Security & Privacy

Data Poisoning : add a backdoor (through a physical key)

(add poison training point)

Backdoor with trigger / triggerless

Expressed as a bilevel optimization problem:

$$\boxed{X_p^* = \underset{X_p}{\operatorname{argmin}} \, L_{adv}(X_t, y_{adv}; \theta^*(X_p))}$$

X_p : poisoned data that we add

L_{adv} : how well we do at attacking our targets X_t

$$\theta^*(X_p) = \underset{\theta}{\operatorname{argmin}} \, L_{train}(X_c \cup X_p, Y; \theta)$$

Approximating solutions to bilevel optimization problems

"Metapoison" attack:

unroll stochastic gradient descent updates

$$\theta_1 = \theta_0 - \alpha \nabla_\theta L_{train}(X_c \cup X_p, Y; \theta_0)$$

$$\begin{cases} \theta_2 = \theta_1 - \alpha \nabla_\theta L_{train}(X_c \cup X_p, Y; \theta_1) \\ X_p^{i+1} = X_p^i - \beta \nabla_{X_p} L_{adv}(X_t, y_{adv}; \theta_2) \end{cases}$$

θ: a differentiable function of X_p

and we can take gradients

Approximating solutions to bilevel opt problems

How can we solve this?

Idea: instead of the argmin, write down the gradient descent updates and 'unroll' stochastic gradient descent updates.

$$\theta_1 = \theta_0 - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(X_c \cup X_p, Y; \theta_0)$$
$$\theta_2 = \theta_1 - \alpha \nabla_\theta \mathcal{L}_{\text{train}}(X_c \cup X_p, Y; \theta_1)$$
$$X_p^{i+1} = X_p^i - \beta \nabla_{X_p} \mathcal{L}_{\text{adv}}(x_t, y_{\text{adv}}; \theta_2),$$

Now θ is a (differentiable) function of X_p and we can take gradients.

This is called the "Metapoison" attack

[Huang+ 2020]

Poison Type:

$$\begin{cases} \text{No Overlap} \\ \text{With Overlap} \end{cases}$$

Aside: What's the state of empirical results in data poisoning? (vision)

Data poisoning is actually pretty brittle: what breaks data poisoning attacks

- Data augmentation / changing to SGD / transfer / ResNets
- Constraining for imperceptibility via l_∞
- Black box attacks
- Flipping the target image

	CIFAR-10			TinyImageNet		
	Transfer		From Scratch	Transfer		From Scratch
Attack	WB	BB		WB	BB	
FC	22.0	7.0	1.33	49.0	2.0	4.0
CP	33.0	7.0	0.67	14.0	1.0	0.0
BP	**85.0**	8.5	2.33	**100.0**	**10.5**	**44.0**
WiB	-	-	**26.0**	-	-	32.0
CLBD	5.0	6.5	1.00	3.0	1.0	0.0
HTBD	10.0	**9.5**	2.67	3.0	0.5	0.0

Attacks are viable, but not as good as we had seen [Schwarzchild+ 2020]

Provable methods for data poisoning mitigation

$$P = (1-\varepsilon) P_{clean} + \varepsilon Q$$

Data poisoning $\Longleftrightarrow$

An adversary arrives and adds samples from an arbitrary distribution Q with the number of samples up to ε times the clean dataset

Recap and future threats

Practical, easy poisoning attacks exist for downstream, fine-tuned models

Metapoison style attacks work for fine-tuned models

Defenses (provable and otherwise) are still an open problem

Data poisoning LMs – not yet seen, but likely in the future

LMs : privacy risk

Aggregation: combine multiple, public sources of information

Accessibility: make sensitive, public information more available

Privacy Attacks!

Memorization of public facts

aggregation

$$\Downarrow$$

provable guarantees

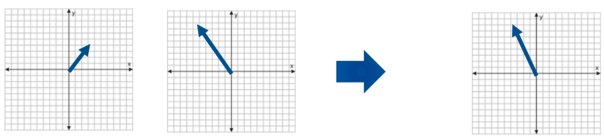
Differential Privacy:

A formal privacy guarantee for a randomized algorithm

Differential privacy with deep learning (DP-SGD)

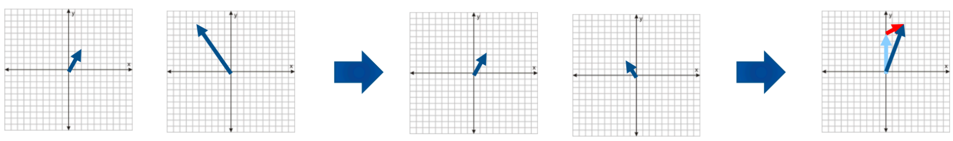
Q: How can we apply this to deep neural networks?

SGD:



Compute gradients Sum and update

Differentially private SGD



Compute gradients Clipping Sum, noise and update

Mixed results for DP w/ deep neural nets in NLP

Prior attempts to apply DP to large neural models in NLP (via DPSGD) have often failed.

Example: Kerrigan et al – trained language generation models on reddit data

Input: “Bob lives close to the..”

Non-private outputs: “station and we only have two miles of travel left to go”

Private output ($\epsilon = 100$): “along supply am certain like alone before decent exceeding”

Why did things fail? (The dimensionality hypothesis)

1. Large language models have ~ 300 million parameters. That is *a lot* of things to privatize
2. Theory says differential privacy performance should degrade with dimension $\sqrt{d}/n$
3. Most (if not all) successful DP methods relied on low-dimensional statistics.

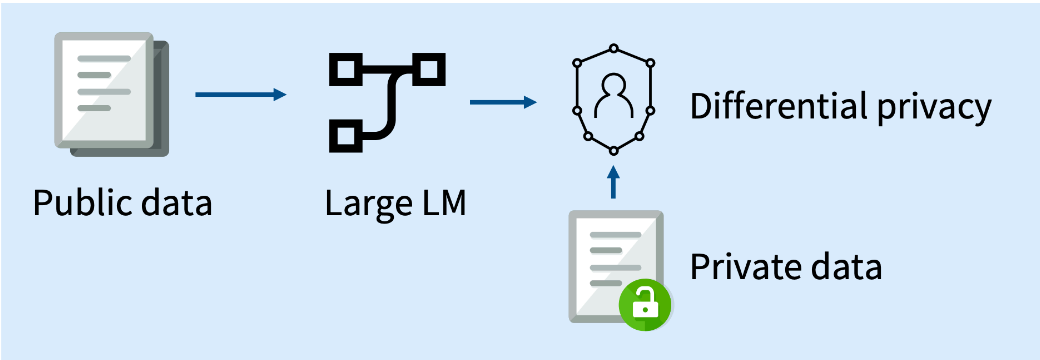
Differential privacy with large language models

Training large language models from scratch with DP

Open problem – large model size poses statistical + computational issues

Using a public language model to build a private downstream model

This is possible!



Fine if tuned right

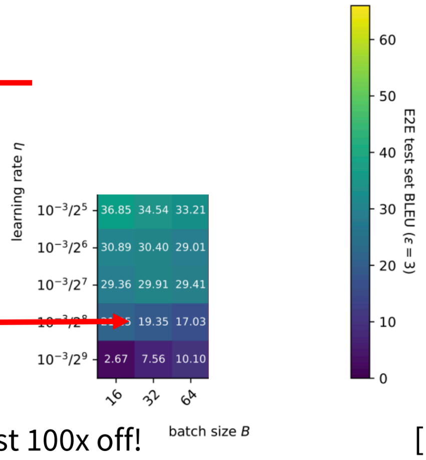
Language model performance – fine if tuned right

Identifying the problem: using *non-private* hyperparameters for *private* optimization

Solution: a way of predicting DP-SGD performance via ‘signal-to-noise’ ratios

Optimal hyperparameters
Found via a ‘signal-to-noise ratio’ analysis

Typical hyperparameters



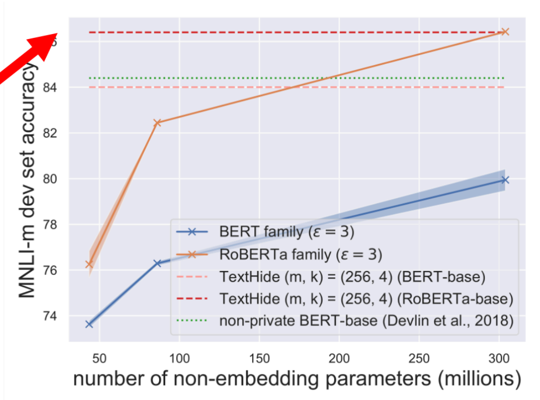
‘Naive’ choices were almost 100x off!

[Li+ 2021]

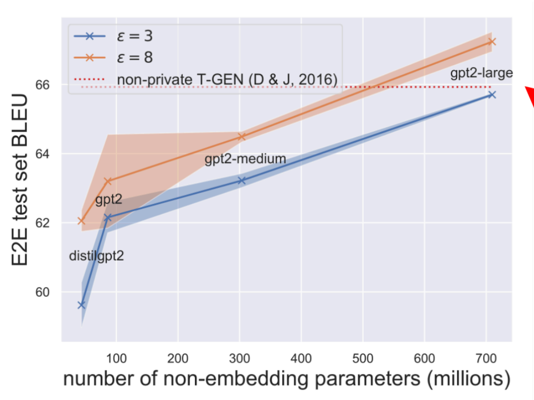
Bigger models are better private learners

DP-SGD (which people ruled out) beats nonprivate baselines + heuristic privacy notions

Heuristic
Privacy method



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



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

Non-private
baseline

Privacy:

Pre-trained, large language models are key to privacy

In the non-private case, pre-training is a small gain (5 BLEU points on E2E)

Metric	DP Guarantee	Gaussian DP + CLT	Compose tradeoff func.	full	LoRA	prefix	RGP	top2	retrain
				Method					
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

For private learning, the difference is **huge**:

- unusable (15 BLEU) when trained from scratch
- usable (61.5 BLEU) when privately fine-tuning a base LM.