DP-MAC: The Differentially Private Method of Auxiliary Coordinates for Deep Learning

Frederik Harder[1] **Jonas Köhler**[2] **Max Welling**[3] **Mijung Park**[4]

Abstract

Developing a differentially private deep learning algorithm is challenging, due to the difficulty in analyzing the *sensitivity* of objective functions that are typically used to train deep neural networks. Many existing methods resort to the stochastic gradient descent algorithm and apply a *pre-defined* sensitivity to the gradients for privatizing weights. However, their slow convergence typically yields a high cumulative privacy loss. Here, we take a different route by employing the *method of auxiliary coordinates*, which allows us to independently update the weights per layer by optimizing a *per-layer* objective function. This objective function can be well approximated by a low-order Taylor's expansion, in which sensitivity analysis becomes tractable. We perturb the coefficients of the expansion for privacy, which we optimize using more advanced optimization routines than SGD for faster convergence. We empirically show that our algorithm provides a decent trained model quality under a modest privacy budget.[1]

1 Introduction

While providing outstanding performance, it has been shown that trained deep neural networks (DNNs) can expose sensitive information from the dataset they were trained on [1, 2, 3, 4]. In order to protect potentially sensitive training data, many existing methods adopt the notion of privacy, called *differential privacy* (DP) [5]. Differentially private algorithms often comprise a noise injection step (e.g. during the training process), which is generally detrimental to performance and leads to a trade-off between privacy and utility. The amount of noise necessary for a desired level of privacy depends on the *sensitivity* of an algorithm, a maximum difference in its output depending on whether or not a single individual participates in the data. In DNNs, the sensitivity of an objective function is often intractable to quantify, since data appears in the function in a nested and complex way. In addition, such models have thousands to millions of parameters, one needs to saveguard, and require many passes over the dataset in training. As a result, providing meaningful privacy guarantees while maintaining reasonable performance remains a challenging task for DNNs.

One existing approach to this problem, DP-SGD [6, 7, 8], avoids complicated sensitivities, by applying a pre-defined sensitivity to the gradients, which are then perturbed with Gaussian noise before updating the weights to ensure DP. This work also introduces the *moments accountant* (MA) [6], a useful method for computing cumulative privacy loss when training for many epochs (a formal introduction to this method is found in Appendix A). In another line of recent work [9, 10, 11], DP training is achieved by approximating the nested objective function through Taylor approximation and perturbing each of the coefficients of the approximated loss before training.

1,2,4: Max Planck Institute for Intelligent Systems, 3: University of Amsterdam

[1]We updated this current manuscript by fixing an implementation error, which was part of the implementation to produce the results we presented at the PPML2018 workshop. For detailed comments on what changes we made, see Appendix.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.
Workshop on Privacy Preserving Machine Learning

In this paper, we combine the benefits of these two approaches. We modify the algorithm called the *method of auxiliary coordinates* (MAC), which allows independent weight updates per layer, by framing the interaction between layers as a local communication problem via introducing auxiliary coordinates [12]. This allows us to split the nested objective function into per-layer objective functions, which can be approximated by low-order Taylor's expansions. In this case the sensitivity analysis of the coefficients becomes tractable.

2 DP-MAC

2.1 The Method of Auxiliary Coordinates

Here we provide a short introduction on the MAC algorithm (see [12] for details). Under a fully connected neural net with K hidden layers, a typical mean squared error (MSE) objective is given by

$$E(\mathbf{W}) = \tfrac{1}{2N} \sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n; \mathbf{W})||^2, \tag{1}$$

where $\mathbf{f}(\mathbf{x}_n; \mathbf{W}) = \mathbf{f}_{K+1}(\ldots \mathbf{f}_2(\mathbf{f}_1(\mathbf{x}_n; \mathbf{W}_1) \ldots); \mathbf{W}_{K+1})$. We denote $\mathbf{W}$ as a collection of weight matrices of $(K+1)$-layers, i.e., $\mathbf{W} = \{\mathbf{W}_k\}_{k=1}^{K+1}$, where the size of each weight matrix is given by $\mathbf{W}_k \in \mathbb{R}^{D_{in}^k \times D_{out}^k}$. Each layer activation function is given by $\mathbf{f}_k(\mathbf{x}_n; \mathbf{W}_k) = \mathbf{f}_k(\mathbf{W}_k{}^\top \mathbf{x}_n)$, and $\mathbf{f}_k$ could be any type of element-wise activation functions. In the MAC framework [12], the objective function in eq. 1 is expanded by adding auxiliary variables $\{\mathbf{z}_n\}$ (one per datapoint) such that the optimization over many variables are decoupled:

$$E(\mathbf{W}, \mathbf{Z}; \mu) = E_o(\mathbf{W}, \mathbf{Z}) + \sum_{k=1}^{K} E_k(\mathbf{W}, \mathbf{Z}, \mu), \tag{2}$$

where the partial objective functions at the output layer and at the k-th layer are given by $E_o(\mathbf{W}_{K+1}, \mathbf{Z}_K) = \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{f}_{K+1}(\mathbf{z}_{K,n}; \mathbf{W}_{K+1})||^2$, and $E_k(\mathbf{W}_k, \mathbf{Z}_k, \mathbf{Z}_{k-1}, \mu) = \frac{\mu}{2N} \sum_{n=1}^{N} ||\mathbf{z}_{k,n} - \mathbf{f}_k(\mathbf{z}_{k-1,n}; \mathbf{W}_k)||^2$. Alternating optimization of this objective function w.r.t. $\mathbf{W}$ and $\mathbf{Z}$ minimizes the objective function. In this paper, we set $\mu = 1$, as suggested in [12]. For obtaining differentially private estimates of $\mathbf{W}$, it turns out we need to privatize the $\mathbf{W}$ update steps only, while we can keep the $\mathbf{Z}$ update steps non-private, as has been studied in Expectation Maximization (EM) type algorithms before [13, 14].

To make this process DP, we first approximate each objective function as 1st or 2nd-order polynomials in the weights. Then, we perturb each approximate objective function by adding noise to the coefficients and optimize it for estimating $\mathbf{W}$. How much noise we need to add to these coefficients depends on the sensitivity of these coefficients as well as the privacy loss we allow in each training step. The final estimate $\mathbf{W}_T$ depends on estimates $\mathbf{W}_t$ and $\mathbf{Z}_t$ for all $t < T$ and so we keep the privacy loss per iteration fixed and compute the cumulative loss using the moments accountant.

2.2 DP-approximate to the per-layer objective function

First, we consider approximating the per-layer objective function via the 2nd-order Taylor expansion

$$E_k(\mathbf{W}_k) = \tfrac{1}{2N} \sum_{n=1}^{N} ||\mathbf{z}_{k,n} - \mathbf{f}_k(\mathbf{z}_{k-1,n}; \mathbf{W}_k)||^2 \approx \tfrac{1}{2N} \left[a_k + \sum_{h=1}^{D_{out}^k} \mathbf{w}_{kh}{}^\top \mathbf{b}_{kh} + \sum_{h=1}^{D_{out}^k} \mathbf{w}_{kh}{}^\top \mathbf{C}_{kh} \mathbf{w}_{kh} \right],$$

where $\mathbf{w}_{kh} \in \mathbb{R}^{D_{in}^k}$ is the h-th column of the matrix $\mathbf{W}_k$, and the derivation of each term $a_k, \mathbf{b}_{kh} \in \mathbb{R}^{D_{in}^k}, \mathbf{C}_{kh} \in \mathbb{R}^{D_{in}^k \times D_{in}^k}$ is given below. Here we choose to use the *softplus* function as an example activation function for f, but any twice differentiable function is valid. We introduce a new notation $T_n(\mathbf{w}_{kh})$

$$E_k(\mathbf{W}_k) = \tfrac{1}{2N} \sum_{n=1}^{N} \sum_{h=1}^{D_{out}^k} T_n(\mathbf{w}_{kh}) \tag{3}$$

where $T_n(\mathbf{w}_{kh}) = z_{kh,n}^2 - 2 z_{kh,n} f(\mathbf{w}_{kh}{}^\top \mathbf{z}_{k-1,n}) + \{f(\mathbf{w}_{kh}{}^\top \mathbf{z}_{k-1,n})\}^2$. We then approximate $T_n(\mathbf{w}_{kh})$ by the 2nd-order Taylor expansion evaluated at $\hat{\mathbf{w}}_{kh}$. In the first optimization step, we approximate the loss function by the 2nd-order Taylor expansion evaluated at a randomly drawn $\hat{\mathbf{w}}_{kh} \sim \mathcal{N}(0, I)$. In the consecutive optimization step, we evaluate the loss function at the noised-up estimate $\hat{\mathbf{w}}_{kh}$ obtained from the previous optimization step.

$$T_n(\mathbf{w}_{kh}) \approx T_n(\hat{\mathbf{w}}_{kh}) + (\mathbf{w}_{kh} - \hat{\mathbf{w}}_{kh})^\top \partial T_{nkh} + \tfrac{1}{2} (\mathbf{w}_{kh} - \hat{\mathbf{w}}_{kh})^\top \partial^2 T_{nkh} (\mathbf{w}_{kh} - \hat{\mathbf{w}}_{kh}), \tag{4}$$

where the derivative expressions of $T_n(\mathbf{w}_{kh})$ are given by

$$\partial T_{nkh} =[-2z_{kh,n}f'(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n}) + 2f(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n})f'(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n})]\mathbf{z}_{k-1,n},$$

$$\partial^2 T_{nkh} =[-2z_{kh,n}f''(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n}) + 2f(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n})f''(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n}) + 2\{f'(\hat{\mathbf{w}}_{kh}{}^\top \mathbf{z}_{k-1,n})\}^2]\mathbf{z}_{k-1,n}\mathbf{z}_{k-1,n}{}^\top .$$

From this, we define the coefficients $a_k, \mathbf{b}_{kh}, \mathbf{C}_{kh}$ as: $a_k = \sum_{n=1}^{N}\sum_{h=1}^{D^k_{out}}[T_n(\hat{\mathbf{w}}_{kh}) - \hat{\mathbf{w}}_{kh}{}^\top \partial T_{nkh} + \frac{1}{2}\hat{\mathbf{w}}_{kh}{}^\top \partial^2 T_{nkh}\hat{\mathbf{w}}_{kh}]$, $\mathbf{b}_{kh} = \sum_{n=1}^{N}\left[\partial T_{nkh} - \partial^2 T_{nkh}\hat{\mathbf{w}}_{kh}\right]$, $\mathbf{C}_{kh} = \sum_{n=1}^{N}\frac{1}{2}\partial^2 T_{nkh}$.

Adding Gaussian noise to these coefficients for privacy modifies the objective function by

$$\tilde{E}_k(\mathbf{w}_k) \approx \tfrac{1}{2N}\left[\tilde{a}_k + \sum_{h=1}^{D^k_{out}} \mathbf{w}_{kh}{}^\top \tilde{\mathbf{b}}_{kh} + \sum_{h=1}^{D^k_{out}} \mathbf{w}_{kh}{}^\top \tilde{\mathbf{C}}_{kh}\mathbf{w}_{kh}\right], \tag{5}$$

where $\tilde{a}_k = a_k + \mathcal{N}(0, (\Delta a_k)^2\sigma^2)$, $\tilde{\mathbf{b}}_k = \mathbf{b}_k + \mathcal{N}(0, (\Delta \mathbf{b}_k)^2\sigma^2\mathbf{I})$, $\tilde{\mathbf{C}}_k = \mathbf{C}_k + \mathcal{N}(0, (\Delta \mathbf{C}_k)^2\sigma^2\mathbf{I})$ and the amount of additive Gaussian noise depends on the sensitivity $(\Delta a_k, \Delta \mathbf{b}_k, \Delta \mathbf{C}_k)$ of each term. When using a purely gradient-based optimization routine (e.g. Adam, unlike Conjugate Gradient), we don't have to perturb a_k and in the case of first order approximation $\mathbf{C}_k$ is omitted as well, leaving only $\mathbf{b}_k$. This method is not limited to MSE objectives and in the classification task we use a binary cross-entropy objective analogously in the output layer.

Note that on the first $\mathbf{W}$ step, unperturbed 1st and 2nd-order approximations provide the same gradient. In this case, if we use vanilla SGD to optimize this first order approximation, this boils down to a variant of DP-SGD, which optimizes each layer objective function separately.

Analytic Sensitivities of the coefficients are given in the appendix. Depending on the architecture of a neural network and dataset at hand, these analytic sensitivity bounds can often be loose, in which case we propose to take a more direct approach and bound the sensitivities directly by clipping the norms of the coefficients $\|\partial T_{nk}\|_F \leq \Theta_{\partial T}$ and $\|\partial^2 T_{nk}\|_F \leq \Theta_{\partial^2 T}$. Here, ∂T_{nk} denotes the matrix of ∂T_{nkh} vectors, which are used to compute $\Delta \mathbf{b}_k$ and $\Delta \mathbf{C}_k$. We found that this yields significantly lower bounds in practice and use these clipping thresholds, along with linear Taylor expansion for all experiments listed below.

2.3 Calculation of cumulative privacy loss

Using the moments accountant and the theorem for subsampled Gaussian mechanism given in [6] for composition requires caution, since the log moments of privacy loss are linearly growing, only if we draw fresh noise per new subsampled data. Up to this point our algorithm, unfortunately, draws many noises for many losses given a subsampled data. This is fixed by treating the vectorized coefficients of the perturbed layerwise objectives as a single vector quantity $[a_1, \mathbf{b}_1, \mathbf{C}_1, \cdots, a_K, \mathbf{b}_K, \mathbf{C}_K,]$ which is perturbed in one step. As previously done in [8], we scale down each objective function coefficient by its own sensitivity times the number of partitions $\sqrt{MK}$, where $m = 3$ if $a_k, \mathbf{b}_k, \mathbf{C}_k$ are used to compute the loss (and $m = 1$ if only $\mathbf{b}_k$ is used). This sets the concatenated vector's sensitivity to 1. Then, we add the standard Gaussian noise with standard deviation σ to the vectors and scale up each perturbed quantities by their own sensitivity times $\sqrt{MK}$. Partitioning the vector in this way allows us to effectively consider the sensitivity and clipping bounds in each layer independently. Details are given in appendix I. The DP-MAC algorithm is summarized in Algorithm 1.

Algorithm 1 DP-MAC algorithm

```
Require: Dataset $\mathcal{D}$, total number of iterations $T$, privacy parameter $\sigma^2$, sampling rate $q$
Ensure: $(\varepsilon, \delta)$-DP weights $\{\mathbf{W}_k\}_{k=1}^{K+1}$
  for number of Iterations $\leq T$ do
    1. Optimize eq. 2 for $\mathbf{Z}$
    2. Optimize eq. 5 (noised-up objective) for $\mathbf{W}$
  end for
  Calculate the total privacy loss $(\varepsilon, \delta)$ using moments accountant
```

3 Experiments

Autoencoder We examine the performance of DP-MAC, when training deeper models, in a reconstruction task with a fully connected autoencoder with 6 layers, as used in the original MAC paper [12]. Unlike the original paper, we don't store any $\mathbf{z}$ values but initialized them with a forward pass on each iteration. For this, we use the USPS dataset is a collection of 16x16 pixel grayscale handwritten digits, of which we use 5000 samples

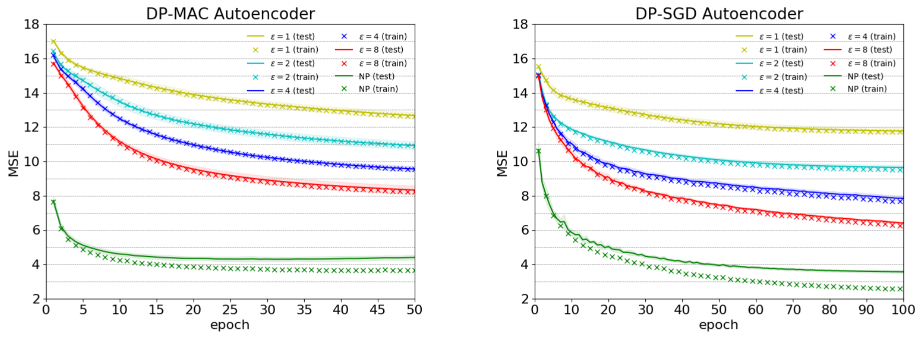

Figure 1: Autoencoder training and test errors. (± 1 stdev. of the latter) averaged over 10 runs each.

for training and 5000 for testing. We provide results for ε values of $1, 2, 4, 8$ with δ =1e-5. For comparison, we train the same model with DP-SGD where we use a single norm bound Θ_g for the full step-wise gradient of the model.

In Fig 1, we observe that DP-MAC lacks behind DP-SGD in both private and non-private settings. We suspect that this is in part owed to vanishing gradient updates in the MAC model. We found that independent of the number of Z updates per iteration, gradient updates in the first and last layer of the model differ by up to 4 orders of magnitude. DP-SGD does not exhibit this problem.

Classifier In addition, we show the performance of our method compared to other existing methods on a classification task on the MNIST digit dataset. We train a classifier with a single hidden layer of 300 units using DP-MAC with noise levels $\sigma = [1.0, 2.8, 8.0]$, which guarantees DP with $\delta = 10^{-5}$ and $\varepsilon = [8, 2, 0.5]$. As in [6], we use a DP-PCA to reduce input dimensionality to 60. Table 2 shows the comparison between our method and the DP-SGD results by [6] as well as the DP convolutional deep belief networks (DP-CDBN) by [11]. Here, our method achieves a comparable test accuracy under the same privacy constraint within a relatively small number of training epochs. Our implementation of both experiments is available at `https://github.com/mijungi/dp_mac`.

	DP-SGD	DP-CDBN	**DP-MAC**
$\varepsilon = 0.5$	0.90	0.92	0.90
# epochs	16	162	10
$\varepsilon = 2$	0.95	0.95	0.95
# epochs	120	162	30
$\varepsilon = 8$	0.97		0.97
# epochs	700		30

Figure 2: Test performance of DP-MAC compared to [6] DP-SGD and DP-CDBN [11] at $\delta = 10^{-5}$

4 Conclusion

We present a novel differentially private deep learning paradigm, DP-MAC, which allows us to compute the sensitivity of the approximate objective functions analytically. Empirically however, we find that directly setting clipping bounds yields significantly lower sensitivities, which leads us to gradient perturbation as a special case. We found that MAC in its current state exhibits vanishing gradient problems in scaling to deeper models, which we believe causes the decrease in test performance compared to regular DP-SGD when training deeper models. Nonetheless, we believe that this work offers an interesting new perspective on the possibilities computing sensitivities in deep neural networks.

References

[1] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232, 2018.

[2] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 587–601, New York, NY, USA, 2017. ACM.

[3] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 909–910, Sept 2015.

[4] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, New York, NY, USA, 2015. ACM.

[5] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, August 2014.

[6] M. Abadi, A. Chu, I. Goodfellow, H. Brendan McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. *ArXiv e-prints*, July 2016.

[7] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2017.

[8] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models without losing accuracy. *CoRR*, abs/1710.06963, 2017.

[9] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional Mechanism: Regression Analysis under Differential Privacy. *ArXiv e-prints*, August 2012.

[10] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction, 2016.

[11] N. Phan, X. Wu, and D. Dou. Preserving Differential Privacy in Convolutional Deep Belief Networks. *ArXiv e-prints*, June 2017.

[12] Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 10–19, Reykjavik, Iceland, 22–25 Apr 2014. PMLR.

[13] Mijung Park, James Foulds, Kamalika Choudhary, and Max Welling. DP-EM: Differentially Private Expectation Maximization. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 896–904, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[14] Mijung Park, James Foulds, Kamalika Chaudhuri, and Max Welling. Variational Bayes In Private Settings (VIPS). *ArXiv e-prints*, November 2016.

[15] Anand D. Sarwate and Kamalika Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE Signal Process. Mag.*, 30(5):86–94, 2013.