Large Language Models Can Be Strong Differentially Private Learners

ICLR 2022

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto

Background: User data and privacy issues in NLP

- There are inherent **conflicts** between data collection and privacy protection for tasks in NLP (e.g., building dialog generation systems)
- Private user data is abundant and of **high quality**. Can we use it directly?

Public data (low quality, large quantity)

Annotator-driven data (high quality, costly)

Private user data (high quality, large quantity)

Background: User data and privacy issues in NLP

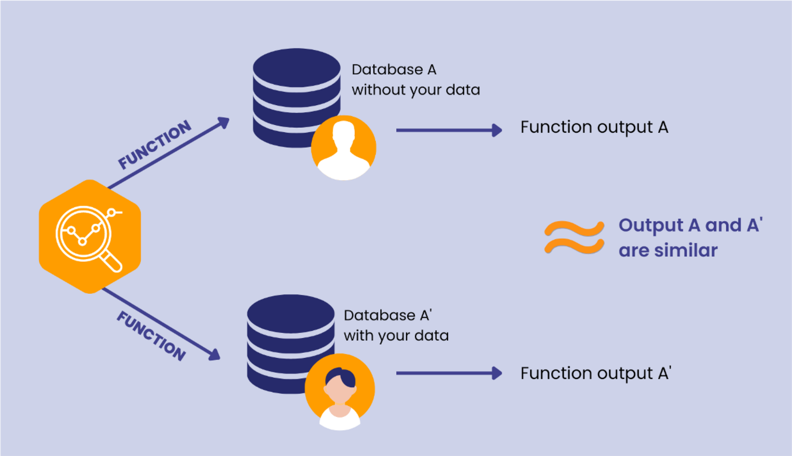
- Directly training large language models (LMs) on private user data can be Problematic
  - Large LMs can memorize training data
  - Data extraction attacks are surprisingly effective for large LMs (Carlini et al., 2021)

Background: Need for privacy-preserving techniques

- We need provable guarantees that models won't leak private data
- Goal
    - Use private data
    - Do not leak them

Background: Differential privacy

- Differential privacy (DP) is a formal privacy guarantee for an algorithm used in US census, in Google analytics, at Apple..
- Past attempts at enforcing DP for vision tasks (via DP-SGD) resulted in models with low utility



This work

- Q: Is it possible to build high quality DP NLP models on moderate amounts of private training data?
- A: Yes!
- This work:
    - Leverage (public) off-the-shelf **pretrained** models and perform fine-tuning with DP-Adam
    - Surprisingly, full fine-tuning-updating **all model parameters** yields strong performance
    - Even more surprisingly, the **larger** the pretrained model, the better the performance of private fine-tuning, unlike what theory for private convex learning prescribes

Overview

- Overall
  - Large language model (transformer-based) can achieve differential privacy
- Contributions
  - Effective: tricks for hyperparameter setting
  - Efficient: ghost clipping

Method overview

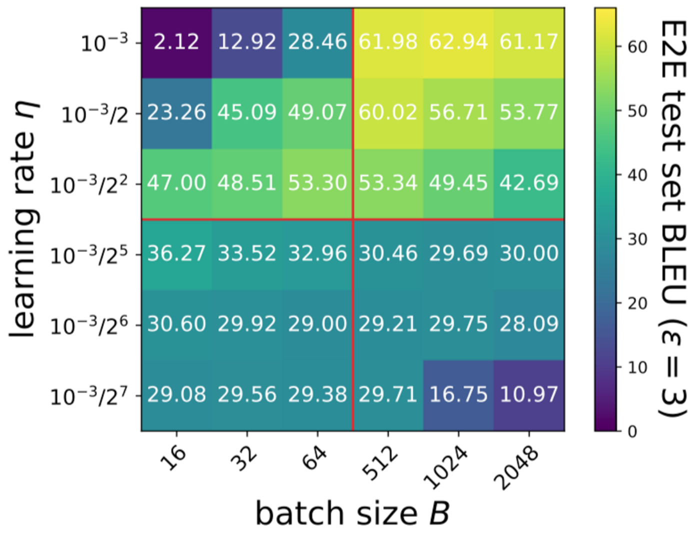
- Effective
    - Hyperparameters
    - Fine-tuning objective
- Efficient
    - Ghost clipping: Clipping per example gradients without instantiating per example gradients.

Effective

- Good hyperparameters
  - DP learning is sensitive to choices of hyperparameters
  - They did a thorough study of how hyperparameters affect performance
  - Good hyperparameters tend to transfer across tasks – we transferred tuning results on one task to all remaining tasks
  - **Totally based on experiment findings**
- Fine-tuning objective
  - Objectives that make learning easy results in better private models
  - They want the fine-tuning objective to be close to the pretraining objective
  - **Alignment**

Hyperparameters

- Batch size and learning rate
- Good batch sizes and learning rates for private learning is different from those typical for non-private learning
- Case 1: Fixed epochs (compute bound)
  - Need large batch size
  - Need large learning rate



Hyperparameters

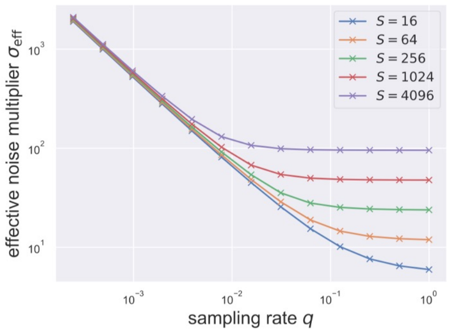
- Batch size and number of epochs
- Case 2: Unconstrained epochs
    - Fix the update steps (large batches, each epoch less updates, more epochs)
    - Jointly scaling both the batch size and number of epochs is almost always better

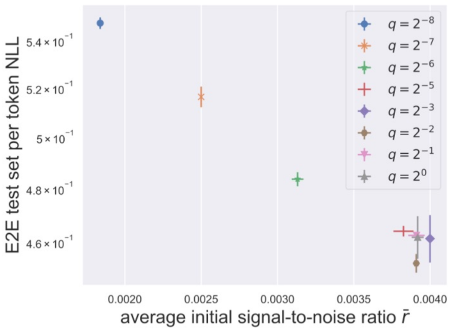
$$\bar{g} = \widetilde{g} + \bar{z}, \quad \widetilde{g} = \frac{1}{B}\sum_{i \in \mathcal{B}} \mathrm{Clip}(\nabla \mathcal{L}_i, C), \quad \bar{z} \sim \mathcal{N}\left(0, C^2 \frac{\sigma^2}{B^2} I_p\right) = \mathcal{N}\left(0, C^2 \frac{\sigma_{\text{eff}}^2}{N^2} I_p\right),$$

- Heuristic explanation:
    - , B is batch size
    - Si $\sigma_{\text{eff}} = \frac{\sigma}{q} = \frac{\sigma N}{B}$
    - Larger r, better perfor $r = \|\widetilde{g}\|_2 / \|\bar{z}\|_2$

Hyperparameters

- Batch size and number of epochs
- Case 2: Unconstrained epochs
    - Fix the update steps (large batches, each epoch less updates, more epochs)
    - Jointly scaling both the batch size and number of epochs is almost always better
    - Larger batch size, larger sampling rate q (just B/N), smaller σ_{eff} , larger r, lower NLL





Fine-tuning task formulation matters

- For classification, CLS-token fine-tuning introduces a discrepancy between pretraining (masked language modeling) and fine-tuning (network on top of CLS)
- This makes the fine-tuning problem slightly difficult

Fine-tuning task formulation matters

- Instead of predicting integer labels, they ask the model to predict textualized labels during fine-tuning
- Example: sentiment classification
  - Given sentence <input>, classify as positive or negative sentiment
  - Construct template "<input> It is [MASK]."
  - Predict [MASK] in the template, where [MASK] is "awesome" or "terrible"
  - Easier fine-tuning problem results in better private models (even with generic templates)

Effective

- Good hyperparameters
  - DP learning is sensitive to choices of hyperparameters
  - They did a thorough study of how hyperparameters affect performance
  - Good hyperparameters tend to transfer across tasks – we transferred tuning results on one task to all remaining tasks
  - **Totally based on experiment findings**
  - **Large batch size**
- Fine-tuning objective
  - Objectives that make learning easy results in better private models
  - They want the fine-tuning objective to be close to the pretraining objective
  - **Alignment**
  - **Templates**
- Then, how to make the model efficient?

Ghost clipping

- DP optimization is costly due to clipping **per example gradients**
- Naively implemented, this step instantiates per example gradients and can be prohibitively costly in memory
- They present a technique for per example gradient clipping **without** instantiating per example gradients for any linear layer in a large Transformer model

Problem: Per-example gradient

$$\bar{g} = \widetilde{g} + \overline{z}, \quad \widetilde{g} = \frac{1}{B}\sum_{i\in\mathcal{B}}\mathrm{Clip}(\nabla\mathcal{L}_i, C), \quad \overline{z} \sim \mathcal{N}\left(0, C^2\frac{\sigma^2}{B^2}I_p\right) = \mathcal{N}\left(0, C^2\frac{\sigma_{\mathrm{eff}}^2}{N^2}I_p\right),$$

- Clip(,) means reweighting
  - Scaling factor: $c_i = \min(1, C/\|\nabla\mathcal{L}_i\|_2)$
  - Reweighted loss: $\sum_i c_i\mathcal{L}_i$
- Challenge:
  - Compute $\|\nabla\mathcal{L}_i\|_2$
- Tricks
  - Per example gradient → Layer by layer gradient
  - Ghost clipping for transformer

Trick 1: Layer by layer gradient

- $\|\nabla \mathcal{L}_i\|_2$ takes a large memory to instantiating
- Observations
  - Neural networks have multi-layers
  - Vector norm: $\|u\|_2 = \|(\|u_1\|_2, \ldots, \|u_k\|_2)\|_2$
- Thus

$$\|\nabla_{W^{(1)}} \mathcal{L}_i\|_2, \ldots, \|\nabla_{W^{(L)}} \mathcal{L}_i\|_2$$

  - We can instantiating one layer each time
  - Compute separately

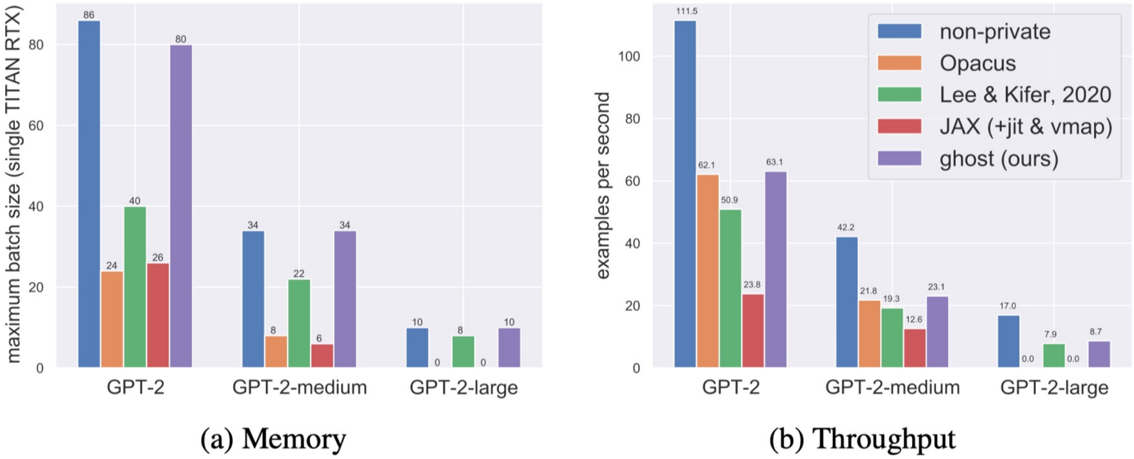
Trick 2: Ghost clipping

- We still need to compute each layer norm $\|\nabla_W \mathcal{L}_i\|_{\mathrm{F}}^2$
- Linear layer
  - a_i input to a linear layer $a \in \mathbb{R}^{B \times T \times d}$
  - g_i gradient of the linear layer $g \in \mathbb{R}^{B \times T \times p}$
  - W weight of the linear layer $W \in \mathbb{R}^{p \times d}$
- Normal way
  - Step 1: $\nabla_W \mathcal{L}_i = g_i^\top a_i \in \mathbb{R}^{p \times d}$. $\mathcal{O}(Bpd)$
  - Step 2: Compute the norm
- Ghost clipping $\|\nabla_W \mathcal{L}_i\|_{\mathrm{F}}^2 = \mathrm{vec}(a_i a_i^\top)^\top \mathrm{vec}(g_i g_i^\top)$
  - Thus not necessary to compute step $\mathcal{O}(BT^2)$

GPT-2, $d \approx 50{,}000$ and $p = 768$ context window $T \leq 1024$.

Ghost clipping

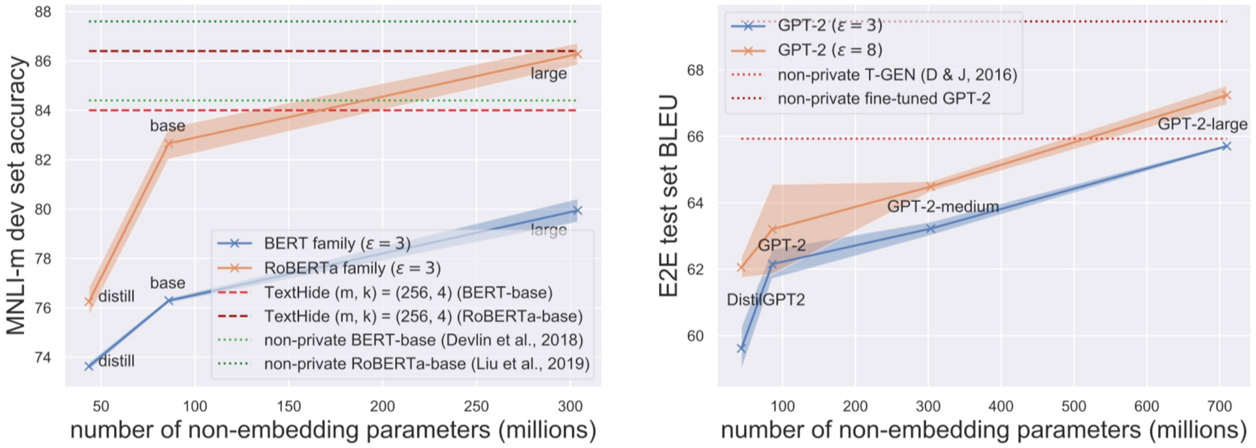
- Performance



- Then, the model is effective and efficient now.

Does high dimensionality degrade performance?

- Do larger models lead to better or worse results?
    - Answer: Larger models are better.
- Are fine-tuning methods that privatize fewer parameters more performant?
    - Answer: Not true in general.



(a) Sentence classification
MNLI-matched (Williams et al., 2018)

(b) Natural language generation
E2E (Novikova et al., 2017)

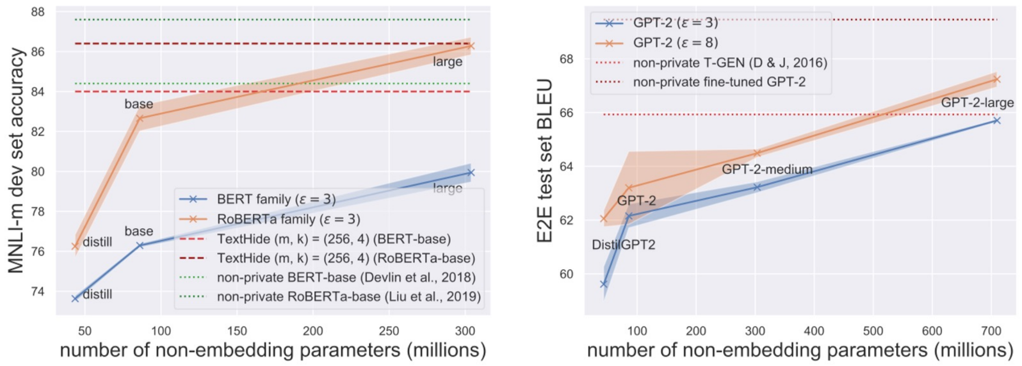
Sentence classification

Table 1: Full fine-tuning larger pretrained models with text infilling has best performance. Results are dev set accuracies. Best numbers based on two-sample test for each privacy level are in bold.

Method	$\epsilon = 3$				$\epsilon = 8$			
	MNLI-(m/mm)	QQP	QNLI	SST-2	MNLI-(m/mm)	QQP	QNLI	SST-2
RGP (RoBERTa-base)	-	-	-	-	80.5/79.6	85.5	87.2	91.6
RGP (RoBERTa-large)	-	-	-	-	86.1/86.0	86.7	90.0	93.0
full (RoBERTa-base)	82.47/82.10	85.41	84.62	86.12	83.30/83.13	86.15	84.81	85.89
full (RoBERTa-large)	85.53/85.81	**86.65**	88.94	90.71	86.28/86.54	**87.49**	89.42	90.94
full + infilling (RoBERTa-base)	82.45/82.99	85.56	87.42	91.86	83.20/83.46	86.08	87.94	92.09
full + infilling (RoBERTa-large)	**86.43/86.46**	86.43	**90.76**	**93.04**	**87.02/87.26**	87.47	**91.10**	**93.81**
$\epsilon \approx$ (Gaussian DP + CLT)	2.52	2.52	2.00	1.73	5.83	5.85	4.75	4.33
$\epsilon \approx$ (Compose tradeoff func.)	2.75	2.75	2.57	2.41	7.15	7.16	6.87	6.69

Results overview

- For classification, DP fine-tuning can outperform TextHide (InstaHide for text)
- For generation, DP fine-tuning can outperform strong non-private baselines
- Larger and better pretrained models result in better fine-tuned performance



(a) Sentence classification (b) Natural language generation

Generation

- Epsilon, smaller, better

Table 2: Full fine-tuning performs on par with or outperforms others methods that execute gradient update in low dimensional spaces. Results are on E2E from fine-tuning GPT-2.

Metric	DP Guarantee	Gaussian DP + CLT	Compose tradeoff func.	Method full	LoRA	prefix	RGP	top2	retrain
BLEU	$\epsilon = 3$	$\epsilon \approx 2.68$	$\epsilon \approx 2.75$	**61.519**	58.153	47.772	58.482	25.920	15.457
	$\epsilon = 8$	$\epsilon \approx 6.77$	$\epsilon \approx 7.27$	**63.189**	**63.389**	49.263	58.455	26.885	24.247
	non-private	-	-	69.463	69.682	68.845	68.328	65.752	65.731
ROUGE-L	$\epsilon = 3$	$\epsilon \approx 2.68$	$\epsilon \approx 2.75$	**65.670**	**65.773**	58.964	65.560	44.536	35.240
	$\epsilon = 8$	$\epsilon \approx 6.77$	$\epsilon \approx 7.27$	**66.429**	**67.525**	60.730	65.030	46.421	39.951
	non-private	-	-	71.359	71.709	70.805	68.844	68.704	68.751

Dialog Generation

Table 3: Fine-tuning with DP-Adam yields high quality chit-chat dialog generation models.

Model	DP Guarantee	Gaussian DP +CLT	Compose tradeoff func.	Metrics		
				F1 ↑	Perplexity ↓	Quality (human) ↑
GPT-2	$\epsilon = 3$	$\epsilon \approx 2.54$	$\epsilon \approx 2.73$	15.90	24.59	-
	$\epsilon = 8$	$\epsilon \approx 6.00$	$\epsilon \approx 7.13$	16.08	23.57	-
	non-private	-	-	17.96	18.52	-
GPT-2-medium	$\epsilon = 3$	$\epsilon \approx 2.54$	$\epsilon \approx 2.73$	15.99	20.68	-
	$\epsilon = 8$	$\epsilon \approx 6.00$	$\epsilon \approx 7.13$	16.53	19.25	-
	non-private	-	-	18.64	15.40	-
DialoGPT-medium	$\epsilon = 3$	$\epsilon \approx 2.54$	$\epsilon \approx 2.73$	**17.37**	**17.64**	2.82 (2.56, 3.09)
	$\epsilon = 8$	$\epsilon \approx 6.00$	$\epsilon \approx 7.13$	**17.56**	**16.79**	3.09 (2.83, 3.35)
	non-private	-	-	19.28	14.28	3.26 (3.00, 3.51)
HuggingFace (ConvAI2 winner)	non-private	-	-	19.09	17.51	-
HuggingFace (our implementation)	non-private	-	-	16.36	20.55	3.23 (2.98, 3.49)
Reference	-	-	-	-	-	3.74 (3.49, 4.00)

Summary

- Large LMs can be **effectively** fine-tuned under DP if hyperparameters and the fine-tuning objective are set right
- Full fine-tuning large LMs under DP can be memory **efficient**
- Better and **larger** pretrained models yield improved private fine-tuning results

Thank you!

Q & A