{:}\mathcal{D} \rightarrow \mathcal{R}^d$, its sensitivity is:*

$$S(f) = \max_{D \sim D'} ||f(D) - f(D')||_1.$$

where $||\cdot||_1$ is the L_1 norm; $D \sim D'$ means that the database differ in one tuple.

Note that $S(f)$ does not depend on the data but is a property of the function f. $S(f)$ captures the magnitude by which a single individuals data can change the function f in the worst case, and therefore, *the uncertainty in the response* that we must introduce in order to hide the participation of a single individual. The sensitivity of a function gives an upper bound on how much we must perturb its output to preserve privacy. Other notions of sensitivity like *smooth sensitivity* [17] guarantee (ϵ-δ)-DP by relating the amount of noise not only to the query result but also to the database.

The Laplace Mechanism. The Laplace Mechanism is based on the idea of computing the output of a function f, and then perturb each coordinate with noise drawn from the Laplace distribution. The scale of the noise will be calibrated to the sensitivity of f (i.e., $S(f)$) divided by ϵ, which represents the privacy budget.

Proposition 3 (Laplace Mechanism) *For a function $f{:}D \rightarrow \mathcal{R}^d$, the mechanism that returns $f(D)+\mathbf{z}$, where each $z_i \in \mathbf{z}$ is drawn from $Lap(S(f) \mid \epsilon)$ satisfies ϵ-differential privacy [7].*

The Gaussian Mechanism. The Gaussian mechanism reduces the probability of very large noise while guaranteeing (ϵ-δ)-DP.

Proposition 4 (Gaussian Mechanism) *For a function $f{:}D \rightarrow \mathcal{R}^d$, the mechanism that returns $f(D) + \mathbf{z}$, where each $z_i \in \mathbf{z}$ is drawn from $\mathcal{N}(0, S(f)^2\sigma^2)$[1] with $\sigma > exp(-(\sigma\epsilon)^2/2)/1.25$ and $\epsilon < 1$, satisfies (ϵ-δ)-DP [8].*

Non-numeric Queries. For non-numeric queries, achieving DP requires the definition of some utility function to sample one of its outcomes according to the probability distribution close to the optimum. This approach is referred to as the *exponential mechanism* [8].

[1] In this case the sensitivity makes usage of the L_2 norm.

2.4 Composing Differentially Private Mechanisms

So far we have discussed the general approach to guarantee DP when considering a single access to the data. Nevertheless, one usually needs to perform multiple application of DP-algorithms and thus accessing multiple times to the data with the risk of increasing privacy breaches. Investigating the impact of multiple data accesses requires to understand how DP mechanisms compose. In other words, when accessing databases multiple times via differentially private mechanisms, each of which having its own privacy guarantees, how much privacy is still guaranteed on the union of those outputs? [11]

In what follows we focus on (ϵ-δ)-DP and report the detail the main composition results by considering a scenario involving T=10K steps. To guarantee that each data access is (ϵ-δ)-DP the standard deviation in the Gaussian mechanism should be set as follows: $\sigma = \frac{\sqrt{log 1/\delta}}{\epsilon}$ [8].

Basic composition. This is the most simple way of investigating the impact of multiple access to data by the same mechanism. Basically, when one needs to perform T steps, the overall process is ($T\epsilon$, $T\delta$)-DP; in other words, the privacy parameters of the individual mechanisms basically sum up [8]. By assuming σ=4 and δ=10^{-5}, each step is (1.2, 10^{-5})-DP and thus after 10K steps the composition gives (12000, .1)-DP.

Advanced Composition. Advanced composition of (ϵ-δ)-DP mechanisms (see e.g., [9]) significantly improve the overall privacy cost. In this case we have that after T steps the process is ($\epsilon\sqrt{T log 1/\delta}$, $T\delta$)-DP. Thus after 10K steps the composition gives (360, .1)-DP.

Amplification by Sampling. The results of advanced composition is further improved by sampling [12]. By considering a batch size q, each step is (2qϵ,qδ)-DP. As an example if q is 1% of the data, with σ=4 and δ=10^{-5} we have that each step is (.024,10^{-5})-DP. When composing T step we have (2q$\epsilon\sqrt{T log 1/\delta}$, q$T\delta$)-DP and thus for T=10.000 the process is (10, .001)-DP.

Optimal Composition. Kairouz et al. [11] recently provide lower bounds for composition. It is previously known that it is sufficient to add Gaussian noise with variance $O(T\Delta^2 \log(1/\delta)/\epsilon^2)$ to each step in order to ensure (ϵ,δ)-DP after T steps. Authors show that this condition can be improved by a log factor. In particular, if each step is (ϵ_0,δ_0)-DP, then the composition of T steps gives $(T{\epsilon_0}^2 + \sqrt{2T\epsilon_0^2 log(e + \sqrt{T\epsilon_0^2}/\tilde{\delta})}, T\delta_0 + \tilde{\delta})$-DP assuming $\epsilon_0 \leq 0.9$ and a slack $\tilde{\delta} > 0$.

Accountant mechanism. A different approach to keep track of the privacy spent during the course of multiple access to sensitive data makes usage of an *accountant mechanism* [16]. This approach has been recently used by Abadi et al. [1] to train neural networks under DP. Authors leverage the moments of the noise distribution and a bound on the gradient value to obtain a more refined result about the composition than the advanced composition. In this case, the result after T steps and considering a batch size q becomes (2q$\epsilon\sqrt{T}$, δ)-DP. Thus, we have that fort T=10.000 steps and q=1% the overall process is (1.25, 10^{-5})-DP.

3 An Overview of the State of the Art

In this section we review applications of DP for Deep Neural Networks by distinguishing between *functional* and *non-functional* approaches.

Functional Approaches. The general idea of differential privacy is to introduce noise. However, it has been observed that it is not easy to adapt this approach in some tasks like logistic regression (e.g., [21]). Chaudhuri et al. [5] came up with a new way of achieving DP in these contexts: *instead of perturbing the results of the regression analysis, one can perturb the objective function and then optimize the perturbed objective function.* The overall idea is reported in Fig. 1 (a).

The original idea of Chauduri has been further refined into the Functional Mechanism (FM) [21]. The FM also perturbs the objective function of regression analysis and then release model parameters that minimize such perturbed function. The approach has been applied to both linear and logistic regression; in this latter case by resorting to a second order Taylor expansions of the error function. Authors show that the amount of noise to be injected in order to guarantee DP basically depends on the number of dimensions of the input dataset. The FM has been recently applied to train autoencoders under DP [18].

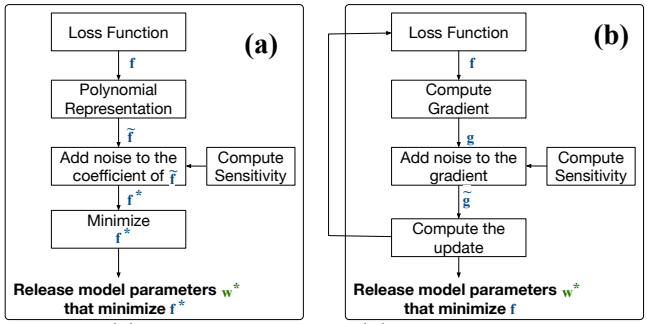

Fig. 1. Functional (a) and non-functional (b) approaches for learning with DP.

Non-Functional Approaches. Non-functional approaches determine the model parameters that minimize the original loss function instead of its perturbed and approximated version. In the context of neural networks, there are a few attempts of using DP; these approaches are mainly based on *line search methods* such as the (Stochastic) Gradient Descent (SGD). An overview of non-functional approaches is provided in Fig. 1 (b). The idea is to start with some initial values of the model parameters and then iteratively update these parameters following the opposite direction of the gradient, after adding noise. We showed in Section 2.4 how to determine the amount of noise in the Gaussian mechanism under different types of composition. One of the early approaches in learning via SGD and DP has been proposed by Rajkumar et al. [19]. This approach performs Gaussian perturbation of the overall multiparty objective to achieve (ϵ-δ)-DP. The goal is that multiparty want to compute a gradient based minimization of the objective function in a differentially-private way. Privacy is achieved by perturbing the gradient information with *two sources of noise*, the second being

needed to avoid an attacker to reconstruct the true minimizer (optimal weights) of the objective function. We already mentioned the work by Abadi et al. [1] that use a SGD-based algorithm and find a tighter bound on the composition of DP mechanism than previous approaches (e.g., [20]. As a side note, we shall also mention that the functional approach described by Phan et al.[18] to train deep autoencoders also resorts to the SGD to fine tune the parameters.

4 Neural Networks and Differential Privacy

In this section we describe our experience on applying the functional mechanism to the multilayer perceptron (see Section 2.1). Our algorithm involves the following four main phases: (i) find a polynomial representation of the objective function; (ii) compute the sensitivity; (iii) add noise to the polynomial representation; (iv) minimize the perturbed objective function.

Polynomial representation of the objective function. We consider the following 2nd-order Taylor expansion $\hat{E}(\mathcal{D}, \mathbf{w})$ of the error function $E(\mathcal{D}, \mathbf{w})=\sum_{\mathbf{x}\in\mathcal{D}}$.

$$\hat{E}(\mathcal{D}, \mathbf{w}) \approx E(\mathcal{D}, \hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^T \nabla E(\mathcal{D}, \hat{\mathbf{w}}) + \frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}})^T \nabla^2 E(\mathcal{D}, \hat{\mathbf{w}})(\mathbf{w} - \hat{\mathbf{w}})$$

where: $\nabla E(\mathcal{D}, \hat{\mathbf{w}}) \equiv \frac{\partial E}{\partial w_{i,j}}|_{\mathbf{w}=\hat{\mathbf{w}}}$ and $(\nabla^2 E(\mathcal{D}, \hat{\mathbf{w}}))_{i,j} \equiv \frac{\partial^2 E}{\partial w_i \partial w_j}|_{\mathbf{w}=\hat{\mathbf{w}}}$ are the Jacobian and Hessian matrices, respectively. Denoting by $g_{i,j}$ (resp., $h_{i,j}$) an element of the Jacobian (resp., Hessian) matrix, we obtain:

$$g_{i,j} = \sum_{\mathbf{x}\in\mathcal{D}} \delta_{\mathbf{x},i} z_{\mathbf{x},j} \tag{1}$$

$$\delta_{\mathbf{x},i} = z'_{\mathbf{x},i} \sum_{v:i\prec v} \delta_{\mathbf{x},v} w_{v,i} \tag{2}$$

$$h_{i,j} = \sum_{\mathbf{x}\in\mathcal{D}} b_{\mathbf{x},i} z^2_{\mathbf{x},j} \tag{3}$$

$$b_{\mathbf{x},i} = z''_{\mathbf{x},i} \sum_{v:i\prec v} w_{v,i}\delta_{\mathbf{x},v} + (z'_{\mathbf{x},i})^2 \sum_{v:i\prec v} w^2_{v,i} b_{\mathbf{x},v} \tag{4}$$

Equation (1) and Equation (3) are defined recursively by considering the values of δ and b of all units. Equation (2) and Equation (4) define such values for internal units. Since the number of model parameters (weights and biases) of the network can be very large computing the whole Hessian matrix is extremely demanding from a computational point of view (it requires a quadratic numbers operations). Hence, to reduce the computational cost we only consider diagonal elements of the Hessian [6]. As for the output units, their output depends from the specific loss function considered. As an example, when considering the Least Squares Error, we obtain:

$$E(\mathbf{x}, \mathbf{w}) = \frac{1}{2}\sum_{c=1}^{d}(y_{\mathbf{x},c} - t_{\mathbf{x},c})^2$$

$$y_{\mathbf{x},c} = y_c(\mathbf{x}; \mathbf{w})$$

where $y_{\mathbf{x},c}$ is the c-th component given by the network when giving as input the vector $\mathbf{x}$ and $t_{\mathbf{x},c}$ is the c-th target value. Hence, for output units we have:

$$\delta_{\mathbf{x},i} = (\psi(a_{\mathbf{x},i}) - t_{\mathbf{x},i})\psi'(a_{\mathbf{x},i}) \tag{5}$$

$$b_{\mathbf{x},i} = (\psi'(a_{\mathbf{x},i}))^2 + (\psi(a_{\mathbf{x},i}) - t_{\mathbf{x},i})\psi''(a_{\mathbf{x},i})) \tag{6}$$

Compute Sensitivity. In order to determine the amount of noise to be injected into the polynomial representation derived in the previous section we need to estimate its sensitivity. We have the following preliminary result.

Lemma 1. *Let $\mathcal{D}$, $\mathcal{D}'$ be any two databases differing in at most one tuple, and*

$$\hat{E}(\mathcal{D}, \mathbf{w}) = (\sum_{\mathbf{x}\in\mathcal{D}}\sum_{j\prec i}\delta_{\mathbf{x},i}z_{\mathbf{x},j})w_{i,j} + \frac{1}{2}(\sum_{\mathbf{x}\in\mathcal{D}}\sum_{j\prec i}b_{\mathbf{x},i}z^2_{\mathbf{x},j})w^2_{i,j}$$

$$\hat{E}(\mathcal{D}', \mathbf{w}) = (\sum_{\mathbf{x}'\in\mathcal{D}'}\sum_{j\prec i}\delta_{\mathbf{x}',i}z_{\mathbf{x}',j})w_{i,j} + \frac{1}{2}(\sum_{\mathbf{x}'\in\mathcal{D}'}\sum_{j\prec i}b_{\mathbf{x}',i}z^2_{\mathbf{x}',j})w^2_{i,j}$$

the polynomial representation of the error function on $\mathcal{D}$ and $\mathcal{D}'$, respectively. Let $\mathbf{x}$ be an arbitrary tuple. We have that the sensitivity is:

$$||\hat{E}(\mathcal{D}, \mathbf{w}) - \hat{E}(\mathcal{D}', \mathbf{w})||_1 \ \leq 2\sum_{j\prec i}\max_{\mathbf{x}}\left(|\delta_{\mathbf{x},i}z_{\mathbf{x},j}| + |b_{\mathbf{x},i}z^2_{\mathbf{x},j}|\right)$$

Adding noise. After determining the sensitivity, and thus the amount of noise, we shall now provide an algorithm for learning with DP. The algorithm is sketched in Algorithm 1. It start with an initialization of the weights (line 1) and then requires to obtain the Jacobian and (approximated) Hessian matrices representing the building blocks of the polynomial representation of the error function (line 2). At line 3 the noise is injected into such coefficients to obtain a perturbed error function, which is then minimized (according to one of the existing methods) in line 4. The final step consists in releasing the model parameters (the set of weights $\tilde{\mathbf{w}}$) that minimize it.

Theorem 5. *Algorithm 1 satisfies ϵ-differential privacy.*

Proof. Consider two neighbor datasets $\mathcal{D}$ and $\mathcal{D}'$ that differ on the last tuple, $\mathbf{x}_n$ and $\mathbf{x'}_n$ for $\mathcal{D}$ and $\mathcal{D}'$, respectively. The proof proceeds by applying the definition of differential privacy (see Definition 1).

$$\frac{Pr\{\widetilde{E}(\mathbf{w}) \mid \mathcal{D}\}}{Pr\{\widetilde{E}(\mathbf{w}) \mid \mathcal{D}'\}} = \frac{exp\left(\frac{\epsilon\left|\left|\sum_{\mathbf{x}\in\mathcal{D}}\sum_{j\prec i}\left(\delta_{\mathbf{x},i}z_{\mathbf{x},j}+b_{\mathbf{x},i}z^2_{\mathbf{x},j}\right)-(\delta_i z_j+b_i z_j^2)\right|\right|_1}{S(\widetilde{E})}\right)}{exp\left(\frac{\epsilon\left|\left|\sum_{\mathbf{x}'\in\mathcal{D}}\sum_{j\prec i}\left(\delta_{\mathbf{x}',i}z_{\mathbf{x}',j}+b_{\mathbf{x}',i}z^2_{\mathbf{x}',j}\right)-(\delta_i z_j+b_i z_j^2)\right|\right|_1}{S(\widetilde{E})}\right)}$$

$$\leq exp\Big(\frac{\epsilon}{S(\widetilde{E})}||\sum_{\mathbf{x}\in\mathcal{D}}\sum_{j\prec i}\left(\delta_{\mathbf{x},i}z_{\mathbf{x},j} + b_{\mathbf{x},i}z^2_{\mathbf{x},j}\right) - \sum_{\mathbf{x}'\in\mathcal{D}'}\sum_{j\prec i}\left(\delta_{\mathbf{x}',i}z_{\mathbf{x}',j} + b_{\mathbf{x}',i}z^2_{\mathbf{x}',j}\right)||_1\Big)$$

$$= exp\Big(\frac{\epsilon}{S(\widetilde{E})}||\sum_{j\prec i}\left(\delta_{\mathbf{x_n},i}z_{\mathbf{x_n},j} + b_{\mathbf{x_n},i}z^2_{\mathbf{x_n},j}\right) - \sum_{j\prec i}\left(\delta_{\mathbf{x'_n},i}z_{\mathbf{x'_n},j} + b_{\mathbf{x'_n},i}z^2_{\mathbf{x'_n},j}\right)||_1\Big)$$

$$\leq exp\Big(\frac{\epsilon}{S(\widetilde{E})}2\sum_{j\prec i}\max_{\mathbf{x}}\left(|\delta_{\mathbf{x},i}z_{\mathbf{x},j}| + |b_{\mathbf{x},i}z^2_{\mathbf{x},j}|\right)\Big) \text{(by Lemma 1)}$$

$$\leq exp(\epsilon)$$

Algorithm 1 `FunctionalNetDP` (Privacy budget ϵ)

1: Initialize $\mathbf{w}$
2: g,$h \leftarrow$ Polynomial representation of the loss function
3: Find $\widetilde{g}$ and $\widetilde{h}$ via `addNoise`($\mathbf{w}$,ϵ) /* Algorithm 2 */
4: Compute $\tilde{\mathbf{w}}$=$\underset{\mathbf{w}}{argmin}\ \widetilde{E}(\mathcal{D}, \mathbf{w})$
5: **return** $\tilde{\mathbf{w}}$ / set of weights that minimizes $\widetilde{E}(\mathcal{D}, \mathbf{w})$ /

Algorithm 2 `addNoise`(Weights $\mathbf{w}$, privacy budget ϵ)

1: Let $\mathcal{D}$ be the dataset
2: Let $S(\widetilde{E})$ be the sensitivity of the polynomial representation fo the network.
3: Find g via eq. (1) and h via eq. (3) using $\mathcal{D}$
4: g$\leftarrow g/(1, ||g||^2/C)$ /* clip Gradient */
5: h$\leftarrow h/(1, ||h||^2/C)$ /* clip Hessian */
6: $\widetilde{g} \leftarrow g$+$Lap(S(\widetilde{E}) \mid \epsilon)$
7: $\widetilde{h} \leftarrow h$+$Lap(S(\widetilde{E}) \mid \epsilon)$
8: **return** $\widetilde{g}$ and $\widetilde{h}$

4.1 Estimating the amount of Laplacian Noise

Algorithm 1 lays the foundation for achieving differential privacy. We have shown in Lemma 1 that the amount of noise depends on the maximum (over all tuples) sum of the coefficients of δ, z, and b. In turn, these terms depend on the choice of the activation functions for the intermediate units (i.e., ϕ), output units (i.e., ψ) and the value of weights (i.e., $w_{i,j}$). We now provide a finer grained estimation of the noise. Assuming that for both ϕ and ψ the logistic function $\sigma(x)$=$\frac{1}{1+e^{-x}}$ is used, then we have that $\sigma(x) \in [0, 1]$, $\sigma'(x) \in [0, 0.25]$, and $\sigma(x)'' \in [-0.1, 0.1]$.

Moreover, we assume that each $w_{i,j} \in [0, 1]$ and that for each dimension d of the input tuple $\mathbf{x^d} \in [0, 1]$. At this point, to bound the amount of Laplace noise needed, we need to bound the components $\sum_{j \prec i} |\delta_{\mathbf{x},i} z_{\mathbf{x},j}|$ and $\sum_{j \prec i} |b_{\mathbf{x},i} z^2_{\mathbf{x},j}|$ in Lemma 1. Let m be the depth of the network. For each layer $j \in \{1, ..., m\}$ we denote by δ^j (resp., b^j) the coefficient of a generic unit in the layer j. Moreover, s_j represents the number of units at layer j.

Lemma 2. *The noise that the $\sum_{j \prec i} |\delta_{\mathbf{x},i} z_{\mathbf{x},j}|$ component in Lemma 1 contributes is $\leq 0.25^m \times \prod_{j=1}^{j=m} s_j$.*

Proof. The idea is to analyze the layer-wise upperbound of δ and z when considering a generic tuple $\mathbf{x}$. As for z we always have $z \in [0, 1]$ (since we are using the sigmoid activation function). As for δ we have:

$$\delta \leq \begin{cases} 0.25 & \text{if } j{=}m \quad \text{(equation (5))} \\ 0.25^{m-j+1} \times \prod_{q=j}^{q=m} s_q & \text{j}{\neq}\text{m} \quad \text{(equation (2))} \end{cases}$$

The maximum contribution occurs when j=1 in $\prod_{q=j}^{q=m} s_q$.

Lemma 3. *The contribution of the* $\sum_{j \prec i} |b_{\mathbf{x},i} z^2_{\mathbf{x},j}|$ *component in Lemma 1 is:*

$$\leq \prod_{j=1}^{j=m} s_j \Big(0.1^m + 0.001625 \times m \Big)$$

Proof. Similarly to the previous case, we have that:

$$b \leq \begin{cases} 0.1625 & \text{if } j{=}m \quad \text{(equation (6))} \\ 0.1^{m-j+1} \times \prod_{q=j}^{q=m} s_q + & \\ +(0.1)^2(m-j)0.1625 \prod_{q=j}^{q=m} s_q & \text{j}{\neq}\text{m} \quad \text{(equation (4))} \end{cases}$$

The contribution is maximum when $j = 1$

Theorem 6. *For a network of* m *layers with* s_j *units in each layer, where* $j \in [1, n]$*, the amount of Laplacian noise to ensure differential privacy is:*

$$\leq 2 \Big(0.25^m \prod_{q=1}^{q=m} s_q + \prod_{j=1}^{j=m} s_j \times \Big(0.1^m + 0.001625 \times m \Big) \Big)$$

Proof. The idea is to substitute the bounds in the previous lemmas in Lemma 1.

The above analysis shows that the amount of noise to ensure differential privacy is dominated by the number of units in the network.

5 Concluding Remarks and Future Work

We have reported on the usage of differential privacy in neural networks and discussed our preliminary findings in adding Laplacian noise to a low-polynomial representation of the error function. We now discuss some crucial aspects of our approach.

First, we have shown via Lemma 1 that the sensitivity basically depends on the topology of the network. The result about the sensitivity when using the functional mechanism for neural networks is in stark contrast with functional approaches tackling regression (e.g., [21]); in that case the sensitivity was bound by the dimensions of the input data. This comes as no surprise because of the fact that our approach involves a complex topology of interconnected network units (neurons) while regression only makes usage of a single unit, in a sense. To mitigate the impact of sensitivity we bound in Algorithm 2 the values of the gradient and Hessian (lines 5 and 6) before adding noise. Finding a tighter bound for Lemma 1 is in our research agenda.

Second, we pursued the simplest way to ensure DP by adding Laplacian noise to the gradient and the Hessian coefficients (the coefficients of our polynomial representation), and use the noisy version to perform the minimization (line 5 Algorithm 1). We note that the sensitivity in our case is irrelevant to the size of the data set at hand. At the same time we note that the potential effect of

the noise can have different effects on the gradient and Hessian, respectively. In this latter case, even a little amount of noise in the coefficients of the Hessian matrix can lead to large changes on the parameters being updated, with the extreme potential consequence that the Hessian matrix's positive definiteness is destroyed, which implies that a global optimal solution may not be attained at all. One way to approach the problem would be to threshold the eigenvalues of the Hessian to ensure positive definiteness; nevertheless this method might generate coefficients that are far from the original (pre-noise) coefficients. Another approach could be to leverage public data sets for the Hessian, that is, datasets where participants voluntarily disclosed information and we use public and private data sets to compute the gradient.

Third, Algorithm 1 works by approximating the Hessian to its diagonal components as done by other approaches (e.g., [14]). Nevertheless, in some cases the approximation can be very rough leading to poor solutions. This problem may be exacerbated by the noise addition. One way to overcome this problem could be to resort to non-functional approaches.

In this context, an interesting aspect that has not yet been investigated in the context of DP is the usage of 2nd order methods like the Hessian Free (HF) technique [15]. The idea of HF is to approximate the error function around each parameter up to second order (SGD-based methods only take into account the first derivatives of the parameters). Operationally, the HF approach requires gradient estimations, approximations of the Hessian (e.g., Gauss-Newton approximation) and its minimization. This latter step is usually done by using the conjugate gradient (CG). The CG applies minimization along each dimension of the parameter space separately and thus would require a number of iterations equals to the number of dimensions of the parameter space to converge in general. However, it usually makes significant progresses toward a minimum in a much lower number of iterations [15]. We see a number of research challenges for the usage of DP in this setting, among which: (i) investigating the amount of noise needed along each iteration of the HF and the properties of composition; and (ii) comparing this approach with first-order methods (e.g., [1]).

Fourth, in this preliminary analysis we have discussed privacy without considering utility. Our ongoing experimental evaluation shows that while our approach achieves strong privacy guarantees, when the size of the network grows[2] the utility is heavily affected. To mitigate this issue, we are evaluating the impact of privacy on utility when considering the HF technique.

References

1. M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proc. of CCS*, pages 308–318. ACM, 2016.
2. C. C. Aggarwal and S. Y. Philip, editors. *Privacy-Preserving Data Mining*. Springer, 2008.

[2] Anecdotally, when considering more than 4 layers with hundreds of units per layer.

3. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.
4. C. M. Bishop. Pattern recognition. *Machine Learning*, 2006.
5. K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pages 289–296, 2009.
6. Y. L. Cun. *Modeles Connexionistes de l'Apprentissage*. PhD thesis, Universite' de Paris, 1987.
7. C. Dwork. Differential privacy. In *ICALP*, pages 1–12. Springer, 2006.
8. C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.
9. C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *Proc. of FOCS*, pages 51–60. IEEE, 2010.
10. G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
11. P. Kairouz, S. Oh, and P. Viswanath. The Composition Theorem for Differential Privacy. In *ICML*, pages 1376–1385, 2015.
12. S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.
13. H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *JMLR*, 10:1–40, 2009.
14. Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.
15. J. Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, 2010.
16. F. D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. of SIGMOD*, pages 19–30. ACM, 2009.
17. K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proc. of STOC*, pages 75–84. ACM, 2007.
18. N. Phan, Y. Wang, X. Wu, and D. Dou. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI*, pages 12–17, 2016.
19. A. Rajkumar and S. Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *International Conference on Artificial Intelligence and Statistics*, pages 933–941, 2012.
20. S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pages 245–248. IEEE, 2013.
21. J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: regression analysis under differential privacy. *Proceedings of the VLDB Endowment*, 5(11):1364–1375, 2012.