AN EFFICIENT DP-SGD MECHANISM FOR LARGE SCALE NLU MODELS

Christophe Dupuy[1], *Radhika Arava*[1], *Rahul Gupta*[1], *Anna Rumshisky*[1,2]

[1]Amazon Alexa AI, Cambridge, MA, USA
[2]University of Massachusetts, Lowell, MA, USA

ABSTRACT

Recent advances in deep learning have drastically improved performance on many Natural Language Understanding (NLU) tasks. However, the data used to train NLU models may contain private information such as addresses or phone numbers, particularly when drawn from human subjects. It is desirable that underlying models do not expose private information contained in the training data. Differentially Private Stochastic Gradient Descent (DP-SGD) has been proposed as a mechanism to build privacy-preserving models. However, DP-SGD can be prohibitively slow to train. In this work, we propose a more efficient DP-SGD for training using a GPU infrastructure and apply it to fine-tuning models based on LSTM and transformer architectures. We report faster training times, alongside accuracy, theoretical privacy guarantees and success of Membership inference attacks for our models and observe that fine-tuning with proposed variant of DP-SGD can yield competitive models without significant degradation in training time and improvement in privacy protection. We also make observations such as looser theoretical ϵ, δ can translate into significant practical privacy gains.

Index Terms— Differential privacy, membership inference attack

1. INTRODUCTION

Large scale NLP models have contributed significantly to the success of commercial voice assistants like Amazon Alexa, Google Assistant and Siri. They have shown high generalization accuracies for various learning tasks, from question answering [1] to named entity recognition [2]. However, large NLP models can be prone to privacy attacks (e.g. MIA: Membership Inference Attacks [3]) and can leak data used to train these models. In this paper, we focus on measuring and mitigating the privacy risks of these models. Specifically, differentially private (DP) model building algorithms have shown promise in providing defense against privacy attacks [4, 5]. We focus on a specific central differential private mechanism - Differentially Private Stochastic Gradient Descent (DP-SGD) and evaluate its impact on model utility and privacy.

DP-SGD [6] is an extension over the popular stochastic gradient descent algorithm that offers theoretical (ϵ, δ) privacy guarantees [7]. In our work, we make extension to proposals by [8] and [9] and, propose *efficient DP-SGD* (eDP-SGD) suited for GPU training. Specifically, we utilize the noise addition on gradients over data batches proposed by [8] to compute gradients per GPU for enhanced efficiency and clip parameters for each layer of the NLU models used in this study. Inspired from [9], we apply noise decay on gradients computed on each GPU. We apply the combination of these techniques to NLU model fine-tuning (as opposed to full training) on dataset of various sizes. For privacy evaluation, we study the impact of our extension of DP-SGD on MIA success. Previous work on the effects of DP on MIA performance study the two extreme cases where either 1) the DP-model is resilient to MIA but suffers significant degradation in performance [10]; or 2) the DP-model achieves similar performance to the non-private model but does not offer significant gains in privacy [11]. We observe that applying our method with looser theoretical DP guarantees translate into significant reduction in MIA performance. We also report the impact of using DP-SGD and our extension on the training time which, to the best of our knowledge, no prior work has done. We observe that DP-SGD can be prohibitively slower (up to 150 times) than non-private baselines while our method, in the worst case, is slower by a factor of 2.

In this paper, we make the following contributions: (i) We study the impact of applying DP techniques for NLP models trained on large-scale datasets. We present a computationally efficient setting for DP-SGD and provide a comparison in terms of training time, accuracy and privacy of the non-private and DP models. No other study has done this comparison, let alone for large-scale NLP setting. (ii) We demonstrate that using DP-SGD during fine-tuning, one can obtain models with competitive utility (in comparison to models trained with vanilla SGD), while achieving significant gains in protection against privacy attacks and, (iii) Building on existing DP-SGD variants, we propose an extended version of the DP-SGD technique which is computationally-efficient and report significant compute gains over DP-SGD.

2. RELATED WORK

DP-SGD [6] modifies vanilla SGD by clipping the gradients computed over each individual datapoint, followed by accu-

mulation of the clipped gradients over a batch and noise addition. Researchers have applied DP-SGD to reduce memorization in language models [12], data generation [13] and image classification [10]. [14] aim to understand the properties of DP-SGD. Attempts have been made to improve efficiency of DP-SGD by improving communication protocols in a distributed training setting, where individual servers contribute gradient on locally stored data and privacy of data in each local server is desirable [15]. Adaptive variants of DP-SGD in a federated setting have also been proposed [16].

MIA has been studied in a variety of settings, such as MIA with synthetic or noisy data [3], shallow models [17], non-matching shadow and target data sets [18]. DP-SGD while carrying theoretical privacy guarantees has also been demonstrated to provide defense against MIA [10]. In this work, we report both theoretical and MIA based privacy quantifications.

3. EFFICIENT DP-SGD

In this section, we present the eDP-SGD algorithm by modifying the following three techniques and, adapting them for a GPU based training. Given the high risks associated with NLP models revealing private information, and the growing concern over these risks, as well as the scarcity of studies in the domain of privacy-preserving algorithms for NLP, we have chosen some sensitive and proprietary datasets of a voice assistant for our studies. We propose the following modifications to the existing DP-SGD technique.

Micro-batch computations per GPU DP-SGD requires clipping the gradient of every single example in the batch. This induces a significant computational cost since given a batch size B, DP-SGD would require the computation of B gradients, as opposed to computation of one gradient in classic SGD. [19] show that it is possible to group examples in *micro-batches* in the DP-SGD scheme and still maintain DP-guarantees for the resulting model. This manipulation is equivalent to a global Gaussian mechanism and the authors provide the equivalence relationship in their paper. We leverage this work and apply DP-SGD computations to the micro-batch contained within each GPU. Given a batch of data points and n GPUs, we divide the batch into n micro-batches, processed independently on each GPU.

We make another addition to the algorithm suggested by [19], and add scaled noise to gradient computed per micro-batch within the GPU (as opposed to adding noise post the aggregation of gradients from all micro-batches). Given n GPUs, adding a Gaussian noise $\mathcal{N}(0, \sigma^2)$ to gradients per micro-batch is equivalent to adding a noise $\mathcal{N}(0, (n\sigma)^2)$ to the gradients aggregated over the micro-batch. This change relaxes the need for aggregation before noise addition and accelerates computation.

Scaling For large scale models, the magnitude of the gradient varies across parameters in the model. For instance, the magnitude of gradients for parameters in the lower layers can be different compared to those in the upper layers. Hence, using a constant clipping parameter C can either be too aggressive or too weak for a certain set of parameters. In this case, given a set of parameters w^k (e.g. those drawn from the k^{th} layer in a neural network), a strategy that clips gradients with a parameter (C_k) specific to the set w^k is preferable. However, this strategy may again lead to poor privacy guarantees for large variations in C_k assigned to each set of parameters. Inspired by the scaling approach suggested by [19], we compute a scaling factor for each layer, proportional to the norm of gradient calculated on the first iteration for each layer. The scaling is applied to gradient computed over a micro-batch contained in each GPU. Since the model parameters are randomly initialized and that the gradient norm is expected to decrease during training, the norm of the gradient in the first iteration gives a rough estimation of the upper bound of the gradient magnitude throughout training. We scale a constant clipping value C by the factor α_k for each layer. This strategy also reduces the number of hyper-parameters to tune for every new model or dataset.

Noise Decay In DP-SGD, the amount of noise added is the same for all the training iterations with a variance equal to the clipping parameter C times the noise multiplier z, used to compute theoretical DP guarantees. As the magnitude of the gradients is likely to decrease when approaching convergence, adding noise with constant variance can lead to slower convergence [9]; as the magnitude of the noise can be significantly higher than the magnitude of the gradient. In addition, a noise with high variance can wash out the information contained in the gradients after a few epochs, while a noise with low variance would yield low privacy guarantees. To improve convergence, [9] use noise variance reduction at every epoch by scaling the initial noise multiplier z_0 by a decreasing function d_τ (parameterized by τ). In our work, we use this strategy, however, it is applied to gradients computed independently at each GPU. We decay the noise strength as a function of epoch number t using one of the following forms of the multiplier d_τ:

1. Linear decay: $d_\tau = 1/(1 + \tau t)$
2. Exponential decay: $d_\tau = e^{-\tau t}$

Algorithm 1 summarizes eDP-SGD.

4. EXPERIMENTAL SETUP

We focus on Intent Classification (IC) and Named-Entity Recognition (NER) tasks in this work as they are popular in industrial NLP systems [20]. We next describe the datasets used in our experiments.

4.1. Datasets

We use three publicly available datasets - ATIS [21], SNIPS [22] and NLU-EVAL [23] and three additional datasets from

Algorithm 1 eDP-SGD

Require: **GPU Devices:** $\mathrm{GPU}_1, \ldots, \mathrm{GPU}_N$;
Data: Batch $B = (x_1, \ldots, x_{|B|})$; $|B| > N$.
DP-SGD Input: Noise multiplier z_0; Clipping coefficient C; Noise decay $d : \mathbb{N} \mapsto \mathbb{R}_+^*$; Scaling $(\alpha_k)_k$;
Model Input: Loss $L(\cdot, w)$; Epoch t;
Ensure: DP-gradient for optimizer
for (M, GPU) in $\{(M_i, \mathrm{GPU}_i)\}_{i=1,\ldots,N}$ **do**
Send micro-batch M to GPU
Compute gradient:
$\Delta^M \leftarrow \nabla_w \left(\frac{1}{|M|} \sum_{x \in M} L(x, w) \right)$
Scale: $\forall k,\ \Delta_k^M \leftarrow \Delta_k^M / \alpha_k$
Clip: $\Delta^M \leftarrow \min \left(1, \frac{C}{\|\Delta^M\|_2} \right) \Delta^M$
Rescale: $\forall k,\ \Delta_k^M \leftarrow \alpha_k \Delta_k^M$
Set noise multiplier: $z_t \leftarrow z_0 \cdot d(t)$
Add scaled Gaussian noise:
$\Delta^M \leftarrow \Delta^M + \frac{y}{N},\ y \sim \mathcal{N}(0, (C \cdot z_t)^2)$
end for
Aggregate: $\Delta \leftarrow \frac{1}{N} \sum_M \Delta^M$
Return Δ

a leading smart-home company: Communication, Health and Video, each containing roughly 1 million utterances. Example utterances in the internal datasets are: Communication "call my parents at 0123", Health: "refill my aspirin prescription" and Video: "play my favorite movie".

We construct a train/validation/test split for these datasets so that the split ratio is approximately the same (45:5:50). A roughly equal number of datapoints in the train and test set helps us create a balanced evaluation set for training the MIA models. The "member" utterances used to train/evaluate the MIA success are sourced from the IC-NER training sets, and an equal number of "non-member" utterances are sourced from the test set (not used in training).

4.2. Models

We train IC and NER models using the following two architectures. These architectures are chosen to capture training on an LSTM based and a transformer based model. These architectures have consistently pushed state of the art on several tasks [24, 25].

CLC [26]: The CLC model takes as input the concatenation of a token-level character CNN output with token embeddings, followed by a bi-LSTM layer. IC and NER models are a fully connected layer and a CRF layer, respectively, on top of the bi-LSTM layer. We use pre-trained FastText embeddings [27] for inputs of our models. In order to train IC-NER models on the public data, we use the pre-trained embeddings obtained using Wiki-news corpus [1]. In order to train the models on the internal corpora, we train FastText embeddings on a separate set of utterances sourced from the same smart home agent. We use two bi-LSTM layers, each with a hidden dimension of 384.

BERT: [24] We also train IC-NER layers on top a BERT model, pretrained on a combination of Wikipedia articles dataset [2] and One Billion Word corpus [28] using the masked language modeling task. Our BERT model has 4 layers, 12 attention heads, and a hidden dimensions of 312. We use the sum of the CRF loss (NER) and the cross entropy loss (IC) as the optimization objective.

We tuned the value of C from the set $\{0.1, 0.25, 0.5, 1, 2, 5\}$, z_0 from values ranging from 10^{-6} to 1. We tuned τ from the set $\{0.02, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 0.9\}$ for linear decay; and from the set $\{0.01, 0.02, 0.05, 0.08, 0.1, 0.2, 0.35, 0.5\}$ for exponential decay. We use a p3.16xlarge instance[3] with 8 GPUs to train each model. We implemented the model block and the DP scheme with MXnet [29] and we leverage Horovod [30] to boost efficiency.

We report the semantic error rate (SER [20], a normalized measure for IC-NER task) metric as the utility metric (lower is better). We also report the theoretical DP (ϵ, δ) guarantees (for DP models) and the success of an MIA attack captured using the area under ROC curve (AUC) metric. Following [3] structure for MIA, we train a shadow model on a chosen public corpora. The attack model is then trained on the outputs of this shadow model and evaluated on the outputs of the model under attack. Sorted IC and NER output scores are used as features to attack model. As a reminder, during MIA evaluation, train set for the model under attack are used as *member* set, while the test set is used as *non-member* set. Finally, we also report the time take to train the models using SGD/eDP-SGD/DP-SGD. ADAM optimizer is use for all – SGD/eDP-SGD/DP-SGD settings.

5. RESULTS

Table 1 presents the utility/privacy metrics obtained using SGD training, in addition to relative changes in those metrics when trained using various decay schemes in eDP-SGD (for the sake or brevity, we do not report DP-SGD results separately as they are similar to the no decay scheme). From the results, we sometimes observe an improvement in performance (particularly for public corpora), as was also reported by [31]. This is encouraging and we attribute the improvement in performance to eDP-SGD acting as a regularizer. For smaller datasets, it is easier to overfit the CLC and BERT models to the training set. On the larger corpora, the model error rates does not degrade significantly (except for Video). Additionally, either linear or exponential decay offer a better privacy-utility trade-off than no decay on larger datasets where a loss in utility is expected (we embolden a lower utility loss and higher MIA value compared to no decay for

[1] https://fasttext.cc/docs/en/english-vectors.html

[2] https://www.wikipedia.org

[3] https://aws.amazon.com/ec2/instance-types/

Table 1. Results showcasing privacy-utility tradeoff of NLU models trained using eDP-SGD against models trained with SGD. Datasets are arranged by size. ΔSER and ΔMIA represent relative changes w.r.t baseline. δ is set to 5×10^{-4} for public corpora and 5×10^{-5} for Alexa datasets during computation of ϵ using zCDP [9]

Dataset	SGD		No decay			Linear Decay			Exponential Decay		
	SER	MIA-AUC	ΔSER	ΔMIA	$\log_{10}\epsilon$	ΔSER	ΔMIA	$\log_{10}\epsilon$	ΔSER	ΔMIA	$\log_{10}\epsilon$
CLC model											
ATIS	6.3	67.5	-12.9	-8.26	4.4	-10.6	-11.6	5.8	-11.9	-11.3	24.4
SNIPS	13.0	70.6	-1.8	-2.5	3.1	-10.3	1.8	6.5	-6.7	4.4	17.4
NLU-eval	26.7	65.3	-0.7	-1.4	2.0	3.6	-7.6	2.7	1.4	-4.6	22.1
Health	-	-	-1.2	-3.1	3.6	-2.5	-2.3	4.0	-2.7	-1.5	4.3
Comm.	-	-	0.3	-3.2	6.2	0.3	-5.7	7.5	0.5	-5.4	19.1
Video	-	-	4.0	-2.4	4.8	4.4	-3.7	5.9	3.1	-6.0	19.5
BERT model											
ATIS	5.4	56.1	1.3	-2.5	5.9	-9.7	-5.2	6.3	-4.1	-5.8	11.1
SNIPS	12.7	65.3	-21.5	-8.3	2.7	-20.3	-9.2	2.3	-7.1	-8.3	2.7
NLU-eval	24.1	63.6	-2.4	-7.6	2.2	4.6	-11.6	1.7	0.3	-9.5	2.3
Health	-	-	3.2	-4.1	1.9	0.8	-2.9	2.8	1.2	-4.3	2.9
Comm.	-	-	0.3	-4.4	2.1	0.2	-3.4	2.8	0.7	-4.4	3.7
Video	-	-	4.7	-10.1	1.9	4.6	-10.9	1.8	3.3	-8.2	2.3

Table 2. Comparison of training time per epoch for SGD, DP-SGD and eDP-SGD.

Dataset	SGD	DP-SGD	eDP-SGD
	Time/epoch	Multiplicative factor	
CLC model			
ATIS	9.5s	×2.1	×1.04
SNIPS	9.8s	×5.1	×1.01
NLU-EVAL	9.9s	× 7.3	×1.03
Health	21.8s	×153.9	×1.14
Comm.	109.8s	×143.5	×1.15
Video	83.7s	×152.4	×1.2
BERT model			
ATIS	2.5s	×13.0	×1.29
SNIPS	2.8s	×22.8	×1.52
NLU-EVAL	3.5s	×29.2	×1.62
Health	56.5s	×82.0	×2.06
Comm.	398.8s	×73.8	×2.05
Video	276.0s	×80.3	×2.17

communication and video datasets in Table 1). This indicates that decay when applied independently at each micro-batch still improves privacy-utility trade-off, but the observation is limited to larger datasets. We also observe that the theoretical ϵ privacy guarantee seem to have a loose correlation with MIA success rate, where sometimes higher value of ϵ is associated with greater decrease in MIA success rate. Table 2 also demonstrates that our algorithm does not degrade the training time over SGD. On the other hand, DP-SGD can increase training time by factor of up to 150. Overall, these results indicate that task specific fine tuning with eDP-SGD can be used in industrial settings.

6. CONCLUSION

In this work, we propose a variant of DP-SGD that is suited for GPU based training and use it during fine-tuning IC-NER models on multiple datasets. We report training time, accuracy and privacy metrics on two model architectures, and argue that task specific fine-tuning with eDP-SGD is practical for large scale model training from an accuracy and efficiency perspective. We make observations such as our methods can provide a better utility privacy trade-off and provides significant gains in the training time. We also observe that practical MIA guarantees see improvement even when the theoretical ϵ values are high in value.

In the future, one can explore the protection offered by DP-SGD against privacy attacks on recently proposed larger models. We also aim to further analyze the weak correlation between theoretical guarantees and MIA. We can also study the impact of other techniques like regularization and quantization (alongside DP-SGD) on model memorization.

7. REFERENCES

[1] C. Qu, L. Yang, M. Qiu, W. B. Croft, Y. Zhang, and M. Iyyer, "Bert with history answer embedding for conversational question answering," in *Proceedings of the 42nd ACM SIGIR Conference*, 2019, pp. 1133–1136.

[2] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled contextualized embeddings for named entity recognition," in *Proceedings of NAACL*, 2019, vol. 1, pp. 724–728.

[3] R. Shokri, M. Stronati, C. Song, and V. Shmatikov,

"Membership inference attacks against machine learning models," in *IEEE SP Symposium*, 2017, pp. 3–18.

[4] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *ACM-SIAM SODA*, 2019.

[5] S. Rahimian, T. Orekondy, and M. Fritz, "Differential privacy defenses and sampling attacks for membership inference," in *PriML Workshop (PriML)*, 2019, vol. 13.

[6] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM CCS*, 2016.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006, pp. 265–284.

[8] H. B. McMahan, E. Moore, D. Ramage, and B. Agüera y Arcas, "Federated learning of deep networks using model averaging," *arXiv:1602.05629*.

[9] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *IEEE SP Symposium*, 2019, pp. 332–349.

[10] A. Rahman, T. Rahman, R. Laganière, and N. Mohammed, "Membership inference attack against differentially private deep learning model," *TDP*, 2018.

[11] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *USENIX Security Symposium*, 2019, pp. 1895–1912.

[12] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *arXiv:1710.06963*.

[13] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *arXiv:1802.06739*.

[14] X. Chen, S. Z. Wu, and M. Hong, "Understanding gradient clipping in private sgd: A geometric perspective," *NeurIPS*, vol. 33, 2020.

[15] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and H. B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," *arXiv:1805.10559*.

[16] O. Thakkar, G. Andrew, and H. B. McMahan, "Differentially private learning with adaptive clipping," *arXiv:1905.03871*.

[17] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Towards demystifying membership inference attacks," *arXiv:1807.09173*.

[18] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv:1806.01246*.

[19] H. B. McMahan and G. Andrew, "A general approach to adding differential privacy to iterative training procedures," *arXiv:1812.06210*.

[20] C. Su, R. Gupta, S. Ananthakrishnan, and S. Matsoukas, "A re-ranker scheme for integrating large scale nlu models," in *IEEE SLT*. IEEE, 2018, pp. 670–676.

[21] C. Hemphill, J. Godfrey, and G. Doddington, "The atis spoken language systems pilot corpus," in *ACL Speech and Natural Language Workshop*, 1990.

[22] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, et al., "Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces," *arXiv:1805.10190*.

[23] P. Swietojanski X. Liu, A. Eshghi and V. Rieser, "Benchmarking natural language understanding services for building conversational agents," in *IWSDS*, 2019.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*.

[25] M. E. Peters, M. Neumann, M. Iyyer, M.and Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv:1802.05365*.

[26] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *ACL*, 2016, pp. 1064–1074.

[27] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *LREC*, 2018.

[28] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv:1312.3005*.

[29] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv:1512.01274*.

[30] A. Sergeev and M. Del Balso, "Horovod: Fast and easy distributed deep learning in TensorFlow," *arXiv:1802.05799*.

[31] A. A. H. Khatri, *Preventing Overfitting in Deep Learning Using Differential Privacy*, Ph.D. thesis, 2017.