TRAINING LARGE-VOCABULARY NEURAL LANGUAGE MODELS BY PRIVATE FEDERATED LEARNING FOR RESOURCE-CONSTRAINED DEVICES

Mingbin Xu*, Congzheng Song*, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, Leo Liu, Anmol Walia, Alex Jin

Apple

ABSTRACT

Federated Learning (FL) is a technique to train models on distributed edge devices with local data samples. Differential Privacy (DP) can be applied with FL to provide a formal privacy guarantee for sensitive data on device. Our goal is to train a large neural network language model (NNLM) on compute-constrained devices while preserving privacy using FL and DP. However, the noise required to guarantee differential privacy increases as the model size grows, which often prevents convergence. We propose Partial Embedding Updates (PEU), a novel technique to reduce the impact of DP-noise by decreasing payload size. Furthermore, we adopt Low Rank Adaptation (LoRA) and Noise Contrastive Estimation (NCE) to reduce the memory demands of large models on compute-constrained devices. We demonstrate in simulation and with real devices that this combination of techniques makes it possible to train large-vocabulary language models while preserving accuracy and privacy.

1. INTRODUCTION

Language Models (LMs) play an important role in many applications such as Automatic Speech Recognition (ASR) and Keyboard next word prediction. The vast amount of text data on user devices can power the LMs to provide a more accurate and personalized user experience. However, data stored on a user’s device is private and cannot be transmitted to the server. Federated learning (FL) is proposed in [1] to train a global model from a federation of edge devices that transmit gradients - instead of raw data - to the server. Training models on mobile devices is constrained in terms of memory and bandwidth, and thus FL favors models with smaller capacity. Applications like ASR require a prohibitively large number of classes to model labels in the tail of the distribution as those tail cases are of greater interest from the user experience perspective [2]. It is less explored how to train models with larger capacity for such applications on real user devices.

On the other hand, lack of access to the raw data does not necessarily guarantee privacy. Model gradients may reveal some information about user data [3, 4]. Neural networks tend to memorize training data [5, 6]. While differential privacy (DP) [7] meets the privacy demand of SGD-based algorithms [8], the DP-perturbation can quickly overwhelm the gradient signal as the model capacity grows, which creates a dilemma between accuracy and privacy [9]. Existing works [10, 11, 12] demonstrated that it is possible to train LMs with a small number of parameters (less than 2 million parameters) and small vocabulary with FL and DP on real devices. Nonetheless, such small models are unsuitable for applications such as ASR, where a much larger vocabulary size is needed.

In this work, we demonstrate for the first time that LMs with larger capacity (more than 28 million parameters and 100,000 vocabulary size) can be trained privately with FL on real user devices. The larger vocabulary size leads to higher communication cost and sparser gradients in the word embedding layer which raises challenges when training with DP. We address resource limitations, communication cost as well as the sparsity issue in federated learning with partial embedding updates, low-rank adaptation, and noise contrastive estimation, which together overcome the decreasing gradient utilization due to the increasing model size in the differentially private context. We stabilize training and speed up convergence by exponential moving average. We evaluated on both a Stack Overflow next word prediction task as well as a Virtual Assistant ASR task, and the results suggest that our approaches achieved better trade-off between privacy, utility, and resource.

2. PRELIMINARY

Language Modeling is the task of learning the probability distribution over a sequence of words. In this work, we focus on neural network LM architectures based on fixed-size ordinally-forgetting encoding (FOFE) [13]. FOFE is efficient for both training and inference on edge devices, and thus more suitable for real-world FL compared to alternatives such as LSTM [14] or Transformers [15]. Specifically, FOFE encodes the representation of context at timestamp \( t \) based on a simple recursive formula as \( \mathbf{c}_t = \alpha \cdot \mathbf{c}_{t-1} + \mathbf{e}_t \), where \( \mathbf{e}_t \) is the word embedding at timestamp \( t \), and \( \alpha \in (0, 1) \) is a constant forgetting factor to control the influence of the history...
on the current position. We fix $\alpha = 0.7$ in our experiments as suggested by [13]. FOFE is parallelizable through matrix multiplications, making it ideal for LM applications on resource-limited devices.

**Federated Learning** (FL) [1] was introduced to train a user-independent model with data from multiple users while ensuring personal data never leaves the device. At iteration $t$ of FL, the server first samples a subset of devices $\mathcal{C}$ from the population. Each sampled device $i \in \mathcal{C}$ then downloads the shared model parameters $\theta_i$ from the server and locally trains the model with SGD on its own data to produce a local model $\theta_i$. The model difference $\Delta_{t,i} = \theta_i - \theta$ is sent back to the server. The server aggregates the model differences as a local model.

**Differential Privacy** (DP) requirements provide strong protections for sensitive data on device. DP is formally defined as follows:

**Definition 1** Differential Privacy [7] A randomized algorithm $\mathcal{A}: \mathcal{D} \rightarrow \mathcal{R}$ is $(\epsilon, \delta)$-differentially private, if for any pair of neighboring training populations $\mathcal{D}$ and $\mathcal{D}'$ and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$, it holds that

$$\Pr[\mathcal{A}(\mathcal{D}) \in \mathcal{S}] \leq e^\epsilon \cdot \Pr[\mathcal{A}(\mathcal{D}') \in \mathcal{S}] + \delta. \quad (1)$$

A training population $\mathcal{D}'$ is considered the neighbor of $\mathcal{D}$ if $\mathcal{D}'$ can be obtained by adding or removing one user from $\mathcal{D}$, and vice versa. The choice of $(\epsilon, \delta)$ yields a trade-off between accuracy and privacy.

Building a FL framework with DP was first proposed in [17]. The interpretation of Equation 1 in the FL setup is that the participation of an individual user and its contribution cannot be detected or reverse-engineered by any means. We call such systems Private Federated Learning (PFL) hereafter.

To combine DP with FL, in each iteration: 1) each sampled client’s model difference $\Delta_{t,i}$ is clipped to a zero-centered $L_2$ ball with radius $S$ so as to bound the sensitivity of $\Delta_{t,i}$; and 2) a Gaussian noise with covariance matrix $\sigma^2 S^2 \mathbf{I}$ is added to aggregation of $\sum_{i \in \mathcal{C}} \Delta_{t,i}$. The noise standard deviation is calibrated by the moment accountant [8, 18, 19] with fixed sampling rate $q$ (fraction of clients sampled in each iteration), number of training iterations $T$, and privacy budgets $(\epsilon, \delta)$.

### 3. METHODS

**Partial Embedding Updates** The vocabulary size and number of parameters of NNLM result in a huge payload size which puts a massive communication burden on the server. In addition, the distribution of word frequency in natural language is highly skewed and long-tailed, and signal-to-DP-noise ratio (SNR) for rare words in a large vocabulary is thus insignificant. We propose partial embedding updates (PEU), a sampling-based FL training for the word embeddings, for reducing the payload size and increasing the SNR for rare words. At a high level, the server samples $m$ words from the vocabulary. Each device trains the full model but only sends back the scaled updates for the sampled words. DP noise is added to the update of these sampled $m$ words instead of all vocabulary. Updates of words that are not sampled are dropped, with no noise added. Thus the overall amount of noise added is greatly reduced. Let

- $R(w)$ be the probability of word $w$ being sampled in each central iteration and the scaling constant for updates of $w$,
- $Z \in \{0, 1\}$ be a Bernoulli random variable where $P(Z = 1) = R(w)$, and
- $E_w$ and $E^0_w$ be the word vector of $w$ after and before local update, respectively.

which allows us to derive the unbiased estimator for the update of $w$ with PEU as

$$\mathbb{E}_{Z \sim P}\left[\frac{Z}{R(w)} (E_w - E^0_w)\right] = P(Z = 0) \cdot 0 + \frac{P(Z = 1)}{R(w)} (E_w - E^0_w) \quad (2)$$

with $R(w) = P(w \in \mathcal{W})$ where $\mathcal{W}$ is a set of $m$ words sampled with some distribution $Q$ without replacement. $R(w)$ does not seem to have a closed-form distribution and $Q(w)$ is used to approximate $R$.

The dimension reduction of PEU benefits the PFL system in two important ways. First, it decouples the model complexity requirement and the network requirement. The number $m$ of sampled words can be determined by the payload budget. Second, it increases the overall SNR because DP noise grows with the size of the model and tail words are less affected by DP noise as they are less likely to be sampled.

**Low-Rank Adaptation** [20] developed low-rank adaptation (LoRA), a generalized reparameterization approach to mitigate the immense memory requirement of fine-tuning large neural networks. Since the major goal of LoRA is shrinking the trainable dimension, it benefits PFL in a similar fashion. Prior to training, each dense matrix $W \in \mathbb{R}^{m \times n}$ is reformulated as a summation $W' = W + L \cdot R$, where $L \in \mathbb{R}^{m \times r}$ is zero-initialized and $R \in \mathbb{R}^{r \times n}$ is randomly initialized. Typically $r$ is one or two orders of magnitudes smaller than $m$ and $n$. $L$ and $R$ are trainable, and $W$ is frozen during client local training. LoRA can effectively reduce the size of model update from $O(mn)$ to $O(r(m + n))$ which not only decreases...
the amount of traffic sent to the server but also the amount of DP noise added to the model.

**Noise Contrastive Estimation** (NCE) [21] is a self-normalized approach that approximates SoftMax over large vocabulary efficiently [22]. If a word $w$ is sampled from the corpus with auxiliary label $D = 1$, and $k$ other words are sampled from unigram $Q$ with auxiliary label $D = 0$. The probability of $D$ is a mixture of $P_D(w, c)$ and $Q(w)$, and $P_D(w, c)$ represents the probability of the word $w$ given the context $c$, modeled by the LM parameter $\theta$:

$$
P(D = 0 | c, w) = \frac{k \times Q(w)}{P_D(w, c) + k \times Q(w)}$$

$$
P(D = 1 | c, w) = \frac{P_D(w, c)}{P_D(w, c) + k \times Q(w)}$$

(3)

where $P_D(w, c)$ is the dot product of penultimate and $w$'s embedding with self-normalization. The LM distribution is learned by maximizing the log-likelihood of $D$:

$$
\text{Loss}_{NCE_k} \approx \sum_{(w, c) \in \text{corpus}} \log P(D = 1 | c, w) + \sum_{i=1}^{k} \log P(D = 0 | c, w_i)).$

While the initial purpose of NCE is to reduce computation, its memory saving is more appealing to PFL. The memory requirement for gradient and intermediate output are controllable by batch size and sampled words, independent of the embedding matrix size.

### 4. EXPERIMENTS

**Stack Overflow** This dataset is derived from Stack Overflow and preprocessed by the Tensorflow API. Its statistics are summarized in Table 3. We build a 3rd-order FOFE [13]

1 We assume SoftMax is implemented as in-place as possible and reuses output buffer for backpropagation. The gap is much larger in practice.

2 Available at https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data

baseline from scratch with BMSGD [23]. We apply our proposed methods in the PFL context. We train the LM from random initialization for 5,000 iterations, and the other hyper-parameters are summarized in Table 2. The model includes a 256-dimension word embedding of 100k vocabulary and 4 fully-connected layers of 768 nodes. We tie the weights of the output layer to the input word embedding to reduce the number of parameters [24]. The model has in total more than 28 million parameters, with $90\%$ of the parameters from the word embedding layer. We evaluate model performance using perplexity (PPL) and use a non-private centrally trained model as the baseline.

**Speech Recognition on Virtual Assistant** In addition to next word prediction, we evaluate our approach on automatic speech recognition (ASR) using word error rate (WER).

We experiment with a hybrid ASR system that consists of acoustic model (AM), pronunciation dictionary and LM. Each component is represented as a weighted finite-state transducer (WFST) [25] and the recognition process is formulated as a beam search on the composed WFSTs. The training corpus contains a large number of randomly sampled and anonymized automatically-transcribed virtual assistant queries and is supplemented with templated utterances based on knowledge base statistics. We sampled 750 million sentences for LM training. Separately, we curated 30 million anonymized human-labeled virtual assistant queries, from which we sampled 20k queries to form a PPL test set and another 20k to form a WER test set. The remaining queries are used as AM training data.

The baseline is trained identically to Stack Overflow LMs, except that virtual assistant data is used. We build a weaker

### Table 1. Word embedding memory requirement between SoftMax and NCE during training, assuming a $100k \times 256$ word embedding and a $32\text{utterances/batch} \times 16\text{words/utterance}$ minibatch with vanilla SGD.

| Hyper-parameter | SoftMax | NCE |
|-----------------|---------|-----|
| embedding parameters | 98 MB | 98 MB |
| gradient | 98 MB | < 1 MB |
| outputs | 200 MB | < 5 MB |
| total | $\approx 300$ MB | $\approx 100$ MB |

### Table 2. Common hyper-parameter values shared throughout the experiments.

| Hyper-parameter | Value |
|-----------------|-------|
| $(\epsilon, \delta)$ | $(2, 10^{-6})$ |
| sampling rate $q$ | $2 \times 10^{-3}$ |
| $L_2$ norm clipping bound | 0.3 |
| batch size | 16 utterances |
| number of local epochs | 1 |
| local learning rate | 0.1 |
| server learning rate | 0.1 |
| server optimizer | FedAdam [16] |
| cohort size | 8000 |
| NCE noise size | 1024 |

### Table 3. Stack Overflow dataset summary.

| Hyper-parameter | Value |
|-----------------|-------|
| distinct users | train | dev | test |
| examples | 342k | 38k | 204k |
| utterances/batch | 135m | 16m | 16m |
Table 4. Performance on Stack Overflow.

| setup             | payload | PPL  |
|-------------------|---------|------|
| non-private baseline | -       | 50.96|
| full model         | 112M    | 66.47|
| LoRA(48)           | 21M     | 85.24|
| LoRA(64)           | 28M     | 80.13|
| PEU(5k)            | 16M     | 64.77|
| PEU(10k)           | 21M     | 65.06|
| PEU(20k)           | 31M     | 65.97|
| PEU(5k)+LoRA(64)   | 3.6M    | 77.79|
| PEU(10k)+LoRA(64)  | 4.8M    | 77.43|

Table 5. Performance on randomly sampled and anonymized virtual assistant queries. Payload is same as Table 4.

| setup             | PPL  | WER  |
|-------------------|------|------|
| non-private baseline | 67.33| 3.97 |
| am-ref             | 71.81| 4.29 |
| full model         | 67.19| 4.09 |
| LoRA(48)           | 67.67| 4.10 |
| LoRA(64)           | 67.56| 4.10 |
| PEU(5k)            | 67.12| 4.08 |
| PEU(10k)           | 67.10| 4.09 |
| PEU(20k)           | 67.04| 4.08 |
| PEU(10k) + LoRA(64)| 67.47| 4.09 |
| real device        | 67.60| 4.13 |
| real device        | 67.46| 4.12 |

baseline that uses only the AM training data, referred to as am-ref. This will also serve as the initial checkpoint for PFL experiments. We follow [26]’s recipe to produce an AM, which is used in all experiments. Specifically, no ngram-LM is interpolated with NNLM. We start by simulating with the non-IID nature of the data present in FL, where we separate the data into 500K partitions with the algorithm described in [27] and treat each partition as an individual user device. Finally, we replicate the results of the simulation on real user devices. For both simulation and real device experiments, we fine-tune the LM for 200 iterations.

**Result Analysis** Results of various configurations are summarized in Table 4 and 5. For both datasets, PEU-enabled settings converge to similar or better performance as their disabled counterparts. Particularly in Stack Overflow, it outperforms LoRA significantly with the same payload budget, which shows that PEU is more suitable for training from scratch. Compared to full model update, PEU dramatically reduces the memory footprint and payload. More importantly, because the reduction is irrelevant to the prediction space, larger models potentially benefit more from PEU. PEU is compatible with LoRA.

In virtual assistant ASR task, the combined setting drops the payload to less than 5% of the full model update, while matching the utility of the non-private server-based baseline. This shows that our proposed sparsification and compression techniques can significantly reduce the communication cost in PFL and are more suitable for fine-tuning. Finally, we replicate the results of the simulation on real user devices, shown in the last two rows of Table 5. The results on real user devices show that our proposed method is not only theoretically sound, but also feasible in production.

5. RELATED WORKS

Federated learning with differential privacy was first formalized in [17] and showed that a small RNN-based LM with 1M parameters can be trained with PFL in simulation. Later works extended the above work by applying PFL on millions of real devices for mobile keyboard prediction where the model is a LSTM [14] with 10,000 word vocabulary [10, 11, 12]. In contrast to these works that focused on a smaller LM, for the first time, we demonstrated that a much larger LM with more than 28M parameters and 10 times larger vocabulary can be trained on real devices with PFL.

More recent work has demonstrated that large LMs can be trained with DP-SGD with reasonable utility. Large BERT models [28] can be pretrained with DP-SGD with carefully tuned hyperparameters [29]. Various techniques based on LoRA were proposed in [30] to reduce the amount of DP noise for fine-tuning large LMs. Different tricks in [31] were effective for fine-tuning large LMs with DP-SGD without using any sparsification or compression. These large LMs are Transformer-based models [15] where the majority of model parameters are in the dense layers instead of the embedding layers. Our work, on the other hand, focus on training models with large vocabulary size in the federated setting where 90% of the model parameters are in the embedding layers. The sparsity of the embedding layer gradients raised challenges in training with DP that the previous works did not consider.

6. CONCLUSION

With partial embedding updates, low-rank adaptation, noise contrastive estimation and exponential moving average on trained parameters, we train large-vocabulary language models using private federated learning on resource-constrained mobile devices. We achieved parity between experiments in simulation and real devices, and for the first time demonstrated that LMs with large number of parameters and large vocabulary can be trained privately with FL with performance comparable to non-private server baseline. Our approaches are attractive for applications such as keyboard prediction and ASR that need to learn a language model from user devices in a privacy-preserving manner.
7. REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, et al., “Communication-efficient learning of deep networks from decentralized data,” in AISTATS, 2017.

[2] Mandana Saebi, Ernest Pusateri, Aaksha Meghawat, et al., “A discriminative entity-aware language model for virtual assistants,” in Interspeech, 2021.

[3] Luca Melis, Congzheng Song, Emiliano De Cristofaro, Vitaly Shmatikov, et al., “Exploiting unintended feature leakage in collaborative learning,” in S&P, 2019.

[4] Ligeng Zhu, Zhijian Liu, and Song Han, “Deep leakage from gradients,” in NeurIPS, 2019.

[5] Nicholas Carlini et al., “The secret sharer: Evaluating and testing unintended memorization in neural networks,” in USENIX Security, 2019.

[6] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, et al., “Extracting training data from large language models,” in USENIX Security, 2021.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith, “Calibrating noise to sensitivity in private data analysis,” in TCC, 2006.

[8] Martín Abadi, Andy Chu, Ian J. Goodfellow, et al., “Deep learning with differential privacy,” in CCS, 2016.

[9] Raef Bassily, Adam Smith, and Abhradeep Thakurta, “Private empirical risk minimization: Efficient algorithms and tight error bounds,” in FOCS, 2014.

[10] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, et al., “Federated learning for mobile keyboard prediction,” arXiv, 2018.

[11] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays, “Federated learning for emoji prediction in a mobile keyboard,” arXiv, 2019.

[12] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, et al., “Training production language models without memorizing user data,” arXiv, 2020.

[13] Shiliang Zhang Hui Jiang, Mingbin Xu, et al., “The fixed-size ordinally-forgetting encoding method for neural network language models,” in ACL, 2015.

[14] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” Neural computation, 1997.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al., “Attention is all you need,” NeurIPS, 2017.

[16] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, et al., “Adaptive federated optimization,” in ICLR, 2021.

[17] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang, “Learning differentially private recurrent language models,” in ICLR, 2018.

[18] Ilya Mironov, “Rényi differential privacy,” in CSF, 2017, pp. 263–275.

[19] Ilya Mironov, Kunal Talwar, and Li Zhang, “Rényi differential privacy of the sampled gaussian mechanism,” CoRR, 2019.

[20] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, et al., “Lora: Low-rank adaptation of large language models,” CoRR, 2021.

[21] Michael Gutmann and Aapo Hyvärinen, “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models,” in AISTATS, 2010.

[22] Andriy Mnih and Yee Whye Teh, “A fast and simple algorithm for training neural probabilistic language models,” in ICML, 2012.

[23] Kai Chen and Qiang Huo, “Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering,” in ICASSP, 2016.

[24] Ofir Press and Lior Wolf, “Using the output embedding to improve language models,” in EACL, 2017.

[25] Mehryar Mohri, Fernando Pereira, and Michael Riley, “Weighted finite-state transducers in speech recognition,” Comput. Speech Lang., 2002.

[26] Zhen Huang, Tim Ng, Leo Liu, et al., “SNDCNN: self-normalizing deep cnns with scaled exponential linear units for speech recognition,” in ICASSP, 2020.

[27] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown, “Measuring the effects of non-identical data distribution for federated visual classification,” CoRR, 2019.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al., “Bert: Pre-training of deep bidirectional transformers for language understanding,” in NAACL, 2019.

[29] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi, “Large-scale differentially private bert,” arXiv, 2021.

[30] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyn A Inan, et al., “Differentially private fine-tuning of language models,” in ICLR, 2022.

[31] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto, “Large language models can be strong differentially private learners,” in ICLR, 2022.