Towards Hyperparameter Transfer for Differentially Private Optimization

Tianze Wang
Department of Statistics
Rutgers University
tianze.wang@rutgers.edu

Zhiqi Bu*
Independent Researcher
woodyx218@gmail.com

Linjun Zhang*
Department of Statistics
Rutgers University
zlj11112222@gmail.com

Abstract

Scaling laws and hyperparameter tuning remain central challenges in training large-scale models, particularly under differential privacy (DP), where optimization is further complicated by noise injection and per-sample gradient clipping. While Maximal Update Parametrization (μP) allows for zero-shot hyperparameter transfer in standard training, we show that it fails under DP constraints due to the distinct spectral properties of high-dimensional noise versus low-rank gradients. In this work, we introduce DP-μP, a theoretically grounded framework that extends μP to the privacy-preserving setting. By analyzing the interaction between gradient clipping, noise injection, and feature learning, we derive a Signal-to-Noise Ratio (SNR) condition that governs layer-wise learning rate scaling. Combining this with our proposed spectral scaling formulation, we provide specific scaling rules for DP-SGD, DP-Adam, and DP-Muon optimizers. Empirically, we demonstrate that DP-μP enables effective zero-shot transfer of learning rates across model widths on diverse architectures (MLP, ViT, GPT-2), matching or exceeding the performance of individually tuned baselines while significantly reducing computational costs.

1 Introduction

The success of large-scale models such as GPT [33, 3], LLaMA [37], Qwen [41], and DeepSeek [24], trained with billions of parameters and trillions of tokens, has underscored the importance of effective hyperparameter tuning [19, 18, 21]. Empirically, model performance is sensitive to choices like learning rate, momentum, and weight decay, necessitating fine-grained and computationally intensive tuning. However, the resources used for tuning hyperparameters grow exponentially with model size under grid or random search, posing a development for scalable model training [34, 11].

Maximal Update Parametrization (μP) [42] has emerged as a compelling solution to hyperparameter tuning in large-scale models. By reparameterizing neural networks to preserve update magnitudes across widths [43, 44], μP enables zero-shot transfer of hyperparameters, such as learning rate, from small models to significantly larger ones without performance degradation. μP achieves this by ensuring consistent layer-wise updates that stabilize activations and weights across both training iterations and model layers. In contrast to empirical scaling laws, which can be inaccurate, model-specific, or computationally expensive, μP offers a theoretically grounded, architecture-agnostic framework for scalable training and robust feature learning [45].

In this work, we investigate μP in differentially private (DP) optimization [1] to enable hyperparameter (learning rate) transfer in privacy-preserving deep learning. To put this into perspective, a well-known empirical tradeoff exists between model utility and privacy risk. On the one hand, larger models typically achieve better performance in tasks such as image classification and language modeling

*Co-corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

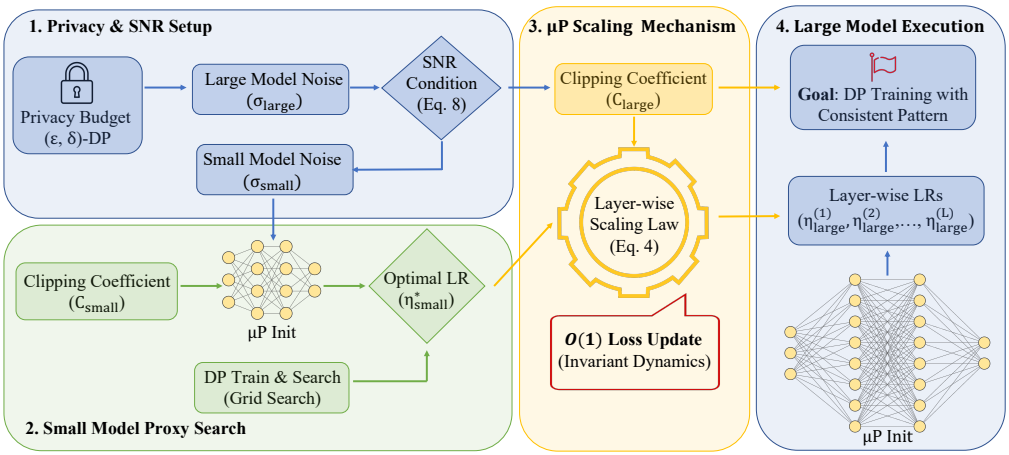


Figure 1: **DP-μP Training Workflow.** We employ μP initialization to stabilize the initial state and align the Signal-to-Noise Ratio (SNR) to derive the appropriate noise parameters for the small proxy model. After identifying the optimal base learning rate on the proxy, we apply layer-wise scaling to the large model to ensure $\mathcal{O}(1)$ loss updates, resulting in invariant training dynamics across model scales. A numerical example is provided in Appendix C.6.

[16, 3]. On the other hand, these models are also more prone to memorizing training data, which can be extracted through membership inference attacks [35] or prompt-based data extraction techniques [10]. To protect data privacy while retaining model utility, DP can be equipped with any optimizer. Moreover, DP has proven effective in settings such as continual learning and fine-tuning [14, 5, 49], making it a practical and increasingly adopted approach for privacy-preserving machine learning.

Specifically, we consider DP optimizers equipped with layer-wise clipping [6, 8, 15, 28]. Our objective is to quantify the layer-wise stabilization required to ensure robust learning rate transfer in DP optimization. In particular, we focus on the noise-dominated regime, where the spectral norm of the injected privacy noise overwhelms the gradient signal. This is an inevitable scenario that naturally arises in large models due to the high dimensionality of the noise matrix [38], and is a standard setting commonly analyzed in DP deep learning [40, 50, 23]. However, we emphasize that noisy learning frameworks (including both DP training and zero-order optimization [27]) can still achieve meaningful utility even when noise dominates in magnitude. We mark our contributions as follows:

1. **[Spectral condition for DP]** We characterize the spectral properties of the noise-dominated regime using random matrix theory: the full-rank nature of high-dimensional Gaussian noise, in contrast to the stable low-rank structure of the gradient signal, induces fundamentally different spectral dynamics that dictate the optimal scaling rules.
2. **[Signal-to-noise condition for DP]** We derive a signal-to-noise ratio (SNR) condition under the μP framework, which, together with the spectral condition, enables learning rate transfer across model sizes. We demonstrate this for three optimizers: DP-SGD, DP-Adam, and DP-Muon, establishing novel scaling rules for both clipping-only and DP settings, summarized in Table 1.
3. **[Zero-shot hyperparameter transfer]** We present a comprehensive empirical study, including coordinate checks to support our scaling method. We demonstrate that our method effectively transfers optimal learning rates from small to large models, matching or exceeding the performance of individually tuned baselines.

2 Preliminary

2.1 Notations

We use the big-Θ notation to describe asymptotic scaling, where $f(n) = \Theta(g(n))$ if there exist positive constants c, C such that

$$c \cdot g(n) \leq f(n) \leq C \cdot g(n), \text{ for all sufficiently large } n.$$

Table 1: Summary of layer-wise learning rate (lr) scaling rules for μP under different privacy settings. Note that for DP-SGD, the scaling explicitly accounts for the noise-to-batch ratio.

Optimizer	NonDP lr scaling [44]	Clipping-only lr scaling	DP lr scaling
SGD	$\frac{n_l}{n_{l-1}}$	$\sqrt{\frac{n_l}{n_{l-1}}}$	$\frac{\sqrt{n_l/n_{l-1}}}{\frac{\sigma_{\text{DP}}}{B}(\sqrt{n_l}+\sqrt{n_{l-1}})}$
Adam	$\frac{1}{n_{l-1}}$	$\frac{1}{n_{l-1}}$	$\frac{\sqrt{n_l/n_{l-1}}}{\sqrt{n_l}+\sqrt{n_{l-1}}}$
Muon	$\sqrt{\frac{n_l}{n_{l-1}}}$	$\sqrt{\frac{n_l}{n_{l-1}}}$	$\sqrt{\frac{n_l}{n_{l-1}}}$

We denote the Frobenius norm by $\|\boldsymbol{W}\|_F := \sqrt{\sum_{i,j} \boldsymbol{W}_{ij}^2}$, and the spectral norm (or operator norm) by $\|\boldsymbol{W}\|_* := \max_{\mathbf{x}\neq 0} \|\boldsymbol{W}\mathbf{x}\|_F / \|\mathbf{x}\|_F$. Finally, we use $\asymp$ to indicate asymptotic equivalence up to constant factors.

2.2 Deep Learning and Gradient Descent

Consider a neural network composed of L layers, where each layer transforms the output of the previous one. A simplified forward pass through the network can be represented layer-wise as follows:

$$\mathbf{a}_l = \mathbf{a}_{l-1}\boldsymbol{W}_l, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_l} = \frac{\partial \mathbf{a}_l}{\partial \boldsymbol{W}_l}^\top \frac{\partial \mathcal{L}}{\partial \mathbf{a}_l} = \mathbf{a}_{l-1}^\top \frac{\partial \mathcal{L}}{\partial \mathbf{a}_l} \tag{1}$$

where l denotes the layer index. Here, $\mathbf{a}_l \in \mathbb{R}^{B\times n_l}$ represents the activation at layer l for a mini-batch of size B, with $\mathbf{a}_0$ being the input data, and $\boldsymbol{W}_l \in \mathbb{R}^{n_{l-1}\times n_l}$ is the weight matrix of the l-th layer. Following the convention in μP literature [44, 45], we omit bias terms and nonlinear activation functions (e.g., ReLU or GELU) between layers for notational simplicity. We highlight that our analysis also works without such simplification and explain it in Appendix A.1.

Training neural networks typically involves iterative updates using gradient-based optimization algorithms (e.g. SGD, Adam and Muon). At each iteration t, the weights are updated according to:

$$\boldsymbol{W}_{l,t+1} = \boldsymbol{W}_{l,t} + \Delta \boldsymbol{W}_{l,t} := \boldsymbol{W}_{l,t} - \eta_l \boldsymbol{g}_{l,t} \tag{2}$$

where η_l is the learning rate associated with layer l, and $\boldsymbol{g}_{l,t} = g(\nabla_{\boldsymbol{W}_{l,t}}\mathcal{L})$ is the pre-conditioned gradient of the mini-batch loss function $\mathcal{L}$. The function $g(\cdot)$ defines the optimizer-specific transformation applied to the raw gradient. For example, $g(\boldsymbol{x}) = \boldsymbol{x}$ corresponds to standard SGD; $g(\boldsymbol{x}) = \text{sign}(\boldsymbol{x})$ corresponds to SignSGD or the gradient update in Adam-like optimizers; $g(\boldsymbol{x}) = \boldsymbol{U}^\top \boldsymbol{V}$ (with $\boldsymbol{x} = \boldsymbol{U}^\top \boldsymbol{S}\boldsymbol{V}$ being the singular value decomposition) corresponds to Muon and Shampoo.

2.3 Maximal Update Parametrization (μP)

To ensure feature learning, μP stabilizes each element of $\mathbf{a}_l$ at $\Theta(1)$ in two dimensions: layer-wise and iteration-wise. Mathematically, μP normalizes the weights to satisfy the following scales:

$$\overbrace{\underbrace{\|\mathbf{a}_l\|_2 = \Theta(\sqrt{n_l}) \Leftarrow \|\boldsymbol{W}_{l,t}\|_* = \Theta(\sqrt{n_l/n_{l-1}})}_{\|\boldsymbol{W}_{l,t}\|_* = \Theta(\sqrt{n_l/n_{l-1}}) \Leftarrow \|\Delta\boldsymbol{W}_{l,t}\|_* = \Theta(\sqrt{n_l/n_{l-1}})}}^{\text{Desideratum 1 in [45] (Feature learning)}} \tag{3}$$

Condition 1 in [45] (Spectral scaling)

Note that (3) can be derived via mathematical induction (detailed in Appendix A.1), provided the following specifications are met:

1. weight initialization with specific variance (known as the spectral parametrization in [45]), so that at $t = 0$, $\|\boldsymbol{W}_{l,0}\|_* = \Theta(\sqrt{n_l/n_{l-1}})$.
2. layer-wise learning rate η, so that $\|\Delta\boldsymbol{W}_{l,t}\|_* = \Theta(\sqrt{n_l/n_{l-1}})$ for any $\boldsymbol{g}_{l,t}$.

Crucially, it has been established that preserving this feature learning regime is fundamental to enabling hyperparameter transfer across model widths [44, 45]. For residual connections in transformers, since μP meticulously scales the residual outputs to ensure the activation norms remain stable, the framework extends to residual architectures without loss of generality. Consequently, to achieve this transferability in the differentially private setting, we determine the **layer-wise learning rate** specifically to enforce these spectral scaling laws. This requirement leads to the condition formalized in the following lemma.

Lemma 2.1 (Spectral condition via layer-wise learning rate [45])**.** For $\boldsymbol{g}_l$ from any optimizer, the spectral scaling condition in (3) is equivalent to setting

$$\eta_l = \|\Delta \boldsymbol{W}_l\|_* / \|\boldsymbol{g}_l\|_* = \Theta(\sqrt{n_l / n_{l-1}} / \|\boldsymbol{g}_l\|_*) \tag{4}$$

We can leverage Lemma 2.1 to derive Table 1 by analyzing $\|\boldsymbol{g}_l\|_*$ under different optimziers. To see this, we elaborate in Section 3 for both DP and non-DP optimizers, with a special focus on the influence of DP operations (i.e. per-sample gradient clipping and Gaussian noising).

2.4 Differentially Private Optimization

Recall the formal definition of (ϵ, δ)-DP.

Definition 2.2. [13] A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if, for any two neighboring datasets D and D' that differ in exactly one datapoint, and for all $\mathcal{S} \subset \text{Range}(\mathcal{M})$, it holds that:

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta. \tag{5}$$

Differentially private (DP) optimization applies any optimizer with the **privatized gradient**:

$$p(\nabla_{\boldsymbol{W}} \mathcal{L}) = \frac{1}{B} \sum_{i=1}^{B} c_i \nabla_{\boldsymbol{W}} \mathcal{L}_i + \frac{\sigma_{\text{DP}}}{B} N(0, I),$$

hence transforming SGD to DP-SGD, as well as AdamW [22, 26], Muon [20], GeN [9, 25], etc. Here $\nabla_{\boldsymbol{W}} \mathcal{L}_i$ is the per-sample gradient, c_i is a constant to perform gradient clipping (e.g. $c_i = \frac{1}{\|\nabla_{\boldsymbol{W}} \mathcal{L}_i\|_F}$), B is the batch size and σ_{DP} is the noise level.

We study the layer-wise automatic clipping [7, 47], for the l-th layer,

$$\boldsymbol{p}_l := p(\nabla_{\boldsymbol{W}_l} \mathcal{L}) = \frac{1}{B} \sum_i \frac{\nabla_{\boldsymbol{W}_l} \mathcal{L}_i}{\sqrt{D} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_F} + \frac{\sigma_{\text{DP}}}{B} N(0, I_l) \tag{6}$$

where D is the number of layers. We focus on this type of method for two reasons: (I) automatic clipping avoids hyperparameter tuning for the per-sample clipping threshold, and (II) we choose layer-wise clipping [29] because μP requires a layer-wise analysis, and because layer-wise clipping is the most efficient and scalable clipping style in distributed training [4, 7]. To summarize, the pre-conditioned and privatized gradient is $\boldsymbol{g}_l = g(\boldsymbol{p}_l)$ for any optimizer g, which has the same level of DP guarantee by the post-processing property of DP.

3 μP for DP optimizers

In this section, we analyze the spectral scaling properties of DP optimizers to address the unique challenges introduced by the privacy mechanism. We demonstrate that the interaction between clipping and noise invalidates standard μP transfer rules, necessitating a rigorous derivation of clipping thresholds and noise levels that satisfy a specific SNR constraint. Ultimately, we establish that learning rate transfer under DP requires two conditions to hold simultaneously: the μP spectral condition, which keeps the update at the feature-learning scale, and the layer-wise SNR condition, which keeps the share of signal in that update consistent across widths. The SNR condition is necessary, and together with the spectral condition it is sufficient (Proposition 3.6). Our analysis relies on the following fact and assumption.

Fact 3.1 (Norm equivalence for per-sample gradient)**.** $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_* = \|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_F$. Since the gradient is computed on a per-sample basis (equivalent to $B = 1$), it is a rank-1 matrix.

Assumption 3.2. $\left\| \frac{1}{B} \sum_{i=1}^{B} g(\boldsymbol{p}_i) \right\|_* = \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \|g(\boldsymbol{p}_i)\|_* \right)$. This geometric alignment assumption for gradients within a mini-batch is widely used for analyzing large-batch training [48, 39]. We emphasize that this assumption is practical, with empirical validation provided in Appendix B.

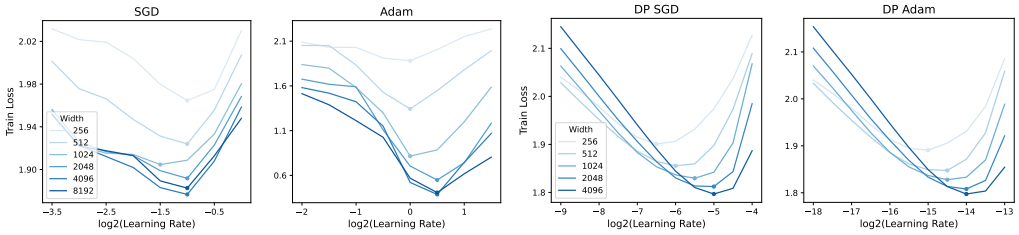
3.1 Standard Optimizers with Per-Sample Gradient Clipping

Begin with analyzing DP optimizers without noise, which effectively reduces to standard optimizers with per-sample gradient clipping, we isolate the scaling behavior induced by clipping.

Proposition 3.3. *The spectral norms of the clipped gradient updates $\boldsymbol{g}(\boldsymbol{p}_l)$ under different optimizers satisfy:*

$$\|g(\boldsymbol{p}_l)\|_* = \begin{cases} \Theta(1), & \textit{SGD / Muon}, \\ \Theta\left(\sqrt{n_l n_{l-1}}\right), & \textit{Adam}. \end{cases} \tag{7}$$

Substituting these estimates into Lemma 2.1 yields the scaling rules listed in the second column of Table 1. We empirically validate these derivations in Figure 2a, which demonstrates successful learning rate transfer across model widths.



(a) Clipping Only (Success) (b) Noisy Norm Scaling (Failure)

Figure 2: **Comparison of hyperparameter transfer scaling rules (μP) on a 5-layer MLP (CIFAR-10).** (a) Under the clipping-only regime (no noise), loss curves align across widths, validating μP. (b) Scaling by the spectral norm of the noisy update fails to align curves, demonstrating the failure of standard μP under differential privacy.

For input layer, the feature norm follows $\|\mathbf{a}_1\| = \Theta(\|\boldsymbol{W}_1\|_* \cdot \|\boldsymbol{x}\|)$. This behavior depends on the data norm $\|\boldsymbol{x}\|$, distinguishing from hidden layers where scaling is governed solely by the width of the preceding layer. We provide a detailed derivation of this boundary condition in Appendix A.4.

Remark 3.4. For non-noisy optimizers, the spectral condition in Lemma 2.1 is necessary and sufficient for transfer. Fundamentally, μP operates by scaling gradient updates to preserve feature learning as width increases. Gradient clipping operates as a scaling adjustment and thus remains within the μP framework. Therefore, standard μP guarantees apply directly provided feature learning holds.

3.2 DP Optimizers with Per-Sample Gradient Clipping and Noise

However, the spectral condition in Lemma 2.1 alone is insufficient for DP optimizers. In the privacy-preserving setting, the total update norm is often dominated by the injected noise rather than the signal. Scaling rules derived solely from the total spectral norm fail to preserve the feature learning dynamics, causing loss curves diverge significantly across widths (Figure 2b). We formally characterize this insufficiency in the following Lemma 3.5.

Lemma 3.5 (Spectral condition is insufficient for DP optimizers)**.** In the noise-dominated regime of SGD, the signal update vanishes as width increases.

Proof. Let the signal $\boldsymbol{s}_l$ and noise $\boldsymbol{\xi}_l$ be defined as:

$$\boldsymbol{s}_l = \frac{1}{B}\sum_{i=1}^{B} \frac{\nabla_{\boldsymbol{W}_l}\mathcal{L}_i}{\sqrt{D}\,\|\nabla_{\boldsymbol{W}_l}\mathcal{L}_i\|_F}, \quad \boldsymbol{\xi}_l \sim \frac{\sigma_{\mathrm{DP}}}{B}\mathcal{N}(0, I_{n_l \times n_{l-1}})$$

Consider the noise-dominated regime, where the spectral norm of the update is dominated by the random matrix contribution. Using standard random matrix theory results, we have:

$$\|\boldsymbol{s}_l + \boldsymbol{\xi}_l\|_* = \Theta\left(\|\boldsymbol{\xi}_l\|_*\right) = \Theta\left(\frac{\sigma_{\mathrm{DP}}}{B}\left(\sqrt{n_l} + \sqrt{n_{l-1}}\right)\right).$$

The first equality holds as $\|\boldsymbol{s}_l\| = \Theta(1)$ due to automatic clipping, which remains non-vanishing even when iterates near stationary. After applying the scaling suggested in Lemma 2.1, the effective

update to the weight matrix W_l with respect to the signal scales as:

$$\|\eta_l \boldsymbol{s}_l\|_* = \Theta\left(\sqrt{\frac{n_l}{n_{l-1}}}\frac{\|\boldsymbol{s}_l\|_*}{\|\boldsymbol{\xi}_l\|_*}\right) = \Theta\left(\frac{\sqrt{n_l/n_{l-1}}}{\sqrt{n_l}+\sqrt{n_{l-1}}}\right).$$

This scales as $\Theta(1/\sqrt{n})$, which implies that the signal contribution vanishes as the width increases, thereby violating the feature learning scales required for transfer. □

Next, we introduce Proposition 3.6, which establishes a layer-wise SNR condition for hyperparameter transfer under DP. We show that this condition is necessary for transfer, and that it is also sufficient when combined with the layer-wise learning rate of Lemma 2.1. To satisfy this condition at every layer, we replace the global clipping coefficient with a layer-dependent one, parametrized by D_l.

Proposition 3.6 (SNR condition is necessary and sufficient for DP optimizers)**.** *We define the spectral noise-to-signal ratio $\mathcal{C}(\cdot)$ for the l-th layer as a function of the noise multiplier σ, clipping coefficient $1/\sqrt{D_l}$, and layer widths:*

$$\frac{\|\boldsymbol{\xi}_l\|_*}{\|\boldsymbol{s}_l\|_*} \asymp \frac{\sigma\sqrt{D_l}}{B}\left(\sqrt{n_l}+\sqrt{n_{l-1}}\right) := \mathcal{C}(\sigma, D_l, n_l, n_{l-1}).$$

Let c_l^{target} denote this value calculated using the target model's configuration $(\sigma^{\text{target}}, D_l^{\text{target}}, n^{\text{target}})$, and c_l^{proxy} denote the corresponding value for the proxy model $(\sigma^{\text{proxy}}, D_l^{\text{proxy}}, n^{\text{proxy}})$. In the noise-dominated regime, hyperparameters tuned on the proxy transfer to the target if and only if

$$c_l^{\text{proxy}} = c_l^{\text{target}}, \quad \forall l \in \{1, \dots, L\}.$$

This equality implies that the effective noise intensity relative to the signal is preserved layer-wise, independent of the model scale.

Proof. We first show sufficiency: under the SNR condition defined in Proposition 3.6, the effective signal contribution to the weight update remains independent of the variable width n_l, thus enabling hyperparameter transfer.

Consider the normalized update step used in μP within the noise-dominated regime. The update direction is normalized by the spectral norm of the total gradient estimate:

$$\Delta \boldsymbol{W}_{\text{norm}} = \frac{\boldsymbol{s}_l + \boldsymbol{\xi}_l}{\|\boldsymbol{s}_l + \boldsymbol{\xi}_l\|_*} \asymp \frac{\boldsymbol{s}_l + \boldsymbol{\xi}_l}{\|\boldsymbol{\xi}_l\|_*}.$$

This approximation follows directly from the triangle inequality, as the spectral norm of the noise significantly exceeds that of the signal (i.e., $\|\boldsymbol{\xi}_l\|_* \gg \|\boldsymbol{s}_l\|_*$). Consequently, the effective contribution of the signal component to this update is $\Delta \boldsymbol{W}_{\text{signal}} \asymp \frac{\boldsymbol{s}_l}{\|\boldsymbol{\xi}_l\|_*}$. Applying the SNR condition established in Proposition 3.6 and substituting it into the expression for the signal component:

$$\|\Delta \boldsymbol{W}_{\text{signal}}\|_* \asymp \frac{\|\boldsymbol{s}_l\|_*}{c_l^{\text{proxy}} \cdot \|\boldsymbol{s}_l\|_*} = \frac{1}{c_l^{\text{proxy}}} = \frac{1}{c_l^{\text{target}}}.$$

Since c_l^{target} is a constant determined solely by the fixed target architecture, the effective signal strength is invariant to the proxy width. This ensures that the feature learning signal is preserved at a consistent scale relative to the target model, allowing hyperparameters tuned on small proxies to be directly transferred.

We next show necessity. Under DP, the noise sets the scale of the update and the signal drives feature learning, so transfer requires both to match across the two models. The learning rate η_l scales the signal and the noise by the same factor, so their ratio c_l does not depend on η_l. Therefore, if $c_l^{\text{proxy}} \neq c_l^{\text{target}}$, no learning rate can match the noise scale and the signal scale simultaneously, and hyperparameter transfer fails. □

Remark 3.7. For batch size transfer, we can treat the ratio $\frac{\sigma_{\text{DP}}}{B}$ as a unified term. This simplifies the derivation of scaling laws, as the effective noise level remains invariant relative to the signal.

3.3 Layer-wise Clipping Parameter

Since the input dimension n_0 (e.g., embedding size) and output dimension n_L (determined by the specific task) are typically fixed, the model widths at the boundaries do not scale proportionally with the hidden layers. Consequently, we cannot rely solely on noise scaling to match the SNR at the input and output layers.

To address this, given a target privacy noise level σ, we treat the layer-wise clipping coefficient as the adjustable parameter to satisfy the SNR condition in the proxy model (characterized by noise σ'). Let D_l and D'_l denote the clipping parameters for the target and proxy models, respectively. We obtain the following system of constraints, which determines the optimal layer-wise parameters:

$$\begin{cases} \left(\frac{\sigma'}{\sigma}\right)^2 \frac{D'_l}{D_l} = \left(\frac{\sqrt{n_{l-1}} + \sqrt{n_l}}{\sqrt{n'_{l-1}} + \sqrt{n'_l}}\right)^2 := f_l, \quad l \in \{1, \dots, L\} \\ \sum_{l=1}^{L} \frac{1}{D_l} = \sum_{l=1}^{L} \frac{1}{D'_l} = 1, \end{cases} \tag{8}$$

The rescaled parameters (σ', D'_i) admit the closed-form solution $\sigma' = \sigma / \sqrt{\sum_{i=1}^{L} \frac{1}{D_i f_i}}$, $D'_i = D_i f_i \cdot \sum_{t=1}^{L} \frac{1}{D_t f_t}$. Equation (8) enforces layer-wise SNR alignment, ensuring that the proxy model effectively mimics the optimization dynamics of the target model despite the difference in width. The LayerNorm operation in Transformers differs from standard linear layers, we present the detailed derivations in Appendix A.6.

In practice, DP-μP proceeds as follows. We first calibrate the noise multiplier of the target model with the privacy accountant and adopt uniform clipping on the proxy. Solving equation (8) then gives the noise multiplier of the proxy and the layer-wise clipping parameters of the target. Finally, we grid search the learning rate on the proxy and map it to the layer-wise learning rates of the target through Lemma 2.1. Appendix C.6 presents this procedure step by step with a worked example.

4 Experiments

In this section, we evaluate the effectiveness of DP-μP in two key aspects: (i) enabling robust hyperparameter transfer across model scales, including variations in width and batch size; and (ii) achieving comparable or superior performance under differential privacy constraints. Following the μP literature, we compare training loss across widths in the main text, and report multi-seed results on the final test accuracy in Appendix C.4.

Our experiments cover two primary settings: (i) image classification, focusing on scale transfer and gradient coordinate analysis, and (ii) text generation. We compare our approach against standard parametrization (SP) using multiple optimizers (SGD, Adam, and Muon). Unless otherwise specified, all differentially private experiments use a fixed privacy budget of $\epsilon = 2$ and $\delta = 10^{-5}$. Following common practice [12, 49] , this budget covers the training of the target model and excludes the privacy cost of the learning rate search on the proxy. DP-μP therefore assumes that the proxy learning rate is either selected privately [30, 31] or obtained from public data [17, 32]; Appendix C.7 gives the end-to-end accounting and shows that the search adds little privacy cost.

Furthermore, all experiments were conducted using a single NVIDIA A100 GPU. This comprehensive setup allows us to assess both the robustness and generality of DP-μP across diverse architectures, tasks, and optimization landscapes.

4.1 Image Classification Pretraining

We begin with experiments on CIFAR-10 to evaluate DP-μP in a controlled setting, focusing on both hyperparameter transfer and optimization performance. Our evaluation spans hyperparameter transfer across width using a 5-layer MLP and Vision Transformer (ViT), complemented by a coordinate-wise gradient analysis under different optimizers.

Figure 3 illustrates the training loss of a 5-layer MLP on CIFAR-10 over 10 epochs. We display the SGD and Muon results here as representative examples, and defer the results for Adam to

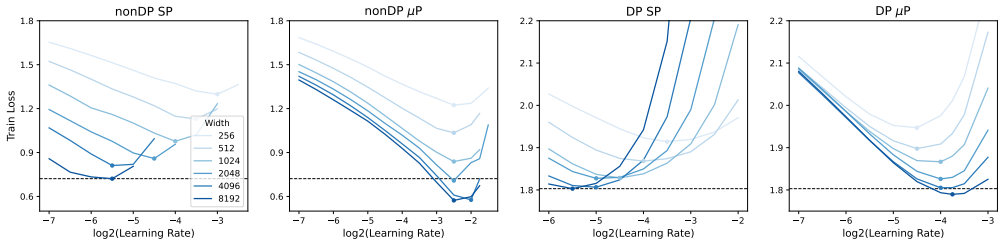


Figure 3: **Comparison between SP and μP (SGD).** We adjust the clipping coefficient and noise to satisfy the SNR condition while keeping the input dimension fixed.

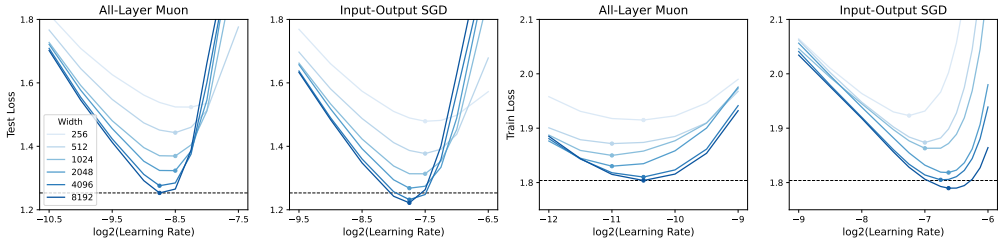


Figure 4: **Comparison between Muon and SGD-Muon.** We evaluate different update rules (Muon vs. SGD) for the Input and Output layers under the μP scaling conditions. The left two panels show the non-private setting (test loss), and the right two panels show the DP setting (training loss).

Appendix C.1. Across all optimizers, μP consistently yields more stable training dynamics and lower final loss compared to standard parametrization (SP).

Regarding the input and output layers in the Muon optimizer, we deviate from the original implementation, which typically employs Adam updates. Instead, we apply either Muon or SGD updates. This modification is necessary because the standard Adam update fails to exhibit hyperparameter transfer in this context, even in non-private settings. Figure 4 compares these two choices in both the non-private and the DP setting. For the non-private results, we report test loss instead of training loss, because Muon overfits significantly in the non-private regime.

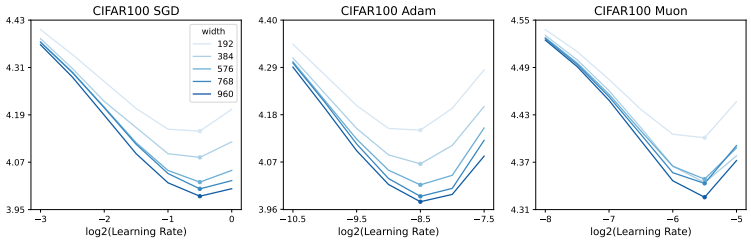


Figure 5: **DP-μP ViT on CIFAR-100.** For Muon, the input and output layers are updated using SGD.

Figure 5 displays the pretraining loss curves for ViT models of increasing width on CIFAR-100, with corresponding results for CIFAR-10 deferred to Appendix C.3. Across all widths, μP enables effective hyperparameter transfer, allowing larger models to be trained using the same learning rate as smaller ones in the DP setting. Furthermore, larger ViTs consistently achieve lower training loss, provided the SNR condition is satisfied.

We also extend our evaluation to batch size transferability. In this setting, we fix the model width and scale the noise multiplier σ proportionally to the batch size. Crucially, to ensure a rigorous comparison of optimization trajectories, we fix the total number of update steps (iterations) across all runs, rather than the number of epochs. The detailed results confirming robust transfer under these conditions are provided in Section C.2.

4.2 Coordinate Check

While the theoretical foundations of DP-μP are well-established, practical implementation remains error-prone. To support correctness, we adopt the coordinate checking procedure introduced by [44]. As illustrated in Figures 6 and 7, we track the l_1 norm of feature activations for specific coordinates over the initial training steps, sweeping model width across a broad range. A correct μP implementation preserves the scale of these activations across varying widths. For comparison, coordinate check results for standard parametrization (SP) are provided in Appendix C.5, where activation norms typically explode as width increases.

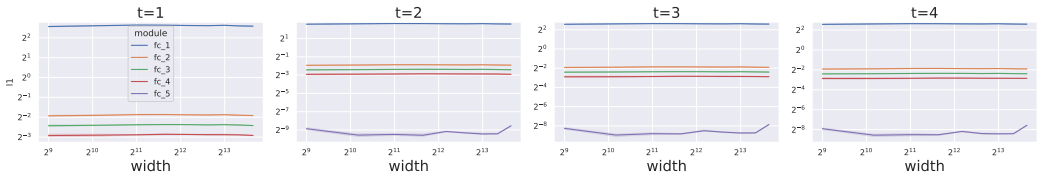


Figure 6: **Coordinate check for MLP (DP-μP).** The evolution of activation l_1 norms across different widths shows consistent scaling, confirming correct implementation.

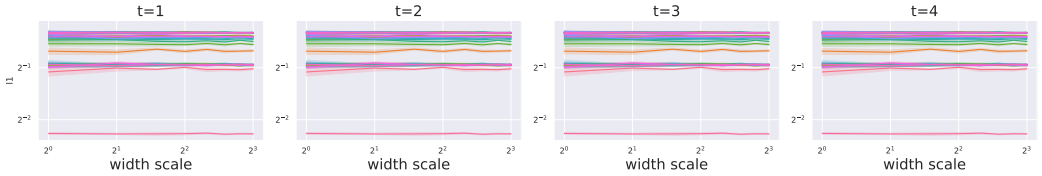


Figure 7: **Coordinate check for ViT (DP-μP).** Similar to the MLP case, the Vision Transformer maintains stable activation scales across varying widths.

4.3 Text Generation Pretraining

To validate our approach on generative tasks, we conduct pretraining experiments using the GPT-2 architecture on the Shakespeare Character dataset. Our target configuration is a GPT-2 Large model (36 layers, 20 heads, $d_{model} = 1280$) with 787.8M parameters. To maximize search efficiency, we determine the optimal learning rate on a small proxy model (5 heads, $d_{model} = 320$, 76.0M parameters), which yields a 90.35% reduction in theoretical computational cost. Empirically, on a single NVIDIA A100 GPU, this reduces the wall-clock execution time per training run from approximately 30 minutes to 11 minutes. Additionally, we evaluate an intermediate model (10 heads, 227.3M parameters) to corroborate the consistency of the loss landscape across widths. As shown in Figure 8, the training loss curves exhibit remarkably consistent patterns across all three model scales for each learning rate, demonstrating successful zero-shot hyperparameter transfer.

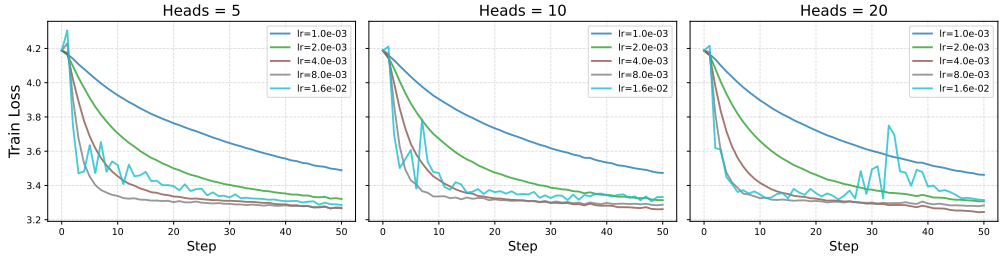


Figure 8: **GPT-2 Pretraining with DP-μP on Shakespeare Character-Level Task.** Loss curves across varying widths (controlled by head counts) exhibit consistent ranking and dynamics. This consistency indicates that the optimal learning rate remains invariant across model sizes, demonstrating successful hyperparameter transfer.

5 Discussion

Maximal update parametrization (μP) has inspired numerous extensions across various architectures and training regimes [2, 46, 51]. Our work introduces DP-μP, a theoretically grounded and empirically validated framework for hyperparameter transfer in differentially private deep learning. By bridging the gap between μP and differential privacy, we resolve the tension between gradient clipping and feature scaling. Specifically, our derivation of layer-wise learning rate scaling rules ensures that the Signal-to-Noise Ratio (SNR) remains consistent across model scales, a critical requirement often overlooked in standard DP optimization.

Empirically, we validate this theoretical framework across diverse settings, including MLP, ViT, and GPT-2 architectures. Our results demonstrate that optimal hyperparameters (e.g., learning rate and clipping threshold) identified on small, inexpensive proxy models transfer zero-shot to large-scale targets. This capability provides a critical alternative to established DP hyperparameter tuning strategies [12, 31], which typically require repeated search iterations or rely heavily on auxiliary public datasets, thereby drastically reducing the computational costs.

6 Limitations and Future Work

While our work establishes robust scaling laws for width and batch size across MLPs, ViTs, and GPT-style transformers, we do not currently address depth scaling. Extending our framework to deeper networks presents a fundamental challenge, as the accumulation of privacy noise across successive layers naturally tends to violate the layer-wise SNR condition required for stable learning. Future work is needed to either derive a depth-scaling mechanism that preserves this SNR or to rigorously establish a theoretical impossibility result.

Finally, our current analysis operates under a noise-dominant regime, an assumption that holds in typical differentially private training where gradient clipping and noise injection often overwhelm the signal. With extremely large batch sizes, such as those used in large-scale private pretraining [36], the noise-to-signal ratio becomes small and the scaling reduces to the clipping-only rules in Table 1. A valuable direction for future research would be to extend this analysis to the transition regime, where signal and noise magnitudes are comparable, to understand if the scaling laws require adjustment when the privacy constraint is relaxed.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Y. Prince, Björn Deiseroth, Andres Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μp: The unit-scaled maximal update parametrization, 2025.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[4] Zhiqi Bu, Justin Chiu, Ruixuan Liu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Zero redundancy distributed learning with differential privacy. In *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*, 2023.

[5] Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. Deep learning with gaussian differential privacy. *Harvard data science review*, 2020(23):10–1162, 2020.

[6] Zhiqi Bu, Ruixuan Liu, Yu-Xiang Wang, Sheng Zha, and George Karypis. On the accuracy and efficiency of group-wise clipping in differentially private optimization, 2023.

[7] Zhiqi Bu, Ruixuan Liu, Yu-Xiang Wang, Sheng Zha, and George Karypis. On the accuracy and efficiency of group-wise clipping in differentially private optimization. *arXiv preprint arXiv:2310.19215*, 2023.

[8] Zhiqi Bu, Hua Wang, Zongyu Dai, and Qi Long. On the convergence and calibration of deep learning with differential privacy, 2023.

[9] Zhiqi Bu and Shiyun Xu. Gradient descent with generalized newton's method. In *The Thirteenth International Conference on Learning Representations*, 2024.

[10] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2021.

[11] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[12] Soham De, Leonard Berrada, Jamie Hayes, Samuel L. Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale, 2022.

[13] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in cryptology-EUROCRYPT 2006: 24th annual international conference on the theory and applications of cryptographic techniques, st. Petersburg, Russia, May 28-June 1, 2006. proceedings 25*, pages 486–503. Springer, 2006.

[14] Ahmad Hassanpour, Majid Moradikia, Bian Yang, Ahmed Abdelhadi, Christoph Busch, and Julian Fierrez. Differential privacy preservation in robust continual learning. *IEEE Access*, 10:24273–24287, 2022.

[15] Jiyan He, Xuechen Li, Da Yu, Huishuai Zhang, Janardhan Kulkarni, Yin Tat Lee, Arturs Backurs, Nenghai Yu, and Jiang Bian. Exploring the limits of differentially private deep learning with group-wise clipping, 2022.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Shlomi Hod, Lucas Rosenblatt, and Julia Stoyanovich. Do you really need public data? surrogate public data for differential privacy on tabular data, 2025.

[18] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[20] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[21] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[23] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners, 2022.

[24] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[25] Ruixuan Liu and Zhiqi Bu. Towards hyperparameter-free optimization with differential privacy. In *The Thirteenth International Conference on Learning Representations*.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[27] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D. Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes, 2024.

[28] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models, 2018.

[29] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[30] Shubhankar Mohapatra, Sajin Sasy, Xi He, Gautam Kamath, and Om Thakkar. The role of adaptive optimizers for honest private hyperparameter selection, 2021.

[31] Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy, 2022.

[32] Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H. Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201, 2023.

[33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[34] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[35] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017.

[36] Amer Sinha, Thomas Mesnard, Ryan McKenna, Daogao Liu, Christopher A. Choquette-Choo, Yangsibo Huang, Da Yu, George Kaissis, Zachary Charles, Ruibo Liu, Lynn Chua, Pritish Kamath, Pasin Manurangsi, Steve He, Chiyuan Zhang, Badih Ghazi, Borja De Balle Pigem, Prem Eruvbetine, Tris Warkentin, Armand Joulin, and Ravi Kumar. Vaultgemma: A differentially private gemma model, 2025.

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[38] Florian Tramèr and Dan Boneh. Differentially private learning needs better features (or much more data), 2021.

[39] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates, 2021.

[40] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization, 2020.

[41] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[42] Greg Yang and Edward J Hu. Feature learning in infinite-width neural networks. *arXiv preprint arXiv:2011.14522*, 2020.

[43] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[44] Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

[45] Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

[46] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks, 2023.

[47] Xiaodong Yang, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Normalized/clipped sgd with perturbation for differentially private non-convex optimization. *arXiv preprint arXiv:2206.13033*, 2022.

[48] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning, 2018.

[49] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models, 2022.

[50] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning, 2021.

[51] Chenyu Zheng, Xinyu Zhang, Rongzhen Wang, Wei Huang, Zhi Tian, Weilin Huang, Jun Zhu, and Chongxuan Li. Scaling diffusion transformers efficiently via μp, 2025.

A Mathematical Derivations

A.1 Handling Bias and Nonlinearities

To justify the simplification in the main text, we analyze the magnitude of the bias terms and the effect of nonlinear activation functions. We show that they do not alter the order of magnitude of the hidden states or gradients with respect to the width n.

A.1.1 Bias Terms.

Let $\mathbf{b}_l \in \mathbb{R}^{n_l}$ be the bias vector at layer l. The pre-activation is given by $\mathbf{a}_l = \mathbf{a}_{l-1}\boldsymbol{W}_l + \mathbf{b}_l$. following the perspective in modern parametrization literature [45], we view the bias vector as a weight matrix $\mathbf{b}_l \in \mathbb{R}^{n_l \times 1}$ connecting an auxiliary layer with width 1 and a constant output of 1 to the l-th hidden layer.

This unification allows us to apply our spectral scaling analysis directly to the bias $\mathbf{b}_l$. A key property utilized here is that for any vector v, its spectral norm is equivalent to its Euclidean norm:

$$\|\mathbf{b}_l\|_* = \|\mathbf{b}_l\|_2 \tag{9}$$

Under the spectral scaling condition, treating $\mathbf{b}_l$ as a weight matrix prescribes $\|\mathbf{b}_l\|_2 = \Theta(\sqrt{n_l})$. Furthermore, this formulation naturally accommodates the SNR condition under layer-wise clipping, where the weight matrix and bias share the same clipping coefficient and noise scale. Assuming proportional width scaling where the aspect ratio between layers is preserved (i.e., $n_{l-1}/n'_{l-1} = n_l/n'_l$), the scaling factor simplifies as follows:

$$\frac{\sigma'}{\sigma}\sqrt{\frac{D'_l}{D_l}} = \frac{\sqrt{n_{l-1}} + \sqrt{n_l}}{\sqrt{n'_{l-1}} + \sqrt{n'_l}} = \frac{\sqrt{n_l}}{\sqrt{n'_l}} \tag{10}$$

The last equality holds because the term $(\sqrt{n_{l-1}/n_l} + 1)$ factors out from both the numerator and the denominator.

A.1.2 Nonlinear Activations

Following the analysis in prior literature [45], we adopt the assumption that the activation function $\phi(\cdot)$ preserves the order of the signal magnitude:

$$\|\phi(\mathbf{a}_l)\|_2 = \Theta(\|\mathbf{a}_l\|_2) \tag{11}$$

This assumption is justified by the design of standard activation functions (e.g., ReLU, GELU), which are constructed to take order-one inputs and yield outputs that neither explode nor uniformly vanish. Consequently, the introduction of nonlinearity does not alter the asymptotic scaling of the network width, allowing our spectral analysis to proceed effectively under the linear approximation.

A.2 Proof of Lemma 2.1

Lemma A.1. For any optimizer gradient update $\boldsymbol{g}_l$, satisfying the spectral scaling condition for feature learning is equivalent to setting the layer-wise learning rate η_l as:

$$\eta_l = \Theta\left(\frac{\sqrt{n_l/n_{l-1}}}{\|\boldsymbol{g}_l\|_*}\right). \tag{12}$$

In this section, we provide a simplified proof of Lemma A.1 (referenced as Lemma 2.1 in the main text) to illustrate the core mechanism. For the complete rigorous derivation and additional theoretical details, we refer readers to [45].

Definition A.2 (Desideratum 1 in [45])**.** Let $\mathbf{a}_l(x) \in \mathbb{R}^{n_l}$ denote the features of input x at layer l of a neural network, and let $\Delta\mathbf{a}_l(x) \in \mathbb{R}^{n_l}$ denote their change after a gradient step. We desire that:

$$\|\mathbf{a}_l\|_2 = \Theta(\sqrt{n_l}) \quad \text{and} \quad \|\Delta\mathbf{a}_l\|_2 = \Theta(\sqrt{n_l}), \tag{13}$$

for layers $l = 1, \ldots, L-1$.

Proof. Recall that the weight update at step t is given by $\Delta \boldsymbol{W}_l = -\eta_l \cdot \boldsymbol{g}_l$. Taking the spectral norm of both sides, we have:

$$\|\Delta \boldsymbol{W}_l\|_* = \eta_l \|\boldsymbol{g}_l\|_*. \tag{14}$$

To ensure feature learning in the limit of infinite width, the spectral scaling condition requires the weight update to scale as:

$$\|\Delta \boldsymbol{W}_l\|_* = \Theta\left(\sqrt{\frac{n_l}{n_{l-1}}}\right). \tag{15}$$

This requirement stems from the property that the change in output activations satisfies $\|\Delta \mathbf{a}_l\|_2 \asymp \|\Delta \boldsymbol{W}_l\|_* \cdot \|\mathbf{a}_{l-1}\|_2$ when the input $\mathbf{a}_{l-1}$ aligns with the top singular vector of the weight update $\Delta \boldsymbol{W}_l$. Since $\|\mathbf{a}_{l-1}\|_2 = \Theta(\sqrt{n_{l-1}})$, ensuring $\|\Delta \boldsymbol{W}_l\|_* = \Theta(\sqrt{n_l/n_{l-1}})$ guarantees that the feature updates scale comparably to the activations themselves ($\|\Delta \mathbf{a}_l\|_2 = \Theta(\sqrt{n_l})$), enabling stable feature learning.

Substituting this condition into the update equation, we solve for η_l:

$$\eta_l \|\boldsymbol{g}_l\|_* = \Theta\left(\sqrt{\frac{n_l}{n_{l-1}}}\right) \implies \eta_l = \Theta\left(\frac{\sqrt{n_l/n_{l-1}}}{\|\boldsymbol{g}_l\|_*}\right). \tag{16}$$

This completes the proof. □

A.3 Derivation of Layer-wise Learning Rate Scaling Rules

Table 2: Summary of layer-wise learning rate scaling rules for μP under different privacy settings. This table restates Table 1 so that the derivation in this section can be read on its own.

Optimizer	**Non-Private**	**Clipping Only**	**Differentially Private (DP)**
SGD	$\frac{n_l}{n_{l-1}}$	$\sqrt{\frac{n_l}{n_{l-1}}}$	$\frac{\sqrt{n_l/n_{l-1}}}{\frac{\sigma_{\text{DP}}}{B}(\sqrt{n_l}+\sqrt{n_{l-1}})}$
Adam	$\frac{1}{n_{l-1}}$	$\frac{1}{n_{l-1}}$	$\frac{\sqrt{n_l/n_{l-1}}}{\sqrt{n_l}+\sqrt{n_{l-1}}}$
Muon	$\sqrt{\frac{n_l}{n_{l-1}}}$	$\sqrt{\frac{n_l}{n_{l-1}}}$	$\sqrt{\frac{n_l}{n_{l-1}}}$

We derive the optimal layer-wise learning rates η_l by ensuring that the spectral norm of the weight update satisfies the Feature Learning condition:

$$\|\Delta \boldsymbol{W}_l\|_* = \Theta\left(\sqrt{\frac{n_l}{n_{l-1}}}\right). \tag{17}$$

First, we recall the assumptions and algebraic facts used in our derivations.

Fact A.3. For rank-1 matrices (such as per-sample gradients in standard networks), the spectral norm is equal to the Frobenius norm: $\|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_* = \|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_F$.

Assumption A.4. We assume that the per-sample gradients are sufficiently aligned across the batch, such that the spectral norm of the sum scales with the sum of the spectral norms:

$$\left\| \frac{1}{B} \sum_{i=1}^{B} g(\boldsymbol{p}_i) \right\|_* = \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \|g(\boldsymbol{p}_i)\|_* \right). \tag{18}$$

Fact A.5. The sign operator preserves the rank-1 structure for outer products: $\text{sign}(uv^\top) = \text{sign}(u)\text{sign}(v)^\top$. Consequently, $\|\text{sign}(uv^\top)\|_* = \|\text{sign}(uv^\top)\|_F = \Theta(\sqrt{n_l n_{l-1}})$.

A.3.1 Non-Private Case (Standard μP)

In the standard non-private setting, we derive the scaling of the gradient spectral norms for SGD and Adam.

SGD: Using the definition of the update and applying our alignment assumption:

$$\begin{aligned}\|g(\boldsymbol{p}_l)\|_* = \left\| \frac{1}{B} \sum_{i=1}^{B} \nabla_{\boldsymbol{W}_l} \mathcal{L}_i \right\|_* &\overset{\text{(A.2)}}{=} \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_* \right) \\ &\overset{\text{(A.1)}}{=} \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \sqrt{\frac{n_{l-1}}{n_l}} \right) = \Theta\left(\sqrt{\frac{n_{l-1}}{n_l}} \right).\end{aligned} \tag{19}$$

SignSGD / Adam: For sign-based optimizers, the update magnitude depends on the full dimension of the matrix due to the element-wise operation:

$$\begin{aligned}\|g(\boldsymbol{p}_l)\|_* = \left\| \frac{1}{B} \sum_{i=1}^{B} \text{sign}(\nabla_{\boldsymbol{W}_l} \mathcal{L}_i) \right\|_* &\overset{\text{(A.2)}}{=} \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \|\text{sign}(\nabla_{\boldsymbol{W}_l} \mathcal{L}_i)\|_* \right) \\ &\overset{\text{(A.3)}}{=} \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \|\text{sign}(\nabla_{\boldsymbol{W}_l} \mathcal{L}_i)\|_F \right) = \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \sqrt{n_{l-1} n_l} \right) = \Theta\left(\sqrt{n_{l-1} n_l} \right).\end{aligned} \tag{20}$$

Muon: The Muon optimizer updates weights using an orthogonalized momentum. The update rule is defined as:

$$\Delta \boldsymbol{W}_l = -\eta_l \cdot g(\boldsymbol{p}_l) := -\eta_l \cdot \mathcal{O}(m(\boldsymbol{p}_l)), \tag{21}$$

where $m(\boldsymbol{p}_l)$ is the momentum accumulator for the gradient, and $\mathcal{O}(\cdot)$ denotes the orthogonalization operator. This operator projects the momentum matrix onto the manifold of orthogonal matrices by solving the following optimization problem:

$$\mathcal{O}(\boldsymbol{M}) = \underset{\boldsymbol{U}:\boldsymbol{U}^\top \boldsymbol{U} = I}{\text{argmin}} \|\boldsymbol{M} - \boldsymbol{U}\|_F. \tag{22}$$

The constraint $\boldsymbol{U}^\top \boldsymbol{U} = I$ explicitly controls the spectral properties of the update. By definition, any matrix $\boldsymbol{U}$ satisfying this constraint has all singular values equal to 1. Consequently, the spectral norm of the update direction is invariant to the choice of momentum or the presence of privacy noise:

$$\|g(\boldsymbol{p}_l)\|_* \equiv \|\mathcal{O}(m(\boldsymbol{p}_l))\|_* = 1 = \Theta(1). \tag{23}$$

To satisfy the feature learning condition $\|\Delta \boldsymbol{W}_l\|_* = \Theta(\sqrt{n_l / n_{l-1}})$, we apply the relationship $\eta_l = \|\Delta \boldsymbol{W}_l\|_* / \|g(\boldsymbol{p}_l)\|_*$ using the spectral norms derived above. These scaling rules correspond directly to the first column of Table 2.

A.3.2 Clipping-Only Case

In this regime, we assume per-sample gradient clipping is active, but noise injection is negligible. The pre-conditioned gradient is defined as $g(p_l)$.

Note that for SignSGD/Adam and Muon, the clipping operation (which scales the gradient by a scalar) does not fundamentally alter the rank or the spectral properties of the update matrix; thus, their scaling orders remain unchanged compared to the non-private setting. However, for SGD, the normalization by the Frobenius norm significantly alters the scaling behavior, as derived below.

SGD: The update is defined as the average of clipped per-sample gradients. By invoking the sufficient alignment assumption (Assumption A.4), we derive:

$$\begin{aligned}\|g(\boldsymbol{p}_l)\|_* = \left\| \frac{1}{B} \sum_{i=1}^{B} \frac{\nabla_{\boldsymbol{W}_l} \mathcal{L}_i}{\sqrt{D} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_F} \right\|_* &\overset{\text{(A.2)}}{=} \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \frac{\|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_*}{\sqrt{D} \|\nabla_{\boldsymbol{W}_l} \mathcal{L}_i\|_F} \right) \\ &\overset{\text{(A.1)}}{=} \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \frac{1}{\sqrt{D}} \right) = \Theta(1),\end{aligned}$$

where D denotes the number of layers, which is independent of the model width.

A.3.3 Differentially Private Case (Noise-Dominated)

In the DP setting, the update is dominated by the injected Gaussian noise $\boldsymbol{\xi} \sim \mathcal{N}(0, I_{n_l \times n_{l-1}})$. We utilize the result from Random Matrix Theory that the spectral norm of a random Gaussian matrix of size $n_l \times n_{l-1}$ scales as $\sqrt{n_l} + \sqrt{n_{l-1}}$.

DP-SGD: The gradient is dominated by noise scaled by $\frac{\sigma_{\text{DP}}}{B}$:

$$\|g(\boldsymbol{p}_l)\|_* \asymp \left\| \frac{\sigma_{\text{DP}}}{B} \boldsymbol{\xi} \right\|_* = \Theta\left(\frac{\sigma_{\text{DP}}}{B} (\sqrt{n_l} + \sqrt{n_{l-1}}) \right). \tag{24}$$

DP-Adam: In the noise-dominated regime, the update geometry of Adam is governed by the injected noise, and the spectral norm of the update direction scales as:

$$\|g(\boldsymbol{p}_l)\|_* = \Theta(\sqrt{n_l} + \sqrt{n_{l-1}}). \tag{25}$$

This differs from the clipping-only case, where the norm scales as $\sqrt{n_l n_{l-1}}$, because the noise does not align across coordinates as the signal gradients do.

DP-Muon: As in the non-private case, the orthogonalization sets all singular values of the momentum to one, regardless of the privacy noise:

$$\|g(\boldsymbol{p}_l)\|_* = \Theta(1). \tag{26}$$

In practice, Muon approximates $\mathcal{O}(\cdot)$ with a few Newton–Schulz iterations, which act only on the singular values and bring them close to one, so the width dependence is unchanged.

A.4 Boundary Condition at the Input Layer

Unlike hidden layers, where the input signal strength depends on the width of the preceding layer n_{l-1}, the input layer receives fixed data samples $\boldsymbol{x} \in \mathbb{R}^{n_0}$. The feature norm at the first layer is bounded by:

$$\|\mathbf{a}_1\| \leq \|\boldsymbol{W}_{in}\|_* \cdot \|\boldsymbol{x}\|. \tag{27}$$

Standard data preprocessing in deep learning (e.g., image whitening or standardization) typically ensures that input features have zero mean and unit variance per dimension. Consequently, the norm of the input vector scales with the square root of the input dimension:

$$\mathbb{E}[\|\boldsymbol{x}\|^2] = \sum_{j=1}^{n_0} \mathbb{E}[x_j^2] = n_0 \cdot \Theta(1) \implies \|\boldsymbol{x}\| = \Theta(\sqrt{n_0}). \tag{28}$$

Given that n_0 is fixed for a specific task, we do not explicitly distinguish this scaling factor in our main notation. However, we note that if the data is standardized differently (e.g., non-unit variance), the learning rate scale for the input layer would need to be adjusted accordingly.

A.5 SNR for Layer Normalization Parameters

For the affine parameters in Layer Normalization (specifically the gain vector $\gamma \in \mathbb{R}^{n_l}$), the operation corresponds to element-wise multiplication. This is mathematically equivalent to matrix multiplication by a diagonal matrix $\boldsymbol{W}_{\mathcal{L}_{\text{norm}}} = \text{diag}(\gamma)$, where $\boldsymbol{W}_{\mathcal{L}_{\text{norm}}} \in \mathbb{R}^{n_l \times n_l}$.

Consequently, the spectral norm of the LayerNorm gradient update matrix, denoted as $g(p_{\mathcal{L}_{\text{norm}}})$, behaves differently from dense weight matrices. The spectral norm of a diagonal matrix is simply the maximum absolute value of its diagonal entries (the infinity norm of the vector):

$$\|g(\boldsymbol{p}_{\mathcal{L}_{\text{norm}}})\|_* = \|\text{diag}(g(\gamma))\|_* = \max_{i \in \{1,\ldots,n_l\}} |g(\gamma)_i|. \tag{29}$$

Unlike dense matrices where the spectral norm typically scales with $\sqrt{n_l}$, the scaling here is governed by *Extreme Value Theory*. Assuming the gradient entries (or the injected DP noise) are independent and identically distributed with sub-Gaussian tails, the expected maximum of n_l variables scales logarithmically:

$$\|g(\boldsymbol{p}_{\mathcal{L}_{\text{norm}}})\|_* = \Theta\left(\sqrt{\ln n_l} \right). \tag{30}$$

To maintain the SNR condition, we require the effective noise magnitude to scale consistently. This implies the following relationship:

$$\frac{\sigma \sqrt{D_k}}{B} \sqrt{\ln(n_k)} = \frac{\sigma' \sqrt{D'_k}}{B} \sqrt{\ln(n'_k)} \iff \left(\frac{\sigma'}{\sigma} \right)^2 \frac{D'_k}{D_k} = \frac{\ln(n_k)}{\ln(n'_k)}, \quad \forall k \in \mathcal{L}_{\text{norm}}. \tag{31}$$

A.6 Scaling Rules for Transformers with LayerNorm

In the main text, Equation (8) establishes the layer-wise constraints for standard linear networks. Here, we formally extend this system to Transformer architectures by explicitly accounting for normalization layers. Let $\mathcal{L}_{\text{linear}}$ denote the set of indices for standard linear layers, and $\mathcal{L}_{\text{norm}}$ denote the set for normalization layers.

Remark A.6. Note that the operation of LayerNorm in Transformers is analogous to diagonal matrix multiplication. However, driven by extreme value theory, the spectral norm of its gradient scales with $\sqrt{\ln n}$ rather than the $\sqrt{n}$ scaling observed in linear layers. Specifically, the expanded system of constraints is given by:

$$\begin{cases} \left(\dfrac{\sigma'}{\sigma}\right)^2 \dfrac{D'_l}{D_l} = \left(\dfrac{\sqrt{n_{in}^{(l)}} + \sqrt{n_{out}^{(l)}}}{\sqrt{n_{in}^{(l)\prime}} + \sqrt{n_{out}^{(l)\prime}}}\right)^2, & \forall l \in \mathcal{L}_{\text{linear}}, \\ \left(\dfrac{\sigma'}{\sigma}\right)^2 \dfrac{D'_k}{D_k} = \dfrac{\ln(n_k)}{\ln(n'_k)}, & \forall k \in \mathcal{L}_{\text{norm}}, \\ \displaystyle\sum_{j \in \mathcal{L}_{\text{linear}} \cup \mathcal{L}_{\text{norm}}} \frac{1}{D_j} = \sum_{j \in \mathcal{L}_{\text{linear}} \cup \mathcal{L}_{\text{norm}}} \frac{1}{D'_j} = 1. \end{cases} \tag{32}$$

B Empirical Support for Assumption 3.2

In this section, we provide empirical validation for Assumption 3.2:

$$\left\| \frac{1}{B} \sum_{i=1}^{B} g(\boldsymbol{p}_i) \right\|_* = \Theta\left(\frac{1}{B} \sum_{i=1}^{B} \|g(\boldsymbol{p}_i)\|_* \right)$$

We evaluate a 5-layer Multi-Layer Perceptron (MLP) trained with DP-SGD and DP-Adam. DP-Muon is omitted from this analysis, as its preconditioning operation inherently maps updates to matrices with a unit spectral norm. Throughout the experiments, we maintain a fixed noise multiplier σ and learning rate, while systematically varying the batch size $B \in \{125, 250, 500, 1000\}$ and the network width $\in \{256, 512, 1024\}$.

Table 3: Empirical spectral norm of the mini-batch average gradients for DP-SGD across varying batch sizes and network widths.

	width = 256				width = 512				width = 1024			
$B =$	125	250	500	1000	125	250	500	1000	125	250	500	1000
Input	1.13	0.56	0.28	0.14	1.25	0.62	0.31	0.16	1.40	0.70	0.35	0.17
First Hidden	0.51	0.25	0.13	0.06	0.72	0.36	0.18	0.09	1.03	0.51	0.26	0.13
Second Hidden	0.50	0.25	0.13	0.06	0.72	0.36	0.18	0.09	1.01	0.51	0.26	0.13
Third Hidden	0.51	0.25	0.13	0.07	0.72	0.36	0.18	0.09	1.02	0.51	0.25	0.13
Output	0.30	0.16	0.08	0.04	0.39	0.20	0.10	0.06	0.56	0.28	0.14	0.07

Table 4: Empirical spectral norm of the mini-batch average gradients for DP-Adam across varying batch sizes and network widths.

	width = 256				width = 512				width = 1024			
$B =$	125	250	500	1000	125	250	500	1000	125	250	500	1000
Input	71.05	71.37	71.22	70.86	77.51	77.70	77.85	77.50	87.65	87.47	87.14	86.90
First Hidden	31.52	31.46	31.57	31.52	44.86	44.57	45.15	44.94	63.45	63.48	63.44	63.71
Second Hidden	30.77	31.80	32.05	31.43	45.21	44.79	44.64	44.81	64.07	63.81	63.86	63.59
Third Hidden	31.96	31.82	31.41	31.55	44.66	44.72	44.82	44.93	63.52	63.70	63.55	63.93
Output	18.64	18.52	18.09	19.00	24.62	24.60	24.79	25.72	34.20	35.04	34.22	34.20

As established in our theoretical analysis, the spectral norm of the DP-SGD gradient scales as:

$$\|g(\boldsymbol{p}_l)\|_* \propto \frac{\sigma}{B}\left(\sqrt{n_l} + \sqrt{n_{l-1}}\right) \tag{33}$$

where σ is the noise multiplier, B is the batch size, and n_l, n_{l-1} denote the input and output dimensions of layer l, respectively. Table 3 empirically confirms this prediction: for a given layer and width (along each row), the spectral norm approximately halves as the batch size B doubles, verifying the expected $1/B$ dependence. Furthermore, across the three width configurations, the values grow proportionally to $\sqrt{n_l} + \sqrt{n_{l-1}}$. Together, these observations strongly validate the predicted scaling behavior for DP-SGD. For DP-Adam, our analysis indicates that the spectral norm scales as:

$$\|g(\boldsymbol{p}_l)\|_* \propto \left(\sqrt{n_l} + \sqrt{n_{l-1}}\right) \tag{34}$$

Table 4 corroborates this behavior. The empirical spectral norm remains remarkably stable across different batch sizes for any given layer and width combination, demonstrating its independence from B. Simultaneously, it correctly scales proportionally to $\sqrt{n_l} + \sqrt{n_{l-1}}$ as the network width increases.

We further repeat the check for DP-Adam on a GPT-2 model trained on Shakespeare-char, varying the width in $\{320, 640, 1280\}$ and the batch size $B \in \{64, 128, 256\}$. As shown in Table 5, the spectral norms are again independent of the batch size and grow with $\sqrt{n_l} + \sqrt{n_{l-1}}$, confirming that the assumption also holds for Transformers. Moreover, DP-μP does not require the alignment to hold exactly: the assumption enters our analysis only through the signal norm, so it suffices that the degree of alignment is comparable between the proxy and the target model.

Table 5: Empirical spectral norm of the mini-batch average gradients for DP-Adam on GPT-2 across varying batch sizes and network widths.

	width = 320			width = 640			width = 1280		
$B =$	64	128	256	64	128	256	64	128	256
Block 5 QKV	48.43	48.38	48.40	68.58	68.70	68.96	97.29	97.69	97.67
Block 5 FC1	53.06	52.87	53.71	75.45	75.86	76.01	107.15	106.98	107.16
Block 5 FC2	53.05	53.73	53.10	75.65	75.36	75.57	107.05	107.52	107.52
Block 10 QKV	48.39	48.36	47.99	69.27	68.85	68.88	97.46	97.68	97.71
Block 10 FC1	53.17	52.91	53.08	75.44	75.97	75.36	107.36	106.92	106.91
Block 10 FC2	53.27	53.20	53.36	75.60	75.44	75.58	107.07	106.94	106.94
Head	26.05	25.50	25.32	32.75	33.01	33.23	44.09	43.04	42.98

C Supplementary Experiments

C.1 Additional MLP Results for Adam

In this section, we provide the supplementary MLP experimental results referenced in the main text. Figure 9 illustrates the training loss of a 5-layer MLP on CIFAR-10 over 10 epochs across varying model widths. We compare SP and μP with Adam under both non-private and differentially private settings.

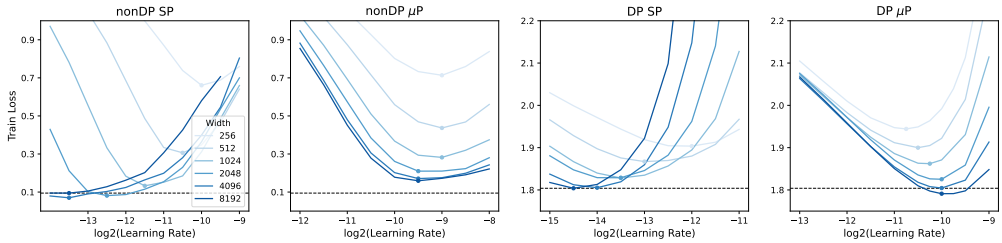


Figure 9: **Comparison between SP and μP (Adam).** Similar to SGD, we adjust the clipping coefficient and noise to satisfy the SNR condition.

C.2 Batch Size Transfer

In this section, we supplement our main analysis with batch size transfer experiments, as illustrated in Figures 10 and 11. For a fixed model width, we adjust the noise multiplier σ proportionally to the batch size. Crucially, to ensure a valid comparison of optimization trajectories, we fix the total number of training iterations across settings rather than the number of epochs.

The noise multiplier and the batch size enter the noise-to-signal ratio only through σ/B, so scaling them together leaves the SNR unchanged. Fixing the number of iterations also keeps the privacy budget approximately fixed as the batch size varies, since the sampling rate grows with B at the same rate as σ. Hence the transfer applies to any batch size, once it is paired with the noise multiplier returned by the privacy accountant.

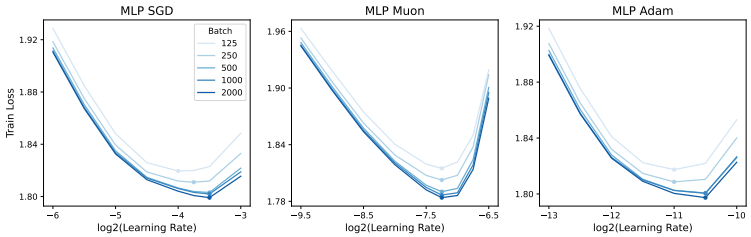


Figure 10: **Batch size transfer of 5-layer MLP on CIFAR-10.** We compare training loss across different batch sizes using SGD, Adam, and Muon optimizers. Curves align when plotted against iterations, confirming effective transfer.

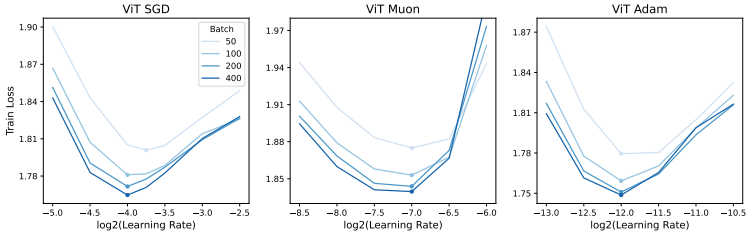


Figure 11: **Batch size transfer of ViT on CIFAR-10.** Similar to the MLP results, the ViT loss curves align across different batch sizes when plotted against iterations, confirming effective transfer for the Transformer architecture.

C.3 ViT Supplementary Results

In this section, we present supplementary evidence demonstrating the failure of Standard Parametrization (SP), alongside the results on CIFAR-10 for Vision Transformers (ViT). As shown in Figure 12, when trained with DP-SGD and DP-Adam, SP models fail to maintain consistent training dynamics across varying widths. In contrast, our DP-μP approach (Figure 13) demonstrates perfect hyperparameter transfer.

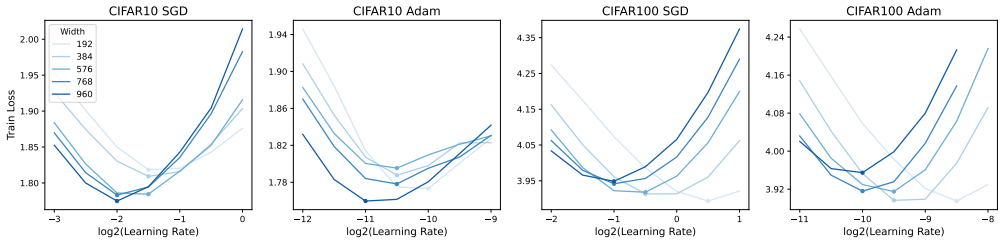


Figure 12: **Failure of Standard Parametrization (SP) on ViT.** As model width increases, the training dynamics under SP become unstable or diverge, failing to exhibit the hyperparameter transfer seen in μP.

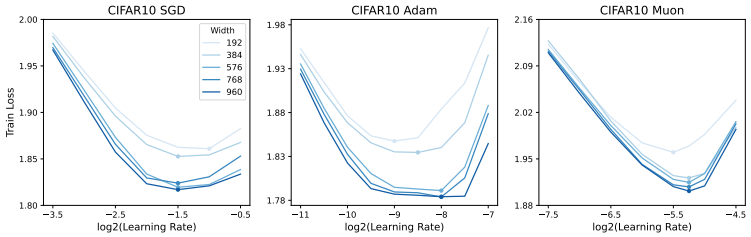


Figure 13: **μP ViT on CIFAR-10.** The optimizer configuration is SGD, Adam and Muon (with SGD boundary updates).

C.4 Final Utility and Robustness across Seeds

We examine whether transfer also holds for the final test accuracy, and whether it is robust to the randomness of the DP noise and the initialization. For the 5-layer MLP and the ViT on CIFAR-10, we grid search the learning rate at every width, select the one achieving the best test accuracy, and repeat each run over 5 seeds fixed in advance. All runs use $(\epsilon, \delta) = (2, 10^{-5})$ for the target model, so the comparison is made at matched privacy. As shown in Table 6, the optimal learning rate under SP drifts with width, while under μP it stays nearly constant. At the largest width, μP reaches slightly higher test accuracy than SP tuned directly at that width, for both SGD and Adam. Table 7 shows the same stability for the ViT, and in both tables the optimal learning rate is stable across seeds. These optimal learning rates differ from those in the training-loss figures, since the learning rate minimizing training loss need not maximize test accuracy.

Table 6: Optimal $\log_2$ learning rate (by test accuracy) at each width for a 5-layer MLP on CIFAR-10, reported as mean $\pm$ std over 5 seeds. Slope β is the regression slope of the optimal $\log_2$ learning rate against $\log_2$ width, and Acc is the test accuracy (%) at the largest width.

width	SP SGD	μP SGD	SP Adam	μP Adam	μP Muon
256	-1.50 ± 0.25	-3.55 ± 0.21	-10.50 ± 0.18	-9.75 ± 0.18	-8.25 ± 0.18
512	-1.70 ± 0.27	-3.60 ± 0.14	-10.90 ± 0.29	-9.65 ± 0.22	-8.75 ± 0.31
1024	-2.25 ± 0.25	-3.25 ± 0.00	-11.10 ± 0.22	-9.75 ± 0.18	-8.55 ± 0.11
2048	-2.85 ± 0.14	-3.15 ± 0.14	-11.55 ± 0.21	-9.15 ± 0.14	-8.75 ± 0.25
4096	-3.45 ± 0.21	-2.90 ± 0.14	-12.30 ± 0.21	-9.10 ± 0.14	-8.65 ± 0.22
8192	-4.00 ± 0.18	-2.90 ± 0.14	-12.85 ± 0.14	-8.90 ± 0.14	-8.65 ± 0.14
Slope β	-0.52	$+0.16$	-0.47	$+0.19$	-0.05
Acc	41.16 ± 0.23	41.77 ± 0.38	41.14 ± 0.29	41.80 ± 0.28	40.76 ± 0.16

C.5 Standard Parametrization Coordinate Check

We visualize the coordinate-wise activation norms for Standard Parametrization in Figures 14, 15 and 16, to contrast with the stable behavior of μP shown in the main text.

Table 7: Optimal $\log_2$ learning rate (by test accuracy) at each width for a ViT on CIFAR-10, reported as mean $\pm$ std over 5 seeds. Slope β and Acc are defined as in Table 6.

width	μP SGD	μP Adam	μP Muon
192	-1.35 ± 0.34	-8.50 ± 0.25	-5.85 ± 0.14
384	-1.40 ± 0.22	-8.05 ± 0.11	-5.70 ± 0.11
576	-1.00 ± 0.35	-8.10 ± 0.29	-5.65 ± 0.14
768	-0.95 ± 0.45	-8.10 ± 0.22	-5.80 ± 0.21
960	-1.15 ± 0.22	-8.30 ± 0.21	-5.65 ± 0.14
Slope β	$+0.16$	$+0.10$	$+0.06$
Acc	33.37 ± 0.43	36.79 ± 0.21	30.08 ± 0.41

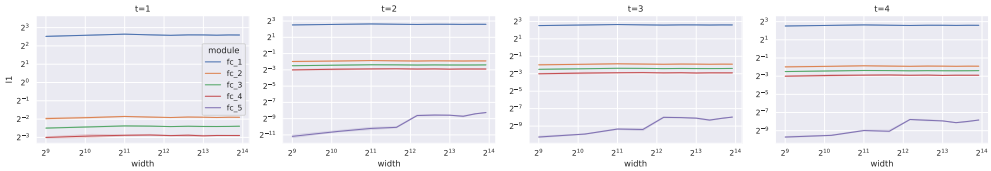


Figure 14: **Coordinate check for MLP (SP) with SNR condition.** Even with the SNR condition applied, standard parametrization often fails to preserve feature scale across widths.

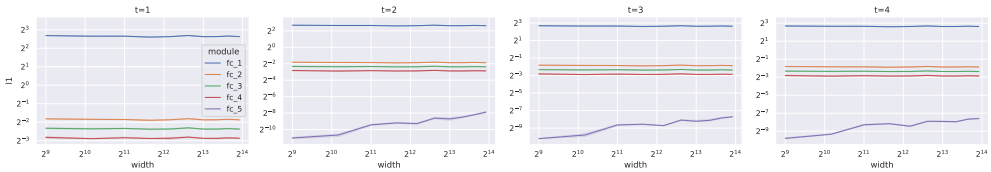


Figure 15: **Coordinate check for MLP (SP) without SNR condition.** Without privacy-aware scaling, feature activations diverge significantly as width increases.

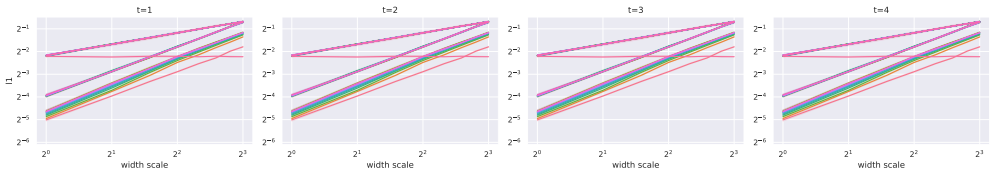


Figure 16: **Coordinate check for ViT (SP).** Similar to the MLP case, SP leads to inconsistent activation scales in Vision Transformers.

C.6 Illustrative Example of Parameter Derivation

In this section, we provide a concrete parameter derivation to illustrate the DP-μP framework depicted in Figure 1. We consider a 5-layer MLP trained on CIFAR-10, where the input dimension is $n_0 = 3072$ ($3 \times 32 \times 32$) and the output dimension is $n_5 = 10$. Our objective is to transfer hyperparameters from a small proxy model with hidden widths $n'_{1,\dots,4} = 128$ to a large target model with widths $n_{1,\dots,4} = 8192$.

Before the search, we fix the proxy width, the target width, the privacy budget, the batch size and the number of training steps, since the last three determine the noise multiplier.

Step 1: Calibrate the noise multiplier of the target. For a target privacy budget of $(\epsilon, \delta) = (2, 10^{-5})$ on the CIFAR-10 dataset ($50,000$ samples), we configure the training with a batch size of

$B = 500$ and train for 10 epochs. Based on these parameters, the privacy accountant computes the required noise multiplier for the large model to be $\sigma \approx 1.0229$, which we approximate as $\sigma \approx 1$.

Step 2: Adopt uniform clipping on the proxy. To facilitate the hyperparameter search on the proxy model, we employ a uniform clipping (e.g. $D'_l = 5$). We use uniform clipping on the proxy in all experiments.

Step 3: Solve the SNR constraints. By solving the SNR constraints defined in Equation (8), we obtain the required noise multiplier for the proxy model:

$$\sigma' \approx 6.91. \tag{35}$$

Simultaneously, we derive the layer-wise scaling coefficients $\frac{1}{\sqrt{\boldsymbol{D}}}$ for the target large model:

$$\boldsymbol{D} \approx (49.9, 3.73, 3.73, 3.73, 5.7) \Longrightarrow \frac{1}{\sqrt{\boldsymbol{D}}} \approx (0.14, 0.52, 0.52, 0.52, 0.42). \tag{36}$$

This specific configuration ensures that the Signal-to-Noise Ratio (SNR) is preserved across widths.

Step 4: Search on the proxy. We proceed to perform hyperparameter tuning on the small proxy model using the derived noise multiplier $\sigma' \approx 6.91$ and a uniform clipping coefficient $1/\sqrt{D'} = 1/\sqrt{5}$. We conduct a grid search to identify the optimal learning rate. For illustrative simplicity, we assume this search yields an optimal value of $\eta^*_{\text{small}} = 1$. Note that we employ a global learning rate for the small model, as the primary objective at this stage is to ensure numerical stability (preventing gradient vanishing or explosion) rather than strictly enforcing layer-wise alignment.

Step 5: Transfer to the target. To determine the layer-wise learning rates for the large target model, we scale the optimal proxy rate $\eta^*_{\text{small}} = 1$ based on the spectral norm ratios, in accordance with Lemma 2.1. The transfer rule is given by:

$$\frac{\eta^{(l)}_{\text{large}}}{\eta^*_{\text{small}}} = \frac{\sqrt{n_l/n_{l-1}}/\left(\sigma \cdot \left(\sqrt{n_l} + \sqrt{n_{l-1}}\right)\right)}{\sqrt{n'_l/n'_{l-1}}/\left(\sigma' \cdot \left(\sqrt{n'_l} + \sqrt{n'_{l-1}}\right)\right)}, \tag{37}$$

which results in $\boldsymbol{\eta}_{\text{large}} \approx (25.28,\ 0.86,\ 0.86,\ 0.86,\ 0.13)$. The rest of the training recipe found on the proxy is reused on the target as it is.

Selecting the batch size. If the batch size is also to be selected, the noise multiplier has to be recomputed for each candidate batch size. Steps 1–5 are then repeated for each candidate, and the best candidate is chosen on the proxy model.

C.7 End-to-End Privacy Accounting

We note that DP-μP serves as a method to transferring hyperparameters for DP optimization without privacy cost. We distinguish the privacy cost of hyperparameter search from that of training the final target model, providing an end-to-end privacy accounting.

Let $\mathcal{D}$ denote the private dataset. We first construct the proxy training configuration according to the DP-μP scaling recipe: the target privacy parameters and the two architectures determine, through the SNR-matching constraints in (8), the proxy noise multiplier and the layer-wise clipping parameters of the target. This step depends only on the privacy parameters and the architectures and does not access $\mathcal{D}$, so it incurs no privacy cost.

Hyperparameter search is then performed only over proxy runs generated under this DP-μP-aligned configuration. The selected proxy learning rate is subsequently mapped to the target-model layer-wise learning rates using the deterministic transfer rule T, given by Lemma 2.1 and illustrated in Appendix C.6.

Privacy cost of hyperparameter search. Suppose the complete search-and-selection mechanism $\mathcal{M}_{\text{search}}$ is $(\epsilon_{\text{search}}, \delta_{\text{search}})$-DP. Importantly, this privacy guarantee must account for the entire search procedure, rather than a single proxy-model training run. For example, if multiple candidate learning rates are evaluated on the same private dataset, the privacy costs of those evaluations and the subsequent selection must be incorporated into $(\epsilon_{\text{search}}, \delta_{\text{search}})$. Alternatively, the search may be performed using public data, in which case $\epsilon_{\text{search}} = \delta_{\text{search}} = 0$.

Privacy cost of target-model training. For every fixed transferred hyperparameter configuration $\hat{\eta}_{\text{target}}$, suppose $\mathcal{M}_{\text{train}}(\mathcal{D}; \hat{\eta}_{\text{target}})$ is $(\epsilon_{\text{train}}, \delta_{\text{train}})$-DP. Although the target-model training procedure depends adaptively on the private output of $\mathcal{M}_{\text{search}}$, adaptive sequential composition shows that the end-to-end mechanism

$$\mathcal{M}_{\text{train}}(\mathcal{D}; T(\mathcal{M}_{\text{search}}(\mathcal{D})))$$

satisfies

$$(\epsilon_{\text{search}} + \epsilon_{\text{train}},\ \delta_{\text{search}} + \delta_{\text{train}})\text{-DP}.$$

Hence, for a prescribed total privacy budget $(\epsilon_{\text{total}}, \delta_{\text{total}})$, it is sufficient to allocate the budgets such that

$$\epsilon_{\text{search}} + \epsilon_{\text{train}} \le \epsilon_{\text{total}}, \qquad \delta_{\text{search}} + \delta_{\text{train}} \le \delta_{\text{total}}.$$

A simple parameterization is

$$\epsilon_{\text{search}} = \gamma\epsilon_{\text{total}}, \qquad \epsilon_{\text{train}} = (1-\gamma)\epsilon_{\text{total}},$$

and

$$\delta_{\text{search}} = \gamma_\delta\delta_{\text{total}}, \qquad \delta_{\text{train}} = (1-\gamma_\delta)\delta_{\text{total}},$$

for predetermined $\gamma, \gamma_\delta \in [0, 1]$ before accessing $\mathcal{D}$.

Tighter accounting. The additive guarantee above follows directly from basic sequential composition and provides the simplest end-to-end accounting rule. In practice, tighter guarantees can be obtained by carrying out both stages in a common privacy accountant. For example, under Rényi differential privacy, if the two stages satisfy $\rho_{\text{search}}(\alpha)$-RDP and $\rho_{\text{train}}(\alpha)$-RDP at order α, respectively, then

$$\rho_{\text{total}}(\alpha) = \rho_{\text{search}}(\alpha) + \rho_{\text{train}}(\alpha),$$

and the composed guarantee can be converted once to $(\epsilon_{\text{total}}, \delta_{\text{total}})$-DP at the end. More refined hyperparameter-tuning or privacy-loss-distribution accountants (such as GDP) may likewise be used without changing our transfer mechanism.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction outline the key challenges addressed, our contributions, and a summary of the experimental findings.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include the limitation of our work in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: See Section 3 and Section A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose the complete training setups, including datasets, model architectures, privacy budgets, and learning rate ranges in Section 4, and provide a detailed numerical walkthrough of the hyperparameter derivation in Appendix C.6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We used the public dataset, and we plan to publicly release our code once the status of the paper is confirmed.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all relevant training details, including datasets, optimizers, and privacy budgets in Section 4, and we provide a comprehensive numerical example of hyperparameter selection and settings in Appendix C.6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For the MLP and ViT experiments, we report the mean and standard deviation over 5 seeds fixed in advance, capturing the randomness of the DP noise and the initialization (Appendix C.4). The GPT-2 experiments are run once due to limited computational resources.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Section 4 and Section 4.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We fully adhere to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section 1.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use standard, publicly available datasets and model architectures, properly citing their original sources in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We will disclose the code after the confirmation of the status of the paper, but currently no new assets is available.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our research does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our research does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.