A Differential Privacy Metric

The (ϵ, δ)-Differential Privacy is the most widely used privacy metric which is defined as

Definition A.1 (Differential Privacy (Dwork et al. 2006b; Dwork 2006; Dwork et al. 2006a))**.** Let $\mathcal{D}^n$ be space of datasets including n samples. A randomized algorithm $M : \mathcal{D}^n \to \mathcal{Z}$ is (ϵ, δ)-DP if for all adjacent[4] $D, D' \in \mathcal{D}^n$ and for all subset of outputs $S \subseteq \mathcal{Z}$, we have

$$P(M(D) \in S) \le e^{\epsilon} P(M(D') \in S) + \delta.$$

If $\delta = 0$, then M achieves pure DP which is stronger than the case $\delta > 0$. To achieve better properties described in Section 3.1, we give a privacy guarantee by a new privacy metric and its proofs here.

A variant of DP is called ω-truncated ρ-Concentrated Differential Privacy (Bun et al. 2018) as follows.

Definition A.2 ((ρ, ω)-tCDP)**.** Let $\omega > 1$ and $\rho > 0$. A randomized algorithm $M : \mathcal{D}^n \to \mathbb{R}$ satisfied (ρ, ω)-tCDP if, for all adjacent inputs $d, d' \in \mathcal{D}^n$,

$$D_\alpha(M(d) \| M(d')) \le \rho\alpha, \ \forall \alpha \in (1, \omega)$$

where $D_\alpha(\cdot\|\cdot)$ denotes the Rényi divergence (Rényi 1961) of order α.

The (ρ, ω)-tCDP provides nice properties for private learning as discussed in Section 3.1. The general usage can be found in the Algorithms 1 and 2. However, the privacy parameters, (ρ, ω), have to be carefully initialized and chosen to satisfy their constraint.

In this section, we first summarize the properties of (ρ, a)-ctCDP and then provide proofs based on the theoretic results of (ρ, ω)-tCDP. Also, we show how to use the properties of the ctCDP to initialize the privacy parameters and add essential constraints in detail.

A.1 Properties of (ρ, a)-ctCDP

We suggest an a-constrained truncated ρ-CDP (or (ρ, a)-ctCDP) based on the (ρ, ω)-tCDP to provide a minimal example with tight composition bound, linear composition function and a simple Gaussian mechanism. It is defined in Definition 3.1. The proposal originates from the scenario where the privacy budget is constrained as a constant, typically as (ϵ, δ)-DP, and a series of properties are required for scalable gradient perturbation (Algorithm 1). Other than ctCDP, more complicated privacy metric can be used on demand. Here, we first summarize the theoretic results of ctCDP with proofs in the following sections. Note that most of the results are identical to tCDP by fixing ω as ω_a. The major difference lies on the Lemma A.5 where the noising mechanism is assumed to be derived from zCDP, resulting in a different constraint on ρ.

Lemma A.1 (Relation to (ϵ, δ)-DP)**.** *Suppose a randomized algorithm M satisfies (ρ_a, a)-ctCDP with $a = \log(1/\delta)/\epsilon$ and ρ_a defined in Definition 3.1. Then M satisfies (ϵ, δ)-DP.*

[4]By adjacent, we mean the two datasets have only one different entry at most.

Lemma A.2 (Relation to tCDP)**.** *Suppose a randomized algorithm M satisfies (ρ, a)-ctCDP. Then M satisfies (ρ, ω_a)-tCDP where ω_a is defined in Definition 3.1.*

Definition A.3 (Sensitivity)**.** Given $f : \mathcal{D}^n \to \mathbb{R}^d$, the L_2 sensitivity of f, denoted by $\Delta_2(f)$ (or simply Δ) is defined as:

$$\Delta_2(f) = \max_{D, D'} \|f(D) - f(D')\|_2 ,$$

where D, D' are adjacent datasets.

Lemma A.3. *The sensitivity of $\|\nabla_t\|_2^2$ is $\Delta_2(\|\nabla_t\|_2^2) = (2|D_t| - 1)C_g^2$.*

Proof. By definition,

$$\max_{D,D'} \left\| \|\nabla_t\|_2^2 - \|\nabla'_t\|_2^2 \right\|_2$$

$$= \max_{D,D'} \left\| \left\| \sum_i \nabla_{t,i} \right\|_2^2 - \left\| \sum_{i \ne m} \nabla_{t,i} \right\|_2^2 \right\|_2$$

where we assuming the m-th sample is eliminated from computing the gradient ∇'_t. Also, we have

$$\left\| \sum_i \nabla_{t,i} \right\|_2^2 = \sum_{i,j} \nabla_{t,i}^\top \nabla_{t,j},$$

$$\left\| \sum_i \nabla'_{t,i} \right\|_2^2 = \sum_{i,j} \nabla_{t,i}^\top \nabla_{t,j} - 2 \sum_j \nabla_{t,m}^\top \nabla_{t,j} + \nabla_{t,m}^\top \nabla_{t,m}.$$

Thus,

$$\left\| \sum_i \nabla_{t,i} \right\|_2^2 - \left\| \sum_i \nabla'_{t,i} \right\|_2^2$$

$$= (2 \sum_j \nabla_{t,j} - \nabla_{t,m})^\top \nabla_{t,m}$$

whose norm is lower than or equal to $(2|D_t| - 1)C_g^2$. Thus, $\Delta_2(\|\nabla_t\|_2^2) = (2|D_t| - 1)C_g^2$. □

Lemma A.4 (Composition & Post-processing)**.** *Let two mechanisms be $M : \mathcal{D}^n \to \mathcal{Y}$ and $M' : \mathcal{D}^n \times \mathcal{Y} \to \mathcal{Z}$. Suppose M satisfies (ρ_1, a)-ctCDP and $M'(\cdot, y)$ satisfies (ρ_2, a)-ctCDP for $\forall y \in \mathcal{Y}$. Then, mechanism $M'' : \mathcal{D}^n \to \mathcal{Z}$ (defined by $M''(x) = M'(x, M(x))$) satisfies $(\rho_1 + \rho_2, a)$-ctCDP.*

The Lemma A.4 is directly given by tCDP when we transform ctCDP to tCDP with fixed ω_a (see Lemma A.2).

Instead of deriving a canonical noise mechanism for the ctCDP, we directly use the Gaussian mechanism theorem of zCDP and amplify its privacy cost in the form of ctCDP by subsampling.

Lemma A.5 (Gaussian mechanism for ρ-zCDP (Bun and Steinke 2016))**.** *Let $f : \mathcal{D}^n \to \mathcal{Z}$ have sensitivity Δ. Define a randomized algorithm $M : \mathcal{D}^n \to \mathcal{Z}$ by*

$$M(x) \leftarrow f(x) + \mathcal{N}(0, \sigma^2).$$

Then M satisfies $\frac{\Delta^2}{2\sigma^2}$-zCDP.

Lemma A.6 ((ρ, a)-ctCDP from ρ-zCDP by privacy amplification through subsampling)**.** *Let* $\omega_a = (1+a)+\sqrt{a(a+1)}$. *Let* $\rho, q \in (0, 0.1]$, $n, N \in \mathcal{Y}$ *with* $q = n/N$ *(sampling rate) and satisfy*

$$\log(1/q) \geq 3\rho(2 + \log_2(1/\rho)), \tag{9}$$

$$0 \leq \rho < \min\left\{\frac{\log(1/q)}{4\omega_a}, \frac{\rho_a}{13q^2}\right\}. \tag{10}$$

Let $M : \mathcal{D}^n \to \mathbb{R}$ *satisfy* ρ*-zCDP. Define the mechanism* $M_s : \mathcal{D}^N \to \mathcal{Y}$ *by* $M_s(x) = M(x_S)$*, where* $x_S \in \mathcal{D}^N$ *is the restriction of* $x \in \mathcal{D}^N$ *to the entries specified by a uniformly random subset* $S \subseteq [N]$ *with* $|S| = n$*.*

The algorithm $M_s : \mathcal{D}^N \to \mathcal{Y}$ *satisfies* $(13q^2\rho, a)$*-ctCDP.*

Remarkably, the Lemma A.6 assumes the sub-routine is ρ-zCDP. Since the subsampling-based privacy amplification happens after the noise mechanism, it is natural to use the Gaussian mechanism (Lemma A.5) which results in a zCDP privacy cost.

These privacy guarantees are derived from (ρ, ω)-tCDP by constraining the range of ρ and ω where ω is simply a constant. By sacrificing the flexibility of the privacy parameters, we can get a single parameter metric which is simple in notation. Moreover, by fixing the ω, we will be able to update the privacy parameters by gradient descent in meta-learning[5].

In this paper, we apply the (ρ, a)-ctCDP to the gradient perturbation scenario (Algorithm 1) where the total privacy cost ρ_{tot} is constrained by (ϵ, δ)-DP. A completed pipeline of private learning includes initialization of the parameters based on the given privacy budget and the protected learning (Algorithm 1). For the convenience of implementation, we provide the detailed steps of the initialization in Algorithm 3 as supplementary to private learning. The algorithm is based on the ctCDP with subsampling. If subsampling is not used, only the step budget needs to be modified as $\rho_0 = \rho_0'$ with the upper bound $\rho_{\text{ub}} = \rho_a$.

A.2 Budget constraint from standard DP

Our motivation for proposing ctCDP is that existing private learning methods are typically compared by performance under a given privacy budget. Therefore, we first introduce a budget constraint using the standard DP, which is translated as bounds on the tCDP parameters.

Theorem A.1 (Transformation of tCDP to DP)**.** *Suppose* M *satisfy* (ρ, ω)*-tCDP. Then, for all* $\delta > 0$ *and all* $1 < \alpha \leq \omega$*,* M *satisfies* (ϵ, δ)*-DP with*

$$\epsilon = \begin{cases} \rho + 2\sqrt{\rho \log(1/\delta)}, & \log(1/\delta) \leq (\omega-1)^2\rho \\ \rho\omega + \frac{\log(1/\delta)}{\omega-1}, & \log(1/\delta) \geq (\omega-1)^2\rho \end{cases}$$

When ϵ and δ are fixed, we want to maximize the available budget ρ and fix ω_a. We consider $\log(1/\delta) \geq (\omega-1)^2\rho$ to find the upper bound of ρ when ω can be maximized, as well. First, we solve the linear function of ρ, i.e., $\epsilon = \rho\omega + \frac{\log(1/\delta)}{\omega-1}$, given some ω. Let

$$a \triangleq \log(1/\delta)/\epsilon$$

[5] Note that the composition of ω in (ρ, ω)-tCDP is not continuously differentiable if all sub-mechanisms have varying ω.

Algorithm 3 Private learning initialization.

(ϵ, δ)-DP, the number of learning iterations (T) estimated from non-private algorithms.

Transformation from DP to ctCDP (Lemma A.1): $a \leftarrow \log(1/\delta)/\epsilon$ and compute ρ_a and ω_a by Definition 3.1
$\rho_{\text{tot}} \leftarrow \rho_a$.
Estimate step budget by uniformly decomposing ρ_{tot} into T steps: $\rho_0' \leftarrow \frac{\rho_{\text{tot}}}{T}$ (Lemma A.4).
Estimate a batch sampling rate q, e.g., $q \leftarrow (\sqrt{|D|} + 10)/|D|$.
$\rho_0 \leftarrow \frac{\rho_0'}{13q^2}$ (subsampling by Lemma A.6)
If ρ_0 and q do not satisfy Eqs. (9) and (10), re-estimate q by choosing the smaller solution to

$$\log(1/q) = 3\frac{\rho_{\text{tot}}}{13q^2}(2 - \log_2(\frac{\rho_0'}{13q^2}))$$
$$\frac{\rho_0'}{13q^2} = \frac{\log(1/q)}{4\omega_a}.$$

Then re-compute ρ_0 using the new q.
Get the upper bound of step budgets: ρ_{ub} (Eq. (10)).
Compute noise scale by Lemma A.5:
estimated step noise: $\sigma_g = \Delta/\sqrt{2\rho_0}$
step noise lower bound: $\sigma_{\min} = \Delta/\sqrt{2\rho_{\text{ub}}}$
Compute batch size $|D_t| \equiv \lfloor qn \rfloor$.
Output: $\rho_{\text{tot}}, \sigma_g, \sigma_{\min}, q, |D_t|$

---

and denote the solution as

$$\rho_{\text{tCDP}} = \frac{\epsilon}{\omega}\left(1 - \frac{\log(1/\delta)/\epsilon}{\omega-1}\right) = \frac{\epsilon(\omega-(a+1))}{\omega(\omega-1)}. \tag{11}$$

Now, we substitute the ρ_{tCDP} into $\log(1/\delta) \geq (\omega-1)^2\rho$ to obtain

$$a\omega \geq (\omega-1)(\omega-(a+1))$$
$$\Rightarrow (1+a) - \sqrt{a(a+1)} \leq \omega \leq (1+a) + \sqrt{a(a+1)}$$

By the definition of tCDP, $\rho_{\text{tCDP}} \geq 0$ and $\omega > \alpha > 1$. Thus,

$$\omega > 1 \text{ and } \omega \geq a+1.$$

Because $a+1 > (1+a) - \sqrt{a(a+1)}$ and $a > 0$, the only solution to $\log(1/\delta) = (\omega-1)^2\rho$ is the upper bound. Now, we denote the upper bound of ω as

$$\omega_a \triangleq (1+a) + \sqrt{a(a+1)}$$

and substitute it into Eq. (11) to get

$$\rho_a \triangleq \epsilon\frac{\sqrt{a(a+1)}}{((1+a)+\sqrt{a(a+1)})(a+\sqrt{a(a+1)})} \tag{12}$$

which is also the solution of $\epsilon = \rho + 2\sqrt{\rho\log(1/\delta)}$.

Until now, we finish the proof of the bounds in Definition 3.1. In addition, it is straightforward to get that (ρ_a, ω_a)-tCDP is equivalent to (ρ_a, a)-ctCDP. Thus, we have the Lemma A.1. We can further extend it for $\rho < \rho_a$.

Lemma A.7 ((ρ, a)-ctCDP to (ϵ, δ)-DP)**.** *Suppose a randomized algorithm M satisfy (ρ, a)-ctCDP for $0 < \rho \le \rho_a$ and $a > 0$, then M satisfies (ϵ, δ)-tCDP with*

$$\epsilon = \frac{(\omega_a - 1)\omega_a}{\omega_a - a - 1}\rho, \ \delta = \exp(-a\epsilon).$$

A.3 Noise mechanism

The canonical noise for tCDP is a Gaussian noise reshaped by a sinh function. We restate the theorem by rearranging the variables.

Theorem A.2 (Sinh-Normal mechanism for (ρ, ω)-tCDP (Bun et al. 2018))**.** *Let $f : \mathcal{D}^n \to \mathcal{Z}$ has sensitivity Δ. Let $\rho = \frac{8\Delta^2}{\sigma^2}$, ω satisfy $\frac{1}{4\omega^2} \le \rho < 16$ and $A = 8\Delta\omega$. Define a randomized algorithm $M : \mathcal{D}^n \to \mathcal{Z}$ by*

$$M(x) \leftarrow f(x) + A \operatorname{arsinh}(\frac{1}{A}\mathcal{N}(0, \sigma^2)).$$

Then M satisfies (ρ, ω)-tCDP.

If $\omega \to \infty$, then $A \to \infty$ in which case the $(\rho, \omega = \infty)$-tCDP is just ρ-zCDP and the Sinh-Normal distribution degrades as the normal distribution. However, due to the truncation of ω, the privacy cost, i.e., $\frac{8\Delta^2}{\sigma^2}$, is not as optimal as ρ-zCDP. Therefore, we use the noise mechanism of ρ-zCDP (Lemma A.5) when $\omega \to \infty$.

A.4 Privacy amplification by subsampling

In stochastic gradient descent, a batch of data subsampled from the whole dataset is used to update models. It is critical for implementing scalable learning algorithms. Because of the randomness of subsampling, it provably reduce the privacy cost. Technically, there are two ways to subsample the batch. One is random sampling without replacement or reshuffling (RF) which is widely used in the non-private deep learning. Yu et al. (2019) proved the composed privacy cost is the maximum of batch costs in one RF epoch. Numerically, each batch is $q\rho$-zCDP if the full batch cost is ρ and batch sample rate is q. In this case, the dynamic budget allocation for batches within one epoch is always worse than the uniform schedule.

The other strategy is the random sampling with replacement (RS), for example, in SGD-MA for the private deep learning (Abadi et al. 2016). Compared to RF, RS injects more randomness and therefore scale down the privacy cost more (Yu et al. 2019), for example, a q^2 factor in the MA. The lack of privacy amplification for RS motivates the development of extensions. Both tCDP (Bun et al. 2018) and the modified zCDP (Yu et al. 2019) spot the issue theoretically and provide similar solutions by truncating the order of Rényi divergence. A privacy amplification of tCDP is given Theorem A.3.

Theorem A.3 (Privacy amplification by subsampling for (ρ, ω)-tCDP)**.** *Let $\rho, q \in (0, 0.1]$ and positive integers n, N with $q = n/N$ and $\log(1/q) \ge 3\rho(2 + \log_2(1/\rho))$. Let $M : \mathcal{D}^n \to \mathbb{R}$ satisfy (ρ, ω')-tCDP for $\omega' \ge \frac{1}{2\rho}\log(1/q) \ge 3$. Define the mechanism $M_s : \mathcal{D}^N \to \mathcal{Y}$ by $M_s(x) = M(x_S)$, where $x_S \in \mathcal{D}^N$ is the restriction of $x \in \mathcal{D}^N$ to the entries specified by a uniformly random subset $S \subseteq [N]$ with $|S| = n$.*

The algorithm $M_s : \mathcal{D}^N \to \mathcal{Y}$ satisfies $(13q^2\rho, \omega'')$-tCDP for $\omega'' = \frac{\log(1/q)}{4\rho}$.

In comparison, the modified zCDP does not have a strict theoretic proof of the scale factor of the privacy cost but empirically shows that $q^2\rho$ works for a wide range of ρ. Here, we use tCDP to derive the range of privacy parameters.

In Theorem A.3, ω'' is a variable depending on the ρ rather than ω'. Thus, we let $\omega' \to \infty$ to degrade (ρ, ω')-tCDP as ρ-zCDP when $\rho \in (0, 16)$.

Recall our target is to simplify the tCDP by eliminating ω. Because the subsampled mechanism also satisfies $(13q^2\rho, \omega_a)$-tCDP if $\omega_a \le \omega''$, we constrain ρ as

$$\rho \le \min\left\{\frac{\log(1/q)}{4\omega_a}, \frac{\log(1/q)}{6}, \frac{\rho_a}{13q^2}\right\} \tag{13}$$

where $\frac{\rho_a}{13q^2}$ comes from the constraint of (ρ, a)-ctCDP on $13q^2\rho$. Typically, when $\omega_a > 1.5$, the $\log(1/q)/6$ can be ignored. Because 1.5 is too small to reach for ω_a in practice, we may assume it is satisfied generally.

B Methodology supplementaries

B.1 Model-based private learning

Here, we provide the formal statement and proof of Theorem 4.1.

Theorem B.1 (Privacy guarantee of model-based gradient descent)**.** *Suppose a gradient-based algorithm Algorithm 1 is protected by Algorithm 2 and $\sigma(\cdot)$ and $\pi(\cdot)$ are crafted fully independently from the private data. The output of the algorithm, i.e., θ_T (assuming the loops stop at step T), is $\hat{\rho}$-ctCDP where $\hat{\rho} \le \rho_{tot}$, if $f_C(\cdot)$, $f_S(\cdot)$ and $\rho(\cdot)$ are defined based on ctCDP properties (Lemmas A.4 to A.6).*

Proof. For brevity, we omit the a in notations. Denote the sub-routine defined in Algorithm 2 is $g_t, \rho_t, z_{t+1} = M_t(\nabla_t, z_t)$ where z_t denotes the hidden states. Then each iteration of private learning in Algorithm 1 can be abstracted as $\theta_{t+1}, z_{t+1} = A_t(\theta_t, g_t, z_t, \rho_t)$. Because of the linear composition, Lemma A.4, the condition $f_C(\rho_{1:t}) > \rho_{\text{tot}}$ can be justified by $\rho_{\text{residual}} > 0$ where $\rho_{\text{residual}} \leftarrow \rho_{\text{residual}} - \rho_t$.

By rearranging variables, without changing the meaning of the mappings, we can write the iteration as $\theta_{t+1}, z_{t+1} = A_t(M_t(\theta_t, z_t))$ where d denotes the private batch data. Suppose (θ_t, z_t) is $\hat{\rho}_t$-ctCDP w.r.t. the dataset and the mapping $M_t(\cdot, \cdot)$ is ρ_t-ctCDP w.r.t. the dataset. Thus, according to Lemma A.4, $\theta_{t+1}, z_{t+1} = A_t(M_t(\theta_t, z_t))$ is $\hat{\rho}_{t+1}$-ctCDP where $\hat{\rho}_{t+1} = \hat{\rho}_t + \rho_t$.

Next, we show $M_t(\cdot, \cdot)$ is ρ_t-ctCDP for $t \le T$ and some $\rho_t < \infty$. According to Lemma A.5, the noised gradient is $1/2\sigma_t^2$-zCDP and the noised gradient norm is $1/2\sigma_g^2$-zCDP (note its sensitivity is proved by Lemma A.3). Further using the Lemma A.6, we can compute

$$\rho_t = 13q^2\left(\frac{1}{2\sigma_t^2} + \frac{1}{2\sigma_g^2}\right) < \infty$$

if σ_t and σ_g are non-zero.

Now we show $A_1(M_1(\theta_1, z_1))$ is ρ_1-ctCDP. Typically, θ_1, z_1 are randomly initialized or constantly zero which are independent from the dataset. Therefore, (θ_1, z_1) is 0-ctCDP. By Lemma A.4, because $M_1(\cdot, \cdot)$ is ρ_1-ctCDP, $A_1(M_1(\theta_1, z_1))$ is ρ_1-ctCDP.

In summary, the output of model-based private learning, i.e., $\theta_T = A_T(M_T(\theta_T, z_T))$ (omitting z_{T+1}), is $\hat{\rho}_T$-ctCDP where

$$\hat{\rho}_T = \hat{\rho}_{T-1} + \rho_T = \sum_{i=1}^{T} \rho_i \leq \rho_{\text{tot}}.$$

□

B.2 Augmented Lagrangian algorithm

Given $\mu_0 > 0$, tolerance $\tau_0 > 0$ (Nocedal and Wright 1999) (Chapter 17), starting point σ_0 and λ^0, the variables are iteratively updated:

1. Line search s such that σ^+ is an approximate minimizer of $\mathcal{L}_{\text{aug}}$ (the gradient norm is less than τ_k):
$$\sigma^+ = \sigma - s\left[\nabla_\sigma F(T, \sigma_T) + \frac{dh}{d\sigma}(z - h(\sigma)/\mu)\right] \quad (14)$$
2. If the final convergence criteria satisfied, stop with approximate solution σ
3. Update Lagrange multiplier:
$$z^+ = z - h(\sigma^+)/\mu \quad (15)$$
4. Choose new penalty parameter $\mu^+ \in (0, \mu)$.

where s is the step size and we let σ be a vector $[\sigma_1, \cdots, \sigma_T]^\top$ or constant scalar. The update on σ^+ can be replaced by another line search, i.e., $\sigma^+ = \arg\min_{\sigma'} \mathcal{L}_{\text{arg}}(\sigma'(s))$ where $\sigma'(s)$ is given by Eq. (14). In practice, we want to avoid the second time of unrolling σ because it is required in Eq. (15). To fix this issue, we proceed with steps 3, 4, first and then finally perform step 1.

B.3 Analysis of the gradients

A generic gradient descent method can be summarized as a set of sequential updates on the parameter θ, i.e.,

$$\begin{aligned}\theta_T &= \theta_1 + \sum_{t=1}^{T-1} g_t \\ &= \theta_1 + \sum_{t=1}^{T-1} \pi(\nabla_t + \sigma_t \nu_t),\ \nu_t \sim \mathcal{N}(0, I).\end{aligned}$$

Assume $\frac{\partial \sigma_t}{\partial \sigma_{t-1}} = 0$ and $\frac{\partial g_t}{\partial \sigma_{t-1}} = 0$. Therefore, we can compute the gradient w.r.t. σ_t as

$$\begin{aligned}\frac{\partial f_T}{\partial \sigma_t} &= \frac{\partial \tilde{\nabla}_t}{\partial \sigma_t} \frac{\partial g_t}{\partial \tilde{\nabla}_t} \frac{\partial f_T}{\partial \theta_T} \\ &= \nu_t^\top \frac{\partial \pi(\tilde{\nabla}_t)}{\partial \tilde{\nabla}_t} \nabla_T \quad (16) \\ &= \frac{1}{\sigma_t}(\tilde{\nabla}_t - \nabla_t)^\top \frac{\partial \pi(\tilde{\nabla}_t)}{\partial \tilde{\nabla}_t} \nabla_T \\ &\approx \frac{1}{\sigma_t}(\pi(\tilde{\nabla}_t) - \pi(\nabla_t))^\top \nabla_T \quad (17)\end{aligned}$$

where $\tilde{\nabla}_t = \nabla_t + \sigma_t \nu_t$ and we approximate in the last term by Taylor expansion. Taking expectation, we can see from Eq. (16) that the gradient is related to the covariance between the noise ν_t and the final gradient ∇_T. Intuitively, if $\frac{\partial \sigma_t}{\partial \sigma_{t-1}} = 0$ and $\frac{\partial g_t}{\partial \sigma_{t-1}} = 0$, the gradient updates on σ_t will increase a lot when $t \ll T$. Together with the observation in Eq. (17), it can be witnessed that the scheduler is decided by the denoising effect of the π.

In our implementation, we use an RNN to model the π and σ which could greatly denoise the updates according to its memory. Furthermore, in Appendix B.6, we give an exact bound of the utility which closely relate the scheduler and projector together.

B.4 Optimality Stability

The optimality of learning scheduler is achieved by vanishing the gradient in Eq. (3). If both of the two terms in Eq. (3) are zero, the second term will be quite unstable since the f_t and $\hat{F}$ are random variables. Especially when the expectation of the gradient is estimated by only few samples, the instability will be a major issue. Here, we focus on the second term and analyze the probability for it to be zero. Assume the optimization of f has converged and the expectation of the gradient is estimated by one sample.

Non-batch algorithms. An optimal case for the non-batch algorithm is $f_t = f_{t'}$ for all $t, t' \in \mathcal{T}$ where $\mathcal{I}_t \neq 0$ and $\mathcal{I}_{t'} \neq 0$. By convergence, we assume the expected loss values cannot be decreased any more since the private updates are totally governed by the noise instead of gradients. Therefore, we can further assume the losses are identically independently distributed with variance $\text{Var}[f_t] \equiv \sigma_f^2$. According to the Chebyshev's inequality, we have, for a constant ξ,

$$P(|f_{t'} - f_t| > \xi) \leq \frac{2\sigma_f^2}{\xi^2}$$

which does not vanish since σ_f is non-zero due to private noise. That means only using one optimization process can barely reach the zero gradient condition.

Batch algorithms. The condition $f_t = f_{t'}$ can be easily extended to the batch case, i.e.,

$$\frac{1}{|\mathcal{B}_j|} \sum_{t \in \mathcal{B}_j} f_t = \frac{1}{|\mathcal{B}_i|} \sum_{t \in \mathcal{B}_i} f_t \quad (18)$$

for all $i, j \in \mathcal{T}_B$ where $\mathcal{I}_i \neq 0$ and $\mathcal{I}_j \neq 0$. Therefore,

$$P\left(\left|\frac{1}{|\mathcal{B}_i|}\sum_{t\in\mathcal{B}_j} f_t - \frac{1}{|\mathcal{B}_i|}\sum_{t\in\mathcal{B}_i} f_t\right| > \xi\right) \leq \frac{2\sigma_f^2}{\xi^2|\mathcal{B}_i|}$$

which has a smaller failure probability if $|\mathcal{B}_i| > 1$. In other words, the batch algorithm is stabler.

B.5 Batch Augmented Lagrangian Algorithm

With the basic augmented objective Eq. (2), we can extend it to the batch case, i.e.,

$$L_b^{\text{aug}}(\sigma; r_b) = \mathbb{E}[\bar{F}_i] - z_b h_b + \frac{\|h_b\|_2^2}{2\mu_b}$$

where $\bar{F}_i$ is the batch-averaged loss defined in Eq. (5). After decomposing the budget constraint into batches, the augmented objective on the whole optimization process has to be replaced by

$$\mathcal{L}_b^{\text{aug}}(\sigma, r; \rho_{\text{tot}}) = \sum_{b=1}^{B} L_b^{\text{aug}}(\sigma; r_b) - z_r h_r + \frac{\|h_r\|_2^2}{2\mu_r},$$

which constrains the b-th-batch privacy cost by r_b and the overall cost by ρ_{tot}. Note that z_b will be gradually reduced to zero when the batch constraint is getting tighter. With the Lagrangian multiplier z_b, the batch scheduler will be allowed to fetch needed budget slightly ignoring the constraint r_b. Therefore, we can define $\hat{\rho}_b = f_C(\{\rho(\sigma_t)\}_{t\in\mathcal{B}_b}) - z_b\mu_b$ to be the batch privacy cost supplemented by the Lagrangian multiplier.

With the constraint decomposition, we can update the σ only using one batch loss $L_b^{\text{aug}}(\sigma; r_b)$ independently if r_b is fixed. Then, we update r by optimizing Eq. (6), i.e.,

$$\mathcal{L}^{\text{aug}}(r; \rho_{\text{tot}}) = \sum_{b=1}^{B} \frac{\|\hat{\rho}_b - r_b\|_2^2}{2\mu_b} - z_r h_r + \frac{\|h_r\|_2^2}{2\mu_r}$$

We conceptually illustrate the enforcement of constraints between batch budget r_b and the global budget ρ_{tot} in Fig. 5. The global budget allocation will be enforced to align the total budget. However, the relation between the batch cost and the batch budget are bi-directions. When we optimize w.r.t. r, the r_b will also be encouraged to align the $\hat{\rho}_b$. When we optimize w.r.t. ρ_t in batch b, the batch privacy cost will be enforced to match the budget r_b.

In the unconstrained batch algorithm, we will use an one-pass fashion to update the meta-model, i.e., the optimizer. That means we do not need to store any batch data (except for the meta-model) that has been used, which could greatly reduce the space complexity. However, for budget constrained L2P, we cannot directly drop the used batches, since the batch state is essential to check if we need to adjust the constraint to fulfill the budget requirement. In Fig. 5, the dependency is represented by the interaction between the batch privacy cost and the constraint.

Rather than a one-pass method, we suggest a two-pass way to update the parameters. First, we update the meta-models with one pass. Second, by fixing the meta-models, we unroll the protected learning, update and store $\hat{\rho}_b$, z_b and μ_b. With the recorded data, we minimize $\mathcal{L}^{\text{aug}}(r; \rho_{\text{tot}})$ w.r.t. r and update corresponding AL variables.

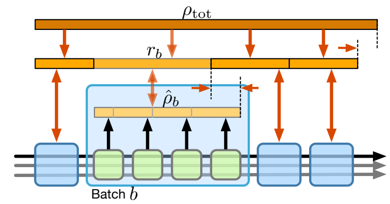


Figure 5: Illustration of the privacy budget constraint in batches. We strip the output of the protectors to be only the privacy costs which are composed into the batch cost $\hat{\rho}_b$ supplemented by the Lagrangian multiplier. The red arrows between r_b and $\hat{\rho}_b$ are the interaction during L2P optimization where they will both be forced to match the value of the other.

B.6 Optimality analysis

In this paper, we cast the searching for the optimal scheduler as a learning problem. Formally, we minimize the objective:

$$\mathbb{E}[\hat{F}(\sigma)] = \mathbb{E}\left[\frac{\sum_{t=1}^{T}\mathcal{I}_t f_t}{\sum_{t=1}^{T}\mathcal{I}_t}\right].$$

We first assume $\mathcal{I}_t$ is a general weight function varying by t. Recall the gradient of $\hat{F}(\sigma)$ w.r.t. σ (Eq. (3)) is

$$\partial\hat{F}(\sigma) = \frac{\sum_{t=1}^{T}\mathcal{I}_t\partial f_t}{\sum_{t=1}^{T}\mathcal{I}_t} + \frac{\sum_{t=1}^{T}(f_t - \hat{F})\partial\mathcal{I}_t}{\sum_{t=1}^{T}\mathcal{I}_t}. \tag{19}$$

By vanishing the gradient, we can get the optimal condition of the L2P objective.

We restrict the non-zero range of $\mathcal{I}_t$ within $[T_\rho - 1, T_\rho]$. Define the weight $\alpha = \mathcal{I}_{T_\rho-1}/(\mathcal{I}_{T_\rho-1} + \mathcal{I}_{T_\rho})$.

We first summarize the major results in Theorem B.2, with which we can get an approximated convergence guarantee to some (local or global) solution of our objective f. It gives us the insight that the final gradient norm is upper bounded by the covariance between the accumulated noise variables (transformed by $\frac{\partial g_t}{\partial\tilde{\nabla}_t}$) and the final gradient. The upper bound will be improved when we train the projector on a fixed scheduler. In short words, to improve the utility, the projector training have to denoise the protected updates which reduces the covariance between the ν_t and the g_t. And the covariance between ν_t and f_{t+1} is reduced meanwhile. As a result, we will see $\mathbb{C}_{T_\rho-1}$ approaching zero.

Definition B.1 (L-smooth function)**.** A differentiable objective function $f : \Theta \times \mathcal{X} \to \mathbb{R}$ is L-smooth over $\theta \in \Theta$ with respect to the norm $\|\cdot\|$ if for any $x \in \mathcal{X}$ and $\theta_1, \theta_2 \in \Theta$, we have:

$$\|\nabla f(\theta_1, x) - \nabla f(\theta_2, x)\|_* \leq L\|\theta_1 - \theta_2\|,$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. If $\|\cdot\|$ is l_2-nrom, this yields

$$f(\theta_1, x) - f(\theta_2, x) \leq \nabla^\top f(\theta_2, x)(\theta_1 - \theta_2) + \frac{L}{2}\|\theta_1 - \theta_2\|^2.$$

Theorem B.2 (Utility bound of a stationary L2P protector)**.** *Suppose f is L-smooth and σ is independent from the noise* *variables ν_t. If the σ is a stationary point of the constrained optimization problem, the following is satisfied:*

$$\mathbb{E}\left\|\nabla_{T_\rho-1}\right\|^2 \leq \sigma^2_{\zeta_{T_\rho-1}} p - \frac{2L}{C_\sigma}(\alpha \mathbb{C}_{T_\rho-1} + (1-\alpha)\mathbb{C}_{T_\rho}), \tag{20}$$

where C_σ is a constant depending on the scheduler σ (Eq. (24)), $\mathbb{C}_T$ represents the covariance between the noise and the true gradient (Eq. (32)) and $\sigma_{\zeta_{T_\rho-1}}$ is the upper bound of reduced noise variance (Eq. (27)).

Proof. Generally, we assume the $\mathcal{I}_t$ be a function of h_t whose gradient is

$$\partial \mathcal{I}_t = \frac{\partial \mathcal{I}_t}{\partial \sigma} = \frac{\partial \mathcal{I}_t}{\partial h_t}\frac{\partial h_t}{\partial \sigma}.$$

Let us first look into $\frac{\partial h_t}{\partial \sigma_t}$ which is

$$\frac{\partial h_t}{\partial \sigma} = \sum_{\tau=1}^{t} \frac{\partial \rho_\tau}{\partial \sigma_\tau}\frac{\partial \sigma_\tau}{\partial \sigma} = -\sum_{\tau=1}^{t} \frac{\Delta^2}{\sigma_\tau^3}\frac{\partial \sigma_\tau}{\partial \sigma} = -\sum_{\tau=1}^{t} \frac{2\rho_\tau}{\sigma_\tau}\frac{\partial \sigma_\tau}{\partial \sigma}$$

$$\frac{\partial h_t}{\partial \sigma} = \frac{\partial h_{t-1}}{\partial \sigma} - \frac{2\rho_t}{\sigma_t}\frac{\partial \sigma_t}{\partial \sigma}$$

if the ctCDP is utilized. Because $\mathcal{I}_t$ is always centered around some real value t_0 for $h_{t_0} = 0$ (by continuous approximation), we may assume $\frac{\partial \mathcal{I}_t}{\partial h_t}$ of different signs on the different sides of t_0.

Let the gradient be zero and rearrange the variables.

$$\frac{1}{Z_T}\sum_{t=1}^{T}\mathcal{I}_t \partial f_t = -\frac{1}{Z_T}\sum_{t=1}^{T}(f_t - \hat{F})\partial \mathcal{I}_t \tag{21}$$

where $Z_T = \sum_{t=1}^T \mathcal{I}_t$. Let T_ρ be the integer such that $h_{T_\rho} = \xi > 0$ and $h_{T_\rho - 1} = \xi' = \xi - \rho_T < 0$. Denote the left-hand-side and right-hand-side of Eq. (21) as lhs and rhs.

We restrict the non-zero range of $\mathcal{I}_t$ within $[T_\rho - 1, T_\rho]$. Then,

$$\mathcal{I}_{T_\rho} = 1 - \gamma h_{T_\rho} = 1 - \gamma\xi$$

$$\mathcal{I}_{T_\rho-1} = 1 + \gamma h_{T_\rho-1} = 1 + \gamma\xi' = 1 + \gamma\xi - \gamma\rho_{T_\rho}$$

whose summation is $2 + \gamma(\xi' - \xi) = 2 - \gamma\rho_{T_\rho}$ and gradients are:

$$\partial \mathcal{I}_{T_\rho} = -\gamma\frac{\partial h_{T_\rho}}{\partial \sigma} = \gamma\sum_{\tau=1}^{T_\rho}\frac{2\rho_\tau}{\sigma_\tau}\frac{\partial \sigma_\tau}{\partial \sigma} \tag{22}$$

$$\partial \mathcal{I}_{T_\rho-1} = \gamma\frac{\partial h_{T_\rho-1}}{\partial \sigma} = -\gamma\sum_{\tau=1}^{T_\rho-1}\frac{2\rho_\tau}{\sigma_\tau}\frac{\partial \sigma_\tau}{\partial \sigma}. \tag{23}$$

Since $\alpha = \mathcal{I}_{T_\rho-1}/(\mathcal{I}_{T_\rho-1} + \mathcal{I}_{T_\rho})$,

$$\begin{aligned}\text{rhs} &= -\frac{1}{Z_T}\sum_{t=1}^{T}(f_t - \hat{F})\partial\mathcal{I}_t \\ &= -\frac{1}{Z_T}[(1-\alpha)\partial\mathcal{I}_{T_\rho-1} - \alpha\partial\mathcal{I}_{T_\rho}](f_{T_\rho-1} - f_{T_\rho})\end{aligned}$$

If σ is independent from the noise variables ν_t, e.g., uniform schedule, the coefficient is a constant, i.e.,

$$C_\sigma = -\frac{1}{Z_T}[(1-\alpha)\partial\mathcal{I}_{T_\rho-1} - \alpha\partial\mathcal{I}_{T_\rho}] \tag{24}$$

based on which we can get the expectation,

$$\mathbb{E}[\text{rhs}] = C_\sigma \mathbb{E}[f_{T_\rho-1} - f_{T_\rho}]. \tag{25}$$

If f is L-smooth (Definition B.1),

$$\mathbb{E}\left[f_{T_\rho} - f_{T_\rho-1}\right] \leq \mathbb{E}[\nabla^\top_{T_\rho-1} g_{T_\rho-1} + \frac{L}{2}\left\|g_{T_\rho-1}\right\|^2].$$

Define $\zeta_T \triangleq L g_T + \nabla_T$. Since π is just a variant of the SGD, the direction of g_T should be opposite to the ∇_T. Therefore,

$$\zeta_t = L(\pi(\nabla_t + \sigma_t\nu_t) - \frac{\nabla_t}{L}), \tag{26}$$

which represents the difference between the projected updates (g_T) and the gradient descent update with the step size $\frac{1}{L}$. A rational guess is that the $\zeta_{T_\rho-1}$ is the residual noise noise after the denoising operation, π. Thus, it is rational to assume the $\mathbb{E}\left\|\zeta_{T_\rho-1}\right\|^2$ is bounded as

$$\mathbb{E}\left\|\zeta_{T_\rho-1}\right\|^2 \leq \sigma^2_{\zeta_{T_\rho-1}} p \tag{27}$$

for some parameter $\sigma_{\zeta_{T_\rho-1}}$ depending on the scheduler where p is the dimension of θ. Then

$$g_t = -\frac{1}{L}\nabla_t + \frac{1}{L}\zeta_t,$$

which leads to

$$\begin{aligned}\mathbb{E}\left[f_{T_\rho} - f_{T_\rho-1}\right] &\leq -\frac{1}{2L}\mathbb{E}\left\|\nabla_{T_\rho-1}\right\|^2 + \frac{1}{2L}\mathbb{E}\left\|\zeta_{T_\rho-1}\right\|^2 \\ &\leq -\frac{1}{2L}\mathbb{E}\left\|\nabla_{T_\rho-1}\right\|^2 + \frac{\sigma^2_{\zeta_{T_\rho-1}} p}{2L}.\end{aligned}$$

Thus,

$$\begin{aligned}\mathbb{E}\left\|\nabla_{T_\rho-1}\right\|^2 &\leq \sigma^2_{\zeta_{T_\rho-1}} p + 2L\mathbb{E}\left[f_{T_\rho-1} - f_{T_\rho}\right] \\ &= \sigma^2_{\zeta_{T_\rho-1}} p + 2L\mathbb{E}[\text{rhs}]/C_\sigma.\end{aligned} \tag{28}$$

Thus, we complete the discussion of the rhs.

To the left-hand-side of Eq. (19), we first calculate the derivatives of the loss functions,

$$\partial f_T = \left(\sum_{t=1}^{T-1}\frac{\partial \sigma_t}{\partial \sigma}\nu_t^\top \frac{\partial g_t}{\partial \tilde{\nabla}_t}\right)\nabla_T. \tag{29}$$

Define a random variable as

$$V_T^\top \triangleq \sum_{t=1}^{T}\frac{\partial \sigma_t}{\partial \sigma}\nu_t^\top\frac{\partial(-g_t)}{\partial \tilde{\nabla}_t}, \tag{30}$$

where the negative sign is added because g_t is usually the opposite to the $\tilde{\nabla}_t$, for example, $g_t \propto -\tilde{\nabla}_t$ in SGD. Now we substitute Eqs. (29) and (30) into Eq. (19) to obtain

$$\text{lhs} = -\alpha V^\top_{T_\rho-2}\nabla_{T_\rho-1} - (1-\alpha)V^\top_{T_\rho-1}\nabla_{T_\rho}.$$

For brevity, we rewrite the expectation as

$$\mathbb{E}[\text{lhs}] = -\alpha \mathbb{C}_{T_\rho - 1} - (1-\alpha)\mathbb{C}_{T_\rho}, \tag{31}$$

where we define

$$\mathbb{C}_t = \mathbb{E}[V_{t-1}^\top \nabla_t] = \sum_{i=1}^{p} \mathrm{Cov}(V_{t-1,i}, \nabla_{t,i}), \tag{32}$$

where we utilize $\mathbb{E}V_{T-1} = 0$ because ν_t is an i.i.d. Gaussian random vector. Therefore, $\mathbb{C}_T$ represents the covariance between two vectors and will be zero only when the two vectors are uncorrelated. Combining Eqs. (25), (28) and (31), we can get Eq. (20). $\square$

Uniform scheduler. We assume the $\sigma_t \equiv \sigma$ where the scheduler degrades as a constant σ. Therefore, with Eqs. (22) and (23), we have

$$T_\rho = \lceil \rho_{\text{tot}}/\rho \rceil, \; \xi = \rho T_\rho - \rho_{\text{tot}}$$
$$\mathcal{I}_{T_\rho} = 1 - \gamma\xi, \; \mathcal{I}_{T_\rho - 1} = 1 + \gamma\xi - \gamma\rho$$
$$\partial \mathcal{I}_{T_\rho} = \gamma T_\rho \frac{2\rho}{\sigma}, \; \partial \mathcal{I}_{T_\rho-1} = -\gamma(T_\rho - 1)\frac{2\rho}{\sigma}$$

where $\rho = \Delta^2/2\sigma^2$. In addition, we have

$$f_{T_\rho} - \hat{F} = \alpha(f_{T_\rho} - f_{T_\rho - 1})$$
$$f_{T_\rho - 1} - \hat{F} = -(1-\alpha)(f_{T_\rho} - f_{T_\rho - 1}).$$

Substitute what we have into Eq. (21) giving

$$\begin{aligned}\text{rhs} &= \frac{1}{Z_T}\left[\alpha T_\rho + (1-\alpha)(T_\rho - 1)\right]\gamma \frac{2\rho}{\sigma}(f_{T_\rho-1} - f_{T_\rho}) \\ &= C_\sigma (f_{T_\rho - 1} - f_{T_\rho}),\end{aligned}$$

where we update the constant C_σ from Eq. (24) as

$$C_\sigma = \frac{T_\rho - (1-\alpha)}{2 - \gamma\rho}\frac{2\rho\gamma}{\sigma}.$$

If $\gamma \in (0, 2/\rho)$, then $C_\sigma > 0$. In our implementation, the condition always holds since $\gamma = 1/\rho_{\text{tot}} < 2/\rho$.

Taking the expectations of lhs and rhs, we have:

$$-\alpha \mathbb{C}_{T_\rho - 1} - (1-\alpha)\mathbb{C}_{T_\rho} = C_\sigma \mathbb{E}[f_{T_\rho - 1} - f_{T_\rho}], \tag{33}$$

where $\mathbb{C}_T$ is given by substituting $\frac{\partial \sigma_t}{\partial \sigma} = 1$ into Eq. (30) and its definition, i.e.,

$$\mathbb{C}_T = \mathbb{E}\left[\sum_{t=1}^{T-1} \nu_t^\top \frac{\partial(-g_t)}{\partial \tilde{\nabla}_t} \nabla_T\right].$$

Analysis of batch algorithm. Recall the objective for the batch algorithm (Eq. (8)) is

$$\mathcal{L}^{\text{aug}}(r) = \sum_{b=1}^{B} \frac{1}{2\mu_b}\|\hat{\rho}_b - r_b\|_2^2 + \frac{\sum_{b \in \mathcal{T}_B} \mathcal{I}_b \bar{F}_i}{\sum_{b \in \mathcal{T}_B} \mathcal{I}_b}$$

where $Z_B = \sum_{b \in \mathcal{T}_B} \mathcal{I}_b$ and we use $\mathcal{I}_b$ to denote $\mathcal{I}(h_b)$. Generally, we assume the batch budget is scheduled by parameterized model $r(\cdot)$ or r for simplicity, e.g., LSTMs. In addition, assume $\Delta B = 1$, $\alpha = \mathcal{I}_{B_\rho - 1}/\mathcal{I}_{B_\rho - 1} + \mathcal{I}_{B_\rho}$. Vanishing $\frac{\partial \mathcal{L}^{\text{aug}}}{\partial r}$ causes

$$\sum_{b=1}^{B} \frac{1}{\mu_b}(r_b - \hat{\rho}_b)\frac{\partial r_b}{\partial r} = -\frac{\sum_{b \in \mathcal{T}_B}(\bar{F}_b - \hat{F})\partial \mathcal{I}_b}{Z_B}, \tag{34}$$

where $Z_B = \sum_{b \in \mathcal{T}_B} \mathcal{I}_b$ and we define the notation $\overline{x_t}^b = \frac{1}{m}\sum_{t \in \mathcal{B}_b} x_t$ for any variables x_t related to the step t. In addition, we need to make the gradient of batch objective be zero, i.e.,

$$0 = \frac{1}{m}\sum_{t \in \mathcal{B}_b} \frac{\partial f_t}{\partial \sigma} + \frac{1}{m\mu_b}(\hat{\rho}_b - r_b)\sum_{t \in \mathcal{B}_b} \frac{\partial \rho_t}{\partial \sigma}, \tag{35}$$

where we let the batch size, $|\mathcal{B}_b|$, be m for any b. If the equalities hold in Eqs. (34) and (35), we can extend non-batch utility bound, Theorem B.2, to the batch version in Theorem B.3. Compared to the non-batch result, the batch utility bound is extended by the average of steps in batches. For example, $\mathbb{C}$ is replaced by $\overline{\mathbb{C}}$.

Theorem B.3 (Utility bound of batch L2P protector)**.** *If f is L-smooth and σ and r are independent from the noise variables ν_t, then we have:*

$$\mathbb{E}\overline{\|\nabla_t\|^2}^{B_\rho - 1} \le p\overline{\sigma^2_{\zeta_t}}^{B_\rho - 1} - \frac{2L}{C_r}\sum_{b=1}^{B} \frac{\overline{\mathbb{C}'_t}^b}{\overline{C'_{\sigma,t}}^b}\frac{\partial r_b}{\partial r}, \tag{36}$$

where $C_{\sigma,b}$ is a constant depending on the scheduler σ and the batch b (Eq. (42)), C_r is a constant depending on the batch scheduler r, $\mathbb{C}_t$ represents the covariance between the noise and the true gradient (Eq. (32)), and σ_{ζ_t} is the upper bound of reduced noise variance (Eq. (27)).

Proof. We can easily get the derivative $\partial \mathcal{I}_b$ based on Eqs. (22) and (23):

$$\partial \mathcal{I}_{B_\rho} = \gamma \sum_{b=1}^{B_\rho} \frac{\partial r_b}{\partial r}, \; \partial \mathcal{I}_{B_\rho - 1} = -\gamma \sum_{\tau=1}^{B_\rho - 1} \frac{\partial r_b}{\partial r}. \tag{37}$$

Still, we use lhs and rhs to denote the two sides of the Eq. (34). From the non-batch analysis, we can extend Eq. (25) as

$$\mathbb{E}[\text{rhs}] = C_r \mathbb{E}[\bar{F}_{B_\rho - 1} - \bar{F}_{B_\rho}] \tag{38}$$

$$C_r = -\frac{1}{Z_B}[(1-\alpha)\partial \mathcal{I}_{B_\rho - 1} - \alpha \partial \mathcal{I}_{B_\rho}] \tag{39}$$

where we still assume $r(\cdot)$ is independent from the private noise which makes C_r constant.

Consider the case when the f is L-smooth. Thus,

$$\mathbb{E}\left[f_t - f_{t-m}\right] \le \mathbb{E}[\nabla_{t-m}^\top g_{t-m} + \frac{L}{2}\|g_{t-m}\|^2],$$

for all t in $\mathcal{B}_{B_\rho}$. Averaging over t, we get:

$$\begin{aligned}\mathbb{E}\left[\bar{F}_{B_\rho} - \bar{F}_{B_{\rho-1}}\right] &\le \mathbb{E}[\frac{1}{m}\sum_{t \in \mathcal{B}_{B_\rho - 1}} \nabla_t^\top g_t + \frac{L}{2}\|g_t\|^2] \\ &\le -\frac{1}{2L}\mathbb{E}\overline{\|\nabla_t\|^2}^{B_\rho - 1} + \frac{p\overline{\sigma^2_{\zeta_t}}^{B_\rho - 1}}{2L},\end{aligned}$$

where we make use of ζ_t defined in Eq. (26) and its bound in Eq. (27). Combine this with Eqs. (34) and (38) to get

$$\overline{\mathbb{E}\|\nabla_t\|^2}^{B_\rho-1} \leq p\overline{\sigma^2_{\zeta_t}}^{B_\rho-1} + \frac{2L}{C_r}\sum_{b=1}^{B}\frac{1}{\mu_b}(r_b-\hat{\rho}_b)\frac{\partial r_b}{\partial r}. \tag{40}$$

To find the value of $\frac{1}{\mu_b}(r_b-\hat{\rho}_b)$, we need to use Eq. (35) which gives:

$$\frac{1}{\mu_b}(r_b-\hat{\rho}_b) = \frac{1}{m\overline{C'_{\sigma,b}}}\sum_{t\in\mathcal{B}_b}\frac{\partial f_t}{\partial\sigma}, \tag{41}$$

$$\overline{C'_{\sigma,t}}^b = \frac{1}{m}\sum_{t\in\mathcal{B}_b}\frac{\partial\rho_t}{\partial\sigma}. \tag{42}$$

According to Eqs. (29) and (30), it can be attained that

$$\mathbb{E}[\frac{1}{\mu_b}(r_b-\hat{\rho}_b)] = -\frac{\overline{\mathbb{C}'_t}^b}{\overline{C'_{\sigma,t}}^b} \tag{43}$$

where we modify Eqs. (30) and (32) as

$$\mathbb{C}'_t = \mathbb{E}[{V^b_{t-1}}^\top\nabla_t] = \sum_{i=1}^{p}\mathrm{Cov}(V^b_{t-1,i},\nabla_{t,i})$$

$$V^b_{T,i} = \sum_{t\in\mathcal{B}_b,t\leq T}\frac{\partial\sigma_t}{\partial\sigma}\nu_t^\top\frac{\partial(-g_t)}{\partial\tilde{\nabla}_t},\ T\in\mathcal{B}_b.$$

Substituting it into Eq. (40), we can get Eq. (36). This thus completes the proof. □

B.7 Implementation details

In this section, we present the implementation details for the the projector and scheduler models. We use the Long-Short Term Memory (LSTM) networks as the backbone models.

Constrain Noise-Scale Prediction. To stablize the L2P training, we explicitly constrain the range of the noise scale by using a Sigmoid activation in the scheduler. In addition, assuming the sigmoid output of the LSTM is y, we scale the output as

$$\sigma_{\min} + 2(\sigma_g - \sigma_{\min})y,$$

which is constrained in $(\sigma_{\min}, 2\sigma_g)$. The σ_{lb} is the lower bound of noise scale which derived from the upper bound of privacy budget, e.g., ρ_a for (ρ, a)-ctCDP. The σ_g is estimated by uniformly scheduling budgets. Generally, we will expect the predicted σ is centered around σ_g and is not too large, e.g., larger than $2\sigma_g$, which will violate the utility greatly. With the constraint, the noise prediction will not fluctuate significantly.

Coordinate-wise LSTM. Following the implementation in (Andrychowicz et al. 2016), we share the parameters of LSTM for all optimized parameters. Therefore, a small LSTM can work for optimizing large-scale neural networks.

Incremental Pre-training. Training an L2P model from scratch may suffer from a great amount of DP noise such that no useful information can be learned. For simple tasks, pre-training without noise can mitigate this noise gap since it could avoid some random optimization exploration at the beginning. For complicated tasks, e.g., deep neural networks or large-scale models, the gap between L2L models and high-privacy L2P models can still be huge. The DP noise is added without considering the scale of the model. Specifically, when the size of model parameters increases and the scale of their every coordinate decreases meanwhile, the DP noise will not change if the clipping norm is fixed. Thus, the noise is relatively amplified. Especially for deep models, the small coordinates may greatly affect the model performance and thus deep models are more sensitive to DP noise. Therefore, neither a scratch nor an L2L model could be robust enough as an initialization for the L2P model. Instead, we suggest an incremental pre-training in which the privacy scale ϵ will incrementally increase from 0.

C Additional experiments

C.1 Quadratic optimization

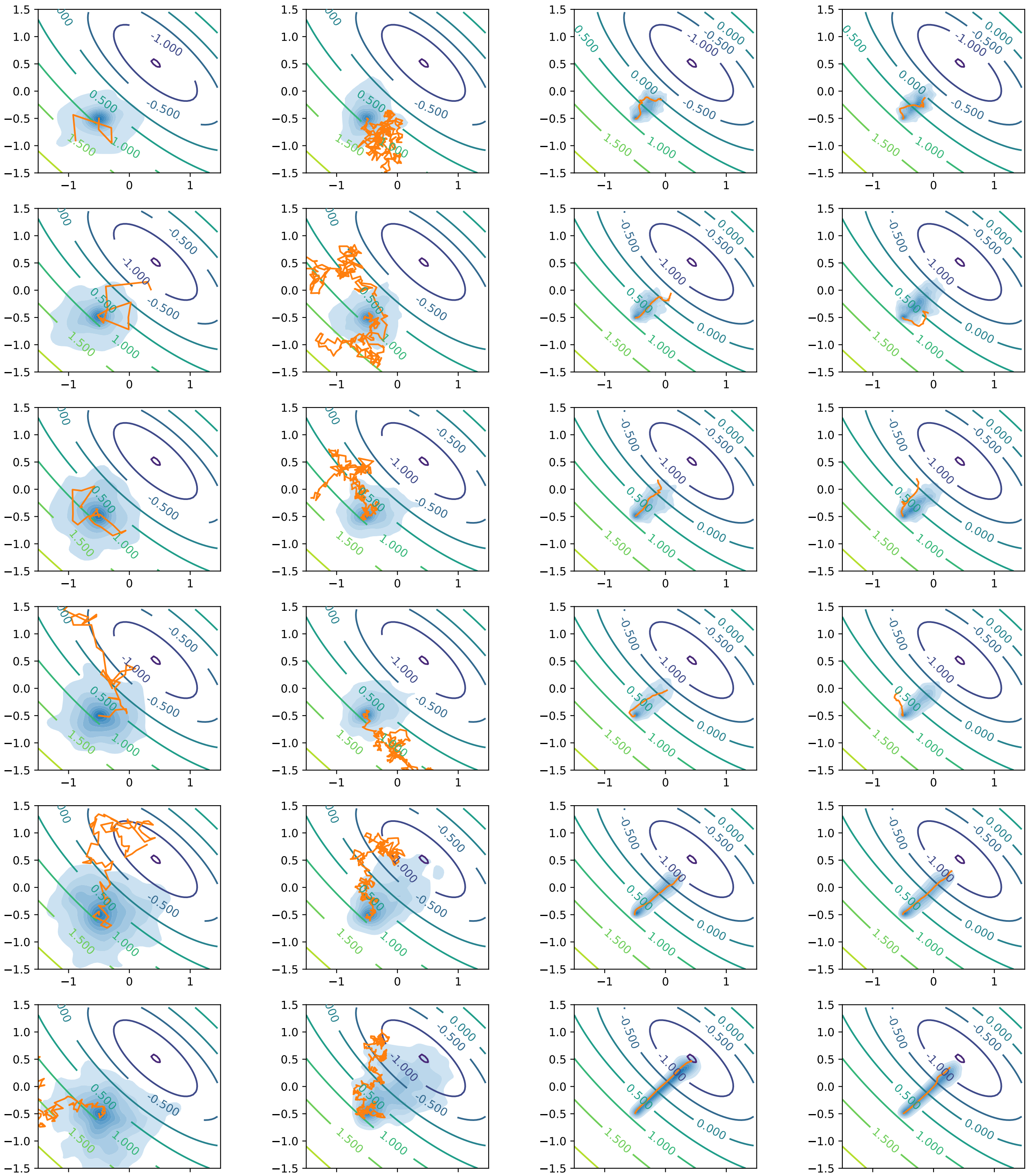
Setup. To show the optimality of L2P training, we compare different algorithms by non-privately tuning them. Formally, given a fixed size of privacy budget, we train or tune a private optimizer on the quadratic optimization problems:

$$\min_\theta f(\theta) = \sum\nolimits_{i=1}^{60}\|W_i\theta - y_i\|_2^2 + 0.001\,\|\theta\|_2^2\,,$$

with random constants $W_i \in \mathbb{R}^{2\times 2}$ and $y_i \in \mathbb{R}^2$ for $i \in \{1,\dots,60\}$. We note that the tuning/meta-training is non-private such that we can see if the L2P can converge to the best private optimizer on the auxiliary datasets in comparison to baselines.

L2P-Proj (L2P with only projector) and L2P models are trained independently. Hence we can see the effect of adaptive perturbation. All optimizers are only tested on identical W, y and initial variables. The L2P and L2P-Proj are trained with normally randomized W and y for 200 epochs after they are pre-trained without noise in the same way and the best model are selected with the lowest loss when their privacy budgets are used up in validation. The iteration numbers for SGD-Adv and L2P-Proj are chosen in range $\{10, 20, 30, 40, 50, 60\}$ which are enough for convergence of such quadratic problems. The step size is chosen from 0.001 to 0.02 with 20 choices for SGD-Adv, while AGD uses the line search in the same range with 20 choices.

Results. In Fig. 8, four optimization methods are compared at the same $(0.05, 10^{-8})$-DP. As shown in Fig. 8, the proposed L2P converges to the zone close to but not exactly at the noise-free optimal solution. The optimization algorithms stop before reaching the optimal, because of the imposed budget constraint. Recall that the model at the optimal solution may leak sensitive information. We see that L2P guides the optimization toward the optimal by adjusting the update directions. More importantly, L2P-Proj reduces the noise magnitude, uses more step budget but converges in less steps. Because L2P-Proj has omitted no budget scheduler, it stop in a different spot. In comparison, the SGD-Adv algorithm randomly walks in a rather large region. Though AGD reduces variances relatively, it barely finds the correct optimization direction.



(a) SGD-Adv (b) AGD (c) L2P-Proj (d) L2P

Figure 6: Comparison of the convergence $(\epsilon, 10^{-8})$-DP with ϵ varying as $0.05, 0.1, 0.2, 0.4, 0.8, 1.6$ from top to bottom.

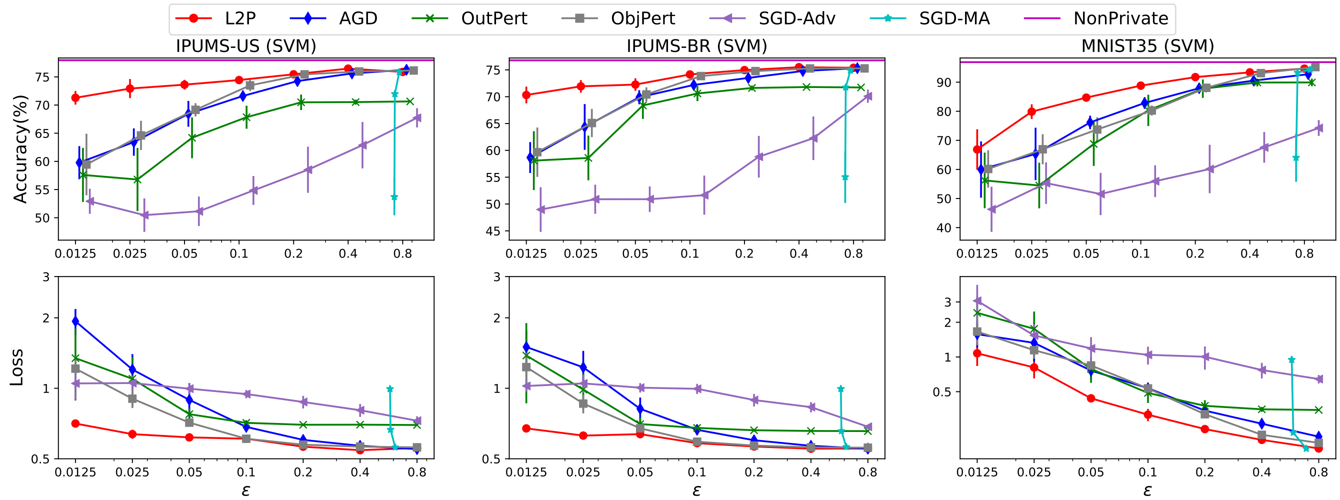
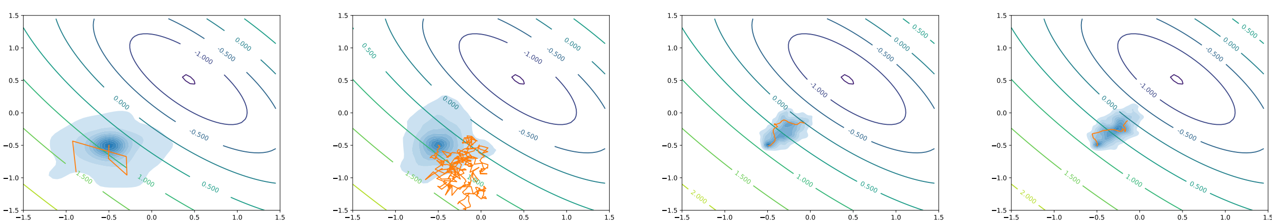


Figure 7: Test performance (top) and training loss values (bottom) by varying ϵ of SVMs classifiers on IPUMS and MNIST datasets. The error bar presents the size of standard deviations. For better visualization of error bars, some virtual horizontal offsets are added to every point.



(a) SGD-Adv (b) AGD (c) L2P-Proj (d) L2P

Figure 8: Comparisons of $(0.05, 10^{-8})$-DP algorithms on a quadratic problem. Solid contour lines illustrate the loss values. The trajectory distribution of 100 repeated optimizations are shown in blue shadowed contours. Sampled trajectories are plotted in orange.

Additional quadratic optimization results for different ϵ are shown in Fig. 6. Because the quadratic problem uses very few data and its gradients are in small scale, the optimization will be very sensitive to the noise. In this case, SGD-Adv rarely find the proper directions to go. In contrast, adaptive DP algorithms perform better. L2P-Proj behaves similarly to the L2P. However, when ϵ gets smaller, L2P is capable to use the budget more efficiently such that it can converge to a better position. Meanwhile, L2P-Proj cannot adaptively adjust its step budgets which make the execution length shorter. AGD shows some ability to correct the noised directions but it fails when the privacy constraint is higher.

C.2 Experiments of generalization to different distributions

In this section, we provide additional experiments for evaluating the generalization ability of L2P.

Experiments of SVMs. The results are reported in Fig. 7. The results are similar to the Logistic.

C.3 Classification on MNIST35 datasets with non-convex objectives and varying ϵ

In addition to convex objectives, we also evaluate our models on a popular non-convex model, neural networks. The evaluated network includes two layers of 20 and 2 units (for binary classification), respectively. The layers are connected with sigmoid activations. The loss is computed by the cross-entropy function.

Different from logistic and SVM models, the patterns of optimizing a neural network could be hard to learn for L2P. The first issue is that the relative magnitude of noise w.r.t. the gradient coordinates is enlarged when the size of the gradient increases. For MNIST35 images of $28 \times 28 = 784$ pixels and a network with 20 units in the first layer, the number of connection weights could be 20×784 which is 20 times of an SVM model. Since a constant L2 sensitivity, e.g., 2, is expected, the gradient norm will be less than 2, which makes each coordinate much smaller while the number of coordinates increases. Meanwhile, the scale of noise will not change for each coordinate, which means it increases in a relative way. This issue makes private learning methods hard to achieve the same utility performance under the same privacy requirement. As a result, we adjust the clipping norm to 2 which can slightly reduce the noise scale.

The second issue has been discussed in L2L (Andrychowicz et al. 2016). The optimization of the L2L or the L2P projector will encounter numerical issues for that the weights in different layers of the optimized networks have different magnitude. Since the optimization often focuses on the large elements, it will cause the optimizer barely updated and thus no convergence can be witnessed. As suggested by Andrychowicz *et al.*(Andrychowicz et al. 2016), either scaling the gradient by logarithm and sign mapping or scaling the output of the L2L by a constant can mitigate the issue. The former method augments the insignificant information in the gradients, while the second one resembles the learning rate such that the predicted updates will not fluctuate too much. To avoid that the gradient values are overwhelmed by noise and the L2P model absorb useless information, we recommend the latter method and use the scaling constant as 0.05.

In addition, optimizing small gradient coordinates with noise could be challenging. Incremental pre-training introduced in Appendix B.7 could reduce the hardness step by step through the step could be flexible. For example, when trained with $\epsilon = 0.2$, the L2P should be initialized by L2P without noise. When trained with $\epsilon = 0.05$, the L2P should be initialized by L2P with $\epsilon = 0.2$ instead. Other experimental settings follow the same principles in previous ones.

For a more precise trade-off between utility and privacy losses, a tuning of the privacy loss coefficient is necessary. A recommended range of the coefficient is $\{500, 1000, 5000\}$. Using the training data to monitor the convergence curves will be helpful for choosing a proper coefficient. Meanwhile, the estimated number of iterations which determines the initial privacy cost should be selected in $\{30, 60, 100, 400\}$. For a small ϵ, a small iteration number will be more helpful for the convergence.

In the last column of Fig. 3, we compare the DP ϵ against the utility metrics, accuracy and loss, on the MNIST35 dataset. The ObjPert and OutPert are excluded since it is designed for convex problems only[6]. Because some methods cannot converge in optimizing the network due to above-mentioned computation difficulties with a large noise, we adjust the range of ϵ. It can be seen that for similar low privacy conditions, L2P can train models with higher accuracy in most cases. The ϵ of SGD-MA increases slowly after 0.8 and its left boundary is given when the the number of iteration is 1, for which only a narrow range is available for presentation.

Notably, when $\epsilon > 0.6$, the performance of SGD-MA is better than other methods except the L2P and AGD, which is quite distinct from previous experimental results. Because SGD-MA is originally designed for optimizing deep models (Abadi et al. 2016), the moment accountant method is used for calculating the privacy level ϵ is more suitable for

[6]Though the OutPert is claimed to be capable for nonconvex problems with SGD algorithm (Zhang et al. 2017), the algorithm requires a constant of β-smoothness which can not be easily obtained or designed for neural networks.

Table 1: Space and time complexity on different batch sizes. #unroll represents the number of unrolled steps in L2P training. 'full' means a full unrolling in one batch while 40 and 20 denotes the sizes of mini-batches.

	Memory (Mb)			Epoch time (sec)		
#unroll	full	40	20	full	40	20
200	474	282	250	19	21	21
400	730	282	250	41	90	89
600	1242	282	250	57	102	173
1000	1843	282	250	164	188	190
1200	2066	282	250	172	248	266
1600	2266	282	250	418	357	350

mini-batch optimization. In other words, the noise scale increases slower by ϵ using SGD-MA. Since the L2P uses the same batch privacy estimation, it is rational to see the L2P could share the benefit in optimization. When $\epsilon \geq 0.8$, SGD-MA outperforms other methods. It is because the moment accountant of privacy costs can lead to a tighter bound of compositions than ρ-zCDP used by L2P and AGD when ϵ increases. But ρ-zCDP can provide a more convenient and efficient way to compute the privacy cost explicitly. Moment accountant has to compute the privacy cost by iterating over the moment orders which is relatively slow. Though L2P does not outperform in accuracies when $\epsilon > 0.8$, it has obviously lower training losses. It means L2P can optimize the losses better within less iterations, which might be local optimal, though.

C.4 Scalability

When extending the meta-training of L2P from non-constrained optimization to the constrained one, a critical issue is the scalability of the algorithm. Here we compare the time and space complexity of the batch and non-batch L2P algorithms to give a view of the issue.

Setup. The memory usage is measured by the GPU memory through the 'nvidia-smi' command on a Ubuntu 16 system with a TITAN X GPU and CUDA 10.1 driver. The program is written using TensorFlow 1.15[7] and allocates memory on need. The time is measured by the process time of one epoch averaged on 100 epochs. We use a 4-layer MLP and MNIST2 dataset for demonstration of budget-constrained optimization of schedulers. Because the memory usage grows nonlinearly due to the TensorFlow allocation, it is slightly more (around 20 to 50 Mb) than the true value while the trend is not affected.

Results. We empirically show the time and space complexity versus the unrolling length in Table 1. We see that the memory size increased quickly using the full batch while mini-batch does not need extra memory. Instead, mini-batch trade the memory with higher but acceptable time complexity. Experiments for larger networks (e.g., 128 layers) are included in the supplementary. Experimental results suggest that when a longer unrolling and larger network (e.g., 1000

[7]https://www.tensorflow.org/

steps for 128-layer network) are needed, slowly increasing the batch size will be beneficial to fit the algorithm into a limited GPU memory.

D Discussion

a) Query Efficiency. In comparison to AGD (Lee and Kifer 2018), the proposed L2P requires fewer times in querying the datasets to obtain the private model, because that AGD needs to query the dataset at each regret. To see this, we assume that the unit privacy cost of one query is ϵ. At each iteration, AGD conducts two queries, including one query for objective and the other for the gradient, if no regrets occur. Once one regret occurs, at least one additional query is required. On the other hand, in L2P the regret query happens in the training process of protector on auxiliary learning tasks: when a bad gradient g_t causes a lower loss at L_t, the effects will be back-propagated to the LSTM cells, as shown in Fig. 1. Since there are no privacy concerns in training the protector, the back-propagation is more accurate than the random objective queries used in AGD.

b) From Noised Model Training to Optimizer Training. In many learning algorithms, the noise-injected training, e.g., dropout training (Wager, Wang, and Liang 2013), has shown to be a useful way to improve the robustness or generalization of an algorithm. Especially if there are infinitely many additional noised samples for training, the classification performance can be improved against specific noise test environment and both in linear space (Maaten et al. 2013) and in nonlinear one (Hong, Chen, and Lin 2018). A critical difference between traditional noise-gradient-based DP algorithm and noised training is the number of noised samples in noised training or gradients in DP [8]. Because the constraint of privacy budget, the allowed training step is limited. In other words, the number of noised gradients is far away from infinity. Thus, the DP training can only result in a degraded model.

Since, in DP, the noised component is the gradient which is the input of an optimizer, we propose to improve the optimizer by training it with noise. It is a direct extension of the noised training except that we also train the noise variance which is related to the privacy budget.

c) The Denoising Effect of Utility Projector The projector in L2P is a denoising post-processing step which does *not* expose the original data, though. The guarantee is given in Lemma A.4. Denoising is not new in this area which has been studied in different directions. Recently, Balle and Wang enhanced the one-time query utility on Gaussian mechanism by calibration and statical denoising (Balle and Wang 2018). They proved that a scaling factor on the query result could lead to a smaller expected distance between the private output and the original one. Though their method is the analytic noising mechanism, it lacks necessary precise composition theory for multiple queries (e.g., a learning algorithm) in comparison to their baseline moment accountants (Abadi et al. 2016). Earlier, Barak *et al.* (Barak et al. 2007) and Hay et al. (Hay et al. 2009) show that accurate estimation can be achieved by enforcing table releases and graph degree sequences to be consistent. Karwa *et al.* make use of the knowledge of the noise distribution to efficiently infer a DP graph. In addition, the idea integrating prior into the Baysian inference from private outputs is formulated in (Williams and Mcsherry 2010). Bernstein *et al.* use Expectation-Maximization to denoise the parameter of a class of probablisitc graphical model (Bernstein et al. 2017). When a target solution is sparse, it is also possible to project linear regression model to a known l_1-ball which improves the resultant error.

Among these work, Balle and Wang's work (Balle and Wang 2018) and Lee and Kifer's work (Lee and Kifer 2018) is the first to adaptively perturb the outputs. Balle and Wang chose to scale the outputs with a factor adapted to the size of private outputs. This idea is also reflected in our adaptive perturbation where the step noise variance is adjusted according to the private gradient norm. Differently, the variance is adaptively calibrated according to an additional query to an alternative objective. Also, this is leveraged in our method while the objective query happens in auxiliary training before a private execution.

d) Protecting L2P Training Data. When there are very difficult learning tasks and hard to identify public auxiliary learning tasks, one may want to use some private data for auxiliary learning, which may cause privacy concerns when using protector in the sensitive learning. In such a case, the training of L2P protector should also be done in a private learning setting, e.g., perturbing the gradients or objective functions through classical privacy-preserving algorithms.

e) Choice of Auxiliary Tasks. The L2P protector as well as the learning-to-learn (Andrychowicz et al. 2016) are in fact performing transfer learning methods that gain gradient knowledge from auxiliary tasks and apply to a target learning task, with and without privacy consideration respectively. We see from our experiments that even though arbitrary choices of auxiliary tasks can deliver promising protectors, more relevant ones can further bring significant performance gains. This points out an important direction for future work, i.e., how to quantify the task relatedness so we can use high-performance protectors for a given learning task.

f) The availability of a public auxiliary dataset similar enough to the private one.

Prior than our paper, public dataset has been suggested for tuning hyper-paramters of private learning algorithms (Wu et al. 2017). However, they did not state how to access the public data and the affect of using different auxiliary datasets. Our method extend the setting for practical purpose. In practice, choosing public auxiliary dataset may not be a trivial work which greatly affect the performance. Here, we show the affects in experiments and with some primitive criteria, we can select useful auxiliary dataset easily. More complicated methods could be developed based on our primitive settings. For example, use cross validation to verify the effectiveness of the auxiliary datasets and extract more non-private information from the target private datasets for accurate auxiliary dataset selection.

More reasons can support the usage of auxiliary datasets in private learning. First off, the availability of auxiliary datasets is the main assumption of this paper and however this is a

[8] Since noised samples can lead to noised gradients, we put them in approximately equivalent position here.

rather common assumption used by other lines of work, such as learning-to-learn (L2L), where the learning trajectories from other tasks are leveraged. **2)** Secondly, for most learning tasks in real-world there *are* similar publicly available datasets, such as electornic medical records or computer vision tasks, on which we can construct auxiliary learning tasks. **3)** Moreover, the proposed L2P framework is learning momentum experiences from other optimization problems, instead of heavily relying on similar datasets, we therefore can leverage a wide spectrum of auxiliary optimization tasks *of the same class*. For example, a quadratic programming (QP) task may benefit from optimization procedures of many other QP, sometimes even a random QP problem of the same size according to our empirical study (Fig. 2). **4)** To evaluate the influence of the choice of auxiliary datasets, an experiment comparing different subsets of MNIST classes is conducted in Fig. 4. The experiment is constructed to simulate the scenario that both the auxiliary and protected datasets are used for binary classification task with same losses. It turns out that visually similar class sets, e.g., $\{4, 6\}$ (auxiliary) to $\{3, 5\}$ (protected), yields better accuracies while less similar ones still show performance above the best baseline.