DIFFERENTIALLY PRIVATE FINE-TUNING OF LANGUAGE MODELS

Privacy Preserving Machine Learning Articles Presentation

Article

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., ... & Zhang, H. (2021). Differentially Private Fine-tuning of Language Models. arXiv preprint arXiv:2110.06500.

Authors

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., ... & Zhang

Presented by

Zakaria Boulkhir & Omar Iken

Master 2 Data Science

Université de Lille

January 13, 2022

Outline

Fine-tuning Large Language Models

Transformer-based Large Language Models (LLMs)

Examples:

- BERT [Devlin et al.]
- GPT [Radford et al.]

Fine-tuning

1. Pre-train the model on a large public dataset.
2. Fine-tune weights for downstream tasks.

Applications

- Email reply suggestion.
- Sentence completion in text editors.
- Language translation.

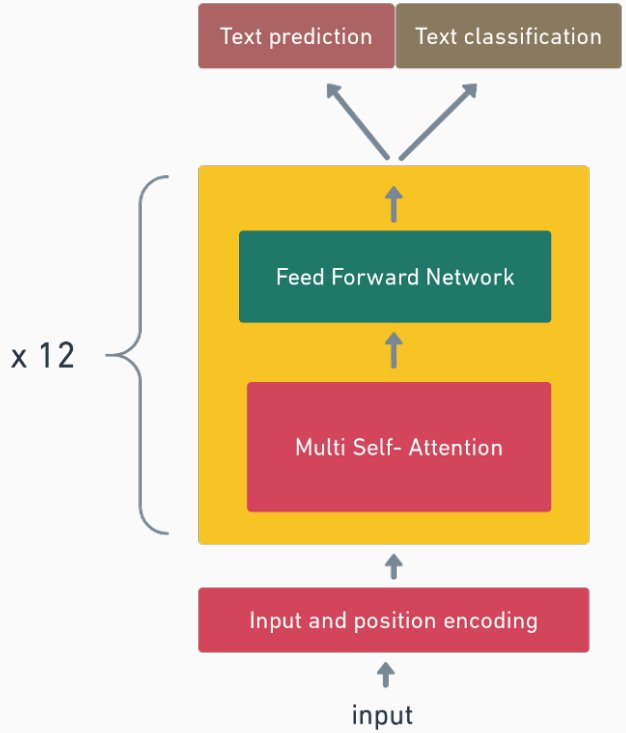


Figure 1: Structure of GPT [Maryam Fallah]

Issue

The immense size of LLMs makes it impractical to fine-tune the full model and store a separate copy of the parameters for hundreds of downstream tasks.

Parameter-efficient methods

To alleviate the issues of storage and computational cost for fine-tuning

Examples:

- Adapter fine-tuning [Houlsby et al. (2019)].
- Prefix-tuning [Li and Liang (2021)].
- Intrinsic dimensionality [Aghajanyan et al. (2020)].

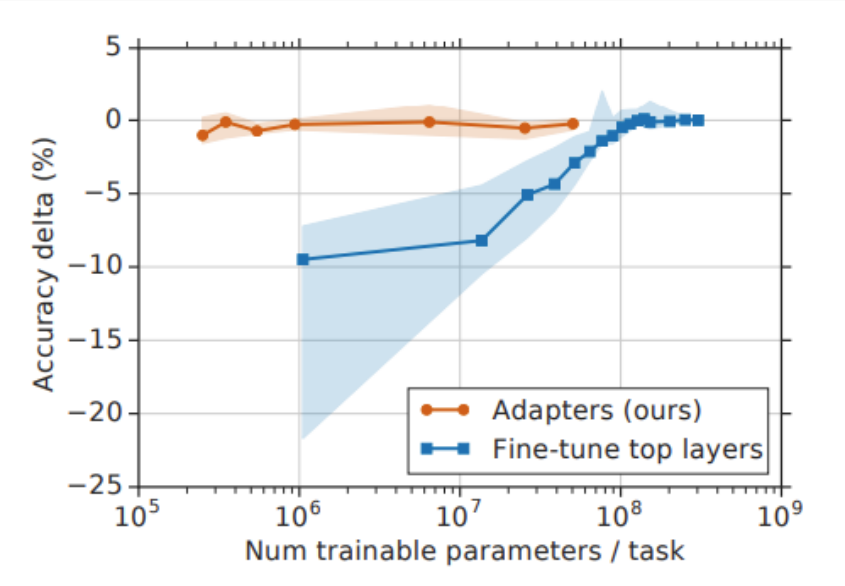


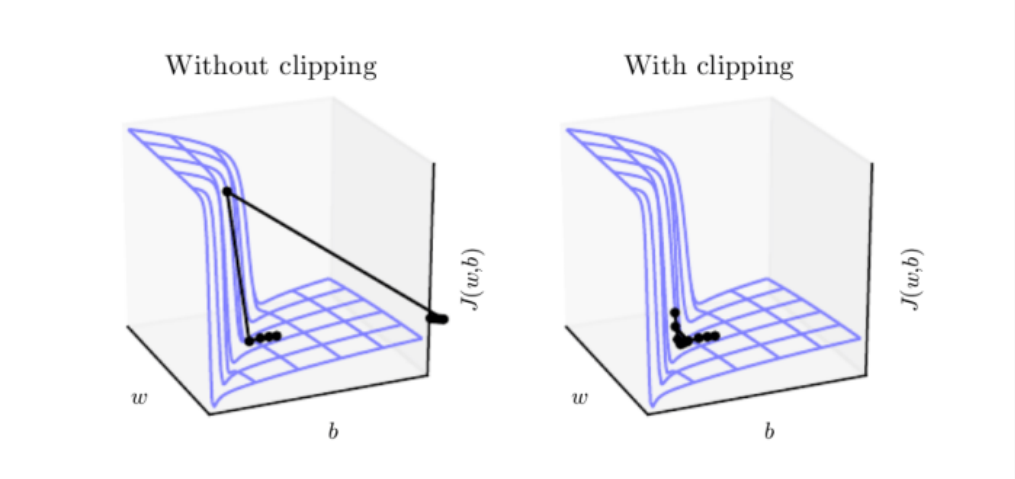
Figure 2: Trade-off between accuracy and number of trained tasks-pecific parameters [Houlsby et al. (2019)].

Works in non-private settings. BUT, what about the context of private learning ?

Meta-Framework (Introducing Privacy)

Prior Algorithms: Full Fine-tuning via DPSGD

Gradient clipping: significant computational and memory overheads in most implementations.



Gaussian noise addition: grows as the square-root as the number of model parameters.

BUT

Runing DPSGD on all the weights $\Longrightarrow$ adding noise in all directions

$\Longrightarrow$ unlearning the knowledge learned during pre-training

Prior Algorithms: Reparametrized Gradient Perturbation (RGP)

Exploits the implicit low-rank structure in the gradient updates of SGD to substantially improve upon DPSGD.

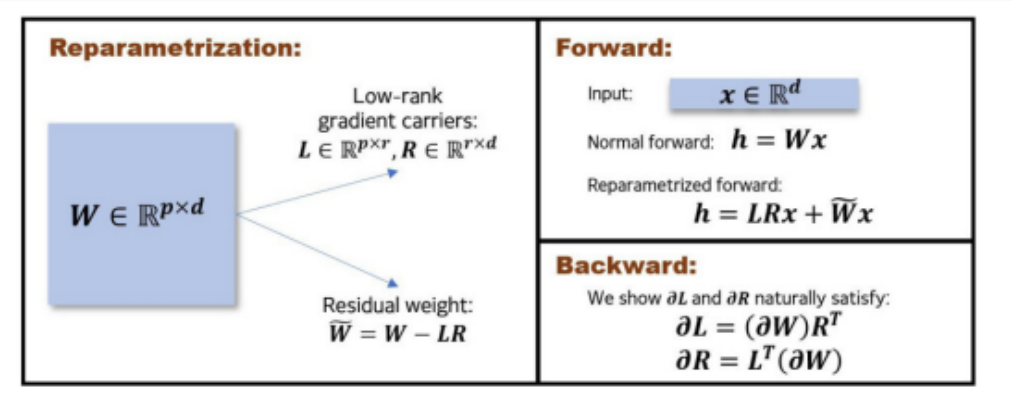


Figure 3: [Yu et al. (2021b)]

While this low-dimensional projection loses some of the signal in gradient, it turns out to contain enough to still achieve high accuracy.

Low-dimensional gradients $\Longrightarrow$ reducing the memory consumption & noise introduced.

BUT

Although RGP uses a low-rank update at each step, its accumulated update is not of low stable rank and hence can not be compressed into small plug-in modules.

- The low-rank subspaces of RGP are different at different updates
- The accumulated update of RGP contains all the added noises, which are of high stable rank.

Meta-framework

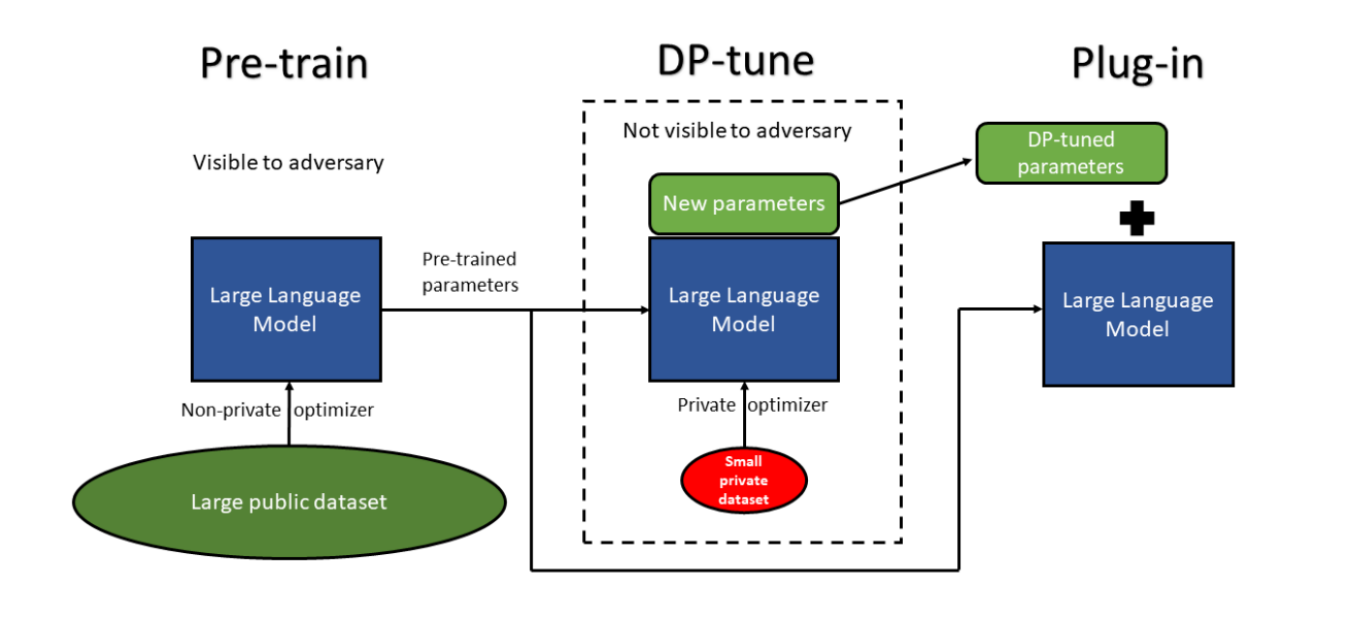


Figure 4: An illustration of the framework

1. The model is pre-trained on a large, public dataset.
2. New parameters are introduced and privately fine-tuned on a smaller, private, task-specific dataset. The original parameters are frozen during this process.
3. The fine-tuned new parameters are plugged-in to the model for downstream tasks.

Meta-Framework

Fine-tuning Model

$$f_{FT}(W_{PT}, \theta; x) \tag{1}$$

- $f(W_{PT}; x)$: pre-trained model and W_{PT} its weights.
- $dim(\theta) \ll dim(W_{PT})$
- **Initialization condition:** $f_{FT}(W_{PT}, \theta_0; x) = f(W_{PT}; x)$
- **Additive fine-tuning methods:** $f_{FT}(W_{PT}, \theta; x) = f_{FT}(W_{PT} + \pi(\theta); x)$

Main Advantages

- Updates only a small fraction of parameters (between 0.05% and 1%).
- Stores a single pre-trained model that can be used for different downstream tasks by setting only a tiny fraction of parameters.
- Learning is performed in a lower dimension, and thus more appropriate for memory usage.
- Unlike RGP, it keep the pre-trained weights frozen during the learning process.
- Interesting applications for distributed learning, e.g., federated learning.

Instantiating The Meta-framework

Fine-tuning via Low-Rank Adaptation (LoRA) [Hu et al., (2021)]

$$W^i = W^i_{PT} + \underbrace{L^i R^i}_{\text{low-rank correction term}} \tag{2}$$

- $W_i \in \mathbb{R}^{a \times b}$, $L_i \in \mathbb{R}^{a \times r}$, $R_i \in \mathbb{R}^{r \times b}$.

Fine-tuning via Adapters [Houlsby et al. (2019)]

$$A(x) = U(\tau(D(x))) + x \tag{3}$$

- $U \in \mathbb{R}^{d \times r}, D \in \mathbb{R}^{r \times d}$: up & down-projection, τ: non-linear activation function.
- $r \ll d$

Fine-tuning via Compacter [Mahabadi et al. (2021)]

$$M_\ell = \sum_{i=1}^{n} A_i \otimes (S_i^\ell T_i^\ell) \tag{4}$$

- $A_i \in \mathbb{R}^{n \times n}$, $S_i^\ell \in \mathbb{R}^{a/n \times k}$, $T_i^\ell \in \mathbb{R}^{k \times b/n}$.

Experiments / Main results

Fine-Tuning for Language Understanding Tasks

Method		MNLI	SST-2	QQP	QNLI	Avg.	Trained params
Full	w/o DP	87.6	94.8	91.9	92.8	91.8	100 %
	DP	53.1	82.6	74.4	63.9	68.5	
RGP	DP	80.1	91.6	85.5	87.2	86.1	100%
Adapter	DP	83.4	**92.5**	85.6	**87.5**	**87.3**	1.4%($r = 48$)
Compacter	DP	82.6	92.3	84.7	85.1	86.2	0.055%($r = 96, n = 8$)
LoRA	DP	**83.5**	92.2	**85.7**	87.3	87.2	0.94%($r = 16$)
Full	w/o DP	90.2	96.4	92.2	94.7	93.4	100%
RGP	DP	86.1	93.0	86.7	90.0	88.9	100%
Adapter	DP	87.7	93.9	86.3	90.7	89.7	1.4%($r = 48$)
Compacter	DP	87.5	94.2	86.2	90.2	89.5	0.053%($r = 96, n = 8$)
LoRA	DP	**87.8**	**95.3**	**87.4**	**90.8**	**90.3**	0.94%($r = 16$)

Table 1: Benchmarking results for different model representations and different language understanding tasks. The first bit of the results concern the model RoBERTa-Medium, while the last bit is for the Large model.

Fine-tuning for Natural Language Generation (NLG)

Method	Val perp	BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2-Small + DP	4.51	63.8	7.19	39.5	67.5	1.87
GPT-2-Medium + DP	4.02	65.5	8.45	42.7	67.9	2.23
GPT-2-Large + DP	3.87	**66.7**	**8.63**	**44.0**	67.8	**2.33**
GPT-2-XL + DP	**3.79**	66.1	8.53	43.0	**68.1**	2.28
GPT-2-Medium	3.19	70.4	8.85	46.8	71.8	2.53
GPT-2-Large	3.06	70.4	8.89	46.8	72.0	2.47
GPT-2-XL	3.01	69.4	8.78	46.2	71.5	2.49

Method	Val perp	BLEU	MET	TER
GPT-2-Small + DP	3.82	38.5	0.34	0.53
GPT-2-Medium + DP	3.30	42.0	0.36	0.51
GPT-2-Large + DP	3.10	43.1	0.36	**0.5**
GPT-2-XL + DP	**3.00**	**43.8**	**0.37**	**0.5**
GPT-2-Medium	2.67	47.1	0.39	0.46
GPT-2-Large	2.89	47.5	0.39	0.45
GPT-2-XL	2.83	48.1	0.39	0.46

Tables 2, 3: Different metric results for language generation tasks. The first table are those on E2E NLG task and table 2 on the DART dataset.

Conclusion & Perspectives

Conclusion & Perspectives

Contributions

- One of the main contribution is testing the setting of Fine-tuning and parameter-efficient under the privacy setting.
- The establishment of a setting that works for the private setting.
- Empirically proving that the larger the model, the better the performance under this same setting.
- Proposing a simpler, sparser, and faster algorithm for differentially private fine-tuning of large-scale pre-trained language models.
- The first to fine-tune GPT-2-XL using DP (the largest model -with 1.5B parameters- trained thus far using DP).

Perspectives

- Why the success of parameter efficient fine tuning methods work ?
- The list of proposed methods for instantiating the framework is non-exhaustive, so one can search and try other methods.
- The framework works mostly for additive fine-tuning methods, so what about non-additive methods?
- The choice of hyperparameters could be improved and can be done separately for each specific task.