μP^2: Effective Sharpness Aware Minimization Requires Layerwise Perturbation Scaling

Moritz Haas[1] **Jin Xu**[2] **Volkan Cevher**[3,4] **Leena Chennuru Vankadara**[4]

[1]University of Tübingen, Tübingen AI Center* [2]University of Oxford*
[3]LIONS, EPFL* [4]AGI Foundations, Amazon

Abstract

Sharpness Aware Minimization (SAM) enhances performance across various neural architectures and datasets. As models are continually scaled up to improve performance, a rigorous understanding of SAM's scaling behaviour is paramount. To this end, we study the infinite-width limit of neural networks trained with SAM, using the Tensor Programs framework. Our findings reveal that the dynamics of standard SAM effectively reduce to applying SAM solely in the last layer in wide neural networks, even with optimal hyperparameters. In contrast, we identify a stable parameterization with layerwise perturbation scaling, which we call *Maximal Update and Perturbation Parameterization* (μP^2), that ensures all layers are both feature learning and effectively perturbed in the limit. Through experiments with MLPs, ResNets and Vision Transformers, we empirically demonstrate that μP^2 is the first parameterization to achieve hyperparameter transfer of the joint optimum of learning rate and perturbation radius across model scales. Moreover, we provide an intuitive condition to derive μP^2 for other perturbation rules like Adaptive SAM and SAM-ON, also ensuring balanced perturbation effects across all layers.

1 Introduction

Sharpness Aware Minimization (SAM) (Foret et al., 2021) and its variants (Kwon et al., 2021; Müller et al., 2024) improves generalization performance across a wide range of neural architectures and datasets (Chen et al., 2021; Kaddour et al., 2022). In the SAM formulation, we minimize a given loss L between our prediction and the data y as a function of the architecture's weights W, where an adversary simultaneously maximizes the same loss by perturbing the weights within a budget ρ.

A standard SAM update for an L-hidden layer multi layer perceptron (MLP) is given by

$$W_{t+1}^l = W_t^l - \eta_l \nabla_{W^l} \mathcal{L}\left(f\left(\xi_t; W_t + \boldsymbol{\varepsilon}_t\right), y_t\right), \quad \text{with} \quad \boldsymbol{\varepsilon}_t^l = \rho \cdot \frac{\nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t)}{\|\nabla_{\mathbf{W}} \mathcal{L}(f(\xi_t; W_t), y_t)\|_F}, \tag{SAM}$$

where $\mathbf{W} = [W^1, \dots, W^{L+1}]$, t is the iteration count and $\boldsymbol{\varepsilon}_t^l$ denotes the perturbation in the l-th MLP layer with width $n \in \mathbb{N}$, and where we define an L-hidden layer MLP iteratively via

$$h^1(\xi) := W^1 \xi, \qquad x^l(\xi) := \phi(h^l(\xi)), \qquad h^{l+1}(\xi) := W^{l+1} x^l(\xi), \qquad f(\xi) := W^{L+1} x^L(\xi),$$

for inputs $\xi \in \mathbb{R}^{d_{\text{in}}}$ with trainable weight matrices $W^1 \in \mathbb{R}^{n \times d_{in}}$, $W^l \in \mathbb{R}^{n \times n}$ for $l \in [2, L]$, and $W^{L+1} \in \mathbb{R}^{d_{\text{out}} \times n}$. We call h^l preactivations, x^l activations, and $f(\xi)$ output function. Despite the inherent difficulty of non-convex, non-concave optimization, SAM is quite successful in practice.

On the other hand, the steadily growing scale of foundation models has sparked considerable interest in scaling laws of model size and dataset size (Kaplan et al., 2020; Zhai et al., 2022). To

*This work was conducted during Moritz', Jin's and Volkan's time at Amazon. Correspondence to: mo.haas@uni-tuebingen.de

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

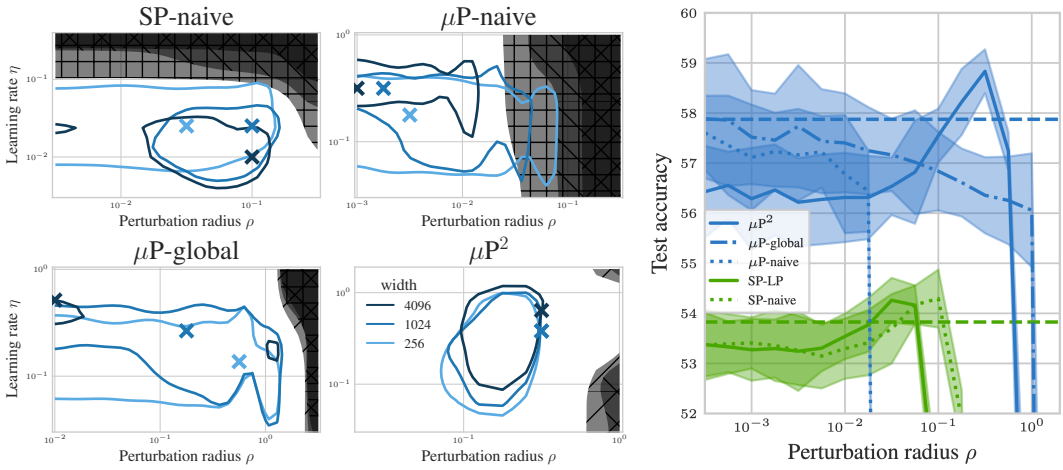


Figure 1: *Left* **(Only μP^2 transfers both η and ρ)**: Test accuracy as a function of learning rate η and perturbation radius ρ of a 3-layer MLP trained with SAM on CIFAR10 for various widths and in different parameterizations (see subplot title), averaged over 3 independent runs.'$\times$' denotes the optimum. Blue contours (the darker, the wider) denote the region within 1% of the optimal test accuracy smoothened with a Gaussian filter. Grey regions (the lighter, the wider) denote the unstable regime below 30% test accuracy. 'naive' denotes no perturbation scaling, 'global' denotes global perturbation scaling $\rho = \Theta(n^{-1/2})$. *Right* **(μP^2 achieves the best generalization performance)**: Same as left but sliced at the optimal learning rate of each parameterization for width 4096. Dashed horizontal lines denote the base optimizer SGD in SP (green) and in μP (blue), respectively. Average and 2σ-CI from 16 independent runs. SP-LP denotes SP with layerwise perturbation scaling.

rigorously understand learning dynamics under width scaling, Yang and Hu (2021) have recently provided general infinite-width theory for SGD, which has since been shown to be a good model for understanding the properties of large models (Vyas et al., 2024). Yang and Hu (2021) show that standard parameterizations (SP), including He or LeCun initialization (He et al., 2015; LeCun et al., 2002) with a global learning rate, do not learn features in the infinite-width limit.

Instead, a different scaling of layerwise initialization variances and learning rates, termed *Maximal Update Parameterization* (μP), is necessary to achieve feature learning in all layers in wide networks. A crucial practical benefit of μP is the transferability of the optimal learning rate across model scales (Yang et al., 2022). This can drastically reduce computational costs as it allows to tune hyperparameters on smaller representative models and then to train the large model only once.

Contributions. In this paper, we adopt a scaling perspective to understand SAM's learning dynamics. Using the Tensor Programs framework (Yang, 2019; Yang and Hu, 2021; Yang and Littwin, 2023), this work provides the first infinite-width theory for SAM with important practical consequences:

1. We show that training an MLP with the standard (SAM) update rule is equivalent to applying perturbations only in the last layer in the infinite-width limit, even if the perturbation radius is properly tuned. This holds for any width-dependent scaling of layerwise initialization variances and learning rates, including SP and μP.
2. We demonstrate that the optimal perturbation radius can shift significantly in μP (Figure 1).
3. We postulate that jointly transferring the optimal learning rate η and perturbation radius ρ requires width-independent feature learning and *effective perturbations in every layer* in the infinite-width limit. We show that this can be achieved with *layerwise scalings* of the perturbation radius, and provide a complete characterization of perturbation scaling parameterizations into four regimes: unstable, vanishing, nontrivial and effective perturbations.
4. We derive the *Maximal Update and Perturbation Parameterization* (μP^2) that achieves both feature learning and effective perturbations in all layers in the infinite-width limit. We empirically demonstrate that μP^2 is the first parameterization to achieve hyperparameter transfer in both learning rate η and perturbation radius ρ (Figure 1).
5. We provide a versatile (spectral) scaling condition ($*$) applicable to architectures such as ResNets and Vision Transformers (ViTs), and to various SAM variants like SAM-ON and Adaptive SAM (ASAM), and any SAM updates modeled in a Tensor Program.

2 Background and related work

We here provide a short summary of related work. A more detailed account is provided in Appendix B.

Sharpness Aware Minimization. SAM was motivated as an inductive bias towards flatter minima and it provably reduces properties of the Hessian that are related to sharpness in simpler settings (Bartlett et al., 2023; Wen et al., 2023; Monzio Compagnoni et al., 2023). However a full understanding of why SAM works so well remains elusive (Andriushchenko et al., 2023b; Wen et al., 2024). For example, applying SAM on only the normalization layers (SAM-ON) often improves generalization further despite increasing sharpness (Müller et al., 2024). A plethora of SAM variants have recently been proposed with the purpose of even stronger performance or reducing SAM's computational and memory complexity. We focus on two variants of Adaptive SAM (ASAM) (Kwon et al., 2021) which achieve the overall strongest results in Müller et al. (2024) (see Appendix F.4 for more details).

Tensor Programs. We build on the Tensor Programs framework (Yang, 2019; Yang and Hu, 2021; Yang and Littwin, 2023; Yang et al., 2022, 2023b), which covers many modern deep learning architectures, optimization algorithms and arbitrary abc-parameterizations. Each abc-parameterization is essentially defined by a layerwise scaling of initialization variance and learning rate as a function of network width. Beyond pure infinite-width limits, the simple $\frac{1}{\sqrt{L}}$-scaling allows depth-scaling in ResNets and unlocks hyperparameter transfer across depths (Hayou et al., 2021; Li et al., 2021; Bordelon et al., 2023; Yang et al., 2023b). Noci et al. (2022, 2024) provide infinite width and depth analyses for Transformers with the goal of preventing rank collapse.

Look-LayerSAM (Liu et al., 2022) already considers layerwise perturbation scaling with the goal of preserving good performance under large batch training. However, achieving μP^2 with Look-LayerSAM requires nontrivial layerwise learning rate and perturbation rescaling (see Appendix B).

3 SAM induces vanishing perturbations in wide neural networks

This section shows that under the standard (SAM) update rule, weight perturbations induced by SAM vanish in the infinite-width limit in every layer except the output layer. We later demonstrate that other SAM variants also selectively perturb other subsets of layers. For enhanced readability of some formulae, we use colors to distinguish four regimes of perturbation behaviour: Unstable, vanishing, nontrivial and effective perturbations.

While our theory covers any stable parameterization including He and LeCun initializations, for concreteness and for the clarity of exposition, we first present our results for MLPs under μP:

$$\text{initialize} \quad W^1 \sim \mathcal{N}(0, 1/d_{in}),\ W^l \in \mathbb{R}^{n\times n} \sim \mathcal{N}(0,1/n) \text{ for } l \in [2, L],\ W^{L+1} \sim \mathcal{N}(0, 1/n^2)$$
$$\text{with layerwise SGD learning rates} \quad \eta_1 = \eta n,\ \eta_l = \eta,\ \text{for } l \in [2, L],\ \eta_{L+1} = \eta n^{-1}.$$

By analyzing the infinite-width behaviour of the SAM update rule, we show that the training dynamics under standard (SAM) become unstable as the network width increases. This result is first stated informally below in Proposition 1 and then more formally in the next section.

Proposition 1 (Instability of standard SAM parameterization in wide neural networks). *Under μP with the standard (SAM) update rule and default perturbation given in (SAM), the output function becomes unbounded after the first update step in the infinite-width limit for any fixed, positive learning rate $\eta > 0$ and perturbation radius $\rho > 0$.*

Hence, to achieve stable optimization, it is necessary to introduce some width-dependent perturbation scaling ρn^{-d} for some suitable $d > 0$. To understand the layerwise scaling behaviour of SAM under this scaling, we define the notion of *vanishing perturbations*.

Vanishing perturbations. The weight perturbation $\boldsymbol{\varepsilon}^l$ perturbs the l-th layer's activations as

$$x^l + \tilde{\delta}x^l = \phi((W^l + \boldsymbol{\varepsilon}^l)(x^{l-1} + \tilde{\delta}x^{l-1})),$$

where $\tilde{\delta}x^l$ denotes the perturbation of the l-th layer's activations accumulated from the weight perturbations $\{\boldsymbol{\varepsilon}^{l'}\}_{l'\in[l]}$ in all previous layers. We say a layer l has *vanishing perturbations* if $\tilde{\delta}x^l \to 0$ as the width approaches infinity. This occurs if the weight perturbations in all previous layers are too small when measured in spectral norm, that is if $\|\boldsymbol{\varepsilon}^{l'}\|_*/\|W^{l'}\|_* \to 0$ for all $l' \in [l]$.

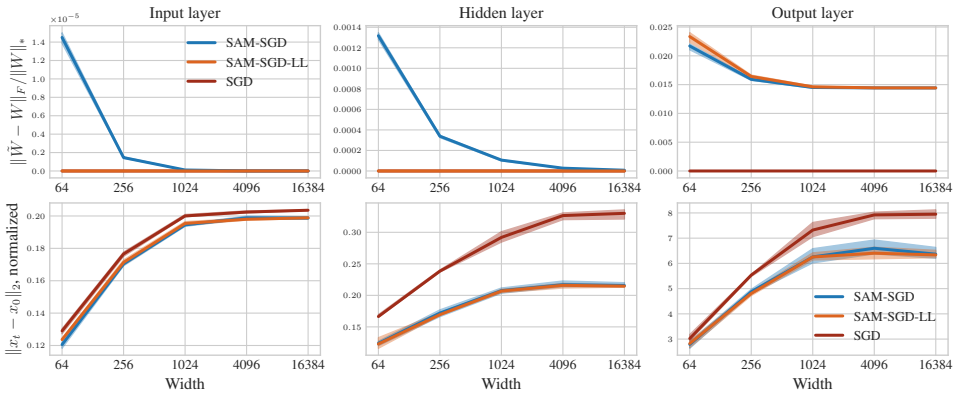


Figure 2: **((SAM) effectively only perturbs the last layer)** Layerwise weight perturbations (top) and normalized activation updates $\|\Delta x^l\|_2$ (bottom) for SAM, last-layer SAM and SGD as a baseline across widths after training a 3-layer MLP in μP with global perturbation scaling $\rho = \Theta(n^{-1/2})$ for 20 epochs on CIFAR10. Average and CI are computed from 4 independent runs. Perturbations are normalized by the weight spectral norm to measure their effect on the layer's output. Activation updates are normalized by $\sqrt{\dim(\Delta x^l)}$ to measure coordinatewise updates. We provide more neural network statistics in Appendix H.1.

Informally, Proposition 2 below shows that for every choice of a decay parameter $d > 0$, either the training dynamics of SAM are unstable or all the hidden layers of the network have vanishing perturbations in the limit. The formal results are stated in the next section.

Proposition 2 (Global perturbation scaling is unstable or induces vanishing perturbations). *Fix $\rho > 0$ and $t \in \mathbb{N}$. Let $\mathring{f}_t$ denote the infinite-width limit of the output function after training an MLP of width n with the SAM update rule (SAM) with perturbation radius ρn^{-d} for t steps. If $d < 1/2$, then output perturbations blow up, and $\mathring{f}_t$ is unstable. If $d > 1/2$, then the perturbations in all layers vanish and $\mathring{f}_t$ corresponds to the limit after t steps of SGD. If $d = 1/2$, then only the last layer is effectively perturbed, all other layers have vanishing perturbations.*

Figure 2 shows statistics of an MLP trained with (SAM) with global width-dependent scaling $\rho n^{-1/2}$ versus the same MLP trained with SAM where *only the last-layer weights are perturbed* and $\boldsymbol{\varepsilon}^l = 0$ for all $l \in [L]$. As predicted by Proposition 2, both training algorithms produce equivalent training dynamics, already at moderate width, and last-layer perturbations are scaled correctly.

Remark 3 (Practical implications). According to Proposition 2, for sufficiently wide models, any performance gains from standard SAM are primarily due to applying the SAM update rule to the last layer even with a properly tuned perturbation radius. This implies that, when applying the standard SAM update rule (SAM), one can remove the inner backward pass beyond the last layer and nearly recover the computational cost of SGD. However, it may be undesirable for optimal generalization if only the last layer is perturbed (Figure 1). ◀

Layerwise perturbation scaling. In the next section, we show that correcting the (SAM) update rule to achieve *effective perturbations* in every single layer requires introducing additional hyperparameters — layerwise width-dependent scaling of the perturbation radius. This is similar in spirit to μP which corrects standard parameterization by introducing layerwise scaling of the learning rates. We postulate that achieving width-independent scaling of both updates and perturbations is a necessary condition for hyperparameter transfer under SAM. We also lay all theoretical foundations and derive the stable parameterization, we call *maximal update and perturbation parameterization* (μP^2) that achieves both **feature learning and effective perturbations in all layers in the infinite-width limit**.

Figure 1 shows that μP^2 is indeed the first parameterization to achieve hyperparameter transfer in the optimal joint choice of (η, ρ), while also achieving the best generalization performance.

General perturbation scaling condition. For intuitive understanding and a generalization to other perturbation rules, a simple condition for achieving effective perturbations in any layer follows from our results: **in every layer, perturbations should scale like updates in μP**.

The reason is that both updates and perturbations are gradient-based $\nabla_{W^l}\mathcal{L} = \nabla_{h^l}\mathcal{L} \cdot (x^{l-1})^\top$, and thus low-rank and correlated with the incoming activations x^{l-1}. Therefore updates and perturbations introduce the same LLN-like scaling factors, and require the same layerwise scaling corrections. Like Yang et al. (2023a), we can rephrase this condition in terms of weight spectral norms to: For a weight matrix $W_t^l \in \mathbb{R}^{\texttt{fan\_out}\times\texttt{fan\_in}}$, its update δW_t^l and its perturbation $\boldsymbol{\varepsilon}_t^l$, it should hold at all times t that

$$\|\boldsymbol{\varepsilon}_t^l\|_* = \Theta\left(\|\delta W_t^l\|_*\right) = \Theta\left(\|W_t^l\|_*\right) = \Theta\left(\sqrt{\texttt{fan\_out}/\texttt{fan\_in}}\right). \tag{$*$}$$

with big-O notation that only tracks dependence on network width (Definition C.1). We discuss the spectral perspective in more detail in Appendix F.7.

4 Sharpness Aware Minimization in the infinite-width limit

4.1 Characterization of layerwise perturbation scaling: Unstable, vanishing, nontrivial and effective perturbations

To systematically and rigorously understand the width-scaling behaviour of neural networks trained under the SAM update rule, we propose a new class of parameterizations, which we refer to as *bcd-parameterizations*. Motivated by the analysis in Section 3, the class of bcd-parameterizations naturally extends abc-parameterizations (Yang and Hu, 2021) by including layerwise scaling of the perturbation radius. By setting all weight multiplier exponents $a_l = 0$, we do not need to modify the MLP architecture and recover representatives of each abc-parameterization that capture their essence and condense all equations: Ignoring numerical considerations (Blake et al., 2024), each abc-parameterization is essentially a layerwise initialization and learning rate scaling. The effects of weight multipliers on SAM are more nuanced than for SGD or Adam (see Remark 12 and Appendix F.6).

To study the infinite-width behaviour of networks trained with SAM in any bcd-parameterization, we utilize the theoretical framework of NE⊗OR⊤programs (Yang et al., 2023b). We write the two forward and backward passes for each SAM update (ascent/perturbation step then descent/update step) using the NE⊗OR⊤computation rules and rigorously track all relevant scalings as provided by the NE⊗OR⊤master theorem. All proofs are provided in Appendix E. The full formal result statements can be found in Appendix D. Further theoretical considerations and generalizations around perturbation scaling are provided in Appendix F.

Assumptions. For clarity of exposition, we present our main results for MLPs. Their extension to other architectures is discussed in Appendix F.5. For all of the results in this section, we assume that the used activation function is either `tanh` or σ-`gelu` for $\sigma > 0$ sufficiently small. For small enough $\sigma > 0$, σ-`gelu` (Definition C.9) approximates ReLU arbitrarily well. We also assume constant training time as width $n \to \infty$. We assume batch size 1 for clarity, but our results can be extended without further complications to arbitrary fixed batch size as well as differing fixed batch sizes for the ascent/perturbation and the descent/update step, as sometimes used for SAM (Foret et al., 2021). Considering small perturbation batch size is practical, as it has been observed to enhance SAM's generalization properties (Andriushchenko and Flammarion, 2022).

Definition 4 (bcd**-parametrization).** A *bcd-parametrization* $\{b_l\}_{l\in[L+1]} \cup \{c_l\}_{l\in[L+1]} \cup \{d_l\}_{l\in[L+1]} \cup \{d\}$ defines the training of an MLP with SAM in the following way:

(a) Initialize weights iid as $W_{ij}^l \sim \mathcal{N}(0, n^{-2b_l})$.
(b) Train the weights using the SAM update rule with layerwise learning rates,

$$W_{t+1}^l = W_t^l - \eta n^{-c_l} \nabla_{W^l}\mathcal{L}\left(f\left(\xi_t; W_t + \boldsymbol{\varepsilon}_t\right), y_t\right),$$

with the scaled perturbation $\boldsymbol{\varepsilon}_t$ via layerwise perturbation radii,

$$\boldsymbol{\varepsilon}_t := \rho n^{-d} \frac{v_t}{\|v_t\|}, \quad \text{with} \quad v_t = (v_t^1, \ldots, v_t^{L+1}), \quad v_t^l := n^{-d_l} \cdot \nabla_{W^l}\mathcal{L}(f(\xi_t; W_t), y_t), \tag{LP}$$

W.l.o.g. we set $\|v_t\| = \Theta(1)$, which prevents nontrivial width-dependence from the denominator. This imposes the constraints: $d_1 \geq 1/2 - \min(b_{L+1}, c_{L+1})$, $d_l \geq 1 - \min(b_{L+1}, c_{L+1})$ for $l \in [2, L]$, and $d_{L+1} \geq 1/2$, with at least one equality required to hold (see Appendix E.1.3). The normalization $v_t/\|v_t\|$ removes one degree of freedom from $\{d_l\}_{l\in[L+1]}$ via the equivalence $\{d_l'\}_{l\in[L+1]} \cong \{d_l\}_{l\in[L+1]}$ iff there exists a $C \in \mathbb{R}$ such that $d_l' = d_l + C$ for all $l \in [L+1]$. ◀

Stability. To ensure that the training dynamics of SAM are well-behaved with scale, we require *bcd*-parameterizations to satisfy conditions of stability. Perturbed weights $\tilde{W}^l = W^l + \boldsymbol{\varepsilon}^l$ induce perturbed activations $x^l + \tilde{\delta}x^l$ and a perturbed output function $\tilde{f}_t(\xi) := f_{\tilde{W}_t}(\xi)$. We call a *bcd*-parameterization *stable* (Definition C.3) if the hidden activations have width-independent scaling $\Theta(1)$ at initialization and during training, and neither the updates nor the perturbations $\tilde{\delta}x^l$ of the activations or output logits $\tilde{f}_t - f_t$ blow up at any point in training.

For stating the conditions that characterize the class of stable *bcd*-parameterizations, we define the *maximal feature perturbation scaling* $\tilde{r}$ of a *bcd*-parameterization as

$$\tilde{r} := \min(b_{L+1}, c_{L+1}) + d + \min_{l=1}^{L}(d_l - \mathbb{I}(l \neq 1)).$$

Similar to the maximal feature update scaling r from Yang and Hu (2021), $\tilde{r}$ describes how much the last hidden-layer activations are perturbed as a function of width, $x^L + \tilde{\delta}x^L = \Theta(n^{-\tilde{r}})$. Hidden-layer activation perturbations do not explode with width if and only if $\tilde{r} \geq 0$. The output perturbations not to blow up if and only if $d + d_{L+1} \geq 1$ and $b_{L+1} + \tilde{r} \geq 1$. In particular, this implies that any stable *bc*-parameterization together with naive perturbation scaling $d_l = d = 0$ for all $l \in [L+1]$ is *unstable due to blowup in the last layer*. We formally state the stability characterization in Theorem D.2. Ideally, we will later require width-independent perturbation scaling which is attained iff $\tilde{r} = 0$.

Effective SGD dynamics. Within the class of stable parameterizations, there are parameterizations in which perturbations in the output vanish in the infinite-width limit at any point during training. In other words, SAM training dynamics collapses to SGD dynamics with scale. We are mostly interested in the opposing class of parameterizations with non-vanishing perturbations. We characterize this class in Theorem 6 and refer to them as *perturbation nontrivial* (Definition 5).

Definition 5 (**Perturbation nontriviality**)**.** We say that a stable *bcd*-parametrization is *perturbation nontrivial* if there exists a training routine, $t \in \mathbb{N}_0$ and $\xi \in \mathbb{R}^{d_{\text{in}}}$ such that $\tilde{\delta}f_t(\xi) := f_{\tilde{W}_t}(\xi) - f_{W_t}(\xi) = \Omega(1)$. Otherwise, the *bcd*-parametrization is *perturbation trivial*. ◀

Theorem 6 (**Perturbation nontriviality characterization**)**.** *A stable bcd-parametrization is perturbation nontrivial if and only if* $d + d_{L+1} = 1$ *or* $\min(b_{L+1}, c_{L+1}) + \tilde{r} = 1$.

For the class of stable and perturbation nontrivial *bcd*-parameterizations, SAM learning is both stable and deviates from SGD dynamics. A natural question to ask here is: what should be the ideal SAM behaviour in the infinite-width limit? To address this question, we make the following crucial distinction between *non-vanishing* and *effective perturbations*.

Non-vanishing versus effective perturbations. Recall that the weight perturbation $\boldsymbol{\varepsilon}^l$ perturbs the l-th layer's activations as

$$x^l + \tilde{\delta}x^l = \phi((W^l + \boldsymbol{\varepsilon}^l)(x^{l-1} + \tilde{\delta}x^{l-1})),$$

where $\tilde{\delta}x^l$ denotes the perturbation of the l-th layer's activations accumulated from the weight perturbations $\{\boldsymbol{\varepsilon}^{l'}\}_{l' \in [l]}$ in all previous layers. Therefore, perturbations $\tilde{\delta}x^l$ can stem both from weight perturbations $\boldsymbol{\varepsilon}^{l'}$ in a previous layer $l' < l$ and/or from weight perturbations $\boldsymbol{\varepsilon}^l$ in the current layer l. Intuitively, if we perturb a layer, we want this to affect the next layer's activations and thereby have a nontrivial effect on the output function. Otherwise one can simply set the layer's perturbations to 0 by design and not change the learning algorithm in the infinite-width limit. This motivates the definition of *effective perturbations*, which demands the weight perturbations of the current layer to contribute non-vanishingly. From the weight perspective ($*$), effective l-th layer perturbations are achieved if and only if weight perturbations scale like the weights in spectral norm, $\|\boldsymbol{\varepsilon}^l\|_* / \|W^l\|_* = \Theta(1)$. Without an effective perturbation $\boldsymbol{\varepsilon}^l$ of the l-th layer, this layer does not inherit SAM's inductive bias towards low spectral norm of the Hessian or enhanced sparsity and does not improve generalization performance. We provide empirical evidence for these claims in Appendix H.2. Therefore a distinction between *non-vanishing* and *effective perturbations* is crucial.

Definition 7 (**Non-vanishing perturbations**)**.** For $l \in [L]$, we say that a stable parameterization has *non-vanishing perturbations in the l-th layer* if there exists a $t \in \mathbb{N}$ such that $\tilde{\delta}x_t^l = \Omega(1)$. ◀

Definition 8 (**Effective perturbations**)**.** For $l \in [L+1]$, we say that a stable parameterization *effectively perturbs the l-th layer* if there exists a $t \in \mathbb{N}$ such that $\boldsymbol{\varepsilon}_t^l(x_t^{l-1} + \tilde{\delta}x_t^{l-1}) = \Theta(1)$, where $x_t^0 + \tilde{\delta}x_t^0 = \xi_t$. ◀

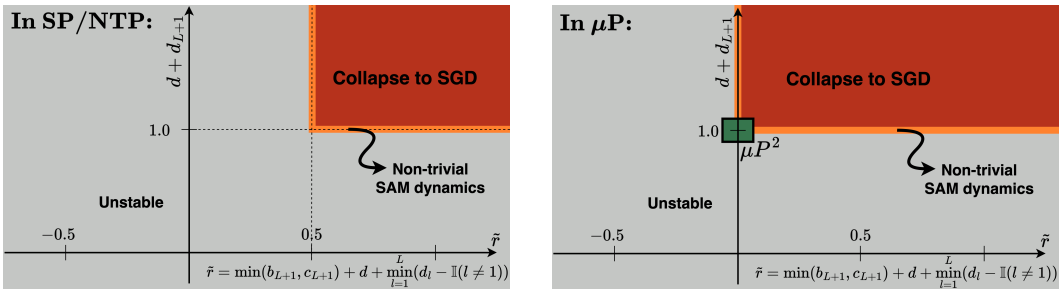


Figure 3: **(Perturbation phase characterization of bcd-parameterizations)** Given a choice of layerwise initialization and learning rate scalings $\{b_l, c_l\}_{l\in[L+1]}$, the maximal feature perturbation scaling $\tilde{r}$ and the last-layer perturbation scaling $d + d_{L+1}$ completely determine whether a *bcd*-parameterization is unstable, has effective SGD dynamics, effective perturbations in some but not all layers or effective perturbations in all layers. In SP or NTP (left), there does not exist a choice of perturbation scalings that achieves effective perturbations in all layers, whereas in μP (right), there is a unique choice as provided in Theorem 11.

Theorem 9 provides a characterization of stable *bcd*-parameterizations with vanishing perturbations in any given layer.

Theorem 9 (**Vanishing perturbation characterization**). *For any $l_0 \in [L]$, the following statements are equivalent:*

(a) A stable bcd-parametrization has vanishing perturbations in layer l_0.
(b) A stable bcd-parametrization has vanishing perturbations in layer l for all $1 \leq l \leq l_0$.
(c) $\tilde{r}_{l_0} := \min(b_{L+1}, c_{L+1}) + d + \min_{m=1}^{l_0}(d_m - \mathbb{I}(m \neq 1)) > 0$.

It follows from Theorem 9 that any stable *bcd*-parameterization that performs updates in the original gradient direction (i.e., $d_l = C$ for all $l \in [L+1]$ for some $C \in \mathbb{R}$) has vanishing perturbations in all input and hidden layers $l \in [L]$, and the last layer $l = L+1$ is effectively perturbed if and only if $d = 1/2$. This covers the case of both standard and maximal update parameterizations with global scaling of the perturbation radius discussed in Section 3. Negating the conditions of Theorem 9 implies that a stable *bcd*-parameterization has non-vanishing perturbations in layer l_0 if and only if $\tilde{r}_{l_0} = 0$. Achieving *effective perturbations* is a stronger requirement for which Theorem 10 provides the necessary and sufficient conditions.

Theorem 10 (**Effective perturbation characterization**). *For $l \in [L]$, a stable bcd-parametrization effectively perturbs the l-th layer if and only if $\min(b_{L+1}, c_{L+1}) + d + d_l - \mathbb{I}(l \neq 1) = 0$.*

A stable bcd-parametrization effectively perturbs the last layer if and only if $d + d_{L+1} = 1$.

4.2 Maximal Update and Perturbation Parameterization (μP^2)

We postulate that just as the optimal learning rate transfers across widths under μP for SGD and Adam due to non-vanishing width-independent feature evolution in all layers, the optimal learning rate and perturbation radius may be jointly transferable across widths if additionally the weight perturbations induce width-independent perturbations of the activations in all layers. Here, we show that, for every stable initialization and learning rate scaling with $b_{L+1} \geq 1$, there exists a unique stable layerwise perturbation scaling which effectively perturbs every single layer. We term this layerwise perturbation scaling $\{d_l\}_{l\in[L+1]} \cup \{d\}$ the Maximal Perturbation Parameterization (MPP). This concludes the phase characterization of perturbation scaling behaviours (Figure 3).

Theorem 11 (**Maximal Perturbation Parameterization (MPP)**). *Consider any stable bcd-parametrization $\{b_l\}_{l\in[L+1]} \cup \{c_l\}_{l\in[L+1]} \cup \{d_l\}_{l\in[L+1]} \cup \{d\}$. If $b_{L+1} < 1$, then there does not exist a stable choice of $\{d_l\}_{l\in[L+1]} \cup \{d\}$ that achieves effective perturbations before the last layer. If $b_{L+1} \geq 1$, then up to the equivalence $d'_l = d_l + C$, $C \in \mathbb{R}$, $\forall l \in [L+1]$, the unique stable choice $\{d_l\}_{l\in[L+1]} \cup \{d\}$ that effectively perturbs all layers $l \in [L+1]$ is given by*

$$d = -1/2, \qquad d_l = \begin{cases} 1/2 - \min(b_{L+1}, c_{L+1}) & l = 1, \\ 3/2 - \min(b_{L+1}, c_{L+1}) & l \in [2, L], \\ 3/2 & l = L+1. \end{cases} \tag{1}$$

	Perturbed under global scaling?			For effective perturbations with μP^2:			
	Input, biases, norm.	Other hidden layers	Output layer	Global ρ	Input-like	Hidden-like	Output-like
SAM	✗	✗	✓	$n^{1/2}$	$n^{1/2}$	$n^{-1/2}$	$n^{-3/2}$
Layer. ASAM	✗	✓	✗	1	1	n^{-1}	1
Elem. ASAM	✓	✓	✓	$n^{1/2}$	1	1	1
SAM-ON	✓	-	-	$n^{1/2}$	1	-	-

Table 1: **(Layerwise perturbation scaling for effective perturbations in μP)** Without layerwise perturbation scaling (*left*), each SAM variant perturbs a different subset of layers at large width $n \to \infty$, but we provide the unique layerwise perturbation rescaling μP^2 (*right*) that achieves effective perturbations in all layers. This parameterization transfers both the optimal η and ρ across widths.

Maximal Update and Perturbation Parameterization μP^2. To achieve feature learning in every layer and hyperparameter transfer in the learning rate, μP is the unique[2] choice of layerwise initialization variance and learning rate scalings $\{b_l, c_l\}_{l\in[L+1]}$ (Yang and Hu, 2021). Together with Theorem 11, this shows that there exists a *unique*[2] bcd-parameterization that achieves both feature learning and effective perturbations in all layers, we call *maximal update and perturbation parametrization*, μP^2 for short. Now that we have found a parameterization that achieves width-independent scaling of both activation updates and activation perturbations, μP^2 fulfills essential necessary conditions for hyperparameter transfer to occur in both η and ρ.

Remark 12 (Achieving μP^2 with weight multipliers). Appendix F.6 covers the extension of our results to nontrivial weight multipliers. We show that, for each choice of weight multipliers $\{a_l\}_{l\in[L+1]}$, there is a unique[2] choice of bcd-hyperparameters that achieves effective perturbations in all layers. But unlike for SGD or Adam, these parameterizations lead to slightly different training algorithms, because differing subsets of layers contribute non-vanishingly to the joint gradient normalization term $\|\nabla_{\mathbf{W}}\mathcal{L}\|_F$ in (SAM). The term $\|\nabla_{\mathbf{W}}\mathcal{L}\|_F$ couples all layers so that there do not exist layerwise but only layer-coupled equivalence classes for (SAM). Most importantly, **instead of adapting (SAM), we can adapt the architecture with the weight multipliers $n^{-a_l} \cdot W^l$ with**

$$a_l = -1/2 \cdot \mathbb{I}(l = 1) + 1/2 \cdot \mathbb{I}(l = L + 1) \qquad (a\text{-}\mu P^2)$$

to achieve effective perturbations in all layers with naive perturbation scaling such that all layers contribute non-vanishingly to the joint gradient norm (Appendix F.6). Unfortunately, this choice of weight multipliers is not compatible unit scaling considerations (Blake et al., 2024). ◀

Alternative perturbation scaling definitions. Scaling equivalent to (a-μP^2) can be achieved without multipliers by scaling the numerator and denominator terms in (LP) independently, and choosing to scale all denominator terms to be width-independent (see perturbation rule (DP) and Appendix F.7 for more details). The ablations in Appendix H.4 suggest that this has a negligible effect on the optimal generalization performance of μP^2, but can be more stable given suboptimal hyperparameters. Gradient normalization in each layer separately is uncommon and performs slightly worse (Appendix H.5). Appendix F.2 discusses further considerations that led to Definition 4.

Trivial, lazy, and feature learning regimes. A small last-layer initialization variance $b_{L+1} \geq 1$ is required for stable feature learning. Theorem 11 shows that $b_{L+1} \geq 1$ is also required for effective hidden-layer perturbations. Beyond this condition, the choice of $\{b_l\}$ and $\{c_l\}$ is decoupled from that of perturbation scalings $\{d_l\} \cup \{d\}$ for stable bcd-parameterizations, because the scale of the activations of a layer l is entirely determined by the scale of initialization b_l and learning rates c_l, given stability. Consequently, whether a parameterization is *trivial*, in the *lazy regime,* or in the *feature learning regime* is independent of the choice of d_l's provided that all stability constraints are met. A complete characterization of these regimes for the class of bc-parameterizations has been provided in Yang and Hu (2021) and remains unchanged for the class of stable bcd-parameterizations. For completeness, formal definitions and the corresponding results are stated in Appendices C and D.

4.3 Generalizations to other architectures and SAM variants

Generalization to other architectures. Our results can be extended to other common layer types, that are representable as a NE⊗OR⊤program, including all ResNet and Transformer components (Appendix F.5). All considered layer types behave like input, hidden or output layers. Most importantly, normalization layer weights and biases scale like input layer weights to the input 1.

Generalization to other SAM variants. We would like to find the correct layerwise perturbation scaling without writing out the NE⊗OR⊤program for every perturbation rule individually. Formally justified by our proof in Appendix E, we rephrase our equivalent spectral scaling condition ($*$) from Section 3 to: maximal stable perturbations are achieved in μP if and only if $\boldsymbol{\varepsilon}^l = \Theta(\delta W^l)$. This condition holds as soon as weight updates δW^l and perturbations $\boldsymbol{\varepsilon}^l$ are both correlated with the incoming activations x^{l-1}, for example if both are gradient-based. Table 1 summarizes the application of this condition to two ASAM variants that perform well empirically but cannot be written as a NE⊗OR⊤program. Additional details are provided in Appendix F.4. We demonstrate that these scalings perform well and transfer hyperparameters in the next section. Note that for hidden layers in μP^2, it holds that $\boldsymbol{\varepsilon}^l = \Theta(n^{-1})$ but $W^l = \Theta(n^{-1/2})$ entrywise, due to large initialization, showing that it is crucial to compare perturbations to updates or to measure weight scalings in spectral norm.

5 The maximal update and perturbation parameterization μP^2 achieves hyperparameter transfer and improved generalization

In this section, we provide experimental results showing that μP^2 is the first parameterization to achieve hyperparameter transfer in both η and ρ across architectures, and that μP^2 also improves generalization – even after multi-epoch training to convergence. We train MLPs and ResNets (He et al., 2016) on CIFAR10 (Krizhevsky et al., 2009) and Vision Transformers (ViTs) (Dosovitskiy et al., 2021) on Imagenet1K (Deng et al., 2009). While we directly implement bcd-parameterizations for MLPs and ResNets in PyTorch (Paszke et al., 2019), we use the `mup`-package (Yang et al., 2022) as a basis for ViT experiments. Pseudocode and a spectral derivation of our μP^2-implementation for ViTs, which is equivalent to (a-μP^2), are provided in Appendix F.7. All experimental details are stated in Appendix G and all supplemental experiments can be found in Appendix H.

Comparing candidate parameterizations in MLPs. Figure 1 shows test accuracy as a function of learning rate and perturbation radius for MLPs of varying width. While previous μP-literature mostly focuses on the more immediate transfer in training error, for SAM it is crucial to consider optimality in test error as the perturbation radius acts as a regularizer, so that optimality in test error typically coincides with suboptimal training error. In SP, the regime of stable learning rates shrinks with width. In μP without perturbation scaling, the regime of stable perturbation radii shrinks. In μP with global perturbation scaling, the regime of stable ρ remains invariant under width scaling, but there is no significant improvement of SAM beyond SGD, so that the optimal perturbation radius fluctuates within its stable regime due to noise. Only μP^2 consistently achieves hyperparameter transfer across widths, and achieves significant improvement over its base optimizer SGD in μP at scale. The full hyperparameter landscapes are provided in Appendix H.3.

ρ-transfer in ViTs. Figure 4 shows that the optimal perturbation radius transfers for ViT-S/16 on Imagenet1K trained with SAM in μP^2. While Andriushchenko and Flammarion (2022, Appendix E.3) observe diminishing benefits of SAM at large widths in SP, here the improvements beyond the base optimizer AdamW in μP are particularly large.

ρ-transfer for SAM variants in μP^2. Figure 6 shows that training a ResNet-18 in μP^2 achieves hyperparameter transfer in ρ for all considered SAM variants with varying width. μP with global perturbation scaling (μP-global) has a width-invariant stability threshold in ρ and the optimal ρ clearly shifts toward that threshold. It would be interesting to see whether this shift continues with larger width and leads to suboptimal performance of μP-global in wider

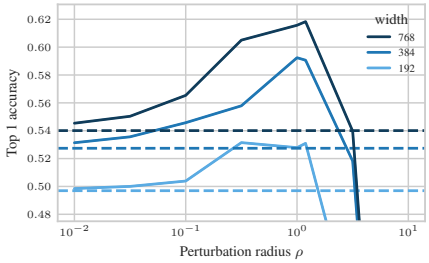


Figure 4: **(ρ-transfer in ViTs)** Training a ViT with SAM in μP^2 on ImageNet1K from scratch for 100 epochs yields ρ-transfer and large improvements over AdamW in μP (dashed lines).

[2]Strictly speaking, unique up to smaller last-layer initialization $b_{L+1} \geq 1$.

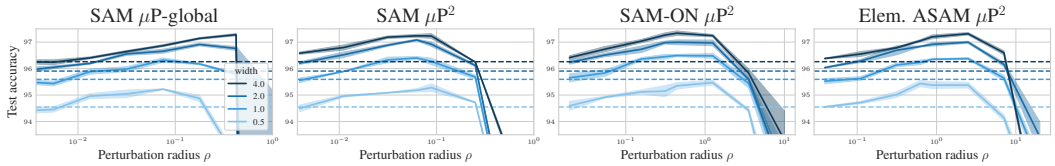


Figure 6: **(ρ-transfer of ASAM variants in μP^2)** Test error as a function of perturbation radius ρ after 200 epochs of training a ResNet-18 in μP^2 on CIFAR10 with various SAM variants (see subplot title). CI over 2 independent runs. Darker lines correspond to larger width multipliers. Other hyperparameters are tuned at base width multiplier 0.5. μP^2 achieves transfer in ρ and large improvements over the base optimizer (dashed lines) SGD in μP with momentum and weight decay.

	SAM global	SAM μP^2	SAM-ON μP^2	Elem. ASAM μP^2
SP	$97.00_{\pm 0.03}(+0.96)$	$97.00_{\pm 0.03}(+0.96)$	$\mathbf{97.29}_{\pm 0.06}(\mathbf{+1.26})$	$97.15_{\pm 0.01}(+1.11)$
μP	$\mathbf{97.19}_{\pm 0.05}(\mathbf{+0.93})$	$\mathbf{97.23}_{\pm 0.08}(\mathbf{+0.97})$	$\mathbf{97.34}_{\pm 0.08}(\mathbf{+1.08})$	$\mathbf{97.32}_{\pm 0.05}(\mathbf{+1.06})$

Table 2: **(Performance of μP^2)** Average test accuracy$_{\pm \text{standard deviation across 4 runs}}$ (+ improvement of SAM over SGD) for ResNet-18 with width multiplier 4 on CIFAR10 using SGD as a base optimizer. In bold, all parameterizations within a 2σ-CI from the best-performing variant SAM-ON in μP^2.

ResNets. Table 2 shows that all SAM variants perform similarly well in μP^2, some slightly outperforming the best-performing variant SAM-ON in SP. This suggests that for ResNets, even with a proper layerwise balance, normalization layer perturbations may suffice, and performance differences in SP are primarily caused by varying degrees to which the normalization layers are perturbed.

Without providing an explanation, Müller et al. (2024, Section 5.3) observe that only SAM-ON and elementwise ASAM sufficiently perturb normalization layers in SP. Table 1 (*left*) explains these observations by showing that only these two SAM variants effectively perturb normalization layers under global perturbation scaling. Table 1 (*right*) also provides full control over which layers to perturb. For transferring the optimal ρ with SAM-ON in μP, our theory predicts the global scaling $\rho = \Theta(n^{1/2})$ which is confirmed by our empirical observations (Figure 6). However, properly understanding the role of normalization layer perturbations remains an important question for future work. Note that we report results after fine-tuning all hyperparameters. The performance gain of μP^2 over SP and μP-global is likely much higher in larger models, for which fine-tuning is infeasible and the lack of feature learning and effective perturbations is more pronounced. Even under optimal HPs, μP^2 appears to stabilize SAM's training dynamics compared to SP (Figure 5).

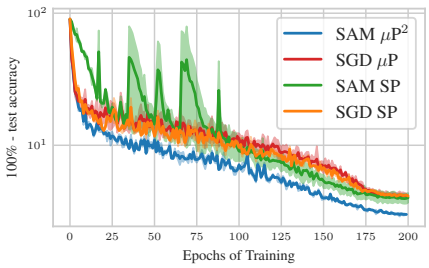


Figure 5: **(Stable training dynamics)** SAM in μP^2 stabilizes training dynamics for a ResNet-18 with width multiplier 2.

6 Future work

This study may serve as an inspiration of how scaling theory can be used to understand and improve training procedures in minimax optimization and beyond. To reach a fully practical theory of deep learning, it will be necessary to take data distributions and training dynamics into account in more detail than it is possible with current Tensor Program theory (Everett et al., 2024). Existing Tensor Program theory assumes constant batch size and training time, and does not make statements about generalization. For example, we observe that ResNets in SP can sometimes display HP transfer in η and ρ after training to convergence (Appendix H.3.2). This contradicts the infinite-width theory from Yang and Hu (2021) which predicts output blowup under large learning rates, and it shows that the exact conditions which enable hyperparameter transfer in practice are not fully understood. It also remains unclear how to optimally adapt (SAM) when increasing network depth. We plan to address some of these questions in upcoming work.

References

Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning (ICML)*, 2022. Cited on page 5, 9, 17, 37, 38, 48.

Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. Sharpness-aware minimization leads to low-rank features. *arXiv:2305.16292*, 2023a. Cited on page 38, 53.

Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023b. Cited on page 3, 17.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022. Cited on page 17.

Peter L. Bartlett, Philip M. Long, and Olivier Bousquet. The dynamics of sharpness-aware minimization: Bouncing across ravines and drifting towards wide minima. *Journal of Machine Learning Research (JMLR)*, 24(316):1–36, 2023. Cited on page 3, 17.

Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020. Cited on page 17.

Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. Chinchilla scaling: A replication attempt. *arXiv:2404.10102*, 2024. Cited on page 17.

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. *arXiv:2407.17465*, 2024. Cited on page 5, 8, 45, 49.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. *arXiv:2309.16620*, 2023. Cited on page 3, 16.

Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. *arXiv:2106.01548*, 2021. Cited on page 1, 17.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations (ICLR)*, 2020. Cited on page 17.

Yan Dai, Kwangjun Ahn, and Suvrit Sra. The crucial role of normalization in sharpness-aware minimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024. Cited on page 17, 39.

Yann N Dauphin, Atish Agarwala, and Hossein Mobahi. Neglected hessian component explains mysteries in sharpness regularization. *arXiv:2401.10809*, 2024. Cited on page 17.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Cited on page 9, 51, 70.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. Cited on page 17.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. Cited on page 9.

Katie E Everett, Lechao Xiao, Mitchell Wortsman, Alexander A Alemi, Roman Novak, Peter J Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. Cited on page 10, 49.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations (ICLR)*, 2021. Cited on page 1, 5, 17, 37, 48.

Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. Cited on page 16.

Soufiane Hayou and Greg Yang. Width and depth limits commute in residual networks. In *International Conference on Machine Learning (ICML)*, 2023. Cited on page 17.

Soufiane Hayou, Eugenio Clerico, Bobby He, George Deligiannidis, Arnaud Doucet, and Judith Rousseau. Stable resnet. In *International Conference on Artificial Intelligence and Statistics*, 2021. Cited on page 3, 16.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE international conference on computer vision (ICCV)*, 2015. Cited on page 2, 51.

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 (CVPR)*, 2016. Cited on page 9.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Comput.*, 9(1):1–42, 1997. Cited on page 17.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv:2203.15556*, 2022. Cited on page 17.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. Cited on page 16.

Yiding Jiang*, Behnam Neyshabur*, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations (ICLR)*, 2020. Cited on page 17.

Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. When do flat minima optimizers work? *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022. Cited on page 1, 17.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020. Cited on page 1, 17.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. Cited on page 9, 51, 70.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning (ICML)*, 2021. Cited on page 1, 3, 17, 37, 39, 48.

Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, 2002. Cited on page 2.

Mufan Li, Mihai Nica, and Dan Roy. The future is log-gaussian: Resnets and their infinite-depth-and-width limit at initialization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. Cited on page 3, 16.

Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 3, 17.

Philip M. Long and Peter L. Bartlett. Sharpness-aware minimization and the edge of stability. *arXiv:2309.12488*, 2023. Cited on page 17.

Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018. Cited on page 16.

Enea Monzio Compagnoni, Luca Biggio, Antonio Orvieto, Frank Norbert Proske, Hans Kersting, and Aurelien Lucchi. An SDE for modeling SAM: Theory and insights. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. Cited on page 3, 17, 39.

Maximilian Müller, Tiffany Vlaar, David Rolnick, and Matthias Hein. Normalization layers are all that sharpness-aware minimization needs. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024. Cited on page 1, 3, 10, 17, 35, 37, 39, 41, 48, 51, 52.

Radford M. Neal. *Priors for Infinite Networks*. Springer New York, 1996. Cited on page 16.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems (NeurIPS)*, 35, 2022. Cited on page 3, 16.

Lorenzo Noci, Chuning Li, Mufan Li, Bobby He, Thomas Hofmann, Chris J Maddison, and Dan Roy. The shaped transformer: Attention models in the infinite depth-and-width limit. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024. Cited on page 3, 16.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. Cited on page 9, 70.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016. Cited on page 16.

David Samuel. (Adaptive) SAM Optimizer (PyTorch). `https://github.com/davda54/sam`, 2022. Cited on page 37, 48, 49, 70.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv:1611.01232*, 2016. Cited on page 16.

Sungbin Shin, Dongyeop Lee, Maksym Andriushchenko, and Namhoon Lee. The effects of overparameterization on sharpness-aware minimization: An empirical and theoretical analysis. *arXiv:2311.17539*, 2023. Cited on page 17.

Nikhil Vyas, Alexander Atanasov, Blake Bordelon, Depen Morwani, Sabarish Sainathan, and Cengiz Pehlevan. Feature-learning networks are consistent across widths at realistic scales. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024. Cited on page 2, 16, 60.

Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How sharpness-aware minimization minimizes sharpness? In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. Cited on page 3, 17.

Kaiyue Wen, Zhiyuan Li, and Tengyu Ma. Sharpness minimization algorithms do not only minimize sharpness to achieve better generalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024. Cited on page 3, 17.

Jonathan Wenger, Felix Dangel, and Agustinus Kristiadi. On the disconnect between theory and practice of overparametrized neural networks. *arXiv:2310.00137*, 2023. Cited on page 16.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv:2006.03677*, 2020. Cited on page 51.

Lechao Xiao, Jeffrey Pennington, and Samuel Schoenholz. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning (ICML)*, 2020. Cited on page 16.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. Cited on page 2, 3, 16, 42.

Greg Yang. Tensor programs iii: Neural matrix laws. *arXiv:2009.10685*, 2021. Cited on page 24.

Greg Yang and Edward J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning (ICML)*, 2021. Cited on page 2, 3, 5, 6, 8, 10, 16, 18, 20, 21, 29, 31, 32, 33, 41.

Greg Yang and Etai Littwin. Tensor programs ivb: Adaptive optimization in the infinite-width limit. *arXiv:2308.01814*, 2023. Cited on page 2, 3, 16, 17, 19, 41.

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:2203.03466*, 2022. Cited on page 2, 3, 9, 16, 19, 43, 51, 57, 60, 70.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv:2310.17813*, 2023a. Cited on page 5, 17, 39, 47.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv:2310.02244*, 2023b. Cited on page 3, 5, 16, 17, 43.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations (ICLR)*, 2020. Cited on page 17.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. Cited on page 1, 17.

Appendices

Appendix Contents.

A Notation

Symbol	Meaning
n, η, ρ	width, learning rate, perturbation radius
ϕ, $\mathcal{L}$, (ξ_t, y_t)	activation function, loss function, input and label at time t
$\|v\| := \|v\|_2$, $\|W\| := \|W\|_F$	2-norm as standard for vectors, Frobenius norm as standard for matrices
$\|W\|_*$	spectral norm for matrices (also called operator norm)
W_t^l	trainable weights at time t in layer l
δW_t^l	weight updates at time t in layer l
$\boldsymbol{\varepsilon}_t^l$	weight perturbations at time t in layer l
$\|v_t\|$	norm of the rescaled gradient in the perturbation denominator
h_t^l, x_t^l	preactivations and activations at time t in layer l
δx_t^l	activation updates at time t in layer l
$\tilde{\delta} x_t^l$	activation perturbations at time t in layer l
$\delta f_t, \tilde{\delta} f_t$	update/perturbation of the output function at time t
$\chi_t = \mathcal{L}'(f_t(\xi_t), y_t)$	derivative of loss w.r.t. output function at time t
$\tilde{\delta} W_t^l = \boldsymbol{\varepsilon}_t^l$	weight perturbations at time t in layer l (with $\tilde{\delta}$ for consistency)
$\odot$	elementwise multiplication
$dz_t = \theta_\nabla^{-1} \nabla_z f$	derivative of output function w.r.t. $z \in \{h^l, x^l\}$ at time t, normalized to $\Theta(1)$
$dz_{SAM,t}$	derivative of perturbed output function w.r.t. perturbed $z \in \{\tilde{h}^l, \tilde{x}^l\}$ at time t, normalized to $\Theta(1)$
θ_∇	scaling of the activation gradients
$\theta_l, \tilde{\theta}_l$	update and perturbation scaling of h_t^l and x_t^l
$\theta_{W^l}, \tilde{\theta}_{W^l}$	update and perturbation scaling of W_t^l
$\mathring{\theta}$	limit scaling; under stability, all considered scalings $\mathring{\theta} \in \{0, 1\}$
Z^z	random variable distributed according to the limiting distribution for the entries of the TP vector z specified by the TP Master Theorem

Table A.1: **(Notation)** Overview over notation used in the main paper (top) and in the appendix (bottom).

B Detailed related work

Signal propagation. Our work can be seen as scaling theory with the goal of preventing both vanishing and exploding signals in forward and backward passes, where the analysis of SAM requires considering stability of perturbations in each layer as well. In this sense, we build on a rich literature, often restricted to an analysis at initialization (Schoenholz et al., 2016; Poole et al., 2016; Hanin and Rolnick, 2018; Xiao et al., 2020). For scaling neural networks to infinite depth, residual connections have been found to be beneficial for stabilizing signal propagation while retaining expressivity. The simple $\frac{1}{\sqrt{L}}$-scaling allows depth-scaling in ResNets and unlocks hyperparameter transfer (Hayou et al., 2021; Li et al., 2021; Bordelon et al., 2023; Yang et al., 2023b). Noci et al. (2022, 2024) provide infinite width and depth analyses for Transformers with the goal of preventing rank collapse and attaining a limit that has behaviour consistent with that of moderately large networks.

Tensor Programs. After kernel-based approaches to understand infinite-width limits of neural networks (Neal, 1996; Jacot et al., 2018) and applications of mean-field theory (Mei et al., 2018), the Tensor Program series (Yang, 2019; Yang and Hu, 2021; Yang and Littwin, 2023; Yang et al., 2022, 2023b) marks the first important break through in the theory of large neural networks. The framework covers many modern deep learning architectures, optimization algorithms and arbitrary abc-parameterizations, where each abc-parameterization is essentially defined by a layerwise scaling of initialization variance and learning rate as a function of network width. Yang and Hu (2021) propose the *maximal update parameterization* (μP) and show that it is the unique stable parameterization that achieves feature learning in all layers in the limit of infinite width. In this framework, training neural networks with a global learning rate $\eta > 0$ for all layers and with He or LeCun initialization falls under the category of so called *standard parameterization* (SP). The neural tangent parameterization (NTP), studied in the neural tangent kernel literature, differs but does not achieve feature learning in any layer, and is therefore less useful to describe the behaviour of finite width networks than μP (Wenger et al., 2023; Vyas et al., 2024). Yang and Littwin (2023) characterize stable learning with adaptive optimizers at infinite width into a feature learning versus a (nonlinear) operator regime. SAM

is not covered by the update rule definition in Yang and Littwin (2023) since the nested application of the gradient w.r.t. the weights is not a coordinatewise optimizer anymore. Yang et al. (2023a) show that μP is equivalent to the spectral scaling conditions on the weights $\|\Delta W^l\| = \Theta(\sqrt{n_l/n_{l-1}})$ and $\|\Delta W^l\| = \Theta(\sqrt{n_l/n_{l-1}})$. Hence Bernstein et al. (2020) would have achieved their goal of an optimizer with automatic update scaling, if they had normalized by the spectral instead of the Frobenius norm and multiplied by $\sqrt{\texttt{fan\_out}/\texttt{fan\_in}}$ in each layer. While recent works have considered joint limits of infinite width and depth (Yang et al., 2023b; Hayou and Yang, 2023), the data distribution has not been taken into account in Tensor Program literature. The study of scaling laws of jointly scaling model size, data set size and training time has predominantly been empirical (Kaplan et al., 2020; Zhai et al., 2022; Hoffmann et al., 2022; Besiroglu et al., 2024). Developing theory to inform Pareto optimal trade offs in a principled manner constitutes an important direction for future work.

Sharpness Aware Minimization. Sharpness aware minimization (SAM) (Foret et al., 2021) has shown to be extremely effective and robust in improving generalization performance across a wide range of architectures and settings (Chen et al., 2021; Kaddour et al., 2022). SAM was motivated as an inductive bias towards flatter minima and it has been understood to have an gradient-norm adaptive edge of stability at which it drifts towards minima with smaller spectral norm of the Hessian (Long and Bartlett, 2023; Bartlett et al., 2023). However a full understanding of why SAM works so well remains elusive. While correlations between flatness and generalization have been observed in some settings (Hochreiter and Schmidhuber, 1997; Jiang* et al., 2020), other studies have questioned the usefulness of sharpness as a measure for generalization, especially for modern architectures (Dinh et al., 2017; Andriushchenko et al., 2023b; Wen et al., 2024). Applying SAM on only the normalization layers often even improves generalization in vision tasks depsite increasing sharpness (Müller et al., 2024). Adaptive SAM (ASAM) (Kwon et al., 2021) is a variant of SAM derived from a sharpness definition that is invariant to weight rescalings with respect to a chosen normalization operator that leave the output function invariant. The results in Müller et al. (2024) suggest that two of the most promising normalization operators are elementwise normalization $T^l_w(x) = |W^l| \odot x$ and layerwise normalization $T^l_w(x) = \|W^l\|_F \cdot x$. We state the resulting update rules and a scaling analysis in Appendix F.4. A variant of SAM that is often studied theoretically because of its simplicity does not normalize the gradient of the perturbation. Our theory covers this variant too (Appendix F.3), but Dai et al. (2024) argue that normalizing the gradients for the perturbation is crucial. Monzio Compagnoni et al. (2023) find that unnormalized SAM gets stuck around saddles while SAM slowly escapes through additional Hessian-induced noise. This suggests that the additional effort of analysing the original SAM update rule with gradient normalization is necessary for practically useful theory. Dauphin et al. (2024) draw connections between SAM and other second order optimizers like gradient penalties and weight noise. They show that SAM is able to effectively use second order information implicitly using ReLU, whereas the other two methods close the gap to SAM when using GeLU since they require the localized second order information that GeLU provides in contrast to ReLU. Wen et al. (2023) show that worst-case, ascent and average case sharpness are biased towards minimizing the maximal eigenvalue, minimal non-zero eigenvalue and trace of the Hessian, respectively. With an architecture-agnostic analysis, they show that 1-SAM minimizes the trace of Hessian like average-case sharpness, for small enough η and ρ. Similarly, the theoretical results by Andriushchenko and Flammarion (2022) rely on the assumption that learning rate η and perturbation radius ρ are chosen sufficiently close to 0. Arguably, the empirically optimal choice of η and ρ lies outside of this gradient flow-like regime and has qualitatively different properties (see e.g. edge of stability literature (Cohen et al., 2020; Arora et al., 2022)).

Scaling theory for SAM. Shin et al. (2023) suggest that the generalisation improvement by SAM continues to increase with growing overparametrization. This corroborates empirical observations that performance monotonically improves with scale, and understanding the infinite-width limit is not only of theoretical interest but entails immediate practical benefits.

Liu et al. (2022) introduce Look-LayerSAM with layerwise perturbation scaling for preserving good performance under large batch training for enhanced training parallelization. They use LAMB (You et al., 2020) for layerwise learning rate scaling for large batch training. The update scaling strategy in these kinds of algorithms follows

$$W^l_{t+1} = W^l_t - \eta_t \phi(\|W^l_t\|_F) \frac{\nabla_{W^l}\mathcal{L}}{\|\nabla_{W^l} L\|_F},$$

with some $\phi : \mathbb{R}_+ \to \mathbb{R}_+$ and where $\nabla_{W^l}\mathcal{L}$ may be replaced by Adam's $\frac{m_t}{\sqrt{v_t}+\varepsilon}$. In practice, often simple functions like $\phi(x) = \max(c, \min(x, C))$ or $\phi(x) = x$ are used. The idea is to ensure that the update has the same order of magnitude as the weights. Look-LayerSAM follows an analogous approach for layerwise perturbation scaling. A derivation of μP for LAMB could also yield feature learning in all layers in the infinite-width limit as well as hyperparameter transfer. It certainly requires layerwise learning rate scaling. In the case $\phi(x) = x$, following a heuristic scaling derivation as in Appendix F.4 leads to layerwise learning rate scalings $\eta_1 = \eta_{L+1} = \Theta(1)$ and $\eta_l = \Theta(n^{-1/2})$ for hidden layers $l \in [2, L]$. With a bounded function like $\phi(x) = \max(c, \min(x, C))$, the scalings become $\eta_1 = \Theta(n^{1/2})$, $\eta_{L+1} = \Theta(n^{-1/2})$ and $\eta_l = \Theta(1)$ for hidden layers $l \in [2, L]$. We leave a closer investigation of feature learning and hyperparameter transfer with LAMB and Look-LayerSAM in SP and μP to future work.

C Definitions

In this section, we collect all definitions that do not appear in the main text. With minor modifications, we adopt all definitions from Yang and Hu (2021). If not stated otherwise, limits are taken with respect to width $n \to \infty$.

Definition C.1 (Big-O Notation). Given a sequence of scalar random variables $c = \{c_n \in \mathbb{R}\}_{n=1}^{\infty}$, we write $c = \Theta(n^{-a})$ if there exist constants A, B such that for almost every instantiation of $c = \{c_n \in \mathbb{R}\}_{n=1}^{\infty}$, for n large enough, $An^{-a} \leq |c_n| \leq Bn^{-a}$. Given a sequence of random vectors $x = \{x_n \in \mathbb{R}^n\}_{n=1}^{\infty}$, we say x has coordinates of size $\Theta(n^{-a})$ and write $x = \Theta(n^{-a})$ to mean the scalar random variable sequence $\left\{\sqrt{\|x_n\|^2/n}\right\}_n$ is $\Theta(n^{-a})$. For the definition of $c = O(n^{-a})$ and $c = \Omega(n^{-a})$, adapt the above definition of $c = \Theta(n^{-a})$ by replacing $An^{-a} \leq |c_n| \leq Bn^{-a}$ with $|c_n| \leq Bn^{-a}$ and $An^{-a} \leq |c_n|$, respectively. We write $x_n = o(n^{-a})$ if $n^a \cdot \sqrt{\|x_n\|^2/n} \to 0$ almost surely. ◀

Definition C.2 (Training routine). A *training routine* is a combination of base learning rate $\eta \geq 0$, perturbation radius $\rho \geq 0$, training sequence $\{(\xi_t, y_t)\}_{t\in\mathbb{N}}$ and a continuously differentiable loss function $\mathcal{L}(f(\xi), y)$ using the SAM update rule with layerwise perturbation scaling (LP). ◀

In addition to the stability conditions from the corresponding SGD result, we demand that the activation perturbations do not blow up. Otherwise the perturbations would strictly dominate both the initialization and the updates which makes the perturbation too strong and is avoided in practice.

Definition C.3 (Stability). We say a *bcd*-parametrization of an L-hidden layer MLP is *stable* if

1. For every nonzero input $\xi \in \mathbb{R}^{d_{\text{in}}}\backslash\{0\}$,
$$h_0^l, x_0^l = \Theta_\xi(1),\ \forall l \in [L], \quad \text{and} \quad \mathbb{E}f_0(\xi)^2 = O_\xi(1),$$
where the expectation is taken over the random initialization.
2. For any training routine, any time $t \in \mathbb{N}$, $l \in [L]$, $\xi \in \mathbb{R}^{d_{\text{in}}}$, we have
$$h_t^l(\xi) - h_0^l(\xi), x_t^l(\xi) - x_0^l(\xi) = O_*(1), \quad \text{and} \quad f_t(\xi) = O_*(1),$$
where the hidden constant in O_* can depend on the training routine, t, ξ, l and the initial function f_0.
3. For any training routine, any time $t \in \mathbb{N}_0$, $l \in [L]$, $\xi \in \mathbb{R}^{d_{\text{in}}}$, for the perturbed (pre-)activation $\tilde{h}_t^l := h^l(\tilde{W}_t)$, $\tilde{x}_t^l := x^l(\tilde{W}_t)$ and output function $\tilde{f}_t(\tilde{W}_t)$ we have
$$\tilde{h}_t^l(\xi) - h_t^l(\xi), \tilde{x}_t^l(\xi) - x_t^l(\xi) = O_*(1), \quad \text{and} \quad \tilde{f}_t(\xi) = O_*(1),$$
where the hidden constant in O_* can depend on the training routine, t, ξ, l and the initial function f_0.

◀

Definition C.4 (Nontriviality). We say a *bcd*-parametrization is *trivial* if for every training routine, $f_t(\xi) - f_0(\xi) \to 0$ almost surely for $n \to \infty$, for every time $t > 0$ and input $\xi \in \mathbb{R}^{d_{\text{in}}}$. Otherwise the *bcd*-parametrization is *nontrivial*. ◀

Definition C.5 (**Feature learning**). We say a bcd-parametrization *admits feature learning in the l-th layer* if there exists a training routine, a time $t > 0$ and input ξ such that $x_t^l(\xi) - x_0^l(\xi) = \Omega_*(1)$, where the constant may depend on the training routine, the time t, the input ξ and the initial function f_0 but not on the width n. ◀

Definition C.6 (**Vanishing perturbations**). Let $l \in [L]$. We say that a stable bcd-parametrization *has vanishing perturbations in the l-th layer* if for any training routine, $t \in \mathbb{N}_0$ and $\xi \in \mathbb{R}^{d_{\text{in}}}$, it holds that $\tilde{x}_t^l - x_t^l = o(1)$, and it *has vanishing perturbations in the output* if for any training routine, $t \in \mathbb{N}_0$ and $\xi \in \mathbb{R}^{d_{\text{in}}}$ it holds that $\tilde{\delta} f_t(\xi) := f_{\tilde{W}_t}(\xi) - f_{W_t}(\xi) = o(1)$. ◀

Definition C.7 (**Perturbation nontriviality**). Let $l \in [L]$. We say that a stable bcd-parametrization is *perturbation nontrivial with respect to the l-th layer* if and only if it does not have vanishing perturbations in the l-th layer. A stable bcd-parametrization is *perturbation nontrivial with respect to the output* if it does not have vanishing perturbations in the output. ◀

Definition C.8 (**Effective perturbations**). Let $l \in [L+1]$. We say that a stable bcd-parametrization *effectively perturbs the l-th layer* if there exists a training routine, $t \in \mathbb{N}$ and $\xi \in \mathbb{R}^{d_{\text{in}}}$ such that $\tilde{\delta} W_t^l \tilde{x}_t^{l-1}(\xi) = \Theta(1)$ where $\tilde{\delta} W_t^l$ is defined in (LP) and $\tilde{x}_t^0 = x_t^0 = \xi_t$. ◀

Definition C.9 (σ-**gelu**). Define σ-gelu to be the function $x \mapsto \frac{x}{2}\left(1 + \operatorname{erf}\left(\sigma^{-1}x\right)\right) + \sigma\frac{e^{-\sigma^{-2}x^2}}{2\sqrt{\pi}}$. ◀

In order to apply the Tensor Program Master Theorem, all Nonlin and Moment operations in the NE⊗OR⊤ program, which do not only contain parameters as inputs, are required to be pseudo-Lipschitz in all of their arguments. For training with SGD, this is fulfilled as soon as ϕ' is pseudo-Lipschitz. Both `tanh` as well as σ-`gelu` fulfill this assumption.

Definition C.10 (**Pseudo-Lipschitz**). A function $f : \mathbb{R}^k \to \mathbb{R}$ is called *pseudo-Lipschitz of degree d* if there exists a $C > 0$ such that $|f(x) - f(y)| \leq C\|x - y\|(1 + \sum_{i=1}^k |x_i|^d + |y_i|^d)$. We say f is *pseudo-Lipschitz* if it is so for any degree d. ◀

D Extensive main results

Using the formal definitions from Appendix C, here we provide the full formal statements of all of our main theoretical results together with further details and implications. The proof of all statements is provided in Appendix E. Since SAM evaluates the gradients on perturbed weights, it is not covered by the update rule definition in Yang and Littwin (2023) and an infinite-width analysis requires explicitly deriving the corresponding NE⊗OR⊤ program, scalings and infinite-width limits.

Recall that our definition of bcd-parameterizations extends abc-parameterizations by setting the maximal perturbation scaling to n^{-d} and allowing relative downweighting n^{-d_l} of the global scaling in each layer l. The perturbation scaling does not affect the choice of layerwise initialization variance scalings b_l and the layerwise learning rate scalings c_l. Common bc-parametrizations for SGD are summarized in Table D.1. SAM with SGD as a base optimizer requires the same scalings. Similarly, SAM with Adam as a base optimizer requires the same scalings as Adam (Yang et al., 2022, Table 3). Recall that, for convenience, we require width-independent denominator scaling $\|v_t\| = \Theta(1)$ of the scaled gradient for the perturbation (LP), which imposes the constraints

$$d_1 \geq 1/2 - \min(b_{L+1}, c_{L+1}), \quad d_l \geq 1 - \min(b_{L+1}, c_{L+1}) \text{ for } l \in [2, L], \quad d_{L+1} \geq 1/2. \tag{D.1}$$

All (pre-)activation and function outputs can be thought of as outputs given a fixed input $\xi \in \mathbb{R}^{d_{\text{in}}} \setminus \{0\}$ with $d_{in} \in \mathbb{N}$ fixed, e.g. $f_t := f_{W_t} := f_{W_t}(\xi)$. For the perturbed weights we write $\tilde{W}_t := W_t + \tilde{\delta} W_t$, with $\tilde{\delta} W_t$ defined in (LP) as $\boldsymbol{\varepsilon}_t^l$. Here we write weight perturbations as $\tilde{\delta} W_t^l$ instead of $\boldsymbol{\varepsilon}_t^l$ to show the resemblance to weight updates δW_t^l. Perturbed activations and function outputs at time t are written as $\tilde{x}_t^l(\xi) = x_{\tilde{W}_t}^l(\xi)$ and $\tilde{f}_t(\xi) = f_{\tilde{W}_t^l}(\xi)$. Recall that for all of the results in this section we make the following smoothness assumption on the activation function.

Assumption 1 (**Smooth activation function**). The used activation function is either `tanh` or σ-`gelu` for $\sigma > 0$ sufficiently small. ◀

We define the maximal feature update scale of a bcd-parameterization

$$r := \min(b_{L+1}, c_{L+1}, d + d_{L+1}) + \min_{l=1}^{L}(c_l - \mathbb{I}(l \neq 1)). \tag{D.2}$$

as well as the maximal feature perturbation scale of a bcd-parameterization

$$\tilde{r} := \min(b_{L+1}, c_{L+1}) + d + \min_{l=1}^{L}(d_l - \mathbb{I}(l \neq 1)). \tag{D.3}$$

Stability requires the constraints (a-c) from SGD and additional perturbation stability constraints (d-e) that include the layerwise perturbation scales $\{d_l\}_{l=1,\dots,L+1}$.

Theorem D.2 (**Stability characterization**). *A bcd-parametrization is stable if and only if all of the following are true:*

(a) (Stability at initialization, $h_0^l, x_0^l = \Theta(1)$ for all l, $f_0 = O(1)$)
$b_1 = 0$, $b_l = 1/2$ *for* $l \in [2, L]$ *and* $b_{L+1} \geq 1/2$.
(b) (Features do not blow up during training, i.e. $\Delta x_t^l = O(1)$ for all l)
$r \geq 0$.
(c) (Output function does not blow up during training, i.e. $\Delta W_t^{L+1} x_t^L, W_0^{L+1} \Delta x_t^L = O(1)$)
$c_{L+1} \geq 1$ *and* $b_{L+1} + r \geq 1$.
(d) (Feature perturbations do not blow up, i.e. $\tilde{\delta} x_t^l = O(1)$ for all l)
$\tilde{r} \geq 0$.
(e) (Output function perturbations do not blow up during training, i.e. $\tilde{\delta} W_t^{L+1} \tilde{x}_t^L, W_t^{L+1} \tilde{\delta} x_t^L = O(1)$)
$d + d_{L+1} \geq 1$ *and* $b_{L+1} + \tilde{r} \geq 1$.

The nontriviality and feature learning characterizations from SGD remain unaltered. This is because in the definition of r, it holds that $d + d_{L+1} \geq 1$ (from perturbation stability), and $\min(b_{L+1}, c_{L+1}) \leq 1$ already had to hold for nontriviality in SGD, so that stable perturbation scaling does not affect r.

Theorem D.3 (**Nontriviality characterization**). *A stable bcd-parametrization is nontrivial if and only if $c_{L+1} = 1$ or $\min(b_{L+1}, c_{L+1}) + r = 1$.*

As for nontriviality, the conditions under which a stable, nontrivial parameterization is feature learning in the infinite-width limit are decoupled from the choice of perturbation scalings $\{d_l\}_{l \in [L+1]} \cup \{d\}$. Hence the conditions are the same as for SGD. Below we provide a slightly refined result in terms of the maximal feature update scale r_{l_0} of a bcd-parameterization up to layer l_0 (as provided in the Appendix of Yang and Hu (2021)).

Theorem D.4 (**Feature learning characterization**). *For any $l_0 \in [L]$, the following statements are equivalent:*

(a) A stable, nontrivial bcd-parametrization admits feature learning in layer l_0.
(b) A stable, nontrivial bcd-parametrization admits feature learning in layer l for all $l \geq l_0$.
(c) $r_{l_0} := \min(b_{L+1}, c_{L+1}, d + d_{L+1}) + \min_{m=1}^{l_0}(c_m - \mathbb{I}(m \neq 1)) = 0$.

Consequently, a stable, nontrivial bcd-parametrization admits feature learning (at least in the last layer activations) if and only if $r = 0$.

Remark D.5 (**Effective feature learning**). As for perturbations, feature learning in later layers can be caused by weight updates in earlier layers that propagate through the network. One could demand effective feature learning in the l-th layer as $\delta W_t^l x_t^{l-1} = \Theta(1)$ and it would occur if and only if $\min(b_{L+1}, c_{L+1}, d + d_{L+1}) + c_l - \mathbb{I}(l \neq 1) = 0$. ◀

As for nontriviality, perturbation nontriviality in the output is attained if the constraints for $\tilde{\delta} W_t^{L+1} \tilde{x}_t^L$ or $W_t^l \tilde{\delta} x_t^L$ are exactly satisfied.

Theorem D.6 (**Perturbation nontriviality characterization**). *Let $l \in [L]$. A stable bcd-parametrization is perturbation nontrivial with respect to the l-th layer if and only if*

$$\tilde{r}_l := \min(b_{L+1}, c_{L+1}) + d + \min_{m=1}^{l}(d_m - \mathbb{I}(m \neq 1)) = 0.$$

A stable bcd-parametrization is perturbation nontrivial with respect to the output if and only if $d + d_{L+1} = 1$ or $\min(b_{L+1}, c_{L+1}) + \tilde{r} = 1$.

The converse formulation of the perturbation-nontriviality results characterizes the regime of vanishing perturbations.

Corollary D.7 (Vanishing perturbation characterization). *For any $l_0 \in [L]$, the following statements are equivalent:*

(a) A stable bcd-parametrization has vanishing perturbations in layer l_0.
(b) A stable bcd-parametrization has vanishing perturbations in layer l for all $1 \leq l \leq l_0$.
(c) $\tilde{r}_{l_0} := \min(b_{L+1}, c_{L+1}) + d + \min_{m=1}^{l_0}(d_m - \mathbb{I}(m \neq 1)) > 0$.

A stable bcd-parametrization has vanishing perturbations with respect to all layers and the output function if and only if $d_{L+1} > 1/2$ and $\tilde{r} > \max(0, 1 - b_{L+1})$. This case reduces to the results in Yang and Hu (2021).

For perturbation nontriviality it suffices that the perturbation in any of the previous layers is scaled correctly. For effective perturbations, we need the correct scaling in exactly that layer.

Theorem D.8 (Effective perturbation characterization). *For $l \in [L]$, a stable bcd-parametrization effectively performs SAM in the l-th layer if and only if $\min(b_{L+1}, c_{L+1}) + d + d_l - \mathbb{I}(l \neq 1) = 0$.*

A stable bcd-parametrization effectively performs SAM in the last layer if and only if $d + d_{L+1} = 1$.

The above understanding of all update and perturbation scalings allows us to extract the most important consequences of different choices of perturbation scaling on the learning dynamics. Beyond vanishing hidden layer perturbations, the following theorem shows that the joint gradient norm $\|v_t\|$ can be approximated efficiently without an additional backward pass under global perturbation scaling.

Theorem D.9 (Global Perturbation Scaling). *Given any stable bcd-parametrization $\{b_l\}_{l\in[L+1]} \cup \{c_l\}_{l\in[L+1]} \cup \{d_l\}_{l\in[L+1]} \cup \{d\}$. The parametrization performs updates in the original gradient direction if and only if $d_l = C$ for all $l \in [L+1]$ for some $C \in \mathbb{R}$. In this case, the parametrization has vanishing perturbations in all hidden layers $l \in [L]$, and the last layer $l = L+1$ is effectively perturbed if and only if $d = 1/2$. If $b_{L+1} > 1/2$ (as in μP), the gradient norm is dominated by the last layer and simplifies to,*

$$\|v_t\| = \Theta(n^{1/2-C}), \qquad \|v_t\| - \mathcal{L}'(f_t(\xi_t), y_t)\|x_t^L\| = o(n^{1/2-C}).$$

One might suspect that it is desirable to let all layers contribute non-vanishingly to the gradient norm in the denominator of (LP). The following proposition shows that this should be avoided with our definition of *bcd*-parameterizations. Of course, if we add even more hyperparameters by decoupling numerator and denominator scalings, we can set all contributions to $\Theta(1)$, which is what we do in Appendix F.7.

Proposition D.10 (Balancing gradient norm contributions). *Given any stable bcd-parametrization $\{b_l\}_{l\in[L+1]} \cup \{c_l\}_{l\in[L+1]} \cup \{d_l\}_{l\in[L+1]} \cup \{d\}$. If all layers contribute to the gradient norm non-vanishingly in the limit, i.e. $\|v_t^l\| = \Theta(\|v_t\|)$ for all $l \in [L+1], t \in \mathbb{N}_0$, then the parametrization has vanishing perturbations in all hidden layers $l \in [L]$. Such a parametrization effectively performs SAM in the last layer $l = L+1$ if and only if $d = 1/2$.*

The following theorem provides the unique correct perturbation scaling for any stable *bc*-parameterization with $b_{L+1} \geq 1$.

Theorem D.11 (Perturbation Scaling Choice for Effective Perturbations). *Given any stable bcd-parametrization $\{b_l\}_{l\in[L+1]} \cup \{c_l\}_{l\in[L+1]} \cup \{d_l\}_{l\in[L+1]} \cup \{d\}$. If $b_{L+1} < 1$, then there does not exist a stable choice of $\{d_l\}_{l\in[L+1]} \cup \{d\}$ that achieves effective perturbations before the last layer. If $b_{L+1} \geq 1$, then up to the equivalence $d_l' = d_l + C$, $C \in \mathbb{R}$, $\forall l \in [L+1]$, the unique stable choice $\{d_l\}_{l\in[L+1]} \cup \{d\}$ with effective perturbations in all layers $l \in [L+1]$ is given by*

$$d = -1/2, \qquad d_l = \begin{cases} 1/2 - \min(b_{L+1}, c_{L+1}) & l = 1, \\ 3/2 - \min(b_{L+1}, c_{L+1}) & l \in [2, L], \\ 3/2 & l = L+1. \end{cases} \tag{D.4}$$

In this parameterization, the first layer dominates the gradient norm as

$$\|v_t\| = \Theta(1), \qquad \big|\|v_t^1\| - \|v_t\|\big| = \Theta(n^{-1/2}).$$

Table D.2 summarizes the consequences of Theorem D.11. Together with Theorem D.11, the following proposition suggests that $b_{L+1} = 1$ is a good choice. However $b_{L+1} > 1$ can also induce effective perturbations, as long as d and d_{L+1} are chosen correctly.

	Definition	SP	SP (stable)	NTP (stable)	μP
b_l	$\mathcal{N}(0, n^{-2b_l})$	$\begin{cases} 0 & l = 1, \\ 1/2 & l \geq 2. \end{cases}$	$\begin{cases} 0 & l = 1, \\ 1/2 & l \geq 2. \end{cases}$	$\begin{cases} 0 & l = 1, \\ 1/2 & l \geq 2. \end{cases}$	$\begin{cases} 0 & l = 1, \\ 1/2 & l \in [2, L], \\ 1, & l = L + 1. \end{cases}$
c_l	LR ηn^{-c_l}	0	1	$\begin{cases} 0 & l = 1, \\ 1 & l \geq 2. \end{cases}$	$\begin{cases} -1 & l = 1, \\ 0 & l \in [2, L], \\ 1 & l = L + 1. \end{cases}$
r	Equation (D.2)	-1	1/2	1/2	0
Stable?			✓	✓	✓
Nontrivial?			✓	✓	✓
Feature learning?					✓

Table D.1: **(bc-parametrizations)** Overview over common implicitly used bc-parametrizations for training MLPs without biases in standard parametrization (SP), standard parametrization with maximal stable nonadaptive LR $c = 1$ (SP (stable)), neural tangent parametrization (NTP) and maximal update parametrization (μP).

	Definition	Naive	Global (stable)	Effective
d	ρn^{-d}	0	1/2	$-1/2$
d_l	$n^{-d_l} \nabla_{W^l} \mathcal{L}_t$	1/2	1/2	$\begin{cases} 1/2 - c_\nabla & l = 1, \\ 3/2 - c_\nabla & l \in [2, L], \\ 3/2 & l = L + 1. \end{cases}$
$\tilde{r}$	Equation (D.3)	$c_\nabla - 1/2$	c_∇	0
Stable?		✗	✓	✓
Last layer effectively perturbed?		✗	✓	✓
All layers effectively perturbed?		✗	✗	✓

Table D.2: **(Perturbation scalings)** Overview over important choices of the global perturbation scaling ρn^{-d} and the layerwise perturbation scalings n^{-d_l} for training MLPs without biases with SAM: Naive scaling without width dependence (Naive), maximal stable global scaling along the original gradient direction (Global) and the unique scaling that achieves effective perturbations in all layers (Effective). An extensive overview that characterizes all possible choices of perturbation scaling is provided in Appendix F.1. Recall the gradient scaling $c_\nabla := \min(b_{L+1}, c_{L+1})$.

Proposition D.12 (**Effects of last-layer initialization b_{L+1} on all perturbations**). *If a stable bcd-parametrization with $\min(b_{L+1}, c_{L+1}) \leq 1$ is perturbation nontrivial with respect to any hidden layer $l \in [L]$, it is also perturbation nontrivial with respect to the output.*

Lastly, the following proposition shows that effective perturbations from the first layer propagate through the entire network.

Proposition D.13 (**Perturbations propagate through the forward pass**). *All stable bcd-parametrizations with $d_1 = -\min(b_{L+1}, c_{L+1}) - d$ effectively perturb the first layer and are perturbation nontrivial in all layers.*

Remark D.14 (**Efficiency gains**). The above results may be used for efficiency gains. Given any stable bcd-parametrization, we can compute the maximal layer l_0 such that $\tilde{r}_{l_0} > 0$, and in wide networks do not have to compute SAM perturbations before layer $l_0 + 1$; as soon as $b_{L+1} > 1/2$ (as for μP), the gradient norm for the SAM update rule is approximately given by $\|\nabla L_t\| \approx \mathcal{L}'(f_t(\xi_t), y_t)\|x_t^L\|$, which can directly be computed without an additional backward pass. The practical recommendation from our experiments however is to either use μP^2 or to completely abstain from perturbations. ◀

Remark D.15 (**SAM without gradient normalization**). For the SAM update rule without gradient normalization simply set $d = 0$ and remove the gradient norm constraints (D.1) to arrive at the adapted NE$\otimes$OR$\top$ program and bcd-constraints. Note that standard parametrization gets even more unstable without dividing by $\|\nabla L\| = \Theta(n^{1/2})$, now requiring $d_{L+1} \geq 1$ for stability. Similar to the previous results, this shows that unawareness of bcd-parametrizations requires strongly scaling

down ρ for stability, while vasting computation on vanishing perturbations before the last layer. More details can be found in Appendix F.3. ◀

E Proof of main results

In this section we derive the NE⊗OR⊤ program that corresponds to training a MLP without biases with SAM. For simplicity and clarity of the proof, we prove the one-dimensional case $d_{in} = 1$, $d_{out} = 1$, but an extension to arbitrary but fixed d_{in}, d_{out} is straightforward. Recall Assumption 1 that allows us to apply the Tensor Program Master Theorem and explicitly state the infinite-width limit of training MLPs with SAM in Appendix E.2.

E.1 Tensor program formulation

E.1.1 Tensor Program initialization

We initialize the matrices $W_0^2, \dots, W_0^L$ as $(W_0^l)_{\alpha\beta} \sim \mathcal{N}(0, 1/n)$, which absorbs $b_l = 1/2$.

We initialize the input layer matrix $W_0^1 \in \mathbb{R}^{n\times 1}$ and normalized output layer matrix $\hat{W}_0^{L+1} = W_0^{L+1} n^{b_{L+1}} \in \mathbb{R}^{1\times n}$ as $(W_0^1)_\alpha, (\hat{W}_0^{L+1})_\alpha \sim \mathcal{N}(0, 1)$, as initial vectors should have a distribution that is $\Theta(1)$.

In the NE⊗OR⊤formulation, we write all quantities as $\theta_z z$, where θ_z denotes their scaling n^C for some $C \in \mathbb{R}$ and z therefore has a $\Theta(1)$ distribution. The stability, nontriviality and feature learning conditions then stem from requiring either $\theta_z \to 0$ or $\theta_z = 1$ depending on z and its desired scale.

E.1.2 First forward pass

We denote a definition of a Tensor Program (TP) or NE⊗OR⊤computation as $:=$. Compared to MLPs trained with SGD nothing changes in the first forward pass,

$$h_0^1(\xi) := W_0^1 \xi \quad \text{(NL)}, \quad x_0^l := \phi(h_0^l) \quad \text{(NL)}, \quad h_0^{l+1} := W_0^{l+1} x_0^l. \quad \text{(MatMul)}$$

In the case of MuP, $f_0(\xi) = W_0^{L+1} x_0^L(\xi) \to 0$ defines a scalar in the TP.

Observe the scalings $x_0^1 = \Theta(h_0^1) = \Theta(n^{-b_1}), x_0^l = \Theta(h_0^l) = \Theta(n^{1/2-b_l})$ for $l \in [2, L]$ due to CLT, independence at initialization and $x_0^l = \Theta(h_0^l) = \Theta(1)$ by stability. Hence stability at initialization inductively requires $b_1 = 0$, $b_l = 1/2$ for $l \in [2, L]$ and $b_{L+1} \geq 1/2$.

E.1.3 First backward pass

The chain rule of the derivative remains the same, we just evaluate on different weights compared to standard SGD. We denote the adversarially perturbed weights by $\tilde{W}_t^l$ and the normalized perturbations by $\tilde{\delta} W_t^l$. Before computing the updates we have to compute a full backward pass to determine these perturbed weights for each layer, and then compute a forward pass with these perturbed weights to compute the perturbed preactivations $\tilde{h}_t^l$ that we will need for computing the SAM update. Therefore the NE⊗OR⊤ program for SAM maintains a perturbed copy of all preactivations, activations, last-layer weights and logits just for computing the updates of the actual parameters.

Under MuP, the loss derivative with respect to the function remains $\chi_0 := \mathcal{L}'(f_0(\xi_0), y_0) \to \mathring{\chi}_0 := \mathcal{L}'(0, y_0)$. For the weight perturbation, we need to perform a SGD backward pass,

$$dx_0^L := \hat{W}_0^{L+1}, \quad dh_0^l := dx_0^l \odot \phi'(h_0^l), \quad dx_0^{l-1} := (W_0^l)^T dh_0^l,$$

where $dz := \theta_\nabla^{-1} \nabla_z f$. For SGD (and for SAM, as we will see later) all gradients have scaling $\theta_\nabla := n^{-b_{L+1}}$ in the first step, whereas we overload the notation $\theta_\nabla := n^{-\min(b_{L+1}, c_{L+1})}$ for all later steps. For clarity of presentation assume $b_{L+1} \geq c_{L+1}$ here, the other case follows analogously. For the first step this can be understood from

$$\nabla_{x^L} f_0 = W_0^{L+1} = \Theta(n^{-b_{L+1}}), \qquad \nabla_{h^L} f_0 = \nabla_{x^L} f_0 \odot \phi'(h_0^L) = \Theta(n^{-b_{L+1}}),$$

since $h_0^L = \Theta(1)$ by the stability assumption, and this scale $\Theta(n^{-b_{L+1}})$ propagates through all layers via the chain rule and remains stable in later backward passes. For hidden layer gradients, observe

that

$$\begin{aligned}\nabla_{x^{L-1}} f_t &= (W_t^L)^T \nabla_{h^L} f_t = (W_0^L + \Delta W_t^L)^T \nabla_{h^L} f_t \\ &= \Theta\left((W_0^L)^T \nabla_{h^L} f_t - n^{-c_L} \sum_{s=0}^{t-1} ((\nabla_{h^L} f_s)^T \nabla_{h^L} f_t) x_s^{L-1} \right) \\ &= \Theta(n^{1-2b_L}\theta_\nabla - n^{-c_L}\theta_\nabla^2 n) = \Theta(\theta_\nabla),\end{aligned}$$

where first term's scale stems from the products $(W_0^L)^T W_0^L v = \Theta(n^{1-2b_L} v)$ due to Yang (2021), $b_L = 1/2$ for stability at initialization and $b_{L+1} + c_L \geq 1$ for update stability during training ($r \geq 0$). If we allowed the second term to strictly dominate, the gradient scale would explode iteratively in the backward pass.

The gradient norm. Before computing the weight perturbations, we need to compute the gradient norm for the SAM update. The gradient norm at time t in each layer $l \in [2, L]$ is given by the scalar,

$$\theta_\nabla^{-2} \left\| \frac{\partial L_t}{\partial W^l} \right\|^2 = \sum_{i,j=1}^{n} \left(\chi_t (dh_t^l)_i (x_t^{l-1})_j \right)^2 = \chi_t^2 \|dh_t^l (x_t^{l-1})^T\|_F^2 = \chi_t^2 \big((dh_t^l)^T dh_t^l \big) \big((x_t^{l-1})^T x_t^{l-1} \big),$$

where $\chi_t = \mathcal{L}'(f_t(\xi_t), y_t)$ and we used $\partial h^l / \partial W_{ij}^l = (x_j^{l-1} \delta_{ik})_{k=1,\dots,n}$.

Hence the gradient norm of all weights jointly is given by the unnormalized scalar

$$\|\nabla_w L_t\|^2 = \chi_t^2 \left(n\theta_\nabla^2 \frac{(dh_t^1)^T dh_t^1}{n} (\xi_t^T \xi_t) + \sum_{l=2}^{L} n^2 \theta_\nabla^2 \frac{(dh_t^l)^T dh_t^l}{n} \frac{(x_t^{l-1})^T x_t^{l-1}}{n} + n \frac{(x_t^L)^T x_t^L}{n} \right), \tag{E.1}$$

with scaling $\theta_{\|\nabla\|}^2 = \Theta(n^2\theta_\nabla^2 + n) = \Theta(n)$, because stability at initialization requires $b_{L+1} \geq 1/2$ so that $n^2\theta_\nabla^2 \leq n$. Note that the first layer contributes vanishingly to the gradient norm, the hidden layer gradients only if $b_{L+1} = 1/2$ (equivalently $f_0 = \Theta(1)$) and the last-layer activations always in dominating order. So in μP, in the limit, $\|\nabla_w L_t\| = \mathcal{L}'(f_t(\xi_t), y_t)\|x_t^L\|$. This means that the unscaled gradient always aligns with the last-layer activation. For learning in μP, this dominance is corrected by the layerwise learning rates.

The squared norm of the rescaled gradient is given by

$$\begin{aligned}\|v_t\|^2 = \chi_t^2 \Bigg(& n\theta_\nabla^2 n^{-2d_1} \frac{(dh_t^1)^T dh_t^1}{n} (\xi_t^T \xi_t) \\ & + \sum_{l=2}^{L} n^2 \theta_\nabla^2 n^{-2d_l} \frac{(dh_t^l)^T dh_t^l}{n} \frac{(x_t^{l-1})^T x_t^{l-1}}{n} + n n^{-2d_{L+1}} \frac{(x_t^L)^T x_t^L}{n} \Bigg),\end{aligned} \tag{E.2}$$

with scaling $\theta_v^2 = \Theta(n^{1-2d_1}\theta_\nabla^2 + \sum_{l=2}^{L} n^{2-2d_l}\theta_\nabla^2 + n^{1-2d_{L+1}})$. For simplicity, set $\theta_v = 1$. This raises the constraints $n^{1-2d_1}\theta_\nabla^2 \leq 1$, $n^{2-2d_l}\theta_\nabla^2 \leq 1$ for $l \in [2, L]$ and $n^{1-2d_{L+1}} \leq 1$, which can be rewritten as

$$d_1 \geq 1/2 - \min(b_{L+1}, c_{L+1}), \quad d_l \geq 1 - \min(b_{L+1}, c_{L+1}) \text{ for } l \in [2, L], \quad d_{L+1} \geq 1/2,$$

where at least one equality is demanded to hold in order to attain $\theta_v = 1$. If one of the equalities holds, the respective layer contributes to the norm non-vanishingly in the limit.

Thus, applying the square root and dividing by $\theta_v = 1$ the square root of (E.2) defines a normalized TP scalar.

Perturbations. Stability implies that also the perturbed (pre-)activations and output function remain $\Theta(1)$ and $O(1)$ respectively. Otherwise a SAM training step would induce blowup in the updates. We call this weaker property of just the perturbations *perturbation stability*.

Definition E.1 (Perturbation stability)**.** We call a *bcd*-parametrization *perturbation stable* if and only if $\tilde{h}_t^l, \tilde{x}_t^l = \Theta(1)$ for all $l \in [L]$ and $t \in \mathbb{N}$ and $\tilde{\delta} f_t = O(1)$ for all $t \in \mathbb{N}$. ◀

Mathematically we get the normalized weight perturbations for $l \in \{2, \dots, L\}$,

$$\tilde{\delta} W_0^{L+1} := \frac{\rho\ \chi_0\ x_0^L}{\|v_0\|}, \quad \tilde{\delta} W_0^l = \frac{\rho\ \chi_0\ dh_0^l\ (x_0^{l-1})^T}{\|v_0\|}, \quad \tilde{\delta} W_0^1 = \frac{\rho\ \chi_0\ dh_0^1\ \xi_0^T}{\|v_0\|},$$

which scale as $\tilde{\theta}_{L+1} := \tilde{\theta}_{W^{L+1}} := n^{-(d+d_{L+1})}$, $\Theta(n^{(d+d_l)-b_{L+1}})$ and $\Theta(n^{-(d+d_1)-b_{L+1}})$ respectively. But the NE$\otimes$OR$\top$ program computation rules do not allow to compute matrices $\tilde{\delta}W_0^l, l \in [L]$, therefore we use the weight updates to directly compute the preactivation and activation changes analogous to the t-th forward pass. For all $t \geq 0$, we write

$$\tilde{h}_t^l = h_t^l + \tilde{\theta}_l \tilde{\delta} h_t^l, \qquad \tilde{x}_t^l = x_t^l + \tilde{\theta}_l \tilde{\delta} x_t^l,$$

with the perturbations for $l \in [2, L]$,

$$\begin{aligned}
\tilde{\delta} h_0^1(\xi) :=& \quad + \frac{\rho \chi_0(\xi_0^T \xi) dh_0^1}{\|v_0\|},\\
\tilde{\delta} x_t^l :=& \quad \tilde{\theta}_l^{-1}(\phi(h_t^l + \tilde{\theta}_l \tilde{\delta} h_t^l) - \phi(h_t^l)),\\
\tilde{\theta}_l \tilde{\delta} h_0^l :=& \quad \tilde{\theta}_{l-1} W_0^l \tilde{\delta} x_0^{l-1} + (\tilde{W}_0^l - W_0^l)\tilde{x}_0^{l-1}\\
=& \quad \tilde{\theta}_{l-1} W_0^l \tilde{\delta} x_0^{l-1} + \rho \tilde{\theta}_{W^l} \frac{\chi_0}{\|v_0\|} \frac{(x_0^{l-1})^T \tilde{x}_0^{l-1}}{n} dh_0^l,
\end{aligned}$$

which defines a NonLin operation with the vectors $W_0^l \tilde{\delta} x_0^{l-1}$ and dh_0^l and everything else treated as scalars, and with first backward pass scalings $\tilde{\theta}_{W^1} := n^{-(d+d_1)}\theta_\nabla$, $\tilde{\theta}_{W^l} := n^{1-(d+d_l)}\theta_\nabla$ and $\tilde{\theta}_l := \max(\tilde{\theta}_{l-1}, \tilde{\theta}_{W^l}) = \max_{m=1}^l \tilde{\theta}_{W^m}$, where we used that $\tilde{x}_0^{l-1} = \Theta(1)$ due to perturbation stability. Note that these scalings may implicitly increase when $t > 0$ since $\theta_\nabla = n^{-b_{L+1}}$ gets replaced by $\theta_\nabla = n^{-\min(b_{L+1}, c_{L+1})}$.

The activation perturbations can then simply be defined via the NonLin operation,

$$\tilde{\delta} x_0^l := \tilde{\theta}_l^{-1}(\phi(h_0^l + \tilde{\theta}_l \tilde{\delta} h_0^l) - \phi(h_0^l)),$$

with the same scaling as $\tilde{\delta} h_0^l$.

The perturbation of the scalar output function can simply be defined via the NonLin operation,

$$\tilde{\delta} f_0 := \tilde{W}_0^{L+1} \tilde{x}_0^L - W_0^{L+1} x_0^L = \tilde{\theta}'_{L+1} \frac{\tilde{\delta} W_0^{L+1} \tilde{x}_0^L}{n} + \tilde{\theta}'_{L\nabla} \frac{\hat{W}_0^{L+1} \tilde{\delta} x_0^L}{n},$$

with $\tilde{\theta}'_{L+1} := n\tilde{\theta}_{W^{L+1}}$ and $\tilde{\theta}'_{L\nabla} := n\theta_\nabla \tilde{\theta}_L$.

SAM Update. Finally, we can compute the SAM updates as follows. In the case $\min(b_{L+1}, c_{L+1}) \leq d + d_{L+1}$ the weight perturbation scale is dominated by the weight scale, so that

$$dx_{SAM,0}^L := \hat{W}_0^{L+1} + \tilde{\theta}_{(L+1)/\nabla}\, \tilde{\delta} W_0^{L+1},$$

with $\tilde{\theta}_{(L+1)/\nabla} := \tilde{\theta}_{L+1}/\theta_\nabla \leq 1$, whereas if $\min(b_{L+1}, c_{L+1}) > d + d_{L+1}$ we write

$$dx_{SAM,0}^L := \tilde{\theta}_{\nabla/(L+1)} \hat{W}_0^{L+1} + \tilde{\delta} W_0^{L+1},$$

with $\theta_{\nabla/(L+1)} := \theta_\nabla/\tilde{\theta}_{L+1} \leq 1$. In any case, the scaling of $dx_{SAM,0}^L$ and all other SAM gradients is $\theta_{SAM} := \max(\theta_\nabla, n^{-(d+d_{L+1})}) = n^{-\min(b_{L+1}, c_{L+1}, d+d_{L+1})}$. The other SAM gradients are given by

$$\begin{aligned}
dh_{SAM,0}^l :=& \quad dx_{SAM,0}^l \odot \phi'(\tilde{h}_0^l)\\
dx_{SAM,0}^{l-1} :=& \quad (\tilde{W}_0^l)^T dh_{SAM,0}^l = (W_0^l + \tilde{\theta}_{W^l} \tilde{\delta} W_0^l)^T dh_{SAM,0}^l\\
=& \quad (W_0^l)^T dh_{SAM,0}^l + \rho \theta_{SAM} \tilde{\theta}_{W^l} \frac{\chi_0}{\|v_0\|} \frac{(dh_0^l)^T dh_{SAM,0}^l}{n} x_0^{l-1}.
\end{aligned}$$

where the last line define a NonLin operation in the vectors $(W_0^l)^T dh_{SAM,t}^l$ and x_0^{l-1} and everything else treated as scalars. Consequently, $\nabla_{h_0^l} f|_{\tilde{W}_0}$ is of the same scale as $\nabla_{x_0^l} f|_{\tilde{W}_0}$ and $\nabla_{x_0^{l-1}} f|_{\tilde{W}_0}$ is of the scale $\max(\theta_{SAM}, \tilde{\theta}_{W^l}\theta_{SAM}) = \theta_{SAM}$ since $\tilde{\theta}_{W^l} \leq 1$ is required for perturbation stability.

Note that for SAM's weight updates the loss derivative is also evaluated on the perturbed weights,

$$\tilde{\chi}_0 := \mathcal{L}'(\tilde{W}_0^{L+1} \tilde{x}_0^L, y_0).$$

Constraints on the output function. Assuming $\tilde{x}_0^L = \Theta(1)$ (perturbation stability), we get $\tilde{\chi}_0 = O(1)$ if and only if $\tilde{\delta} W_0^{L+1} = O(n^{-1})$ if and only if $d + d_{L+1} \geq 1$.

We have $\tilde{\chi}_0 = \Theta(1)$ if and only if $\tilde{f}_0 = \tilde{W}_0^{L+1} \tilde{x}_0^L = \Theta(1)$. This can either be caused by changes in the last-layer weights, by non-vanishing initial function $W_0^{L+1} x_0^L$ (if and only if $b_{L+1} = 1/2$) or by $W_0^{L+1} \tilde{\delta} x_0^L = \Theta(1)$, which holds if and only if $b_{L+1} + \tilde{r}_L = 1$ (analogously, $W_0^{L+1} \tilde{\delta} x_0^L = O(1)$ if and only if $b_{L+1} + \tilde{r}_L \geq 1$). The first case requires $\tilde{\delta} W_0^{L+1} = \Theta(n^{-1})$, since $\tilde{\delta} W_0^{L+1}$ and $\tilde{x}_0^L$ are highly correlated. $\tilde{\delta} W_0^{L+1} = \Theta(n^{-1})$ is fulfilled if and only if $d + d_{L+1} = 1$ (the analogue to $c_{L+1} \geq 1$ for stability and $c_{L+1} = 1$ for nontriviality).

Hence perturbation stability of the output function holds only if $d + d_{L+1} \geq 1$ and $b_{L+1} + \tilde{r}_L \geq 1$. Then, perturbation nontriviality holds if and only if $d + d_{L+1} = 1$ or $b_{L+1} + \tilde{r}_L = 1$.

In the t-th backward pass, $b_{L+1} + \tilde{r}_L \geq 1$ will be replaced by the slightly stronger constraint $b_{L+1} + \tilde{r} \geq 1$.

E.1.4 t-th forward pass

Formally, we sum the updates in each step,

$$\hat{W}_t^{L+1} := \hat{W}_0^{L+1} + \theta_{L+1/\nabla} (\delta W_1^{L+1} + \cdots + \delta W_t^{L+1}),$$

where $\delta W_{t+1}^{L+1} := -\eta \ \tilde{\chi}_t \ (\tilde{x}_t^L)^T$ denotes the normalized change in the weights W^{L+1} (as a row vector) of scaling $\theta_{L+1} = \theta_{W^{L+1}} = n^{-c_{L+1}}$ under perturbation stability and nontriviality so that $\hat{W}_t^{L+1}$ scales as $\theta_\nabla = n^{-\min(b_{L+1}, c_{L+1})}$. δW_{t+1}^{L+1} should not be confused with $\tilde{\delta} W_{t+1}^{L+1}$ which denotes the perturbation of the weights at time $t+1$. For every nontrivial stable parametrization we have $\tilde{\chi}_t = \Theta(1)$ and $\tilde{x}_t^L = \Theta(1)$ which requires $\tilde{\theta}_L \leq 1$. In the case $c_{L+1} < b_{L+1}$, we write $\hat{W}_t^{L+1} := n^{-b_{L+1} + c_{L+1}} \hat{W}_0^{L+1} + (\delta W_1^{L+1} + \cdots + \delta W_t^{L+1})$ with the same scaling $\theta_\nabla = n^{-\min(b_{L+1}, c_{L+1})}$.

For preactivations and activations we also sum the changes from each step,

$$h_t^l := h_0^l + \theta_l (\delta h_1^l + \cdots + \delta h_t^l), \qquad x_t^l := x_0^l + \theta_l (\delta x_1^l + \cdots + \delta x_t^l).$$

Using the fact that

$$W_t^1 - W_{t-1}^1 = -\eta \tilde{\chi}_{t-1} \theta_{W^1} dh_{SAM,t-1}^1 \xi_{t-1}^T,$$

yields the normalized preactivation updates

$$\delta h_t^1(\xi) := -\eta \tilde{\chi}_{t-1} dh_{SAM,t-1}^1 \xi_{t-1}^T \xi \quad \text{(NL)},$$

with scaling $\theta_1 = \theta_{W^1} = n^{-c_1} \theta_{SAM} = n^{-c_1 - \min(b_{L+1}, c_{L+1}, d + d_{L+1})}$ as for SGD under perturbation stability and nontriviality where $\tilde{\chi}_{t-1} = \Theta(1)$.

For $l \in [2, L]$, it holds that

$$W_t^l - W_{t-1}^l = -\eta \tilde{\chi}_{t-1} \theta_{W^l} \frac{1}{n} dh_{SAM,t-1}^l (\tilde{x}_{t-1}^{l-1})^T,$$

with the right scaling $\theta_{W^l} = n^{1 - c_l - \min(b_{L+1}, c_{L+1}, d + d_{L+1})}$ as for SGD under perturbation stability $\tilde{x}_{t-1}^{l-1} = \Theta(1)$, so that we get δh_t^l using a telescope sum,

$$\begin{aligned}
\theta_l \delta h_t^l = \quad & W_t^l x_t^{l-1} - W_{t-1}^l x_{t-1}^{l-1} = W_{t-1}^l (x_t^{l-1} - x_{t-1}^{l-1}) + (W_t^l - W_{t-1}^l) x_t^{l-1} \\
= \quad & \theta_{l-1} \left(W_0^l \delta x_t^{l-1} + \sum_{s=1}^{t-1} (W_s^l - W_{s-1}^l) \delta x_t^{l-1} \right) + (W_t^l - W_{t-1}^l) x_t^{l-1} \\
= \quad & \theta_{l-1} \left(W_0^l \delta x_t^{l-1} - \eta \theta_{W^l} \sum_{s=1}^{t-1} \tilde{\chi}_{s-1} \frac{(\tilde{x}_{s-1}^{l-1})^T \delta x_t^{l-1}}{n} dh_{SAM,s-1}^l \right) \\
& - \eta \theta_{W^l} \tilde{\chi}_{t-1} \frac{(\tilde{x}_{t-1}^{l-1})^T x_t^{l-1}}{n} dh_{SAM,t-1}^l,
\end{aligned}$$

which defines a NonLin operation with the vectors $W_0^l \delta x_t^{l-1}, dh_{SAM,0}^l, dh_{SAM,t-1}^l$ and everything else treated as scalars. The scaling is given by

$$\theta_l = \max(\theta_{l-1}, \theta_{W^l}\theta_{l-1}, \theta_{W^l}) = \max_{m=1}^{l} \theta_{W^m} = n^{-r_l},$$

with

$$r_l := \min(b_{L+1}, c_{L+1}, d + d_{L+1}) + \min_{m=1}^{l}(c_m - \mathbb{I}(m \neq 1)),$$

where $\theta_{W^l} \leq 1$ for all $l \in [L]$ for stability. Note that for $l_1 \leq l_2$, it holds that $\theta_{l_1} \leq \theta_{l_2}$, which explains the sufficiency of $\theta_L = n^{-r_L} = n^{-r}$ for the stability of the activation updates.

Activations with the same scaling θ_l can then simply be defined via the NonLin operation

$$\delta x_t^l := \theta_l^{-1}(\phi(h_{t-1}^l + \theta_l \delta h_t^l) - \phi(h_{t-1}^l)).$$

The updates of the output function are scalars defined as

$$\delta f_t := \theta'_{L+1} \frac{\delta W_t^{L+1} x_t^L}{n} + \theta'_{L\nabla} \frac{\hat{W}_{t-1}^{L+1} \delta x_t^L}{n},$$

where $\theta'_{L+1} = n\theta_{L+1} = n^{1-c_{L+1}}$ and $\theta'_{L\nabla} = n\theta_\nabla \theta_L = n^{1-\min(b_{L+1}, c_{L+1}) - r_L}$, where we will see why $W_{t-1}^{L+1} = \Theta(n^{-\min(b_{L+1}, c_{L+1})})$ in the next paragraph. This leads to the constraints $c_{L+1} \geq 1$ and $b_{L+1} + r \geq 1$ for the stability of the output function, where equality in either constraint leads to nontriviality.

E.1.5 t-th backward pass

Perturbations. Due to linearity and stability, the last layer remains

$$dx_t^L := \hat{W}_t^{L+1},$$

with scaling $\theta_\nabla = n^{-\min(b_{L+1}, c_{L+1})}$.

As in the first backward pass, we use the weight updates to directly compute the preactivation and activation perturbations similar to the t-th forward pass but performing SGD instead of SAM in the last step. The SGD backward pass for the perturbation is given by

$$\begin{aligned}
dh_t^l &:= dx_t^l \odot \phi'(h_t^l), \\
dx_t^{l-1} &:= (W_t^l)^T dh_t^l \\
&= \left(W_0^l - \eta\theta_{W^l} \sum_{s=1}^{t} \tilde{\chi}_{s-1} \frac{1}{n} dh_{SAM,s-1}^l (\tilde{x}_{s-1}^{l-1})^T \right)^T dh_t^l \\
&= W_0^l dh_t^l - \eta(n^{1-c_l}\theta_{SAM}\theta_\nabla) \sum_{s=1}^{t} \tilde{\chi}_{s-1} \frac{(dh_{SAM,s-1}^l)^T dh_t^l}{n} \tilde{x}_{s-1}^{l-1},
\end{aligned}$$

with scaling $\max(\theta_\nabla, n^{1-c_l}\theta_{SAM}\theta_\nabla) = \theta_\nabla$, since $n^{1-c_l}\theta_{SAM} \leq 1$ is implied by $r \geq 0$ required for the stability of (pre-)activation updates.

We write $\chi_t = \mathcal{L}'(f_t(\xi_t), y_t)$ for the derivative of the loss with respect to the unperturbed function (which is $\Theta(1)$ under stability and nontriviality), and get

$$\begin{aligned}
\tilde{\delta} h_t^1(\xi) &:= + \frac{\rho \chi_t (\xi_t^T \xi) dh_t^1}{\|v_t\|}, \\
\tilde{\theta}_l \tilde{\delta} h_t^l &:= \tilde{\theta}_{l-1} W_t^l \tilde{\delta} x_t^{l-1} + (\tilde{W}_t^l - W_t^l)\tilde{x}_t^{l-1} \\
&= \tilde{\theta}_{l-1} \left(W_0^l \tilde{\delta} x_t^{l-1} + \sum_{s=1}^{t} (W_s^l - W_{s-1}^l) \tilde{\delta} x_t^{l-1} \right) + (\tilde{W}_t^l - W_t^l)\tilde{x}_t^{l-1} \\
&= \tilde{\theta}_{l-1} \left(W_0^l \tilde{\delta} x_t^{l-1} - \eta(n^{1-c_l}\theta_{SAM}) \sum_{s=1}^{t} \tilde{\chi}_{s-1} \frac{(\tilde{x}_{s-1}^{l-1})^T \tilde{\delta} x_t^{l-1}}{n} dh_{SAM,s-1}^l \right)
\end{aligned}$$

$$+\rho\tilde{\theta}_{W^l}\frac{\chi_t}{\|v_t\|}\frac{(x_t^{l-1})^T\tilde{x}_t^{l-1}}{n}dh_t^l,$$

which defines a NonLin operation with the vectors $W_0^l\tilde{\delta}x_t^{l-1}, dh_{SAM,0}^l, \ldots, dh_{SAM,t-1}^l, dh_t^l$, and where we can now define the definitive scalings $\tilde{\theta}_1 := \tilde{\theta}_{W^1} := n^{-(d+d_1)}\theta_\nabla = n^{-(\min(b_{L+1},c_{L+1})+d+d_1)}$, $\tilde{\theta}_{W^l} := n^{1-(d+d_l)}\theta_\nabla = n^{-(\min(b_{L+1},c_{L+1})+d+(d_l-1))}$ and $\tilde{\theta}_l = \max(\tilde{\theta}_{l-1}, n^{1-c_l}\theta_{SAM}\tilde{\theta}_{l-1}, \tilde{\theta}_{W^l}) = \max_{m=1}^l \tilde{\theta}_{W^m} = n^{-\tilde{r}_l}$ with

$$\tilde{r}_l := \min(b_{L+1}, c_{L+1}) + d + \min_{m=1}^{l}(d_m - \mathbb{I}(m \neq 1)),$$

where we used that $n^{1-c_l}\theta_{SAM} \leq 1$ due to $r \geq 0$ for stability and $\tilde{x}_t^{l-1} = \Theta(1)$ due to perturbation stability. Perturbation stability of the hidden layer (pre-)activations $\tilde{\delta}h^l, \tilde{\delta}x^l = O(1)$ for all $l \in [L]$ holds if and only if $\tilde{r} := \tilde{r}_L \geq 0$ since $\tilde{r}_l \geq \tilde{r}_L$ for all $l \leq L$.

The activation perturbations $\tilde{\delta}x_t^l$ and the perturbation of the output function $\tilde{\delta}f_t$ can be defined exactly as in the first backward pass,

$$\begin{aligned}
\tilde{\delta}x_t^l :=& \quad \tilde{\theta}_l^{-1}(\phi(h_t^l + \tilde{\theta}_l\tilde{\delta}h_t^l) - \phi(h_t^l)),\\
\tilde{\delta}f_t :=& \quad \tilde{W}_t^{L+1}\tilde{x}_t^L - W_t^{L+1}x_t^L = \tilde{\theta}'_{L+1}\frac{\tilde{\delta}W_t^{L+1}\tilde{x}_t^L}{n} + \tilde{\theta}'_{L\nabla}\frac{\hat{W}_t^{L+1}\tilde{\delta}x_t^L}{n},
\end{aligned}$$

with $\tilde{\delta}W_t^{L+1} := \frac{\rho\,\chi_t\,x_t^L}{\|v_t\|}$ and the same scalings $\tilde{\theta}_l$, $\tilde{\theta}'_{L+1} = n^{1-(d+d_{L+1})}$ and $\tilde{\theta}'_{L\nabla} = n\theta_\nabla\tilde{\theta}_L = n^{1-\min(b_{L+1},c_{L+1})-\tilde{r}}$ since $W_t^{L+1} = W_0^{L+1} + \Delta W_t^{L+1} = \max(n^{-b_{L+1}}, n^{-c_{L+1}})$, which yields the slightly stronger constraint (than in the first backward pass) $\min(b_{L+1}, c_{L+1}) + \tilde{r} \geq 1$ for perturbation stability and either $\tilde{\theta}'_{L+1} = 1$ or $\min(b_{L+1}, c_{L+1}) + \tilde{r} = 1$ for perturbation nontriviality.

SAM Update. For each $l \in \{1, \ldots, L\}$, as in the first backward pass, we get

$$dx_{SAM,t}^L := \hat{W}_t^{L+1} + \tilde{\theta}_{(L+1)/\nabla}\,\tilde{\delta}W_t^{L+1},$$

with scaling $\theta_{SAM} = n^{-\min(b_{L+1},c_{L+1},d_{L+1}+1/2)}$ as well as

$$dh_{SAM,t}^l := dx_{SAM,t}^l \odot \phi'(\tilde{h}_t^l).$$

For $dx_{SAM,t}^l$ we again use a telescope sum over the weight changes,

$$\begin{aligned}
dx_{SAM,t}^{l-1} :=& \quad (\tilde{W}_t^l)^T dh_{SAM,t}^l = (W_0^l + \theta_{W^l}\sum_{s=1}^t \delta W_s^l + \tilde{\theta}_{W^l}\tilde{\delta}W_t^l)^T dh_{SAM,t}^l\\
=& \quad (W_0^l)^T dh_{SAM,t}^l - \eta(n^{1-c_l}\theta_{SAM})\sum_{s=1}^t \tilde{\chi}_{s-1}\frac{(dh_{SAM,s-1}^l)^T dh_{SAM,t}^l}{n}\tilde{x}_{s-1}^{l-1}\\
&+\rho(n^{1/2-d_l}\theta_\nabla)\frac{\chi_t}{\|v_t\|}\frac{(dh_t^l)^T dh_{SAM,t}^l}{n}x_t^{l-1},
\end{aligned}$$

which defines a NonLin operation in the vectors $(W_0^l)^T dh_{SAM,t}^l, \tilde{x}_0^{l-1}, \ldots, \tilde{x}_{t-1}^{l-1}, x_t^{l-1}$ and everything else treated as scalars. Note that the scalings remain θ_{SAM}, since $\nabla_{x_t^{l-1}} f|_{\tilde{W}_t} = \Theta(\max(\theta_{SAM}, n^{1-c_l}\theta_{SAM}^2, n^{1/2-d_l}\theta_\nabla\theta_{SAM})) = \Theta(\theta_{SAM})$ under stability, nontriviality, perturbation stability and perturbation nontriviality.

Finally define the loss derivative on the perturbed output function

$$\tilde{\chi}_t := \mathcal{L}'(\tilde{W}_t^{L+1}\tilde{x}_t^L, y_t),$$

and compute the normalized change in W^{L+1},

$$\delta W_{t+1}^{L+1} := -\eta\tilde{\chi}_t\tilde{x}_t^L.$$

E.2 The infinite-width limit

In this section, we apply the Master Theorem's computation rules to derive the marginal distributions Z corresponding to the vectors of the program constructed above. According to the Master Theorem, each such vector z will have roughly iid coordinates distributed like Z^z in the large n limit.

We assume stability holds, so that $\theta \to \mathring{\theta} \in \{0, 1\}$ for all scalars θ in the program.

For the first forward pass, we have

$$Z^{h_0^1(\xi)} = \xi Z^{W_0^1}, \qquad Z^{x_0^l(\xi)} = \phi(Z^{h_0^l(\xi)}), \qquad Z^{h_0^{l+1}(\xi)} = Z^{W_0^{l+1} x_0^l(\xi)}.$$

If $b_{L+1} > 1/2$ then $\mathring{f}_0 = 0$, otherwise if $b_{L+1} = 1/2$ then $\mathring{f}_0$ converges to a nontrivial Gaussian. For the details we refer to Appendix H.4.1 in Yang and Hu (2021), as at initialization their results still hold here.

For the first SGD backward pass, we have

$$Z^{dx_0^L(\xi)} = Z^{\hat{W}_0^{L+1}}, \qquad Z^{dh_0^l(\xi)} = Z^{dx_0^l(\xi)} \phi'(Z^{h_0^l(\xi)}), \qquad Z^{dx_0^{l-1}(\xi)} = Z^{(W_0^l)^T dh_0^l(\xi)},$$

where $\dot{Z}^{dx_0^l(\xi)} = 0$ and $Z^{dx_0^l(\xi)} = \hat{Z}^{dx_0^l(\xi)}$ for all $\xi \in \mathcal{X}$.

For general $t > 0$, we have

$$\begin{aligned}
Z^{dx_t^L(\xi)} &= Z^{\hat{W}_t^{L+1}}, \\
Z^{dh_t^l(\xi)} &= Z^{dx_t^l(\xi)} \phi'(Z^{h_t^l(\xi)}), \\
Z^{dx_t^{l-1}(\xi)} &= Z^{(W_0^l)^T dh_t^l(\xi)} - \eta \mathring{\theta}_{W^l} \sum_{s=1}^{t} \mathring{\chi}_{s-1} \mathbb{E}[Z^{dh^l_{SAM,s-1}} Z^{dh_t^l}] Z^{\tilde{x}_{s-1}^{l-1}},
\end{aligned}$$

where $\mathring{\chi}_s = \mathcal{L}'(\mathring{f}_s(\xi_s), y_s)$ for $s < t$, and $Z^{(W_0^l)^T dh_t^l(\xi)}$ is a $\Theta(1)$ random variable distributed as

$$Z^{(W_0^l)^T dh_t^l(\xi)} = \hat{Z}^{(W_0^l)^T dh_t^l(\xi)} + \sum_{v \in \mathcal{V}:\ W_0^l v \in \mathcal{V}} Z^v \, \mathbb{E} \frac{\partial Z^{dh_t^l(\xi)}}{\partial \hat{Z}^{W_0^l v}}.$$

For all $t \geq 0$, the limit of the gradient norm is given by

$$\|\mathring{v}\| = \mathring{\chi}_t \left(\mathring{\theta}^2_{\|v^1\|} \mathbb{E}[Z^{(dh_t^1)^2}](\xi_t^T \xi_t) + \sum_{l=2}^{L} \mathring{\theta}^2_{\|v^l\|} \mathbb{E}[Z^{(dh_t^l)^2}] \mathbb{E}[Z^{(x_t^{l-1})^2}] + \mathring{\theta}^2_{\|v^{L+1}\|} \frac{(x_t^L)^T x_t^L}{n} \right)^{1/2}, \tag{E.3}$$

where $\mathring{\chi}_t = \mathcal{L}'(\mathring{f}_t(\xi_t), y_t)$, $\theta^2_{\|v^1\|} := n^{1-2d_1} \theta^2_\nabla$, $\theta^2_{\|v^l\|} := n^{2-2d_l} \theta^2_\nabla$ for $l \in [2, L]$ and $\theta^2_{\|v^{L+1}\|} := n^{1-2d_{L+1}}$, and where $\mathring{\theta}^2_{\|v^{L+1}\|} = 1$ if and only if $d_{L+1} = 1/2$ and $\mathring{\theta}^2_{\|v^{L+1}\|} = 0$ if and only if $d_{L+1} > 1/2$, while $\mathring{\theta}^2_{\|v^l\|} = 1$ if and only if $2d_l = 1 + \mathbb{I}(l > 1) - 2\min(b_{L+1}, c_{L+1})$ and $\mathring{\theta}^2_{\|v^l\|} = 0$ if and only if $2d_l > 1 + \mathbb{I}(l > 1) - 2\min(b_{L+1}, c_{L+1})$.

For the last-layer weight perturbations (for $\theta_\nabla \geq \tilde{\theta}_{L+1}$, else $Z^{\hat{\tilde{W}}_t^{L+1}} = Z^{\tilde{\delta} W_t^{L+1}}$) we have

$$Z^{\hat{\tilde{W}}_t^{L+1}} = Z^{\hat{W}_t^{L+1}} + \mathring{\tilde{\theta}}_{(L+1)/\nabla} Z^{\tilde{\delta} W_t^{L+1}}, \qquad Z^{\tilde{\delta} W_t^{L+1}} = \frac{\rho \, \mathring{\chi}_t}{\|\mathring{v}\|} Z^{x_t^L}.$$

Note that $\mathring{\chi}_t$ cancels itself out and we purely get a perturbation in distribution $Z^{x_t^L}$ scaled to have standard deviation ρ.

For all $t \geq 0$ and $l \in [1, L]$, we have

$$Z^{\tilde{h}_t^l} = Z^{h_t^l} + \mathring{\tilde{\theta}}_l Z^{\tilde{\delta} h_t^l}, \qquad Z^{\tilde{x}_t^l} = Z^{x_t^l} + \mathring{\tilde{\theta}}_l Z^{\tilde{\delta} x_t^l},$$

where for $l = 1$,

$$Z^{\tilde{\delta} h_t^1(\xi)} = + \frac{\rho \mathring{\chi}_t (\xi_t^T \xi)}{\|\mathring{v}\|} Z^{dh_t^1}.$$

If $\mathring{\tilde{\theta}}_l = 0$, then

$$Z^{\tilde{\delta}x_t^l} = \phi'(Z^{h_t^l}) Z^{\tilde{\delta}h_t^l},$$

otherwise $\mathring{\tilde{\theta}}_l = 1$ and

$$Z^{\tilde{\delta}x_t^l} = \phi(Z^{\tilde{h}_t^l}) - \phi(Z^{h_t^l}).$$

For $l \geq 2$, we have

$$\begin{aligned} Z^{\tilde{\delta}h_t^l} = \quad & \mathring{\tilde{\theta}}_{(l-1)/l}\, Z^{W_0^l \tilde{\delta}x_t^{l-1}} - \eta \mathring{\theta}_{W^l(\tilde{l}-1)/\tilde{l}} \sum_{s=1}^{t} \mathring{\tilde{\chi}}_{s-1} \mathbb{E}[Z^{\tilde{x}_{s-1}^{l-1}} Z^{\tilde{\delta}x_t^{l-1}}] Z^{dh_{SAM,s-1}^l} \\ & + \rho \mathring{\tilde{\theta}}_{W^l/l} \frac{\mathring{\chi}_t}{\|\mathring{v}\|} \mathbb{E}[Z^{x_t^{l-1}} Z^{\tilde{x}_t^{l-1}}] Z^{dh_t^l}, \end{aligned}$$

where $\tilde{\theta}_{(l-1)/l} = \frac{\tilde{\theta}_{l-1}}{\tilde{\theta}_l}$, $\theta_{W^l(\tilde{l}-1)/\tilde{l}} = \frac{\theta_{W^l}\tilde{\theta}_{l-1}}{\tilde{\theta}_l}$ and $\tilde{\theta}_{W^l/l} = \frac{\tilde{\theta}_{W^l}}{\tilde{\theta}_l}$, and $Z^{W_0^l \tilde{\delta}x_t^{l-1}}$ has the decomposition

$$Z^{W_0^l \tilde{\delta}x_t^{l-1}} = \hat{Z}^{W_0^l \tilde{\delta}x_t^{l-1}} + \sum_{v \in \mathcal{V}:\ (W_0^l)^T v \in \mathcal{V}} Z^v\, \mathbb{E} \frac{\partial Z^{\tilde{\delta}x_t^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T v}}.$$

The perturbed output function has the limit $\mathring{\tilde{f}}_t := \mathring{f}_t + \mathring{\tilde{\delta}} f_t$ with

$$\mathring{\tilde{\delta}} f_t := \mathring{\tilde{\theta}}'_{L+1} \mathbb{E}[Z^{\tilde{\delta}W_t^{L+1}} Z^{\tilde{x}_t^L}] + \mathring{\tilde{\theta}}'_{L\nabla} \mathbb{E}[Z^{\hat{W}_t^{L+1}} Z^{\tilde{\delta}x_t^L}],$$

so that we can define $\mathring{\tilde{\chi}}_t = \mathcal{L}'(\mathring{\tilde{f}}_t(\xi_t), y_t)$ or equivalently $\mathring{\tilde{\chi}}_t = \mathcal{L}'(\mathring{\tilde{\theta}}_{L+1} \mathring{\tilde{\theta}}_L \mathbb{E}[Z^{\hat{\tilde{W}}_t^{L+1}} Z^{\tilde{x}_t^L}], y_t)$.
For the SAM gradients we have

$$\begin{aligned} Z^{dx_{SAM,t}^L} = \quad & Z^{\hat{W}_t^{L+1}} + \mathring{\tilde{\theta}}_{(L+1)/\nabla}\, Z^{\tilde{\delta}W_t^{L+1}}, \\ Z^{dh_{SAM,t}^l} = \quad & Z^{dx_{SAM,t}^l} \cdot \phi'(Z^{\tilde{h}_t^l}) \\ Z^{dx_{SAM,t}^{l-1}} = \quad & Z^{(W_0^l)^T dh_{SAM,t}^l} - \eta \mathring{\theta}_{W^l} \sum_{s=1}^{t} \mathring{\tilde{\chi}}_{s-1} \mathbb{E}[Z^{dh_{SAM,s-1}^l} Z^{dh_{SAM,t}^l}] Z^{\tilde{x}_{s-1}^{l-1}} \\ & + \rho \mathring{\tilde{\theta}}_{W^l} \frac{\mathring{\chi}_t}{\|\mathring{v}\|} \mathbb{E}[Z^{dh_t^l} Z^{dh_{SAM,t}^l}] Z^{x_t^{l-1}}, \end{aligned}$$

where $Z^{(W_0^l)^T dh_{SAM,t}^l}$ is given by

$$Z^{(W_0^l)^T dh_{SAM,t}^l} = \hat{Z}^{(W_0^l)^T dh_{SAM,t}^l} + \sum_{v \in \mathcal{V}:\ W_0^l v \in \mathcal{V}} Z^v\, \mathbb{E} \frac{\partial Z^{dh_{SAM,t}^l}}{\partial \hat{Z}^{W_0^l v}}.$$

Now SAM's (pre-)activation updates are given by

$$Z^{h_t^l} = Z^{h_0^l} + \mathring{\theta}_l (Z^{\delta h_1^l} + \cdots + Z^{\delta h_t^l}), \qquad Z^{x_t^l} = Z^{x_0^l} + \mathring{\theta}_l (Z^{\delta x_1^l} + \cdots + Z^{\delta x_t^l}),$$

with, for $l \in [2, L]$,

$$\begin{aligned} Z^{\delta h_t^1(\xi)} = \quad & -\eta \mathring{\tilde{\chi}}_{t-1} (\xi_{t-1}^T \xi) Z^{dh_{SAM,t-1}^1}, \\ Z^{\delta h_t^l} = \quad & \mathring{\theta}_{(l-1)/l} \left(Z^{W_0^l \delta x_t^{l-1}} - \eta \mathring{\theta}_{W^l} \sum_{s=1}^{t-1} \mathring{\tilde{\chi}}_{s-1} \mathbb{E}[Z^{\tilde{x}_{s-1}^{l-1}} Z^{\delta x_t^{l-1}}] Z^{dh_{SAM,s-1}^l} \right) \\ & - \eta \mathring{\theta}_{W^l/l} \mathring{\tilde{\chi}}_{t-1} \mathbb{E}[Z^{\tilde{x}_{t-1}^{l-1}} Z^{x_t^{l-1}}] Z^{dh_{SAM,t-1}^l}, \end{aligned}$$

where $\theta_{(l-1)/l} := \theta_{l-1}/\theta_l$, $\theta_{W^l/l} := \theta_{W^l}/\theta_l$ and $Z^{W_0^l \delta x_t^{l-1}}$ has the decomposition

$$Z^{W_0^l \delta x_t^{l-1}} = \hat{Z}^{W_0^l \delta x_t^{l-1}} + \sum_{v \in \mathcal{V}:\ (W_0^l)^T v \in \mathcal{V}} Z^v\, \mathbb{E} \frac{\partial Z^{\delta x_t^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T v}}.$$

If $\mathring{\theta}_l = 0$, then

$$Z^{\delta x_t^l} = \phi'(Z^{h_{t-1}^l}) Z^{\delta h_t^l},$$

otherwise $\mathring{\theta}_l = 1$ and

$$Z^{\delta x_t^l} = \phi(Z^{h_t^l}) - \phi(Z^{h_{t-1}^l}).$$

The last-layer SAM weight update is given by

$$Z^{\hat{W}_t^{L+1}} = Z^{\hat{W}_0^{L+1}} + \mathring{\theta}_{L+1/\nabla}(Z^{\delta W_1^{L+1}} + \cdots + Z^{\delta W_t^{L+1}}),$$

with $Z^{\delta W_t^{L+1}} = -\eta \mathring{\tilde{\chi}}_{t-1} Z^{\tilde{x}_{t-1}^L}$.

For $t > 0$, the SAM function update is given by

$$\mathring{f}_t = \mathring{f}_0 + \mathring{\delta} f_1 + \cdots + \mathring{\delta} f_t,$$

with $\mathring{\delta} f_t = \mathring{\theta}'_{L+1} \mathbb{E}[Z^{\delta W_t^{L+1}} Z^{x_t^L}] + \mathring{\theta}'_{L\nabla} \mathbb{E}[Z^{\hat{W}_{t-1}^{L+1}} Z^{\delta x_t^L}]$.

E.3 Concluding the proof of all main results

After writing out the NE⊗OR⊤program and its limit, as well as tracking all scalings, the main results stated in Appendix D all follow from the Tensor Program Master Theorem and from the characterization results in Yang and Hu (2021) in the following way.

Formally Yang and Hu (2021) show feature learning for SGD with small enough learning rate $\eta > 0$ by proving $\partial_\eta^2 \mathbb{E}(Z^{x_1^L(\xi_0)})^2 \neq 0$ at $\eta = 0$, and they show that learning does not occur in the kernel regime by showing $\partial_\eta^3 \mathring{f}_1 \neq 0$, hence $\mathring{f}_1 - \mathring{f}_0$ is not linear in η.

Both $\mathbb{E}(Z^{x_1^L(\xi_0)})^2$ and $\mathring{f}_1$ are defined via NE⊗OR⊤computations and can be written as a composition of additions, multiplications, the expectation operator, applications of ϕ and ϕ', overall applications of infinitely differentiable, pseudo-Lipschitz functions to (Gaussian) random variables, η and ρ. Consequently $\mathbb{E}(Z^{x_1^L(\xi_0)})^2$ and $\mathring{f}_1$ are infinitely often differentiable as a function of both η and ρ, where differentiating the expectation operator is covered in Yang and Hu (2021, Lemma H.39). Since Yang and Hu (2021) cover the case $\rho = 0$, their proofs immediately show the correctness of the derived scalings for SAM as long as $\eta > 0$ and $\rho > 0$ are chosen small enough. Both the gradient evaluation for the perturbation as well as the gradient evaluation for the updates stay arbitrarily close to those of SGD if $\rho > 0$ is chosen small enough. The conditions for stability, nontriviality, feature learning, perturbation nontriviality and effective perturbations now follow from considering the respective scaling.

E.3.1 Proof of Theorem D.2

A *bcd*-parameterization is stable if and only if all scalings in the Tensor Program have the limit $\mathring{\theta} \in \{0, 1\}$, where $\mathring{\theta} = 1$ is required for activations at initialization (for which nothing changes compared to SGD). Potential cancellations are taken care of for sufficiently small $\eta > 0$ and $\rho > 0$ by the argument above. Now collecting all constraints that are already stated in the Tensor Program formulation at the respective step concludes the proof.

E.3.2 Proof of Theorem D.3

A stable *bcd*-parameterization is nontrivial if and only if $\mathring{f}_t = \Theta(1)$ if and only if $\mathring{\theta}'_{L+1} = 1$ or $\mathring{\theta}'_{L\nabla} = 1$.

E.3.3 Proof of Theorem D.4

A stable *bcd*-parametrization is feature learning in layer l if and only if the feature update scaling $\mathring{\theta}_l = 1$ where

$$\theta_l = n^{-r_l}, \quad r_l := \min(b_{L+1}, c_{L+1}, d_{L+1} + 1/2) + \min_{m=1}^{l}(c_m - \mathbb{I}(m \neq 1)).$$

Hence a stable *bcd*-parametrization is feature learning in layer l if and only if $r_l = 0$.

Since for all $l_1 \leq l_2$, it holds that $r_{l_1} \geq r_{l_2} \geq 0$, we get the equivalence for any $l_0 \in [L]$: A stable *bcd*-parametrization is feature learning in layer l_0 if and only if it is feature learning in layer l for all $l \geq l_0$ if and only if $r_{l_0} = 0$.

E.3.4 Proof of Theorem D.6

Given a stable *bcd*-parametrization, perturbation triviality is fulfilled if and only if $\mathring{\tilde{\theta}}'_{L+1} = 0$ and $\mathring{\tilde{\theta}}'_{L\nabla} = 0$, where $\tilde{\theta}'_{L+1} = n^{1/2-d_{L+1}}$ and $\tilde{\theta}'_{L\nabla} = n\theta_\nabla \tilde{\theta}_L = n^{1-\min(b_{L+1},c_{L+1})-\tilde{r}}$, hence if and only if $d_{L+1} > 1/2$ and $\min(b_{L+1}, c_{L+1}) + \tilde{r} > 1$.

In that case, $\mathring{\tilde{f}}_t = \mathring{f}_t$, but $\mathring{f}_t$ may still be affected by non-vanishing SAM perturbations in δW_t^{L+1} and δx_t^L. Only when all SAM perturbations vanish are we effectively only using SGD. By definition, the perturbation scale in the l-th layer vanishes if and only if $\mathring{\tilde{\theta}}_l = 0$, where $\tilde{\theta}_l = n^{-\tilde{r}_l}$ with $\tilde{r}_l = \min(b_{L+1}, c_{L+1}) + 1/2 + \min_{m=1}^{l}(d_m - \mathbb{I}(m \neq 1))$, hence if and only if $\tilde{r}_l > 0$. Since $\tilde{r}_l \geq \tilde{r}_L = \tilde{r}$ for all $l \leq L$, we get $\mathring{\tilde{\theta}}_l = 0$ for all $l \in [L]$ if and only if $\tilde{r} > 0$. Similarly, for any reference layer $l_0 \in [L]$, we get $\mathring{\tilde{\theta}}_l = 0$ for all $l \leq l_0$ if and only if $\tilde{r}_{l_0} > 0$. In words, for any $l_0 \in [L]$, we have vanishing perturbations in layer l_0 if and only if we have vanishing perturbations until layer l_0 if and only if $\tilde{r}_{l_0} > 0$.

Altogether, a stable *bcd*-parametrization has vanishing perturbations if and only if $\tilde{r} > 0$, $d_{L+1} > 1/2$ and $\min(b_{L+1}, c_{L+1}) + \tilde{r} > 1$. This case reduces to the results in Yang and Hu (2021) in the limit. Since stability requires $c_{L+1} \geq 1$ and $\tilde{r} \geq 0$, we can rewrite the equivalence conditions as $d_{L+1} \geq 1/2$ and $\tilde{r} > \max(0, 1 - b_{L+1})$.

E.3.5 Proof of Theorem D.8

Recall $\tilde{\theta}_{W^1} := n^{-(d+d_1)}\theta_\nabla$, $\tilde{\theta}_{W^l} := n^{1-(d+d_l)}\theta_\nabla$ and, for the last layer $\tilde{\theta}_{W^{L+1}} := n^{-(d+d_{L+1})}$.

As opposed to perturbation nontriviality, we are not only interested in $\tilde{\theta}_l = \max(\tilde{\theta}_{l-1}, \tilde{\theta}_{W^l}) = \max_{m=1}^{l} \tilde{\theta}_{W^m} \to 1$, but in a non-vanishing contribution of the perturbations in layer l, i.e. $\mathring{\tilde{\theta}}_{W^l} = 1$ or, for the last layer, $\mathring{\tilde{\theta}}_{L+1} = 1$.

E.3.6 Proof of Theorem D.9

The limit of the gradient norm is defined as a NE⊗OR⊤ program scalar (E.3). Note that for $b_{L+1} > 1/2$, the last-layer scaling strictly dominates all other scalings leading to the simplified gradient norm formula.

Now consider an arbitrary stable choice of layerwise initialization variances $\{b_l\}_{l\in[L+1]}$ and learning rates $\{c_l\}_{l\in[L+1]}$. To fulfill the gradient norm constraints (D.1), we have to choose $d_l = C = 1/2$ for all $l \in [L+1]$, because stability requires $\min(b_{L+1}, c_{L+1}) \geq 1/2$. Now stability of the output function perturbations requires $d \geq 1/2$, where $d > 1/2$ yields vanishing perturbations and $d = 1/2$ yields effective last-layer SAM through the term $\tilde{\delta} W_t^{L+1} \tilde{x}_t^L$. After choosing $d \geq 1/2$, we get $\tilde{r} \geq \min(b_{L+1}, c_{L+1}) \geq 1/2 > 0$ which implies vanishing perturbations in all hidden layers.

E.3.7 Proof of Proposition D.10

To achieve non-vanishing gradient norm contribution of the last layer in (D.1), we need to choose $d_{L+1} = 1/2$, which requires $d \geq 1/2$ for stability of the output function perturbations. Achieving non-vanishing gradient norm contributions of all layers requires $d_1 = 1/2 - \min(b_{L+1}, c_{L+1})$ and $d_l = 1 - \min(b_{L+1}, c_{L+1})$ for $l \in [2, L]$, which results in $\tilde{r} = d \geq 1/2 > 0$ which implies vanishing perturbations in all hidden layers.

E.3.8 Proof of Theorem D.11

Given a stable *bcd*-parametrization, we know $d + d_{L+1} \geq 1$, so that the feature learning constraint r is not affected by any stable choice of $d \cup \{d_l\}_{l\in[L+1]}$. The maximal stable choice of layerwise initialization variances $\{b_l\}_{l\in[L+1]}$ and learning rates $\{c_l\}_{l\in[L+1]}$ that constitute μP is therefore unaffected by the perturbation scalings $d \cup \{d_l\}_{l\in[L+1]}$.

Stability of the output function perturbations requires $b_{L+1} + \tilde{r} \geq 1$. Hence if $b_{L+1} < 1$, then $\tilde{r} \geq 1 - b_{L+1} > 0$, which implies vanishing perturbations in all hidden layers.

From now on consider $b_{L+1} \geq 1$. Recall $c_\nabla := \min(b_{L+1}, c_{L+1})$. In μP, $c_\nabla = 1$, but effective perturbations in all layers can be achieved more generally for $c_\nabla \geq 1$. Choosing $d_1 = 1/2 - c_\nabla$ saturates the gradient norm constraint (D.1). To reach effective perturbations already in the first layer $\tilde{r}_1 = c_\nabla + d + d_1 = 0$, we need $d = -1/2$. For perturbation stability and last-layer effective perturbations, we need $d + d_{L+1} = 1$ which requires $d_{L+1} = 3/2$. Achieving perturbation stability and effective perturbations in all hidden layers requires $\tilde{\theta}_{W^l} = 1$ which is equivalent to $c_\nabla + d + d_l - \mathbb{I}(l \neq 1) = 0$. For $l \in [2, L]$, we therefore need $d_l = 3/2 - c_\nabla$. This choice of $\{d_l\}_{l \in [L+1]}$ achieves effective perturbations in all layers.

To show uniqueness we iterate through all possibilities of saturating the norm bound constraint (D.1). We have considered the cases $d_{L+1} = 1/2$ in (b) leading to vanishing perturbations in all hidden layers and $d_1 = 1/2 - c_\nabla$ in (c) with only one choice for effective perturbations in all layers. Lastly consider $d_l = 1 - c_\nabla$ for $l \in [2, L]$ for non-vanishing gradient contribution of the hidden layers. Note that all hidden layers play the same role in all relevant constraints. Effective perturbations in any hidden layer $l \in [2, L]$ requires $\tilde{\theta}_{W^l} = 1$ for which we need $d = 0$. But then, as $d_1 \geq 1/2 - c_\nabla$, it holds that $\tilde{r}_1 \geq 1/2$ implying vanishing perturbations in the first layer. This shows the uniqueness of (1).

For the gradient norm statements, note that the gradient norm $\|v_t\|$ can be written as a NE⊗OR⊤computation rule (E.2) where the layer scalings in this parameterization are $\Theta(1)$ for the input layer, $\Theta(n^{-1/2})$ for hidden layers and $\Theta(n^{-1})$ for the output layer. Now the Tensor Program master theorem immediately implies the result.

E.3.9 Proof of Proposition D.12

Perturbation nontriviality with respect to any hidden layer is equivalent to $\tilde{r} = 0$. Since $\min(b_{L+1}, c_{L+1}) \leq 1$, we get $\min(b_{L+1}, c_{L+1}) + \tilde{r} \leq 1$. Since stability requires $\min(b_{L+1}, c_{L+1}) + \tilde{r} \geq 1$, we get $\min(b_{L+1}, c_{L+1}) + \tilde{r} = 1$, which implies perturbation nontriviality with respect to the output.

E.3.10 Proof of Proposition D.13

The constraint is the same constraint as in Theorem D.8, which implies effective perturbations in the first layer. Now $\tilde{r}_l \leq \tilde{r}_1 = 0$ implies perturbation nontriviality in all hidden layers due to Theorem D.6.

E.4 Analytic expression of the features after first SAM update

Below we state the analytic expression of the first SAM update, but leave a closer analysis of its fine-grained dynamics in comparison to SGD to future work. Before looking into the effective perturbation regime, we restate Lemma H.37 in Yang and Hu (2021) with a more detailed proof.

First, we define $\ell \in [L]$ as the unique index that satisfies $\theta_L = \cdots = \theta_\ell = 1 > \theta_{\ell-1} \geq \cdots \geq \theta_1$. In words, ℓ is the first layer in which feature learning occurs. Analogously, we define $\tilde{\ell} \in [L]$ as the unique index that satisfies $1 = \frac{\tilde{\theta}_L}{\theta_L} = \cdots = \frac{\tilde{\theta}_{\tilde{\ell}}}{\theta_L} > \frac{\tilde{\theta}_{\tilde{\ell}-1}}{\theta_L} \geq \cdots \geq \frac{\tilde{\theta}_1}{\theta_L}$.

Lemma E.2 (Features after first SGD step). *Defining $Z_t^l := Z^{h_t^l}$, $\gamma^l(\eta) = \mathbb{E}\phi(Z_0^l)\phi(Z_1^l)$ for $l \geq 1$, $\gamma^0 = \xi_0^T \xi$ and $\gamma_{11}^l(\eta) = \mathbb{E}\phi'(Z_0^l)\phi'(Z_1^l)$, we have*

$$Z_1^{\ell-1} = Z_0^{\ell-1}, \ldots, Z_1^1 = Z_0^1,$$

and, for all $l \geq \ell$,

$$Z_1^l = Z_0^l + \mathbb{I}_{l>\ell} \hat{Z}^{W_0^l \delta x_1^{l-1}} + \eta \beta^l Z^{dx_0^l} \phi'(Z_0^l),$$

where β^l is defined recursively by

$$\beta^l = \beta^l(\eta) = -\mathring{\chi}_0 \gamma^{l-1}(\eta) + \beta^{l-1}(\eta)\gamma_{11}^{l-1}(\eta),$$

with $\beta^{\ell-1} = 0$. Note that $\beta^l(0) < 0$ for all $l \geq \ell$.

Proof. By the defining infinite-width equations, assuming $\mathring{\theta}_{W^l/l} = 1$ (so minimal stable choice of c_l),

$$Z_1^l = Z_0^l + \mathring{\theta}_{(\ell-1)/\ell} Z^{W_0^l \delta x_1^{l-1}} - \eta \mathring{\chi}_0 \gamma^{l-1} Z^{dx_0^l} \phi'(Z_0^l).$$

At $l = \ell$, we get $\mathring{\theta}_{(\ell-1)/\ell} = 0$, whereas for $l > \ell$ we get $\mathring{\theta}_{(l-1)/l} = 1$, which results in $\mathring{\theta}_{(\ell-1)/\ell} = \mathbb{I}_{l>\ell}$.

Now, for $l > \ell$, the second term decomposes into $\hat{Z}^{W_0^l \delta x_1^{l-1}}$ and

$$\dot{Z}^{W_0^l \delta x_1^{l-1}} = Z^{dh_0^l} \mathbb{E} \frac{\partial Z^{\delta x_1^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T dh_0^l}}.$$

Since by induction hypothesis,

$$Z^{\delta x_1^{l-1}} = \phi(Z_1^{l-1}) - \phi(Z_0^{l-1}) = \phi\left(Z_0^{l-1} + \mathbb{I}_{l>\ell} \hat{Z}^{W_0^l \delta x_1^{l-1}} + \eta \beta^{l-1} Z^{dx_0^{l-1}} \phi'(Z_0^{l-1})\right) - \phi(Z_0^{l-1}),$$

where $Z^{dx_0^{l-1}} = Z^{(W_0^l)^T dh_0^l}$ is the only dependence on $\hat{Z}^{(W_0^l)^T dh_0^l}$, we get

$$\frac{\partial Z^{\delta x_1^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T dh_0^l}} = \phi'(Z_1^{l-1}) \eta \beta^{l-1} \phi'(Z_0^{l-1}).$$

Plugging the derivative back into the defining equation and noticing that $Z^{dh_0^l} = Z^{dx_0^l} \phi'(Z_0^l)$ concludes the proof. □

An analogous analysis for the perturbation at initialization shows.

Lemma E.3 (Feature perturbation at initialization). *The perturbation trivial layers fulfill*

$$Z^{\tilde{h}_0^{\tilde{\ell}-1}} = Z^{h_0^{\tilde{\ell}-1}}, \dots, Z^{\tilde{h}_0^1} = Z^{h_0^1},$$

and, for all $l \geq \tilde{\ell}$,

$$Z^{\tilde{h}_0^l} = Z^{h_0^l} + \mathbb{I}_{l>\tilde{\ell}} \hat{Z}^{W_0^l \tilde{\delta} x_0^{l-1}} + \rho \tilde{\beta}^l Z^{dx_0^l} \phi'(Z^{h_0^l}),$$

where $\tilde{\beta}^l$ independent of η is defined recursively by

$$\tilde{\beta}^l = \tilde{\beta}^l(\rho) = \frac{\mathring{\chi}_0}{\|\mathring{\nabla} L_0\|} \mathbb{E}[\phi(Z^{h_0^{l-1}}) \phi(Z^{\tilde{h}_0^{l-1}})] + \tilde{\beta}^{l-1} \mathbb{E}[\phi'(Z^{h_0^{l-1}}) \phi'(Z^{\tilde{h}_0^{l-1}})]$$

with $\tilde{\beta}^{\tilde{\ell}-1} = 0$. Note that $\tilde{\beta}^l(0) > 0$ for all $l \geq \tilde{\ell}$.

Remark E.4. If $\tilde{\ell} = 1$, in the definition of $\tilde{\beta}^l$ replace $\mathbb{E}[\phi(Z^{h_0^{l-1}}) \phi(Z^{\tilde{h}_0^{l-1}})]$ by $\xi_0^T \xi$. ◀

Now we are ready to state the closed form expression for the first SAM update.

Lemma E.5 (Features after first SAM update). *Defining $Z_t^l := Z^{h_t^l}$ and $\tilde{Z}_t^l := Z^{\tilde{h}_t^l}$, we have*

$$Z_1^{\ell-1} = Z_0^{\ell-1}, \dots, Z_1^1 = Z_0^1,$$

and, for all $l \geq \ell$,

$$Z_1^l = Z_0^l + \mathbb{I}_{l>\ell} \hat{Z}^{W_0^l \delta x_1^{l-1}} + \eta \beta^l Z^{dx_{SAM,0}^l} \phi'(\tilde{Z}_0^l) + \eta \gamma^l Z^{dh_0^l},$$

where β^l is defined recursively by

$$\beta^l = \beta^l(\eta) = -\mathring{\tilde{\chi}}_0 \mathbb{E}[\phi(\tilde{Z}_0^{l-1}) \phi(Z_1^{l-1})] + \beta^{l-1}(\eta) \mathbb{E}[\phi'(Z_1^{l-1}) \phi'(\tilde{Z}_0^{l-1})],$$

with $\beta^{\ell-1} = 0$, and $\gamma^l = \gamma^l(\eta)$ is recursively defined by

$$\gamma^l := \beta^{l-1} \rho \tilde{\beta}^{l-1} \mathbb{E}[\phi'(Z_1^{l-1}) \phi'(Z_0^{l-1}) \phi''(\tilde{Z}_0^{l-1}) Z^{dx_{SAM,0}^{l-1}}] + \gamma^{l-1} \mathbb{E}[\phi'(Z_0^{l-1}) \phi'(Z_1^{l-1})],$$

with $\gamma^{\ell-1} = \gamma^\ell = 0$.

Remark E.6. If $\ell = 1$, in the definition of β^l replace $\mathbb{E}[\phi(\tilde{Z}_0^{l-1}) \phi(Z_1^{l-1})]$ by $(\xi_{t-1}^T \xi)$. ◀

Proof. By the defining infinite-width equations, for $l \geq \ell$, assuming $\mathring{\theta}_{W^l/l} = 1$ (so minimal stable choice of c_l),

$$Z_1^l = Z_0^l + \mathring{\theta}_{(l-1)/l} Z^{W_0^l \delta x_1^{l-1}} - \eta \mathring{\chi}_0 \mathbb{E}[\phi(\tilde{Z}_0^{l-1})\phi(Z_1^{l-1})] Z^{dx_{SAM,0}^l \phi'(\tilde{Z}_0^l)}. \tag{E.4}$$

At $l = \ell$, we get $\mathring{\theta}_{(\ell-1)/\ell} = 0$ and $\mathring{\theta}_{W^\ell/\ell} = 1$, whereas for $l > \ell$ we get $\mathring{\theta}_{(l-1)/l} = 1$ and $\mathring{\theta}_{W^l/l} = 1$ (under minimal stable choice of c_l), which results in $\mathring{\theta}_{(l-1)/l} = \mathbb{I}_{l>\ell}$. Now, for $l > \ell$, the second term decomposes into $\hat{Z}^{W_0^l \delta x_1^{l-1}}$ and $\dot{Z}^{W_0^l \delta x_1^{l-1}}$. For the rest of the proof it remains to analyse $\dot{Z}^{W_0^l \delta x_1^{l-1}}$. Since by induction hypothesis,

$$\begin{aligned} Z^{\delta x_1^{l-1}} &= \phi(Z_1^{l-1}) - \phi(Z_0^{l-1}) \\ &= \phi\left(Z_0^{l-1} + \mathbb{I}_{l>\ell} \hat{Z}^{W_0^l \delta x_1^{l-1}} + \eta \beta^{l-1} Z^{dx_{SAM,0}^{l-1} \phi'(\tilde{Z}_0^{l-1})} + \eta \gamma^{l-1} Z^{dh_0^{l-1}}\right) - \phi(Z_0^{l-1}), \end{aligned}$$

where $Z^{dx_{SAM,0}^{l-1}} = Z^{(W_0^l)^T dh_{SAM,0}^l} + \rho \mathring{\tilde{\theta}}_{W^l} \frac{\mathring{\chi}_0}{\|\nabla \tilde{L}_0\|} \mathbb{E}[Z^{dh_0^l} Z^{dh_{SAM,0}^l}] Z^{x_0^{l-1}}$ with the second term independent of $(W_0^l)^T$ and by Lemma E.3 we know $\tilde{Z}_0^{l-1} = Z_0^{l-1} + \mathbb{I}_{l-1>\tilde{\ell}} \hat{Z}^{W_0^{l-1} \tilde{\delta} x_0^{l-2}} + \rho \tilde{\beta}^{l-1} Z^{dx_0^{l-1}} \phi'(Z_0^{l-1})$, where only the last term with $Z^{dx_0^{l-1}} = Z^{(W_0^l)^T dh_0^l}$ influences $\dot{Z}^{W_0^l \delta x_1^{l-1}}$, we get

$$\dot{Z}^{W_0^l \delta x_1^{l-1}} = Z^{dh_0^l} \mathbb{E} \frac{\partial Z^{\delta x_1^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T dh_0^l}} + Z^{dh_{SAM,0}^l} \mathbb{E} \frac{\partial Z^{\delta x_1^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T dh_{SAM,0}^l}}, \tag{E.5}$$

with

$$\frac{\partial Z^{\delta x_1^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T dh_{SAM,0}^l}} = \phi'(Z_1^{l-1}) \eta \beta^{l-1} \phi'(\tilde{Z}_0^{l-1}),$$

and, using $Z^{dh_0^{l-1}} = Z^{dx_0^{l-1}} \phi'(Z_0^{l-1}) = Z^{(W_0^l)^T dh_0^l} \phi'(Z_0^{l-1})$, yields

$$\begin{aligned} \frac{\partial Z^{\delta x_1^{l-1}}}{\partial \hat{Z}^{(W_0^l)^T dh_0^l}} &= \phi'(Z_1^{l-1}) \left(\eta \beta^{l-1} Z^{dx_{SAM,0}^{l-1} \phi''(\tilde{Z}_0^{l-1})} \frac{\partial \tilde{Z}_0^{l-1}}{\partial \hat{Z}^{(W_0^l)^T dh_0^l}} + \eta \gamma^{l-1} \phi'(Z_0^{l-1})\right) \\ &= \phi'(Z_1^{l-1}) \left(\eta \beta^{l-1} Z^{dx_{SAM,0}^{l-1} \phi''(\tilde{Z}_0^{l-1})} \rho \tilde{\beta}^{l-1} \phi'(Z_0^{l-1}) + \eta \gamma^{l-1} \phi'(Z_0^{l-1})\right). \end{aligned}$$

Plugging Eq. (E.5) back into the defining equation (E.4) and noticing that $Z^{dh_{SAM,0}^l} = Z^{dx_{SAM,0}^l \phi'(\tilde{Z}_0^l)}$ as well as $Z^{dh_0^l} = Z^{dx_0^l \phi'(Z_0^l)}$ concludes the proof. □

F Generalizations and further perturbation scaling considerations

F.1 Overview over choices of d_l and d

Since for some combinations of architectures and datasets it turns out that performing SAM on a subset of layers performs better than effective perturbations in all layers (Müller et al., 2024), we would like to know how to choose d and d_l to adjust which layers should be effectively perturbed and which should have vanishing weight perturbations. In practice, simply set all perturbations that should vanish to 0 by design, and use the global scaling d and relative scalings d_l from μP^2 for the perturbed layers. This section is instead interested in a complete characterization of all possible choices of $\{d_l\}_{l \in [L+1]}$ and d. The heuristic derivation only requires the gradient norm constraints (D.1) and the perturbation stability constraints that require $\tilde{\delta} W^1 = O(1)$ and $\tilde{\delta} W^l = O(n^{-1})$ for $l > 1$ given by

$$d_l \geq \begin{cases} -c_\nabla - d & \text{if } l \text{ is input-like,} \\ 1 - c_\nabla - d & \text{if } l \text{ is hidden-like,} \\ 1 - d & \text{if } l \text{ is output-like,} \end{cases} \tag{F.1}$$

where a layer is effectively perturbed if and only if equality holds in the respective perturbation stability inequality. This heuristic claim yields the characterization of all phases of the choices of

	Effective perturbations possible			Gradient norm may be dominated by		
	input-like	hidden-like	output-like	input-like	hidden-like	output-like
$d = -1/2$	✓	✓	✓	✓	✗	✗
$d \in (-1/2, 0)$	✗	✓	✓	✓	✗	✗
$d = 0$	✗	✓	✓	✓	✓	✗
$d \in (0, 1/2)$	✗	✗	✓	✓	✓	✗
$d = 1/2$	✗	✗	✓	✓	✓	✓
$d > 1/2$	✗	✗	✗	✓	✓	✓

Table F.1: **(Characterization of perturbation scalings)** Overview over the regimes of all possible choices of d and d_l. A layer is effectively perturbed if and only d_l satisfies (F.1). At least one layer must satisfy equality in its gradient norm constraint (D.1). This table summarizes which layers can exhibit effective perturbations, and which may dominate the gradient norm, given a choice of d. The choice $d < -1/2$ results in perturbation blowup $\tilde{r} < 0$. At the critical $d = -1/2$ (respectively, $d = 0$; $d = 1/2$) a input-like (respectively hidden-like; output-like) layer is effectively perturbed if and only if it dominates the gradient norm. Consequently $d = -1/2$ implies effective perturbations in at least one input-like layer.

perturbation scalings d and d_l in Table F.1 and allows us to formulate a simple rule of how to choose d and d_l given the information which layers should be effectively perturbed, and which should have vanishing weight perturbations.

Choice of perturbation scaling from list of layers to effectively perturb. We denote the set of all layers by $\mathcal{L}$, whereas the subset of layers, which we want to effectively perturb, is denoted by $\mathcal{L}_{SAM} \subseteq \mathcal{L}$.

1. If there exists an input-like layer $l \in \mathcal{L}_{SAM}$, set $d = -1/2$. Input-like layers are effectively perturbed if and only if $d_l = 1/2 - c_\nabla$. Hidden-like (respectively, output-like) layers are effectively perturbed if and only if $d_l = 3/2 - c_\nabla$ (respectively, $d_l = 3/2$). For all layers that have vanishing weight perturbations, do not perturb these weights or choose $d_l > 1/2 - c_\nabla$ for input-like, $d_l > 3/2 - c_\nabla$ for hidden-like and $d_l > 3/2$ for output-like layers.
2. If all input-like layers should have vanishing weight perturbations but there exists a hidden-like layer $l \in \mathcal{L}_{SAM}$, set $d = 0$. Hidden-like layers are effectively perturbed if and only if $d_l = 1 - c_\nabla$. Output-like layers are effectively perturbed if and only if $d_{L+1} = 1$. For all layers that have vanishing weight perturbations, do not perturb these weights, or set $d_l > c_\nabla$ for input-like, $d_l > 1 - c_\nabla$ for hidden-like and $d_l > 1$ for output-like layers (as required by the perturbation stability and gradient norm constraints).
3. If both all input-like and all hidden-like layers have vanishing weight perturbations, but there exists some output-like layer $l \in \mathcal{L}_{SAM}$, then set $d = 1/2$. Output-like layers are effectively perturbed if and only if $d_l = 1/2$. For all layers that have vanishing weight perturbations, do not perturb these weights or set $d_l \geq 1/2 - c_\nabla$ for input-like, $d_l \geq 1 - c_\nabla$ for hidden-like and $d_l > 1/2$ for output-like layers (as required by the perturbation stability and gradient norm constraints).
4. If $\mathcal{L}_{SAM} = \emptyset$, then set $d > 1/2$ or simply perform SGD.

Example F.1 (First-layer-only effective perturbations). Instead of simply using the rule set above, we derive the necessary choice of perturbation scaling from the scaling equalities and the norm constraints (D.1). To achieve first-layer effective perturbations, but trivial weight perturbations in all other layers, we need $\tilde{\theta}_{W^1} = 1$ and $\tilde{\theta}_{W^l} = 0$, for which we will choose $\tilde{\theta}_{W^l} = n^{-1}$. This requires setting

$$d_1 = -(c_\nabla + d), \qquad d_l = 2 - c_\nabla - d, \qquad d_{L+1} = 2 - d,$$

where one of the constraints (D.1) has to be fulfilled. Plugging the above d_l-choices into (D.1) results in the constraints $d \leq -1/2$, $d \leq 1$, $d \leq 3/2$, hence choose $d = -1/2$ so that only the first layer contributes non-vanishingly to the gradient norm. Note that $\tilde{r} = 0$ and output perturbation nontriviality holds if and only if $\min(b_{L+1}, c_{L+1}) = 1$ (as in μP). We apply this perturbation scaling in Appendix H.2 to show that propagating perturbations from early layers are not enough to inherit SAM's inductive bias that leads to improved generalization performance. ◀

F.2 Other ways to introduce layerwise perturbation scaling

Before presenting alternative ways how layerwise perturbation scaling could be accomplished, let us collect desirable properties that a definition should fulfill:

- Layerwise perturbation scaling should enable stable, effective perturbations in every layer.
- The perturbation step should require at most one additional forward and backward pass in each update step.
- The adapted optimization algorithm should recover the original (SAM) algorithm when not using layerwise perturbation scaling.

We start with the simplest case where the perturbations are normalized in each layer separately.

Remark F.2 (SAM with layerwise gradient normalization). For (SAM) with layerwise gradient normalization of the perturbations

$$\boldsymbol{\varepsilon}^l = \rho_l \cdot \nabla_{W^l}\mathcal{L}/\|\nabla_{W^l}\mathcal{L}\|, \tag{LN}$$

where $\|\cdot\|$ may denote the Frobenius or the spectral norm (equivalent under limited perturbation batch size), the spectral scaling condition ($*$) immediately yields the correct layerwise perturbation scaling $\rho_l \overset{!}{=} \Theta(\sqrt{\texttt{fan\_out}/\texttt{fan\_in}})$. ◀

However, in practice, perturbations are usually globally normalized across layers, according to the GitHub repositories provided by Foret et al. (2021); Samuel (2022); Kwon et al. (2021); Andriushchenko and Flammarion (2022); Müller et al. (2024). Preliminary ablations in Appendix H.5 suggest that layer-coupled SAM with global normalization slightly outperforms SAM with layerwise gradient normalization. As our goal in this paper is to study (SAM) as applied in practice, we consider SAM with joint gradient normalization.

A first alternative to Definition 4 could scale perturbations after the joint gradient normalization. Opposed to Definition 4, for this variant the perturbation norm, i.e. the radius of the adversarial ascent ball, is not guaranteed to be ρn^{-d}, but $\rho(\sum_{l\in[L+1]}\rho_l^2)^{1/2}$. The correct perturbation scaling for this version more immediately follows from the condition that perturbations scale like updates.

Remark F.3 (Layerwise perturbation scaling after joint gradient normalization). For (SAM) with joint gradient normalization of the perturbations

$$\boldsymbol{\varepsilon}^l = \rho_l \cdot \frac{\nabla_{W^l}\mathcal{L}}{\|\nabla_{\mathbf{W}}\mathcal{L}\|},$$

the correct perturbation scaling in μP is given by $\rho_l \overset{!}{=} \Theta(n^{1/2}\cdot \texttt{fan\_out}/\texttt{fan\_in})$.

To understand this scaling rule, note that for $b_{L+1} > 1/2$ (such as in μP), the last layer always dominates the gradient norm (see Eq. (E.2) for the TP argument), resulting in the scaling

$$\|\nabla_{\mathbf{W}}\mathcal{L}\|_F \approx \mathcal{L}'(f_t(\xi_t), y_t)\|x^L\| = \Theta(n^{1/2}).$$

Thus, compared to SAM without gradient normalization (Appendix F.3), $\|\nabla_{\mathbf{W}}\mathcal{L}\|_F$ always contributes the scaling $n^{1/2}$. Noting that perturbations should scale like updates, and updates receive the layerwise learning rates $\eta_l \overset{!}{=} \Theta(\texttt{fan\_out}/\texttt{fan\_in})$ concludes the derivation. ◀

In Definition 4, we accept the additional layer-coupling complications that the layerwise gradient scaling before the joint gradient normalization entails in order to analytically control the perturbation radius to ρn^{-d}. To simplify the analysis as much as possible, we will first ensure width-independence of the normalization, so that the layerwise perturbation scaling is not affected by the normalization term. Then, layerwise perturbations should be scaled like updates.

Another alternative to layerwise perturbation scaling as in Definition 4 is motivated by the observation, that in μP^2 with Definition 4, only the first layer dominates the joint gradient norm (Theorem D.11). To let all layers contribute width-independently to the joint gradient norm, we can introduce even more hyperparameters (with limited benefit) by decoupling the numerator and denominator scalings in the perturbation. Opposed to Definition 4, the perturbation norm is again not analytically set with the choice of ρ, but may be smaller. Empirically, we do not observe performance differences due to denominator contribution scaling (Appendix H.4). This is the perturbation scaling we implement for ViTs (see Algorithm 1 for details).

Remark F.4 (SAM with decoupled perturbation numerator and denominator scaling). For (SAM) with perturbations

$$\boldsymbol{\varepsilon}^l = \rho n^{-d_l} \frac{\nabla_{W^l}\mathcal{L}}{\|v\|} \quad \text{with} \quad \|v\|^2 = \sum_{l=1}^{L+1} n^{-2\tilde{d}_l} \|\nabla_{W^l}\mathcal{L}\|^2, \tag{DP}$$

with layerwise perturbation radii $\rho \cdot n^{-d_l}$ and separate gradient norm scaling $n^{-\tilde{d}_l}$. ◀

In all alternatives, nontrivial layerwise perturbation scaling is necessary for effective perturbations in every layer, which necessarily changes the direction away from the original gradient direction. Such a layerwise gradient rescaling can also be achieved by adapting the architecture with width-dependent weight multipliers. The multipliers (a-μP^2) achieve effective perturbations without layerwise perturbation scaling such that all layers contribute non-vanishingly to the joint gradient norm. They rescale the gradients equivalently to (DP) when scaling all denominator terms to be width-independent. See Appendix F.6 for all details about weight multipliers.

Adapting the TP-based analysis. Our TP-based analysis covers all of the above perturbation scaling alternatives with minor adjustments. We just have to replace the normalized TP scalar (E.2). If we want to express $\|\nabla_{\mathbf{W}}\mathcal{L}\|_F$, we just drop all perturbation scaling terms n^{-d_l}. For the examples of (LN) and (DP), we replace (E.2) in each layer separately by the normalized TP scalars,

$$\|\nabla_{W^1}\mathcal{L}_t\| := \chi_t \left(\frac{(dh_t^1)^T dh_t^1}{n} (\xi_t^T \xi_t) \right)^{1/2},$$

with scaling $\theta_{\|\nabla_1\|} = n^{1/2}\theta_\nabla$ for the first layer, where θ_∇ is overloaded to denote $\theta_\nabla = n^{-b_{L+1}}$ in the first step and $\theta_\nabla = n^{-\min(b_{L+1}, c_{L+1})}$ in all later steps (in μP, $\theta_\nabla = n^{-1}$ always),

$$\|\nabla_{W^l}\mathcal{L}_t\| := \chi_t \left(\frac{(dh_t^l)^T dh_t^l}{n} \frac{(x_t^{l-1})^T x_t^{l-1}}{n} \right)^{1/2},$$

with scaling $\theta_{\|\nabla_{L+1}\|} = n\theta_\nabla$ for all hidden layers $l \in [2, L]$, and

$$\|\nabla_{W^{L+1}}\mathcal{L}_t\| := \chi_t \left(\frac{(x_t^L)^T x_t^L}{n} \right)^{1/2}.$$

with scaling $\sqrt{n}$ for the output layer, with respective normalized limits

$$\mathring{\chi}_t (\mathbb{E}[Z^{(dh_t^1)^2}](\xi_t^T \xi_t))^{1/2}, \quad \mathring{\chi}_t (\mathbb{E}[Z^{(dh_t^l)^2}]\mathbb{E}[Z^{(x_t^{l-1})^2}])^{1/2}, \quad \mathring{\chi}_t (\mathbb{E}[Z^{(x_t^L)^2}])^{1/2},$$

where $\mathring{\chi}_t = \mathcal{L}'(\mathring{f}_t(\xi_t), y_t)$.

The adapted scalings can then be tracked as before to derive the maximal stable layerwise perturbation scaling. Consider for example input layers in (LN). In μP, we know $\|\nabla_{W^1}\mathcal{L}_t\| = \Theta(n^{-1/2})$ and $\nabla_{W^1}\mathcal{L}_t = \Theta(n^{-1})$ entrywise. Effective perturbations are achieved with $\boldsymbol{\varepsilon}^1 = \Theta(1)$, so choose $\rho_l = n^{1/2}$ as expected from ($*$). Proceed similarly for all layers and perturbation scaling variants.

F.3 Extension to SAM without gradient normalization

Andriushchenko and Flammarion (2022) and Andriushchenko et al. (2023a) consider the SAM update without normalizing the gradient in the adversarial ascent. The corresponding update rule is given by

$$W_t = W_t - \eta \nabla_w \mathcal{L}(f(\xi_t; W_t + \rho v_t, y_t)), y_t), \qquad v_t = \nabla_w \mathcal{L}(f(\xi_t; W_t)).$$

The NE⊗OR⊤ program for this update rule with arbitrary $v_t^l = n^{-c_l}\nabla_w \mathcal{L}(f(\xi_t; W_t)$ is also easily adapted from the above derivation. Just note that the gradient norm appears in an equation if and only if the perturbation radius ρn^{-d} appears. Without dividing by $\|v_t\|$, the parameter d becomes superfluous. Simply set $d = 0$ and remove the gradient norm constraints (D.1) to arrive at the NE⊗OR⊤ program and bcd-constraints for the update rule without gradient normalization.

Perturbation scaling d_l plays a similar role as learning rate scaling c_l as both scale similar gradients. We get effective perturbations in the l-th layer from the equation $d_l + \min(b_{L+1}, c_{L+1}) = c_l +$

$\min(b_{L+1}, c_{L+1}, d_{L+1})$ in μP, which yields $d_l = c_l$ for all $l \in [L]$ (since $d_{L+1} = 1$ for stability). **In particular, in μP, the correct layerwise perturbation scaling of unnormalized gradients is given by the rule $\frac{\text{fan out}}{\text{fan in}}$ or the squared weight (update) spectral norm $\|W^l\|_*^2$** (Yang et al., 2023a), which could be efficiently approximately tracked with a running power iteration.

Note that Dai et al. (2024) argue that the normalizing the gradients for the perturbation is crucial (in standard parametrization) due to a stabilizing effect and an enhanced drift along manifolds of minima. Monzio Compagnoni et al. (2023) find that unnormalized SAM gets stuck around saddles while SAM slowly escapes through additional Hessian-induced noise. This suggests that the additional effort of analysing the original SAM update rule with gradient normalization is necessary for practically useful theory. From this paper's point of view, the gradient normalization may be adding stability via the $n^{-1/2}$ contribution which allows to scale down ρ less aggressively in practice.

F.4 Extension to Adaptive SAM

Adaptive SAM (ASAM) (Kwon et al., 2021) is motivated by a sharpness definition that is invariant to parameter rescaling operators that leave the output function invariant, and can provide a further improvement over SAM of 0.5% to 1%, depending on the considered vision dataset and model (Müller et al., 2024). Here we consider the two examples of elementwise rescaling operators (with $p = 2$) and layerwise rescaling operators (with $p = 2$), which are the best performing SAM variant in most settings in Müller et al. (2024).

Proposition F.5. *Neither elementwise ASAM, which performs* (SAM) *but using the perturbation rule* (F.4)*, nor layerwise ASAM, which performs* (SAM) *but using the perturbation rule* (F.6)*, can be written as a* NE⊗OR⊤*program.*

Proof sketch. Elementwise ASAM requires an elementwise multiplication of matrices, and layerwise ASAM requires calculating the Frobenius norm of a matrix. A NonLin operation only takes vectors as arguments, so NE⊗OR⊤calculations with a matrix require its multiplication with a vector. But then a single coordinate of the resulting vector contains a mixture of an entire row of that matrix. Since we are only allowed to define random vectors and matrices, and the NE⊗OR⊤master theorem states that coordinates of NE⊗OR⊤vectors behave iid-like, this mixture cannot be disentangled by choosing a structured vector. Hence, already at initialization, the square of individual entries/the Frobenius norm of a random matrix cannot be exactly recovered by a function of matrix-vector products with NE⊗OR⊤vectors. □

Although ASAM is not formally covered by our theory, we still expect that the ASAM perturbations are correlated with the gradient and therefore with the incoming activations, so that heuristically we can still expect LLN-like behaviour and apply our scaling condition. If the perturbation rules still behave LLN-like, then Table 1 summarizes which layers are effectively perturbed under global scaling and provides the unique maximal perturbation scalings for all considered SAM variants. The correct perturbation scaling in μP for other perturbation rules that behave LLN-like can always be derived following the same steps:

1. In μP, it always holds that

$$W^l = \begin{cases} \Theta(1) & l = 1, \\ \Theta(n^{-1/2}) & l \in [2, L], \\ \Theta(n^{-1}) & l = L+1, \end{cases} \quad \text{and} \quad \nabla_{W^l}\mathcal{L} = \begin{cases} \Theta(\theta_\nabla) = \Theta(n^{-1}) & l \leq L, \\ \Theta(1) & l = L+1. \end{cases} \tag{F.2}$$

2. Assuming the normalization term in the denominator is scaled to $\Theta(1)$, track the layerwise scalings of the numerator. Maximal stable perturbations are always achieved with

$$\tilde{\delta}W_t^l = \begin{cases} \Theta(1) & l = 1, \\ \Theta(n^{-1}) & l > 1. \end{cases} \tag{F.3}$$

   This yields constraints for achieving maximal stable perturbations in each layer.
3. Now replace the norm constraints (D.1) by tracking the scalings of each layer's contribution to the update rule's total normalization term.
4. To ensure normalization term scaling $\Theta(1)$, iterate through the layers l:

(a) choose d_l to satisfy its norm constraint,
(b) choose d to induce maximal stable perturbations in that layer,
(c) choose all other $d_{l'}$, $l' \neq l$, minimal to both satisfy its norm constraint as well as perturbation stability $\tilde{\delta}W_t^l = \begin{cases} O(1) & l = 1, \\ O(n^{-1}) & l > 1. \end{cases}$

5. From the above configurations, choose the unique one that yields maximal stable perturbations in all layers.[3]

F.4.1 Elementwise ASAM

If we want to be invariant to elementwise rescaling operators $T_w^l(x) = |W^l| \odot x$ where $x, W^l \in \mathbb{R}^{m \times n}$ and $\odot$ denotes elementwise multiplication, the resulting ASAM perturbation rule (where we introduce (layer-wise) perturbation scalings $\{d\} \cup \{d_l\}_{l \in [L+1]}$) replaces (LP) and is given by

$$\tilde{\delta}W_t^l := \rho n^{-d} \frac{n^{-d_l} |W^l| \odot |W^l| \odot \nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t)}{\|\nabla_{ASAM}^{elem}\|}, \tag{F.4}$$

with normalization

$$\|\nabla_{ASAM}^{elem}\| := \sum_{l=1}^{L+1} n^{-d_l} \left\| |W^l| \odot \nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t) \right\|_F,$$

where the absolute values $|W^l|$ are computed and multiplied elementwise. To find the correct perturbation scalings, we track the typical elementwise scaling of each quantity as before.

Elementwise ASAM in μP. In μP, the layerwise weights and gradients scale as (F.2). For $\|\nabla_{ASAM}^{elem}\| = O(1)$, we therefore replace the constraints (D.1) by the constraints

$$d_l \geq 1/2 - c_\nabla, \text{ for } l \in [L], \qquad d_{L+1} \geq -1/2, \tag{F.5}$$

where we can choose $\{d_l\}_{l \in [L+1]}$ to achieve equality in at least one constraint to achieve $\|\nabla_{ASAM}^{elem}\| = \Theta(1)$.

The layerwise perturbations scale as $\tilde{\delta}W_t^l = n^{-d} \begin{cases} \Theta(n^{-d_1} \theta_\nabla) & l = 1, \\ \Theta(n^{-1-d_l} \theta_\nabla) & l \in [2, L], \\ \Theta(n^{-d_{L+1}} n^{-2}) & l = L+1. \end{cases}$

Stable and nontrivial perturbations in each layer are achieved under condition (F.3), which induces the constraints for optimal layerwise perturbation scaling

$$d + d_l = -c_\nabla, \text{ for } l \in [L], \qquad d + d_{L+1} = -1.$$

Irrespective which of the above norm constraints (F.5) we satisfy, we need $d = -1/2$ to achieve optimal layerwise perturbation scaling. Hence $d = d_{L+1} = -1/2$ and $d_l = 1/2 - c_\nabla$ for $l \in [L]$ is the unique choice of $\{d\} \cup \{d_l\}_{l \in [L+1]}$ modulo norm scaling equivalence that achieves $\Theta(1)$ perturbation scaling in all layers. With this choice all layers contribute non-vanishingly to the gradient norm. In μP $c_\nabla = 1$, so that $d_l = -1/2$ for all $l \in [L+1]$, so that ASAM does not require layerwise rescaling of the gradients, but upscaling of the perturbation by $n^{1/2}$ to achieve nontrivial perturbations in any layer. This may explain why ASAM often outperforms SAM in large models: By only requiring global scaling, ASAM achieves maximal stable perturbations in all layers if the perturbation radius is tuned globally at every width.

If instead of a global gradient norm $\|\nabla_{ASAM}^{elem}\|$, one would want to normalize in each layer separately with $\|\nabla_{ASAM}^{elem,l}\| := n^{-d_l} \||W^l| \odot \nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t)\|_F$, the layerwise perturbation scalings become $\tilde{\delta}W_t^l = n^{-d} \begin{cases} \Theta(n^{-1/2}) & l = 1, \\ \Theta(n^{-3/2}) & l > 1. \end{cases}$ Again, to achieve maximal stable perturbations in all layers we need $d = -1/2$ and no layerwise adaptation of the gradient norm.

[3] There must exist such a choice of $\{d_l\}_{l \in [L+1]}$ and d, because $\{d_l\}_{l \in [L+1]}$ allow to set any relative scalings between layers and d determines the global scaling, which overall allows to set all possible layerwise scalings. Any deviation from the choice that achieves effective perturbations in all layers either results in blowup or a vanishing effect of the weight perturbation in some layer. This shows uniqueness.

F.4.2 Layerwise ASAM

ASAM with layerwise rescaling as in Müller et al. (2024) employs the layerwise transformations $T_w^l(x) = \|W^l\|_F \cdot x$. This ASAM perturbation rule replaces (LP) and is given by

$$\tilde{\delta} W_t^l := \rho n^{-d} \frac{n^{-d_l} \|W^l\|_F^2 \nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t)}{\|\nabla_{ASAM}^{layer}\|}, \tag{F.6}$$

with normalization

$$\|\nabla_{ASAM}^{layer}\| := \sum_{l=1}^{L+1} n^{-d_l} \|W^l\|_F \|\nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t)\|_F.$$

Layerwise ASAM in μP. In μP, we have $\|W^l\|_F = \begin{cases} \Theta(n^{1/2}) & l = 1, \\ \Theta(n^{1/2}) & l \in [2, L], \\ \Theta(n^{-1/2}) & l = L+1. \end{cases}$

Hence, the norm constraints (D.1) are now replaced by

$$d_1 \geq 1 - c_\nabla, \qquad d_l \geq 3/2 - c_\nabla \quad \text{for } l \in [2, L], \qquad d_{L+1} \geq 0.$$

The scale of the perturbation numerator now scales as $\tilde{\delta} W_t^l = n^{-d} \begin{cases} \Theta(n^{-d_1} n \theta_\nabla) & l = 1, \\ \Theta(n^{-d_l} n \theta_\nabla) & l \in [2, L], \\ \Theta(n^{-d_{L+1}} n^{-1}) & l = L+1. \end{cases}$

In μP, achieving maximal stable perturbations (F.3) is therefore equivalent to satisfying the constraints

$$d + d_1 = 0, \qquad d + d_l = 1 \quad \text{for } l \in [2, L], \qquad d + d_{L+1} = 0.$$

Now we can simultaneously satisfy the first- and last-layer norm constraints with $d_1 = 0$ and $d_{L+1} = 0$, while achieving effective perturbations in all layers with $d = 0$ and $d_l = 1$. Satisfying the norm constraint in the hidden layers with $d_l = 1/2$ would imply vanishing perturbations in the first and last layer (by requiring $d \geq 1/2$).

F.5 Representing general architectures and adaptive optimizers as Tensor Programs

Here, we lay out explicitly how to write some of the building blocks in ResNets and ViTs in a Tensor Program and provide further scaling considerations. According to Yang and Hu (2021), it is straightforward to generalize scaling conditions that induce feature learning in MLPs to these other common neural network building blocks. Since perturbations should always scale like updates, the conditions for stable feature learning and those for stable effective perturbations are analogous.

One potential complication in the case of SAM would be a contribution to the joint gradient normalization $\|v_t\|$ that differs from the classical input, hidden or output layer contribution. But we will see that these contributions do not differ for any of the considered layer types.

Layernorm. The Layernorm operation is defined as

$$h_t^{l+1} = \gamma_t^l \frac{x_t^l - \nu_t^l}{\sigma_t^l + \varepsilon} + \beta_t^l,$$

where $\varepsilon > 0$ is a small positive constant, γ_t^l, β_t^l are learnable parameters and $\nu_t^l = \frac{1}{n} \sum_{i=1}^n (x_t^l)_i$ is an Avg operation as in Yang and Littwin (2023, Def. 2.6.1) and $\sigma_t^l = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_t^l - \nu_t^l)^2}$ is a composition of Nonlin, Avg and Nonlin. The parameters γ_t^l, β_t^l can be seen as input weights to the input 1. They should be initialized as $\gamma_0^l = 1$ and $\beta_0^l = 0$. In the forward pass, the layernorm preserves stability $h_t^{l+1} = \Theta(1)$ when $\gamma_t^l + \beta_t^l = \Theta(1)$ except for the Lebesgue nullset of learning rates for which they exactly cancel each other out. Recall the notation $dz = \theta_z^{-1} \partial f / \partial z$, where $\theta_z = n^C$ for some $C \in \mathbb{R}$ denotes the width-dependent scaling. The derivatives are

$$d\beta_t^l = dh_t^{l+1}, \qquad d\gamma_t^l = dh_t^{l+1} \frac{x_t^l - \nu_t^l}{\sigma_t^l + \varepsilon}.$$

These gradients coincide both in shape and scaling with the scaling we expect for an input layer, resulting in the same gradient spectral/Frobenius norm scaling. Continuing the backward pass, using $\frac{\partial \sigma_t^l}{\partial x_t^l} = \frac{x_t^l - \nu_t^l}{n\sigma_t^l}$, we get

$$\begin{aligned} dx_t^l = & \quad dh_t^{l+1}\gamma_t^l \left(\frac{1}{\sigma_t^l + \varepsilon}(I - \frac{1}{n}) - \frac{x_t^l - \nu_t^l}{(\sigma_t^l + \varepsilon)^2} \frac{\partial \sigma_t^l}{\partial x_t^l} \right) \\ = & \quad dh_t^{l+1}\gamma_t^l \left(\frac{1}{\sigma_t^l + \varepsilon}(I - \frac{1}{n}11^T) - \frac{x_t^l - \nu_t^l}{(\sigma_t^l + \varepsilon)^2} \frac{(x_t^l - \nu_t^l)^T}{n\sigma_t^l} \right), \end{aligned}$$

which preserves the order as long as $\gamma_t^l = \Theta(1)$, since $x_t^l = \Theta(1)$, we know $\nu_t^l, \sigma_t^l = \Theta(1)$.

Note that Layernorm removes the necessity to avoid blowup in the activations x_t^l in the forward pass (ignoring potential numerical issues), and always rescales to $\Theta(\max(\gamma_t^l, \beta_t^l))$. However, in the backward pass, a scaling $x_t^l = \Theta(n^c)$, with $c > 0$, results in $dx_t^l = \Theta(n^{-c} dh_t^{l+1}\gamma_t^l)$, hence vanishing gradients. The gradients would only stabilize if $\phi'(h_t^l) = \Theta(h_t^l)$, but no popular activation function has a scale equivariant derivative. Yang (2019) shows how to write Batchnorm and Average Pooling as a Tensor Program.

Convolutions. Convolutional layers can be seen as a collection of dense weight matrices where width corresponds to the number of channels (Yang, 2019). With kernel positions ker, input channels $[n^l]$ and output channels $[n^{l+1}]$, the weights of a stride-1 convolution are given by $\{W_{i\alpha\beta}^l\}_{i \in ker, \alpha \in [n^{l+1}], \beta \in [n^l]}$, so that for each $i \in ker$, $W_i^l \in \mathbb{R}^{n^{l+1} \times n^l}$ is a dense matrix. With $\{x_{i\alpha}^l\}_{i \in pos^l, \alpha \in [n^l]}$, the convolution operation is given by

$$(W^l * x)_{i\alpha} = \sum_{\beta, j : j+i \in pos^l} W_{j\alpha\beta}^l x_{i+j,\beta}^l,$$

which performs MatMul and Avg and where ker, pos^l are assumed to be of fixed size. For ker of fixed size, convolutional weights scale like hidden layer weight matrices, also in Frobenius norm contributing to $\|v_t\|$.

Residual connections. A residual connection propagates the current activation forward, skipping an arbitrarily complex nonlinear block $f_t^l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}$ in between, where f_t^l can depend on time-dependent parameters like a weight matrix. The forward pass can be written as

$$x_t^l = x_t^{l-1} + f_t^l(x_t^{l-1}).$$

If $x_t^l = \Theta(1)$ for all layers l holds in the model without residual connections, it also holds in the model with residual connections. At fixed depth, $f_t^l = o(1)$ should be avoided, as it would hold that $x_t^{l+1} = x_t^l$ in the infinite-width limit and the layer would be superfluous. The derivative of the activations becomes

$$dx_t^{l-1} = dx_t^l + dx_t^l \frac{\partial f_t^l}{\partial x_t^{l-1}},$$

where the second term stays the same as without the residual connection. For the example of f_t^l being a fully connected layer we get $dx_t^{l-1} = dx_t^l + (W_t^l)^T \left(dx_t^l \odot \phi'(W_t^l x_t^{l-1}) \right)$. In this example, the derivative with respect to the weights becomes

$$\frac{\partial f_t}{\partial W_t^l} = dx_t^l \frac{\partial x_t^l}{\partial W_t^l} = dx_t^l \frac{\partial f_t^l}{\partial W_t^l} = (dx_t^l \odot \phi'(W_t^l x_t^{l-1}))(x_t^{l-1})^T,$$

where the residual connection does not alter the functional dependence on dx_t^l and x_t^l compared to a MLP, but implicitly influences the weight gradient since dx_t^l and x_t^l are altered. As for the forward pass, the gradient scaling dx_t^l gets stabilized in the backward pass so that $\frac{\partial f_t^l}{\partial x_t^{l-1}}$ is now allowed to be vanishing with width. Again, we are not aware of an architecture in which that would be desirable. Since a residual connection does not introduce learnable parameters, it interferes in $\|v_t\|$ only implicitly through the stabilized gradients in earlier layers, which can contribute non-vanishingly to $\|v_t\|$ even if later layers are wrongly scaled and their scaling is not adapted.

Adam as a base optimizer. When using Adam or similar adaptive optimizers as a base optimizer, the learning rate should scale as $\Theta(1)$ for input-like layers and biases, and $\Theta(n^{-1})$ for hidden and output

layers (Yang et al., 2022). Yang et al. (2023b) provide proofs for arbitrary optimizers that perform generalized, nonlinear outer products. In the example of Adam, the update rule can be written as

$$\phi(u_\alpha^1, \dots, u_\alpha^k, v_\beta^1, \dots, v_\beta^k) = \sum_i \gamma_i u_\alpha^i v_\beta^k / \left(\sum_i \omega_i (u_\alpha^i v_\beta^i)^2 \right)^{1/2},$$

where γ_i, ω_i are the weights that stem from the moving averages. By using a learning rate of n^{-1} and using the fact that both u and v have approximately iid coordinates of order $\Theta(1)$, the law of large numbers yields $\Theta(1)$ updates of the form

$$\frac{1}{n} \sum_{\beta=1}^{n} \phi(u_\alpha^1, \dots, u_\alpha^k, v_\beta^1, \dots, v_\beta^k) x_\beta = \mathbb{E} \phi(u_\alpha^1, \dots, u_\alpha^k, Z^{v^1}, \dots, Z^{v^k}) Z^x.$$

Any other learning rate scaling would either result in blowup or vanishing updates.

Adaptive optimizers have not been used for the ascent/perturbation step. In the descent/update step, nothing changes compared to unperturbed optimization as long as we ensure stable perturbations.

F.6 Influence of width-dependent weight multipliers on *bcd*-parameterizations

Our definition of *bcd*-parameterizations is convenient because it purely adapts the learning algorithm but not the architecture. **We can also adapt the architecture by using layerwise width-dependent weight multipliers to effectively perturb all layers without any perturbation scaling.** The reason is that layerwise weight multipliers scale the layerwise gradients. Here, we study how the introduction of weight multipliers affects *bcd*-parameterizations.

In this section, we consider L-hidden layer MLPs with weight multipliers $\{a_l\}_{l \in [L+1]}$, width $n \in \mathbb{N}$, inputs $\xi \in \mathbb{R}^{d_{\text{in}}}$, and with outputs $f(\xi) := n^{-a_{L+1}} W^{L+1} x^L(\xi)$ where the activations $x^L(\xi)$ are defined via the iteration

$$h^1(\xi) := n^{-a_1} W^1 \xi, \qquad x^l(\xi) := \phi(h^l(\xi)), \qquad h^{l+1}(\xi) := n^{-a_{l+1}} W^{l+1} x^l(\xi).$$

We define *abcd*-parameterizations in the same way as *bcd*-parameterizations, but instead of MLPs we use MLPs with weight multipliers $\{a_l\}_{l \in [L+1]}$.

Definition F.6 (*abcd*-**parametrization**)**.** An *abcd-parametrization* $\{a_l\}_{l \in [L+1]} \cup \{b_l\}_{l \in [L+1]} \cup \{c_l\}_{l \in [L+1]} \cup \{d_l\}_{l \in [L+1]} \cup \{d\}$ defines the training of an MLP with weight multipliers $\{a_l\}_{l \in [L+1]}$ with SAM in the following way:

(a) Initialize weights iid as $W_{ij}^l \sim \mathcal{N}(0, n^{-2b_l})$.
(b) Train the weights using the SAM update rule with layerwise learning rates,

$$W_{t+1}^l = W_t^l - \eta n^{-c_l} \nabla_{W^l} \mathcal{L}\left(f\left(\xi_t; W_t + \boldsymbol{\varepsilon}_t\right), y_t\right),$$

with the scaled perturbation $\boldsymbol{\varepsilon}_t$ via layerwise perturbation radii,

$$\boldsymbol{\varepsilon}_t := \rho n^{-d} \frac{v_t}{\|v_t\|}, \quad \text{with} \quad v_t = (v_t^1, \dots, v_t^{L+1}), \quad v_t^l := n^{-d_l} \cdot \nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t), \tag{LP}$$

W.l.o.g. we set $\|v_t\| = \Theta(1)$, which prevents nontrivial width-dependence from the denominator. This imposes the constraints:

$$d_1 + a_1 \geq 1/2 - c_\nabla, \qquad d_l + a_l \geq 1 - c_\nabla, \qquad d_{L+1} + a_{L+1} \geq 1/2,$$

with at least one equality required to hold, where $l \in [2, L]$, and where $\nabla_{x^L} f = n^{-a_{L+1}} W^{L+1} = \Theta(n^{-c_\nabla})$ with $c_\nabla = \min(b_{L+1} + a_{L+1}, c_{L+1} + 2a_{L+1})$. The normalization $v_t / \|v_t\|$ removes one degree of freedom from $\{d_l\}_{l \in [L+1]}$ via the equivalence $\{d_l'\}_{l \in [L+1]} \cong \{d_l\}_{l \in [L+1]}$ iff there exists a $C \in \mathbb{R}$ such that $d_l' = d_l + C$ for all $l \in [L+1]$. ◀

F.6.1 *abcd*-equivalence classes

Update scalings behave as in SGD. The weight multiplier n^{-a_l} scales the gradient $\nabla_{W^l} f$ by n^{-a_l}. In the following forward pass, another multiplication of the weight updates with n^{-a_l} leads to the activation update scaling n^{-2a_l}. This can be counteracted by adapting the learning rate scaling. For

abc-parameterizations and SGD training, this induces the layerwise equivalence between parameterizations with (a_l, b_l, c_l) or with $(a_l + \theta_l, b_l - \theta_l, c_l - 2\theta_l)$. The extension of all of our results to Adam as a base optimizer is straightforward, since learning rate scalings and perturbation scalings are decoupled. For Adam, c_l should be adapted to $c_l - \theta_l$.

Again, perturbations with joint gradient normalization complicate matters compared to SGD and Adam. Keeping the gradient norm scalings invariant under $a_l \mapsto a_l + \theta$ would require $d_l \mapsto d_l - \theta$, but keeping the activation perturbation scaling invariant would require $d_l \mapsto d_l - 2\theta$ as for updates. Consequently, an exact equivalence between $abcd$-parameterizations at finite width requires θ to be the same for all layers and the conflicting gradient norm in the denominator and perturbation scaling in the numerator to be accounted for by adapting the global perturbation scaling $d \mapsto d - \theta$ (together with $d_l \mapsto d_l - \theta$). In other words, (SAM) with layer-coupling gradient normalization (LP) does not have layerwise analytical equivalence classes at finite width. Below, we provide two alternative perturbation rules that resolve these complications and recover layerwise equivalence classes. The following lemma formally states the layer-coupled equivalence relation for the perturbation rule (LP). All proofs are provided at the end of this section.

Lemma F.7 ($abcd$**-equivalence classes).** *Let $f_t(\xi)$ denote the output of a MLP in a stable $abcd$-parameterization with weight multipliers $\{a_l\}_{l\in[L+1]}$ after t steps of training with the SAM update rule with layerwise perturbation scaling (LP) using a fixed sequence of batches and evaluated on input ξ. Then for any $\theta \in \mathbb{R}$ and any $C \in \mathbb{R}$, $f_t(\xi)$ stays fixed for all t and ξ if, for all $l \in [L+1]$,*

$$(a_l, b_l, c_l, d_l, d) \text{ is reparameterized to } (a_l + \theta, b_l - \theta, c_l - 2\theta, d_l - \theta + C, d - \theta).$$

Remark F.8 (Infinite-width equivalences). In the infinite-width limit, $abcd$-parameterizations remain equivalent under $(a_l + \theta_l, b_l - \theta_l, c_l - 2\theta_l, d_l - 2\theta_l, d)$ layerwise as long as the set of layers that contribute to the gradient norm non-vanishingly remains invariant. The gradient norm constraints for $\|v^l\| = O(1)$ become

$$d_1 + a_1 \geq 1/2 - c_\nabla, \qquad d_l + a_l \geq 1 - c_\nabla, \qquad d_{L+1} + a_{L+1} \geq 1/2,$$

where $\nabla_{x^L} f = n^{-a_{L+1}} W^{L+1} = \Theta(n^{-c_\nabla})$ with $c_\nabla = \min(b_{L+1} + a_{L+1}, c_{L+1} + 2a_{L+1})$ remains invariant under equivalence transformations. ◀

Remark F.9 (SAM with layerwise gradient normalization). As the layer coupling is induced by the joint gradient normalization in the perturbations, layerwise gradient normalization simplifies the analysis. For (SAM) with layerwise gradient normalization (LN) of the perturbations global perturbation scaling d is superfluous, and there exist layerwise equivalence classes: For any $\{\theta_l\}_{l\in[L+1]} \subset \mathbb{R}$,

$$(a_l, b_l, c_l, d_l) \text{ is equivalent to } (a_l + \theta_l, b_l - \theta_l, c_l - 2\theta_l, d_l - \theta_l).$$

To understand this equivalence, observe that any layerwise gradient scaling is cancelled out by the normalization $\nabla_{W^l}\mathcal{L}/\|\nabla_{W^l}\mathcal{L}\|$. Only the n^{-a_l} factor from subsequent forward passes has to be counteracted. ◀

Remark F.10 (SAM with decoupled perturbation numerator and denominator scaling). A perturbation rule with joint gradient normalization and layerwise equivalence classes can be achieved by introducing even more hyperparameters and decoupling numerator and denominator scalings of each layer. For (SAM) with perturbations (DP) with layerwise perturbation radii $\rho \cdot n^{-d_l}$ and separate gradient norm scaling $n^{-\tilde{d}_l}$, global perturbation scaling d is superfluous, and there exist layerwise equivalence classes: For any $\{\theta_l\}_{l\in[L+1]} \subset \mathbb{R}$,

$$(a_l, b_l, c_l, d_l, \tilde{d}_l) \text{ is equivalent to } (a_l + \theta_l, b_l - \theta_l, c_l - 2\theta_l, d_l - 2\theta_l, \tilde{d}_l - \theta_l).$$

This perturbation rule also allows us to recover an analytical equivalence between trivial weight multipliers $a_l = 0$ for all l, and any other weight multipliers. ◀

F.6.2 μP^2 under non-trivial weight multipliers.

Our goal here is to find the weight multipliers that simplify the necessary perturbation scaling for effective perturbations in all layers as much as possible. The non-existence of layerwise equivalence classes in $abcd$-parameterizations from (LP) is not an issue if we are interested in effective perturbation properties and recovering μP^2 for arbitrary weight multipliers $\{a_l\}_{l\in[L+1]}$, as the equivalence breaks due to varying gradient norm contributions, which are inconsequential for achieving effective perturbations.

As we aim to reproduce μP^2, we restrict ourselves to the μP equivalence class of abc-parameterizations. We do not allow layerwise perturbation scaling and are interested in the maximal stable choice of global perturbation scaling ρn^{-d} to at least achieve non-vanishing perturbations in some layers. The following lemma shows even more: **The choice**

$$a_l = -1/2 \cdot \mathbb{I}(l = 1) + 1/2 \cdot \mathbb{I}(l = L + 1)$$

achieves effective perturbations in all layers with the naive (SAM) **update rule with naive perturbation scaling $\rho \cdot n^0$, and all layers contribute non-vanishingly to the joint gradient norm.** Hence this seems to be a natural choice of weight multipliers for SAM. However, it is in conflict with unit scaling considerations (Blake et al., 2024). Effectively, naive learning rate and perturbation scaling with these multipliers is equivalent to (DP) where all denominator terms are scaled to be width independent, as implemented by Algorithm 1, which resembles our implementation for ViTs. Our ablations in Appendix H.4 suggest that gradient norm contributions have a negligible effect on generalization performance.

Lemma F.11 (Naive perturbation scaling can effectively perturb all layers). *Consider an abcd-parameterization where $\{(a_l, b_l, c_l)\}_{l\in[L+1]}$ are chosen from the μP equivalence class, and where there is some $C \in \mathbb{R}$ such that $d_l = C$ for all $l \in [L + 1]$. This reduces to training a MLP with weight multipliers with* (SAM) *with global perturbation scaling ρn^{-d} for some $d \in \mathbb{R}$. Effective perturbations in all layers are achieved and all layers contribute non-vanishingly to the gradient norm if and only if*

$$a_1 = -d - 1/2, \qquad a_l = -d \quad \text{for } l \in [2, L], \qquad a_{L+1} = -d + 1/2.$$

Achieving μP^2 with the current implementation of the `mup`-package requires both an adaptation of the architecture and of the learning algorithm, as the following lemma shows. Hence the package is not particularly suited for SAM learning in μP^2 when the goal is simple perturbation scaling.

Lemma F.12 (Effective perturbations with the `mup`-package). *Consider an abcd-parameterization where $\{(a_l, b_l, c_l)\}_{l\in[L+1]}$ are chosen from the μP equivalence class, and with the weight multipliers $a_{L+1} = \mathbb{I}(l = L + 1)$ as in the `mup`-package.*

- *(a) (**`mup`-package global scaling effectively perturbs hidden layers**) Under global scaling $d_l = C$, $C \in \mathbb{R}$, for all $l \in [L + 1]$, maximal stable perturbations are achieved with $d = 0$. In this parameterization, hidden layers are effectively perturbed, but input and output layers are not effectively perturbed.*
- *(b) (**μP^2 with the `mup`-package**) Effective perturbations in all layers are achieved with the choice $d = d_1 = d_{L+1} = -1/2$ and $d_l = 1/2$ for $l \in [2, L]$.*

The following lemma covers the general case how to achieve μP^2 given arbitrary weight multipliers.

Lemma F.13 (μP^2 with arbitrary weight multipliers). *Consider an abcd-parameterization where $\{(a_l, b_l, c_l)\}_{l\in[L+1]}$ are chosen from the μP equivalence class. Then effective perturbations in all layers are achieved with the choice $d = \min_{l\in[L+1]}(-a_l - 1/2\mathbb{I}(l = 1) + 1/2\mathbb{I}(l = L + 1))$, and*

$$d_1 = -1 - d - 2a_1, \qquad d_l = -d - 2a_l, \text{ for } l \in [2, L], \qquad d_{L+1} = 1 - d - 2a_{L+1}.$$

The following lemma shows that weight multipliers that achieve μP^2 with naive perturbation scaling under perturbations with layerwise normalization (LN) are exactly the same as the ones for (LP).

Lemma F.14 ((LN) with naive perturbation scaling can effectively perturb all layers). *Consider* (SAM) *with layerwise normalization* (LN)*. Assume $\{(a_l, b_l, c_l)\}_{l\in[L+1]}$ are chosen from the μP equivalence class, and assume there is some $C \in \mathbb{R}$ such that $d_l = C$ for all $l \in [L + 1]$. Then all layers are effectively perturbed if the multipliers are chosen as*

$$a_1 = -1/2 - C, \qquad a_l = -C, \qquad a_{L+1} = 1/2 - C.$$

Proof of Lemma F.14. As derived in Appendix F.7, under $a_l = 0$ for all $l \in [L + 1]$, all layers are effectively perturbed if and only if $d_l = -1/2 \cdot \mathbb{I}(l = 1) + 1/2 \cdot \mathbb{I}(l = L + 1)$. Now we can exploit the layerwise equivalence relation to enforce $d_l = C$ in each layer by adapting all a_l. □

Proof of Lemma F.11. In general, in the abc-equivalence class of μP, the l-th layer's gradient norm is scaled by n^{-a_l}. This induces the generalized gradient norm constraints for $\|\nabla_W L\| = \Theta(1)$,

$$d_1 + a_1 \geq -1/2, \qquad d_l + a_l \geq 0, \qquad d_{L+1} + a_{L+1} \geq 1/2.$$

Effective perturbations are achieved when $\rho n^{-d-d_l-a_l}\nabla_{W^l}L = \Theta(n^{-\mathbb{I}(l>1)})$, which induces the perturbation stability constraints

$$d + d_1 + 2a_1 \geq -1, \qquad d + d_l + 2a_l \geq 0, \qquad d + d_{L+1} + 2a_{L+1} \geq 1,$$

with effective perturbations whenever the equality of the respective layer holds.

Under global scaling, the gradient norm constraints become, for some $C \in \mathbb{R}$,

$$C + a_1 \geq -1/2, \qquad C + a_l \geq 0, \qquad C + a_{L+1} \geq 1/2,$$

and the conditions for effective perturbations become

$$d + C + 2a_1 \geq -1, \qquad d + C + 2a_l \geq 0, \qquad d + C + 2a_{L+1} \geq 1.$$

As $d + C$ is a common term in all layers, we get the relations $a_l = a_1 + 1/2$, $a_{L+1} = a_1 + 1$, so that all gradient norm constraints are simultaneously satisfied with $C = -a_l$ and effective perturbations are achieved in all layers with $d = -a_l$. □

Proof of Lemma F.12. Under the choice $a_l = \mathbb{I}(l = L + 1)$, the gradient norm constraints become

$$d_1 \geq -1/2, \qquad d_l \geq 0, \qquad d_{L+1} \geq -1/2,$$

and the conditions for effective perturbations become

$$d + d_1 \geq -1, \qquad d + d_l \geq 0, \qquad d + d_{L+1} \geq -1.$$

Proof of (a):

Satisfying the gradient norm constraints with global scaling requires $d_l = 0$ for all $l \in [L + 1]$, then the minimal stable choice of d is $d = 0$ which only effectively perturbs hidden layers.

Proof of (b):

The choice $d = -1/2$ and $d_1 = -1/2$ saturates the gradient norm constraint and achieves effective perturbations in the input layer. Then the choice $d_l = 1/2$ and $d_{L+1} = -1/2$ satisfies the gradient norm constraints and achieves effective perturbations in all layers. □

Proof of Lemma F.7. To understand the influence of weight multipliers on updates and perturbations, first note that under an equivalence transformation of all $abcd$-parameters w.l.o.g from $a_l = 0$ for all $l \in [L + 1]$, the scalings of h^l, x^l and of $n^{-a_l}W^l$ remain invariant. This implies that the scalings of $\nabla_{x^L} f = n^{-a_{L+1}}W^{L+1}$, $\nabla_{h^l} f = \nabla_{x^l} f \odot \phi'(h^l)$ and $\nabla_{x^l} f$ for all $l \in [L]$ also remain invariant. Hence the weight gradients, $\nabla_{W^{L+1}} f = n^{-a_{L+1}}x^L$ and $\nabla_{W^l} f = n^{-a_l}\nabla_{h^l} f \cdot (x^{l-1})^\top$ are scaled by n^{-a_l} in each layer.

In the following forward pass, we get

$$h^l = n^{-a_l}(W^l + \Delta W^l)x^{l-1} = n^{-a_l}(W^l - \eta n^{-c_l}\nabla_{W^l}\mathcal{L})x^{l-1},$$

so that activation/output updates and perturbations of layer l are scaled by n^{-2a_l}.

Again, a complication compared to SGD or Adam arises through the gradient normalization of SAM's weight perturbation. If the gradients are simply normalized layerwise $\varepsilon^l = \rho \cdot n^{-d_l} \cdot \nabla_{W^l}\mathcal{L}/\|\nabla_{W^l}\mathcal{L}\|$, the n^{-a_l}-term from the backward pass cancels out, and only in the forward pass we get a scaling n^{-a_l}. Hence an exact layerwise equivalence still exists for SAM with layerwise gradient normalization:

$$(a_l, b_l, c_l, d_l) \text{ is equivalent to } (a_l + \theta_l, b_l - \theta_l, c_l - 2\theta_l, d_l - \theta_l).$$

Under joint gradient normalization (SAM), as we consider in our definition of bcd-parameterizations, keeping the gradient norm scalings invariant under $a_l \mapsto a_l + \theta$ would require $d_l \mapsto d_l - \theta$, but keeping the perturbation scaling invariant would require $d_l \mapsto d_l - 2\theta$ as for updates. Consequently, due to the layer coupling of joint gradient normalization $\|\nabla_{\mathbf{W}}\mathcal{L}\|$, an exact equivalence between $abcd$-parameterizations at finite width requires θ to be the same for all layers and the conflicting gradient norm in the denominator and perturbation scaling in the numerator to be accounted for by $d_l \mapsto d_l - \theta$ and $d \mapsto d - \theta$.

In the infinite-width limit, $abcd$-parameterizations remain equivalent under $(a_l + \theta_l, b_l - \theta_l, c_l - 2\theta_l, d_l - 2\theta_l, d)$ layerwise as long as the set of layers that contributes to the gradient norm non-vanishingly remains invariant. The gradient norm constraints for $\|v^l\| = O(1)$ become

$$d_1 + a_1 \geq 1/2 - c_\nabla, \qquad d_l + a_l \geq 1 - c_\nabla, \qquad d_{L+1} + a_{L+1} \geq 1/2,$$

where $\nabla_{x^L} f = n^{-a_{L+1}} W^{L+1} = \Theta(n^{-c_\nabla})$ with $c_\nabla = \min(b_{L+1} + a_{L+1}, c_{L+1} + 2a_{L+1})$ remains invariant under equivalence transformations. □

Proof of Lemma F.13. First, the choices of d_l ensure that the constraints for effective perturbations from the proof of Lemma F.11 are saturated in each layer. It is left to show, that these choices satisfy the $\|\nabla_W L\| = \Theta(1)$-constraints. For input layers, since $-d \geq a_1 + 1/2$, it holds that $d_1 + a_1 \geq -1/2$. For hidden layers, since $-d \geq a_l$, it holds that $d_l + a_l \geq 0$. For output layers, since $-d \geq a_{L+1} - 1/2$, it holds that $d_{L+1} + a_{L+1} \geq 1/2$. Observe that the minimizer in the definition of d saturates its gradient norm constraint. □

F.7 The spectral perspective on μP^2

While Tensor Programs allow to track the transformations of vectors like activations, Yang et al. (2023a) provide an equivalent formulation in terms of weight matrix spectral norms. They find that the spectral norm measures the effect of a weight update on the activations, under certain non-cancellation assumptions and limited batch size. For all MLP layers, they show that μP is equivalent to achieving the condition

$$\|W_t^l\|_* = \Theta\left(\sqrt{\frac{\texttt{fan\_out}}{\texttt{fan\_in}}}\right) \quad \text{and} \quad \|\Delta W_t^l\|_* = \Theta\left(\sqrt{\frac{\texttt{fan\_out}}{\texttt{fan\_in}}}\right),$$

at all times t, where $W_t^l : \mathbb{R}^{\texttt{fan\_in}} \to \mathbb{R}^{\texttt{fan\_out}}$. This condition is achieved with initialization σ_l, SGD learning rate η_l and Adam learning rate η_l^{Adam} chosen as,

$$\sigma_l = \Theta\left(\frac{1}{\sqrt{\texttt{fan\_in}}} \min\left\{1, \sqrt{\frac{\texttt{fan\_out}}{\texttt{fan\_in}}}\right\}\right), \quad \eta_l = \Theta\left(\frac{\texttt{fan\_out}}{\texttt{fan\_in}}\right), \quad \eta_l^{Adam} = \Theta\left(\frac{1}{\texttt{fan\_in}}\right).$$

This generalizes μP to varying widths inside the network. For varying widths, we adopt the notation $W^l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ with $n_0 = d_{in}$ and $n_{L+1} = d_{out}$, whereas $\texttt{fan\_in}$ and $\texttt{fan\_out}$ always adapt to the weight matrix under consideration.

To understand why the spectral norm is desirable, note that $\Delta W^l = \eta_l \nabla_{h^l} \mathcal{L} (x^{l-1})^\top$ is low rank and aligned with the incoming activations. For batch size 1, we even have rank-1 updates with $\|\Delta W^l\|_* = \eta_l \|\nabla_{h^l} \mathcal{L}\|_2 \|x^{l-1}\|_2$, aligned with the incoming activations x^{l-1}, hence $\|\Delta W^l x^{l-1}\|_2 = \|\Delta W^l\|_* \|x^{l-1}\|_2$. This allows to achieve $\|\Delta x^l\|_2 = \Theta(\sqrt{n_l})$ irrespective of the layer type with $\|\Delta W_t^l\|_* = \Theta(\sqrt{n_l / n_{l-1}})$.

Our simple condition that perturbations should scale like updates, which is rigorously justified by our Tensor Program based proof in Appendix E, now allows to derive the correct perturbation scalings using the spectral weight perspective.

Layerwise perturbations. As a simple starting point, consider a variant of (SAM) that does not globally normalize the gradient of all layers jointly, but uses layerwise normalization (LN), resulting in the layerwise perturbation rule,

$$\varepsilon^l = \rho_l \cdot \nabla_{W^l} \mathcal{L} / \|\nabla_{W^l} \mathcal{L}\|,$$

where $\|\cdot\|$ may denote either the spectral or the Frobenius norm (equivalent under limited perturbation batch size). Without the global normalization, the scalings of all layers are not coupled, and the spectral condition $\|\varepsilon^l\|_* = \Theta(\sqrt{\texttt{fan\_out}/\texttt{fan\_in}})$ immediately requires choosing

$$\rho_l = \rho \cdot \sqrt{\texttt{fan\_out}/\texttt{fan\_in}}$$

for effective perturbations in layer l with width-independent hyperparameter $\rho \geq 0$.

Perturbations with global gradient normalization. Perturbations that are globally normalized across layers have usually been implemented practice according to the GitHub repositories provided

by Foret et al. (2021); Samuel (2022); Kwon et al. (2021); Andriushchenko and Flammarion (2022); Müller et al. (2024). Since we are interested in analysing (SAM) as it is applied in practice, we study variants with joint gradient normalization in more detail. Preliminary ablations in Appendix H.5 suggest that layer-coupled SAM with global normalization slightly outperforms SAM with layerwise gradient normalization. To simplify the analysis as much as possible, we will first ensure width-independence of the normalization, so that the layerwise perturbation scaling is not affected by the normalization term. Then, layerwise perturbations should again be scaled like updates.

Separate denominator scalings. If we allow to scale each denominator term separately from the corresponding numerator term (DP), the perturbation radius in each layer for the numerator can be scaled like updates, $\rho_l = \Theta\left(\frac{\texttt{fan\_out}}{\texttt{fan\_in}}\right)$.

Now, to ensure $\Theta(1)$ in the denominator, each input and hidden-like gradient norm $\|\nabla_{W^l}\mathcal{L}\|_F$, $l \in [L]$, achieves width-independence if it scaled by $\sqrt{\texttt{fan\_out}/\texttt{fan\_in}}$. The same rule applies to biases when understanding them as weights $\mathbb{R} \to \mathbb{R}^{n_l}$ to the input 1. These scalings are derived in the next paragraph. The last-layer gradient norm $\|\nabla_{W^{L+1}}\mathcal{L}\|_F$ should be scaled as $(n_L n_{L+1})^{-1/2}$, and $\|\nabla_{b^{L+1}}\mathcal{L}\|_F$ as $n_{L+1}^{-1/2}$.

If we care about the correct width-independent constants, observe that the learning rate scaling $\eta_{L+1} = \Theta(\texttt{fan\_out}/\texttt{fan\_in})$ induces $\|W^{L+1}\| = \|\Delta W^{L+1}\| = \Theta(\sqrt{n_{L+1}^3/n_L})$. If we wanted to achieve $\Delta W^{L+1} = \Theta(\sqrt{n_{L+1}/n_L})$ we would need $\eta_{L+1} = \Theta(1/\texttt{fan\_in})$. As $n_{L+1} = d_{out}$ is width-independent, $\sqrt{\texttt{fan\_out}/\texttt{fan\_in}}$ would result in the same width-dependent scaling for the last layer, but ignoring large constants can introduce a significant width-independent spectral distortion. For example in ImageNet1K, n_{L+1} is large. By tuning input, hidden and output multipliers such constant distortions may be corrected. The multiplier used in the `mup`-package does not correct this distortion. Using a base width at which SP is recovered may also cement such spectral distortions, if no multipliers are tuned.

Derivation of gradient norm $\|\nabla_{W^l}\mathcal{L}\|_F$ scalings. In this paragraph, $\|\cdot\|$ may denote the Frobenius or spectral norm. As all matrices are of limited rank, both norms scale equivalently. As a first step, $\|\nabla_{h^l}\mathcal{L}\| = \Theta(\frac{1}{\sqrt{n_l}})$ can be reconstructed from

$$\Theta(\sqrt{n_l/n_{l-1}}) = \|\Delta W^l\|_* = \eta_l \|\nabla_{h^l}\mathcal{L}\| \|x^{l-1}\|_2 = n_l/n_{l-1} \cdot \sqrt{n_{l-1}} \|\nabla_{h^l}\mathcal{L}\|.$$

Now, for input and hidden layers, $\|\nabla_{W^l}\mathcal{L}\| = \|\nabla_{h^l}\mathcal{L}\| \|x^{l-1}\|_2 = \Theta(\sqrt{n_{l-1}/n_l})$. Multiplying by the inverse yields width-independent scaling. The output layer gradient $\nabla_{W^{L+1}}\mathcal{L} \in \mathbb{R}^{n_{L+1} \times n_L}$ is given by $(\nabla_{W^{L+1}}\mathcal{L})_{ij} = x_j^L = \Theta(1)$, so that $\|\nabla_{W^{L+1}}\mathcal{L}\| = \Theta(\sqrt{n_L n_{L+1}})$. Biases before the last layer follow the scheme $\|\nabla_{b^l}\mathcal{L}\| = \|\nabla_{h^l}\mathcal{L}\| = \Theta(\sqrt{1/n_l}) = \Theta(\sqrt{\texttt{fan\_in}/\texttt{fan\_out}})$. The last layer bias $\|\nabla_{b^{L+1}}\mathcal{L}\| = \sqrt{n_{L+1}}$ scales width-independently as it should, but needs to be scaled by a different constant $1/\sqrt{\texttt{fan\_out}}$ than earlier layers.

Extensions to ASAM. As ASAM cannot be written as a NE⊗OR⊤program, its scaling can only be derived heuristically. As provided in Table 1 and derived in Appendix F.4, elementwise ASAM scales all layer types correctly in relation to each other, and it suffices to rescale the global perturbation radius by $\sqrt{n_L}$, assuming all width dimensions scale proportionally. For SAM-ON, we only perturb input-like layers such as normalization layers. As the conditions for correct scaling remain the same, the above scalings for input layers in SAM also apply to SAM-ON.

For layerwise ASAM, first note that $\|W_t^l\|_F = \Theta(\|W_0^l\|_F) = \Theta(\sqrt{n_l})$ for input and hidden layers $l \in [L]$. As the numerator contains $\|W_t^l\|_F^2$, it requires the layerwise perturbation scaling $\frac{1}{\texttt{fan\_in}}$. In the denominator, width independence is achieved with the multiplier $\sqrt{\frac{1}{\texttt{fan\_in}}}$, since $\|W^l\|_F \|\nabla_{W^l}\mathcal{L}\|_* = \sqrt{n_l}\sqrt{\frac{n_{l-1}}{n_l}} = \sqrt{n_{l-1}}$. Again, the output layer requires a special treatment. Due to its small initialization, it holds that $\|W^{L+1}\|_F^2 = \|\Delta W^{L+1}\|_F^2 = \Theta(\frac{n_{L+1}^3}{n_L})$. For perturbations that fulfill the spectral condition $\rho_{L+1}\|W^{L+1}\|_F^2 \|\nabla_{W^{l+1}}\mathcal{L}\|_* = \Theta(\sqrt{\frac{n_{L+1}}{n_L}})$, we need to choose $\rho_{L+1} = \rho \cdot \frac{1}{n_{L+1}^3}$ (width-independent, but very small). The last-layer denominator term scales as $\|W^{L+1}\|_F \|\nabla_{W^{l+1}}\mathcal{L}\|_* = \Theta(\sqrt{\frac{n_{L+1}^3}{n_L}} \cdot \sqrt{n_L n_{L+1}}) = \Theta(n_{L+1}^2)$, which is width independent, but

can be a large constant, as for ImageNet1K. The output bias numerator exactly conforms with the correct scaling $\|\nabla_{b^{L+1}}\mathcal{L}\|_F^2 = n_{L+1} = \texttt{fan\_out}/\texttt{fan\_in} = \Theta(1)$.

Note that weight decay may break statements like $\|W_t^l\|_F = \Theta(\|W_0^l\|_F)$ over long training. Everett et al. (2024) have recently observed more generally that scalings may evolve differently over long training than predicted by pure infinite-width TP theory, because alignments evolve dynamically between CLT- and LLN-like behaviour.

Using the `mup`-package. The `mup`-package introduces the output layer weight multiplier n_L^{-1} so that input and output layer learning rates may be scaled by the same width-dependent factor. Hence, only the last-layer scalings change. The scalings of $n_L^{-1}W^{L+1}$ and $n_L^{-1}\Delta W^{L+1}$ remain the unique ones that achieve μP, but $\nabla_{W^{L+1}}\mathcal{L}$ is scaled by n_L^{-1}. This requires adapting the last-layer learning rate η_{L+1} to scale like input layers. For SAM, the last-layer perturbation radius can now be scaled like input layers. That is, assuming proportionally growing width n, in the numerator $\rho_{L+1} = \rho_1 = \rho \cdot n$ and $\rho_l = \rho$ for $l \in [2, L]$, and the gradient norm contributions should be scaled by $\sqrt{n}$ for input and output layers, and by 1 for hidden layers. The Tensor Program perspective on weight multipliers can be found in Appendix F.6. The correct width-independent constants are achieved with the last-layer numerator scaling $\rho_{L+1} = \rho \cdot n_L$ and the last-layer denominator scaling $\sqrt{n_L/n_{L+1}}$, since $\nabla_{W^{L+1}}\mathcal{L} = \Theta(\sqrt{n_{L+1}/n_L})$ and for the numerator we get an additional n_L^{-1} in the forward pass.

For SAM-ON nothing changes, as only input-like layers are perturbed. For elementwise ASAM, ignoring width-independent constants, nothing changes as the weight multiplier n_L^{-1} increases the weight scaling W^{L+1} and decreases the gradient scaling $\nabla_{W^{L+1}}\mathcal{L}$ by the same amount. The additional n_L^{-1}-factor in the numerator is cancelled out by the additional W^{L+1}-factor. For the correct width-independent constants with decoupled numerator and denominator scaling, we would scale the denominator by $\sqrt{n_L/n_{L+1}^3}$ with or without weight multiplier, and scale the numerator by $\rho_{L+1} = \rho \cdot n_L/n_{L+1}^2$ with or without weight multiplier. For the example of layerwise ASAM, we still get for the denominator $\|W^{L+1}\|_F\|\nabla_{W^{l+1}}\mathcal{L}\|_* = \Theta(n_{L+1}^2)$, again because the weights W^{L+1} are scaled up by n_L and the gradient is scaled down by the same amount. In the numerator, the upscaling of the weights also cancels out the downscaling of the gradient and additional n_L^{-1} in the subsequent forward pass, leading to an unchanged $\rho_{L+1} = \rho \cdot n_{L+1}^{-3}$, which is width-independent but potentially leads to numerical issues.

Code for μP^2 with separate denominator scalings. Algorithm 1 provides a PyTorch code example that implements the above μP^2 scalings for SAM, scaling the gradient norm contributions of all layers to $\Theta(1)$ (equivalent to (a-μP^2) together with naive perturbation and learning rate scaling). We adapt the popular SAM implementation Samuel (2022) using the `mup`-package. This code resembles our implementation for the ViT experiments. In the `mup`-package, 'vector-like´ parameters scale as $n \times$ constant or constant $\times\, n$ and include input and output weights. The last-layer multiplier n_L^{-1} is chosen so that input and output layers can be scaled by the same width-dependent factor. On the other hand, 'matrix-like´ parameters scale as $n \times n$ and include hidden weights. The implementation uses a base width at which μP^2 and SP are equivalent; all width-dependent scalings then scale with width-multipliers `width`/`base_width`. This allows to immediately transfer well-performing settings from SP to μP^2.

Let us recapitulate how the μP^2 scaling in the following code arises. The crucial variables to track are `factor`, `group["rho"]` and `group["gradnorm_scaling"]`. For limited batch size, the spectral and Frobenius norm of gradients scale equivalently, and we get, for all $l \in [L]$,

$$\|\nabla_{W^l}\mathcal{L}\|_F = \Theta(\|\nabla_{W^l}\mathcal{L}\|_*) = \Theta\left(\sqrt{\frac{\texttt{fan\_in}}{\texttt{fan\_out}}}\right).$$

We want to scale each weight's contribution in the denominator to be width-independent, hence need the factor $\sqrt{\texttt{factor}}$ with $\texttt{factor} = \texttt{fan\_out}/\texttt{fan\_in}$. For the numerator, the spectral condition ($*$) demands $\|\rho_l \cdot \nabla_{W^l}\mathcal{L}\|_* \overset{!}{=} \Theta(\sqrt{\frac{\texttt{fan\_out}}{\texttt{fan\_in}}})$, so that we need to scale the weight's perturbation radius to $\rho_l = \rho \cdot \texttt{factor}$. Since the `mup`-package sets the last-layer weight multiplier such that input and output layers can be scaled in the same way, the implementation is short. For optimal numerical properties however, this choice of multipliers is sub-optimal (Blake et al., 2024).

```python
import math, torch
from mup import MuAdamW

# specify parameterization
parameterization = 'mupp' # 'sp-naive', 'mup-naive'
# for 'mup-global' use 'mup-naive' and scale rho accordingly

# specify model and hyperparameters
model, lr, rho, weight_decay, last_layer_weight_name = ...



# adapt SAM to allow gradient norm scaling of each weight tensor
class SAM(torch.optim.Optimizer):
  ...

  def grad_norm(self):
    grads = []
    for i, group in enumerate(self.param_groups):
      for p in group["params"]:
        grads.append((group["gradnorm_scaling"] * p.grad).norm(p=2))
    norm = torch.stack(grads).norm(p=2)
    return norm

  @torch.no_grad()
  def first_step(self): # perturbation step before the weight update
    grad_norm = self.grad_norm()
    for group in self.param_groups:
      scale = group["rho"] / (grad_norm + 1e-12)
      for p in group["params"]:
        if p.grad is None: continue
        self.state[p]["old_p"] = p.data.clone()
        e_w = p.grad * scale.to(p)

        p.add_(e_w)  # climb to the local maximum "w + e(w)"



# set width-dependent rho and gradient norm scaling for each weight
param_groups = []
for name, p in model.named_parameters():
  if p.infshape.ninf() == 0 or 'naive' in parameterization:
    factor = 1
  elif p.infshape.ninf() == 1:
    # vector-like
    for d in p.infshape:
      if d.base_dim is not None:
        factor = d.dim / d.base_dim #width
        break
  elif p.infshape.ninf() == 2:
    # matrix-like
    factor = (p.infshape[0].dim/p.infshape[1].dim) * (p.infshape[1].base_dim/
    p.infshape[0].base_dim) # fan_out/fan_in
  else:
    raise NotImplementedError

  group = {
    "params": [p],
    "lr": lr,
    "rho": rho * factor,
    "gradnorm_scaling": math.sqrt(factor),
  }
  param_groups.append(group)


```

```
optimizer = SAM(param_groups,
  base_optimizer=MuAdamW if parameterization=='mup' else torch.optim.
    AdamW, weight_decay=weight_decay)
```

Algorithm 1: Pytorch implementation of μP^2 for SAM using the `mup`-package. Key changes from the original implementation that correct the layerwise perturbation scaling are highlighted with gray boxes. This code decouples the scalings of numerator and denominator terms following (DP), and scales the gradient norm contributions of all layers by `group["gradnorm_scaling"]` in the denominator to be width-independent. The numerator terms `group["rho"]` of all weight tensors are scaled to achieve effective perturbations. This scaling is equivalent to (a-μP^2) together with naive perturbation and learning rate scaling.

G Experimental details

If not mentioned otherwise, experiments use the settings specified in this section.

Implementation details. For MLPs, we exactly implement our Definition 4 of bcd-parameterizations to precisely validate our theoretical results. For ResNets and ViTs, the width varies inside the network, so that we implement the spectral scaling rules derived in Appendix F.7. Like the `mup`-package, we introduce a base width at which SP and μP are equivalent, allowing to immediately transfer setups that perform well in SP. We use the `mup`-package only for ViTs, and our implementation of μP^2 resembles the pseudocode provided in Algorithm 1. For ResNets, we use no width-dependent last-layer multiplier. At initialization, μP differs from SP only through a smaller last layer initialization. For MLPs we exactly implement the bcd-parameterization with $b_{L+1} = 1$. For ResNets and Vits, we initialize the last layer to 0 in μP, which corresponds to $b_{L+1} \to \infty$ and which recovers the limit behaviour $f_0 \to 0$ already at finite width. We are working on making Python code to reproduce all of our experiments publicly available.

MLPs. We train 3-layer MLPs without biases with ReLU activation function for 20 epochs with constant learning rate, using SGD as base optimizer as specified in Definition 4, but allow for SGD batchsize larger than 1, defaulting to batch size 64. We evaluate the test accuracy after every epoch and use the snapshot across training with the best accuracy. This is necessary as the test accuracy is not monotonically increasing across training, while the training accuracy is. For ResNets we do not observe such harmful overfitting. For the standard parametrization, we use He initialization (He et al., 2015) and don't tune multipliers to mimic standard training procedures. For μP, we resort to the optimal multipliers from Yang et al. (2022). We then find the optimal learning rate and perturbation radius for each bcd-parametrization and SAM variant separately.

ResNets. For ResNet18 experiments, we augment the CIFAR10 data with random crops and random horizontal flips, set labelsmoothing to 0.1 and use a cosine learning rate schedule. ResNets in μP have base width 0.5, gradient norm scaling according to Definition 4 and their last layer is initialized to 0. For SP, we again adopt the standard hyperparameters from Müller et al. (2024) by using a momentum of 0.9, weight decay 0.0005, an output multiplier of 1.0, and individually tuned learning rate and perturbation radius for each SAM variant. For μP, at base width multiplier 0.5 compared to the original width, for each SAM variant, we perform a random grid search over the hyperparameters learning rate, perturbation radius, output multiplier $[2^{-8}, 2^{-7}, \dots, 2^{8}]$, weight decay $[0, 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 10^{-2}]$ and momentum $[0, 0.1, 0.4, 0.7, 0.9]$. Learning rate and perturbation radius grids were either set to $[2^{-10}, 2^{-9}, \dots, 2^{1}]$ or centered around recommendations from the literature. The optimal hyperparameter configurations found from at least 150 runs for each SAM variant are summarized in Table G.1. Learning rates and perturbation radii were further tuned with the experiments from Appendix H.3.2.

ViTs. We train ViT-S/16 with 6 layers and 12 attention heads on ImageNet1K (Deng et al., 2009) and a ViT-S/4 with 12 layers and 12 attention heads on CIFAR100 (Krizhevsky et al., 2009) (see Appendix H.6), again adopting the hyperparameter settings from Müller et al. (2024). This means we use AdamW as a base optimizer with warmup and a cosine learning rate decay. For CIFAR100, we use random crops, random horizontal flips and AutoAugment as data augmentations. For Imagenet we use the original preprocessing from Huggingface `vit-base-patch16-224` (Wu et al., 2020). For μP, we tune multipliers at a basewidth 384, initialize the last layer and query weights to 0. By using the μP package, the relative perturbation scalings change as explained in Appendix F.7 and Appendix F.6. Global and naive perturbation scaling in μP now coincide. Here, instead of the

original perturbation scaling Definition 4, we scale the gradient norm contributions of all layers in the denominator to $\Theta(1)$. The hyperparameter choices for ViTs on CIFAR100 and ImageNet are summarized in Table G.2. For μP, the learning rate, perturbation radius, input multiplier, output multiplier and weight decay were tuned using 3 independent runs of Nevergrad `NGOpt` with budget 56 on ImageNet. The same multipliers are used on CIFAR100.

Figures. Whenever multiple runs with independent random seeds are used for training, confidence bands cover the interval from the empirical 2.5%- to the empirical 97.5%-quantile. The line then denotes the average of all runs. When confidence bands are given, but the number of independent runs is not specified, the number of runs defaults to 4.

Computational resources. We ran all of our experiments on Amazon EC2 G5 instances each containing up to 8 NVIDIA A10G GPUs. On a single GPU, our μP^2-SAM training script for MLPs of width 4096 on CIFAR10 takes 502 seconds to run in total (25 seconds per epoch), where data handling takes most of the time. The training times for ResNets and ViTs are presented in Table G.3.

Hyperparam.	SAM		SAM-ON		ResNet18	Elem. ASAM		Layer ASAM	
	SP	μP^2	SP	μP^2	SGD	SP	μP^2	SP	μP^2
Training epochs					200				
Batch size					64				
LR η	0.05	2^{-4}	0.05	2^{-4}		0.05	2^{-4}	0.1	2^{-4}
LR decay					Cosine				
Weight decay					0.0005				
Momentum					0.9				
Labelsmoothing					0.1				
Pert. radius ρ	0.1	2^{-4}	0.5	$5 \cdot 2^{-4}$		2	$10 \cdot 2^{-4}$	0.02	2^{-6}
Output multiplier	1	0.125	1	0.125		1	0.125	1	0.125

Table G.1: **(ResNet-18 hyperparameters for CIFAR10)** Hyperparameters for SP are taken from Müller et al. (2024). Learning rate and perturbation radius are tuned using the experiments in Appendix H.3.2. ResNets in μP have base width 0.5, gradient norm scaling according to Definition 4 and their last layer is initialized to 0.

Hyperparam.	SAM on ImageNet1K		shared	SAM on CIFAR100	
	SP	μP^2		SP	μP^2
Training epochs	100			300	
Batch size			128		
LR η	0.001	0.00226		0.0005	
LR warmup epochs	10			30	
LR decay			Cosine		
Weight decay	0.1	0.0872		0.05	
Labelsmoothing			0.1		
Pert. radius ρ	1	1.1939		0.25	0.25
Input multiplier	1	1.7309		1	1.7309
Output multiplier	1	4.0946		1	4.0946
Layers	6			12	
Attention heads			12		
Patch size	16			4	

Table G.2: **(Vision Transformer hyperparameters)** Hyperparameters for SP are taken from Müller et al. (2024) using AdamW as a base optimizer. ViTs in μP have base width 384, last layer and query weights are initialized to 0 and gradient norm contributions of all layers are scaled to $\Theta(1)$.

	ResNet-18 on CIFAR10				ViT on CIFAR100			ViT on ImageNet1K		
Width multiplier	0.5	1	2	4	0.5	1	2	0.5	1	2
Seconds per epoch	109	161	327	803	209	327	777	2550	4151	9802

Table G.3: **(Training time per epoch)** Training time (in seconds) per epoch of the entire data loading and training pipeline of SAM in μP^2 on a single NVIDIA A10G GPU.

H Supplemental experiments

This section provides more extensive empirical evaluations to validate the claims of the main paper. By naive perturbation scaling (naive) we denote parameterizations that do not adapt any perturbation scalings ($d = d_l = 0$ for all l). Global perturbation scaling (global) denotes the maximal stable scaling n^{-d} of the global perturbation radius that achieves effective perturbations in some layers without layerwise perturbation scaling ($d_l = 0$ for all l).

H.1 SAM is approximately LL-SAM in μP with global perturbation scaling

Figure H.1 compares SAM in μP under global perturbation scaling (μP-global) with SAM under global perturbation scaling where only the last-layer weights are perturbed (LL-SAM) by showing more neural network statistics that are related to SAM's inductive bias and to learning in general. From top-left to bottom right, the statistics are: Frobenius norm of the layerwise weight perturbation (which is closely related to spectral norm as perturbations are low rank); Frobenius norm of the layerwise weight perturbation normalized by the weight spectral norm to upper bound the influence of the perturbations on the output; spectral norm of the weight updates across training scaled by the spectral condition $n^{1/2}$, 1 and $n^{-1/2}$ for input, hidden and output layers respectively; norm of the activation updates for each layer normalized by the square root of the layer's output dimension to measure coordinatewise update scaling; layerwise effective feature ranks measured as in Andriushchenko et al. (2023a) by the minimal amount of singular values to make up 99% of the variance of the activations in a given layer; gradient norm, Hessian spectral norm and Hessian trace of loss with respect to weights; training accuracy, test accuracy after optimally stopping.

Observe that, especially for large widths, global perturbation scaling effectively only perturbs the last layer, as predicted by Theorem 11. Last-layer SAM is more similar to μP-global SAM than SGD on all of the tracked statistics, in particular at large widths. Only perturbing the last layer still affects the gradients in earlier layers so that weight updates and activations change in all layers. We find that SAM in μP with global scaling does not consistently improve generalization performance over SGD, whereas μP^2 does improve over SGD for all widths (Figure H.3). Last-layer perturbation norms coincide by design with the global perturbation radius $n^{-d}\rho$ and their effect on the activations stays $\Theta(1)$ with increasing width as measured in relation to weight spectral norm. Formally the last-layer perturbation norm converges due to

$$\|\tilde{W}^{L+1} - W^{L+1}\|_F = n^{-d}\rho \|\frac{\chi_t x_t^L}{\|v_t\|}\|_F \to n^{-d}\rho \|\frac{x_t^L}{\|x_t^L\|}\|_F = n^{-d}\rho \to 0,$$

where the loss derivative χ_t always cancels out due to the normalization and the global gradient norm $\|v_t\|$ is dominated by the last-layer gradient norm due to the global scaling (Theorem 11). Normalizing the weight perturbations by the weight spectral norm measures the influence of the perturbations on the activations. Note that this influence is also vanishing. Feature ranks stay close to initialization, since random initialization has high rank and training does low effective rank updates. Here we do not observe that SAM reduces the feature rank compared to SGD. The Hessian spectral norm and trace are quite noisy. The last-layer Hessian spectral norm explodes with width in μP, because last-layer learning rate is scaled as n^{-1}, hence the edge of stability explodes. ResNets in μP are more stable, their Hessian spectral norm even shrinks with width (not shown).

Contrast the results for μP-global with the results for μP^2 in Figure H.2 for a comparison with SGD in μP. The Hessian spectral norm is reduced by SAM as you would expect. Additionally μP^2 shows low variability in performance and all other statistics. SAM in μP^2 does not reduce the feature rank compared to SGD in μP. This suggests that the conclusions drawn by Andriushchenko et al. (2023a) do not apply to MLPs in μP.

H.2 Propagating perturbations from the first layer does not inherit SAM's benefits

Here we apply a parametrization that only effectively perturbs the first layer weights (derived in Example F.1). Figure H.2 shows that effective first-layer SAM loses both μP^2 SAM's improvement in test accuracy as well as SAM's inductive bias towards smaller gradient norm and Hessian norm, i.e. lower sharpness in MLPs. This performance deterioration occurs although the perturbation of first-layer SAM has an effect of the same order of magnitude as μP^2 on weight and activation updates in all layers. This shows that mere propagation of weight perturbations from earlier layers cannot

replace effective weight perturbations in each layer in order to benefit from SAM. It is crucial to correctly adjust the layerwise perturbation scaling, and to distinguish between effective perturbations and perturbation nontriviality in each layer.

SAM in μP^2, on the other hand, achieves the correct perturbation and update scaling, has lower final gradient and Hessian spectral norm, improves test accuracy over SGD and has overall lower variance between training runs.

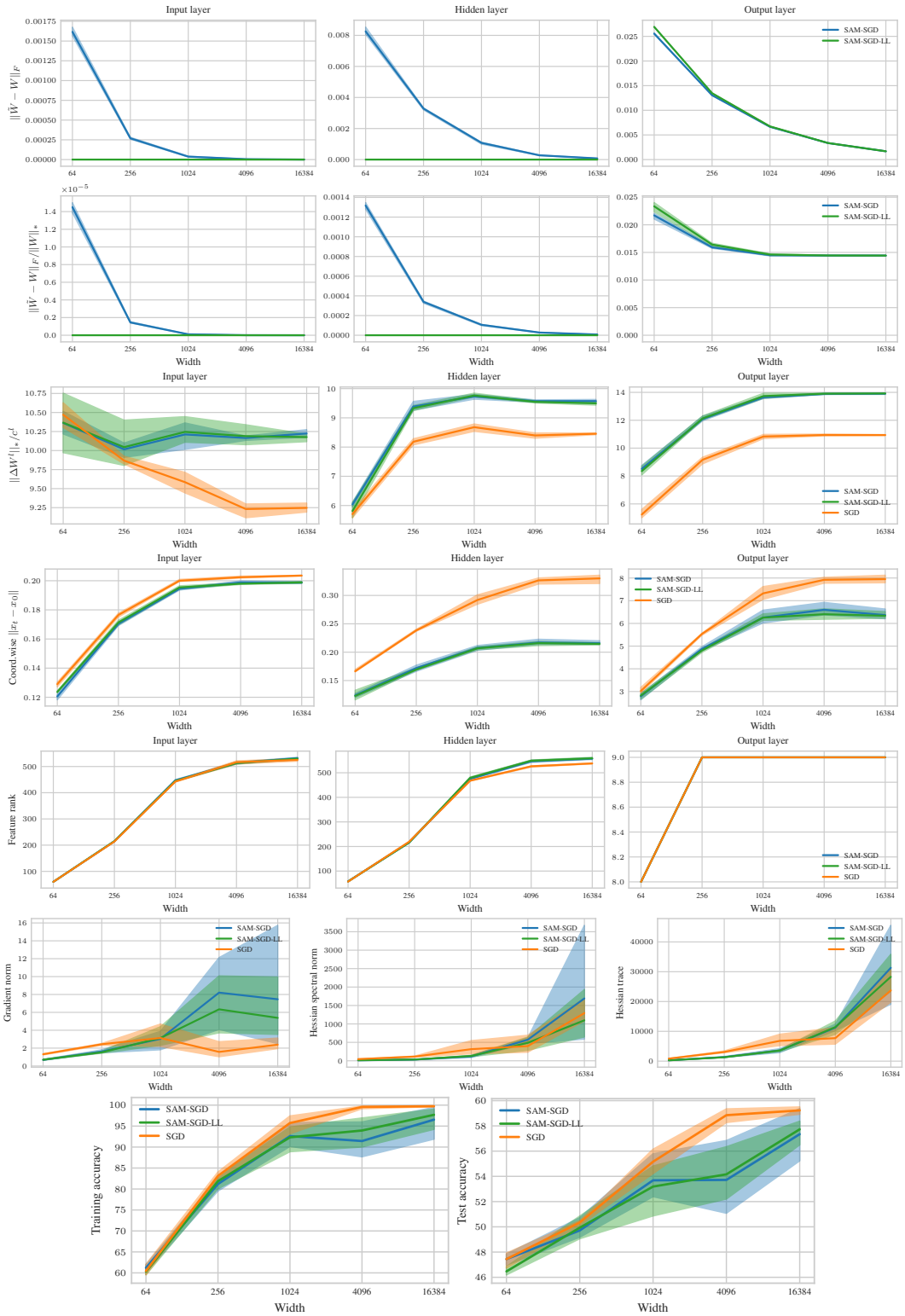


Figure H.1: Several neural network statistics for SAM (blue), LL-SAM (green) and SGD as a baseline (orange) across width after training a 3-layer MLP in μP-global for 20 epochs with the optimal learning rate 0.3432 and perturbation radius 0.2154. The statistics are explained in the text of Appendix H.1.

Figure H.2: Same neural network statistics as in Figure H.1 but SAM-SGD in μP^2 (blue) versus MUP with perturbations scaled to only effectively perturb the first layer weights (green) with SGD in μP as a baseline. The first-layer perturbation parameterization performs worse than μP^2 and results in gradient norm and Hessian norm similar to that of SGD, larger than those of SAM. While the spectral norm of the weights converges to a similar quantity as for μP^2, the effect of the weight changes on the hidden activation updates behaves more like SGD. Feature ranks all look similar.

H.3 Hyperparameter transfer

In this section, we provide supplemental evidence that μP^2 is the unique parameterization that robustly achieves hyperparameter transfer both for the optimal learning rate and the optimal perturbation radius across neural architectures and datasets.

H.3.1 MLPs

Figure H.3 shows that in μP^2 the optimal hyperparameters in terms of test accuracy transfer in both learning rate and perturbation radius at sufficient width. In μP^2, SAM improves over SGD more than in SP, and the overall best test accuracy is achieved with the widest MLPs in μP^2.

While other works focus on hyperparameter transfer in training loss, we are ultimately interested in transfer with respect to test accuracy. Especially under harmful overfitting, the test accuracy is affected by nontrivial interactions between the learning rate and the perturbation radius. While the joint optimum is slightly shifting towards larger learning rate and perturbation radius for small widths, it remains remarkably stable for sufficient width ≥ 1024. Note that slight shifts in the optimal learning rate due to finite width biases have also been observed in earlier works (Yang et al., 2022).

Figure H.4 shows that global perturbation scaling does transfer the same perturbation instability threshold, whereas in μP-naive every fixed perturbation radius becomes unstable at sufficient width (Figure H.7). But in μP-global we do not observe a benefit of SAM over SGD. While the optimal learning rate with respect to the training accuracy transfers, the optimal learning rate with respect to the validation error is smaller for MLPs of moderate widths due to harmful overfitting. How to control for nonmonotonic dependence of the test error on the training error is an important question for future work. Figure H.5 also shows μP-global but with a different choice of input and output multipliers. With these multipliers, networks with width at most 256 perform better in terms of test accuracy than with the other multiplier choice in Figure H.4, but these multipliers have worse width scaling properties. To the best of our knowledge, the issue that optimally tuned hyperparameters on small models may scale worse than slightly suboptimal hyperparameters has not been stated before. This raises the question when and how can we use small models to predict the optimal choice of all hyperparameters jointly in large models.

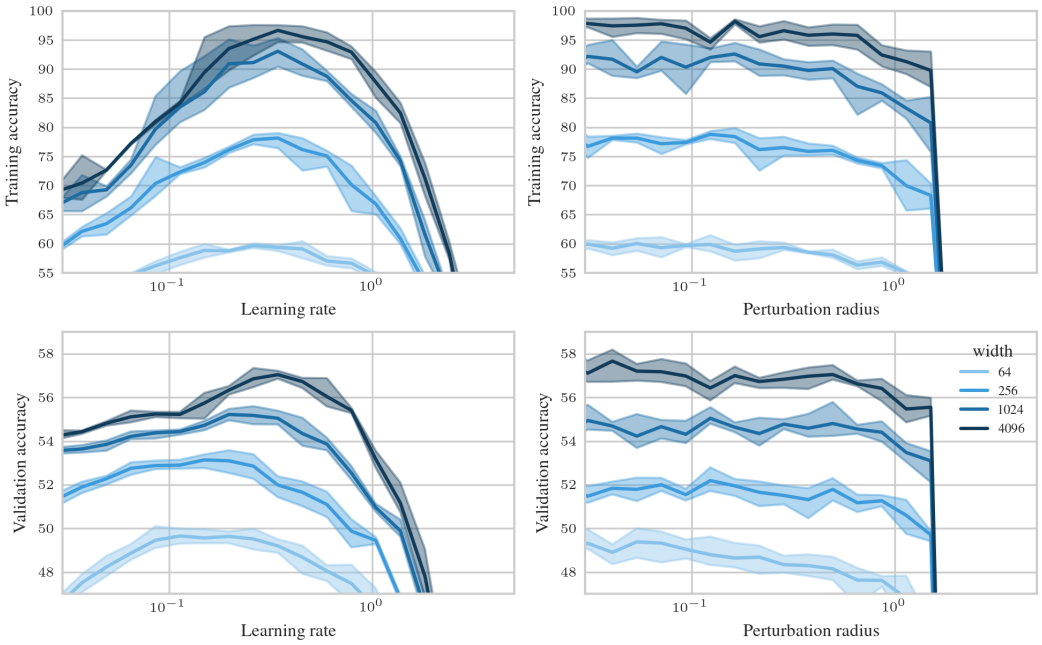


Figure H.4: Training accuracy (top) and test accuracy (bottom) after optimally stopping 20 epoch SAM training as a function of learning rate (left) and perturbation radius (right) in μP-global with the same base learning rate and perturbation radius as in Figure H.9. For global perturbation scaling, we do not observe a benefit of SAM over SGD.

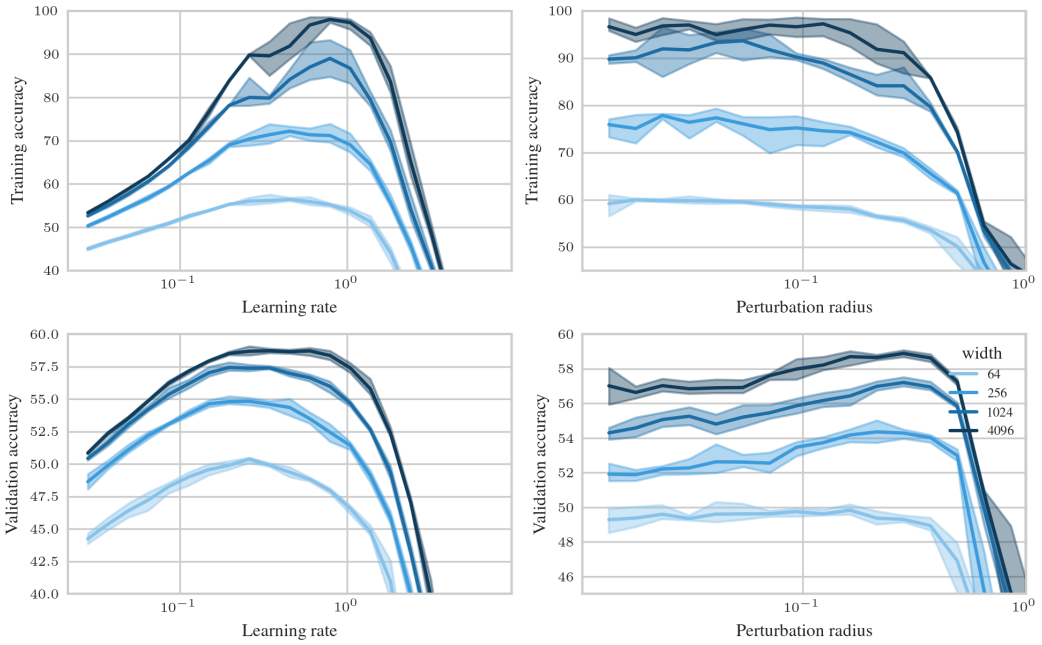


Figure H.3: Training accuracy (top) and test accuracy (bottom) after optimally stopping 20 epoch SAM training as a function of learning rate (left) with perturbation radius $\rho = 0.2154$, and as a function of perturbation radius (right) with learning rate $\eta = 0.4529$ in μP^2. The optimal learning rate transfers. The smaller the perturbation radius the better the training accuracy. For sufficiently wide MLPs, the validation-optimal perturbation radius transfers as well and SAM reduces harmful overfitting.

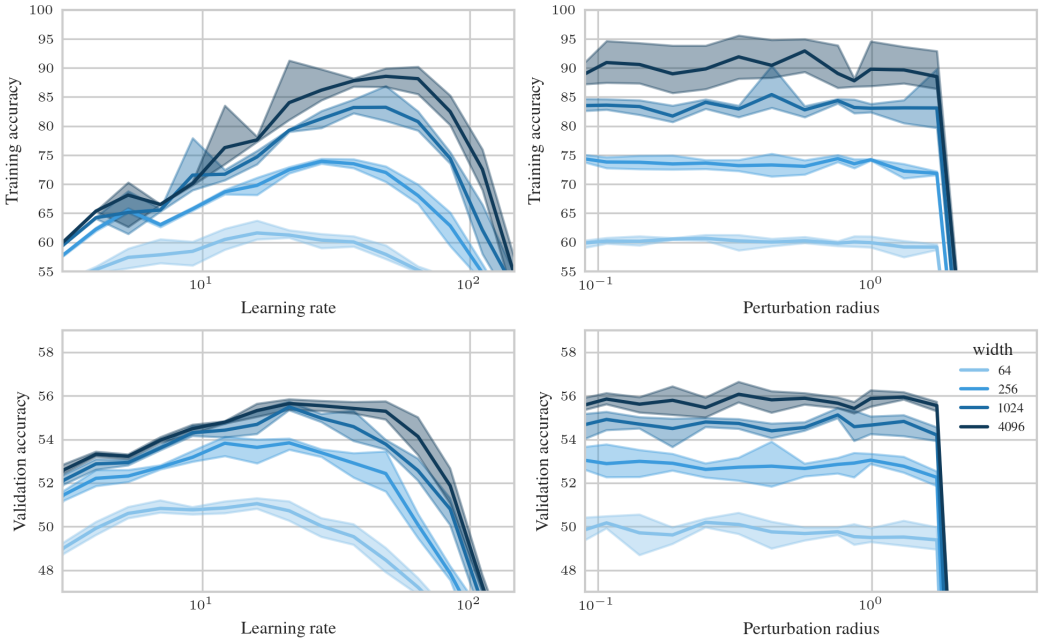


Figure H.5: Same as Figure H.4 but with input multiplier 0.0305 and small output multiplier 0.0098. Note that networks with width at most 256 perform better in terms of test accuracy than with the other multiplier choice in Figure H.4, but the multipliers here have worse width scaling properties. To the best of our knowledge, the issue that optimally tuned hyperparameters on small models may scale worse than slightly suboptimal hyperparameters has not been stated before. This raises the question when and how can we use small models to predict the optimal hyperparameters of large models.

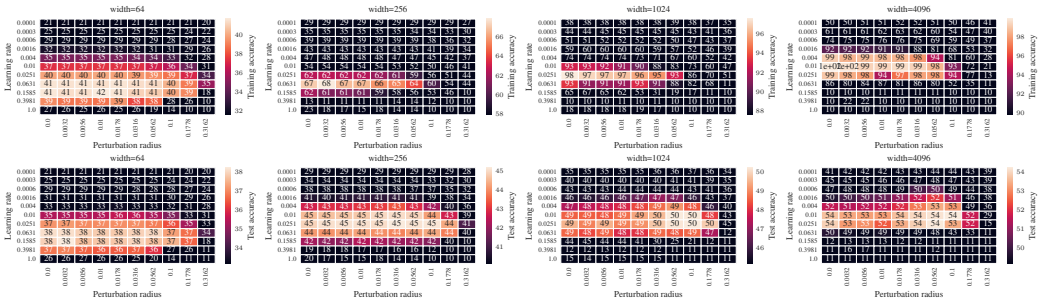


Figure H.6: Mean (over 3 runs) of training accuracy (top) and test accuracy (bottom) after optimally stopping 20 epoch SAM training of a MLP in SP-naive as a function of learning rate and perturbation radius. Neither the optimal learning rate nor the optimal perturbation radius transfers. Every fixed learning rate becomes unstable in sufficiently wide networks. Optimal training and test accuracy are reached on differing hyperparameters due to harmful overfitting in SGD.

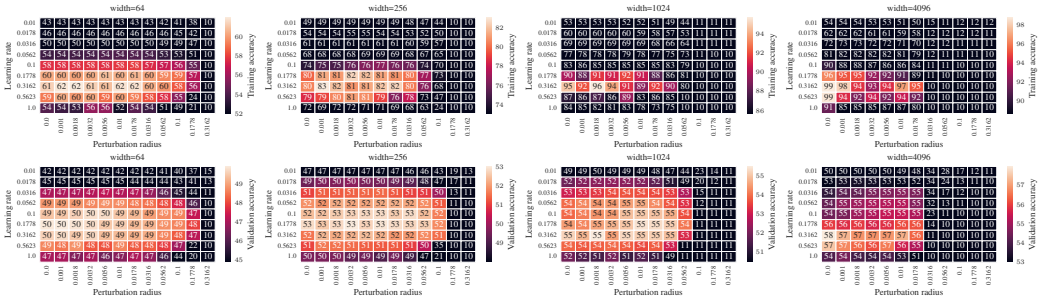


Figure H.7: Mean (over 3 runs) of training accuracy (top) and of test accuracy (bottom) after optimally stopping 20 epoch SAM training of a MLP in μP-naive as a function of learning rate and perturbation radius. The optimal hyperparameters do not transfer. Every fixed perturbation radius becomes unstable in sufficiently wide networks.

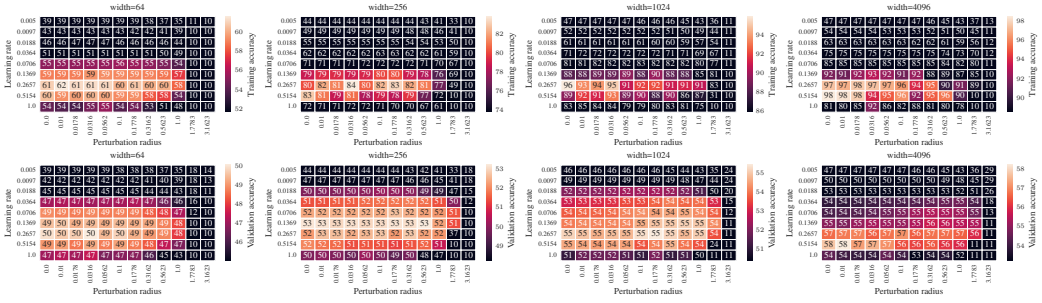


Figure H.8: Mean (over 3 runs) of training accuracy (top) and of test accuracy (bottom) after optimally stopping 20 epoch SAM training of a MLP in μP-global as a function of learning rate and perturbation radius. The global scaling of the perturbation radius by $n^{-1/2}$ compared to μP-naive (Figure H.7) makes the stable regime invariant to width. But the suboptimal layerwise perturbation scaling that only perturbs the last layer does not consistently improve over SGD ($\rho = 0$).

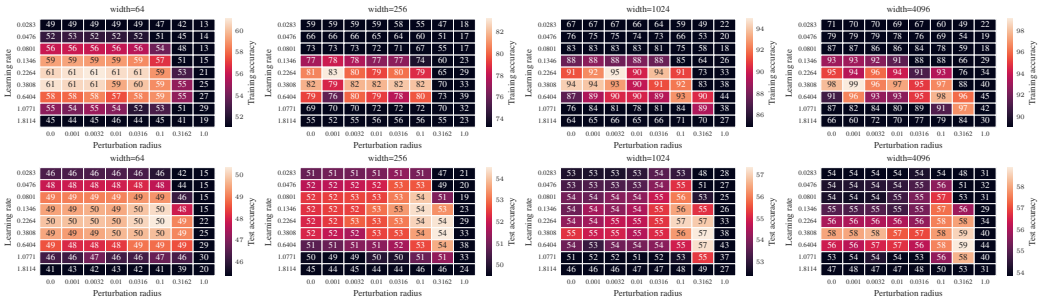


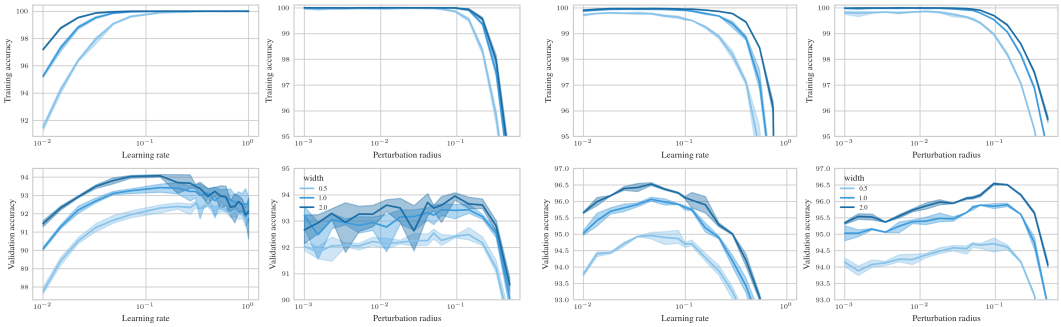
Figure H.9: Mean (over 3 runs) of training accuracy (top) and of test accuracy (bottom) after optimally stopping 20 epoch SAM training of a MLP in μP^2 as a function of learning rate and perturbation radius. At sufficient width, the optimal hyperparameters are stable in terms of test accuracy, even under severe overfitting.

H.3.2 ResNets

In this section, we plot averages and σ-CI from 2 independent runs.

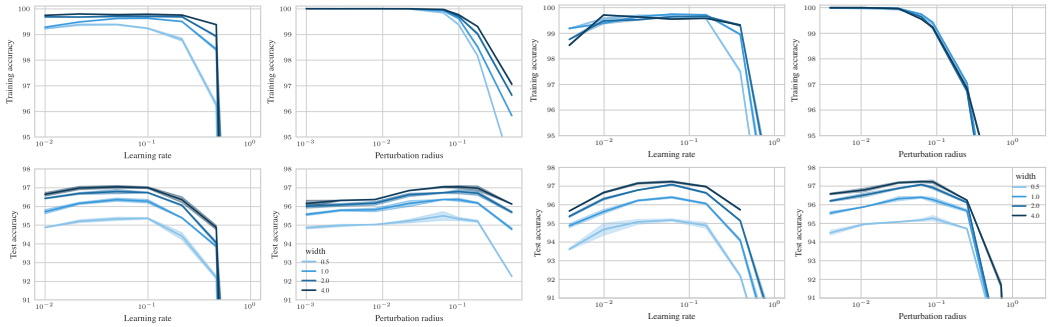
ResNets in μP^2 transfer both the optimal learning rate and perturbation radius for SAM (Figure H.11), SAM-ON (Figure H.13) and elementwise ASAM (Figure H.14), as well as different alternatives of scaling the gradient norm contributions to SAM's denominator (Figure H.17). This suggests correctness of the derived scalings. At width multipliers 2 and 4, μP^2 achieves the same or slightly better test accuracy than SP in all SAM variants.

Figure H.12 shows ResNets trained with SAM in different parameterizations. In ResNets of practical scale, ρ remains quite stable in μP^2 but surprisingly also in SP-NAIVE. In μP, for naive perturbation scaling the regime of stable perturbation radii shrinks, for global perturbation scaling, the optimal perturbation radius shifts, approaching its maximal stable value, which stays invariant to width scaling. Here, it would be interesting to see whether even larger width would lead to suboptimal performance of μP-global. μP^2 is most robust to the choice of ρ and achieves the best test accuracy. Surprisingly, ResNets in SP have very stable hyperparameter transfer across most SAM variants too, as soon as we tune momentum, weight decay and labelsmoothing (Figure H.10). This is in line with previous empirical observations (Figure 16, Yang et al., 2022) but contradicts pure infinite-width theory. Because we are training to convergence, pure infinite-width theory does not adequately describe the training dynamics anymore (Vyas et al., 2024). We plan to study this phenomenon in more detail in upcoming work. The infinite-width theory implies that scaling the width further would eventually break the learning rate transfer.



(a) No momentum, weight decay or labelsmoothing (b) Tuned momentum and weight decay

Figure H.10: Training accuracy (top) and test accuracy (bottom) after optimally stopping 100 epoch SAM training as a function of learning rate and perturbation radius in SP-naive without regularization (left) and with tuned regularization (right) using momentum 0.9, weightdecay 0.0005 and labelsmoothing 0.1. CI denote the minimal and maximal value from 4 independent runs. Without regularization, the optimal learning rate shrinks with width. Given the learning rate, the optimal perturbation radius seems quite stable, but since the optimal learning rate shifts, the performance scales worse than for μP^2 with the fixed learning rate that is tuned on the small model. With optimal regularization, both optimal learning rate and perturbation radius remain remarkably stable. We plan to investigate this mechanism in an upcoming work.



(a) SP, no ρ scaling (b) μP^2

Figure H.11: Training accuracy (top) and test accuracy (bottom) after optimally stopping 200 epoch SAM training as a function of learning rate and of perturbation radius in SP (left) and in μP^2 (right) with optimized momentum 0.9, weight decay $5 \cdot 10^{-4}$ and labelsmoothing 0.1 for both μP^2 and SP. In μP^2, the base learning rate is $\eta = 2^{-4}$ and the base perturbation radius is $\rho = 2^{-4}$, in SP $\eta = 0.05$ and $\rho = 0.1$, respectively. Observe monotonic improvement with width in both training and test error. Optimal hyperparameters transfer across widths, surprisingly in both μP^2 and SP.

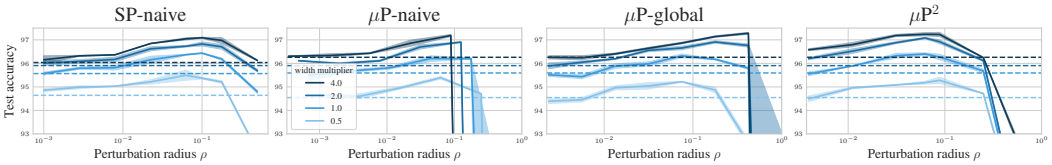


Figure H.12: Test accuracy after optimally stopping 200 epoch SAM training as a function of perturbation radius in various parameterizations. Dashed lines denote the base optimizer SGD with tuned momentum and weight decay in the respective parameterization.

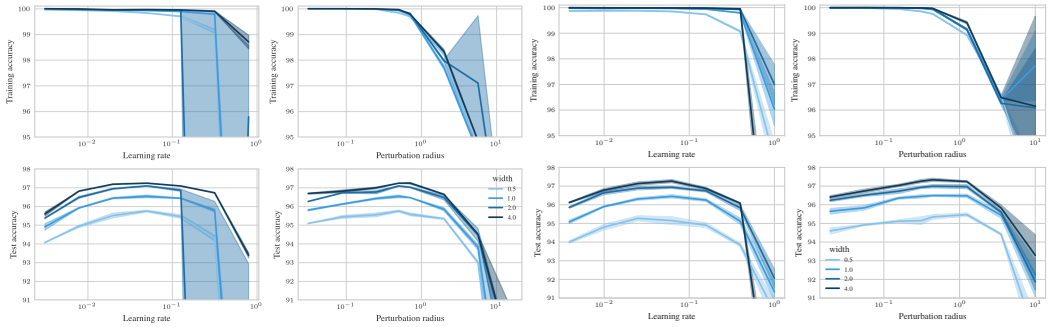
H.3.3 ASAM variants

As we are not aware of any use of ASAM with MLPs in the literature and since the amount of necessary experiments for ViTs exceeds our computational budget, we only show that ResNets trained with the all of the discussed SAM variants in μP^2 transfer the optimal (η, ρ).

For the examples of elementwise ASAM and SAM-ON the global perturbation scaling $n^{1/2}$ suffices to reach μP^2. The stability of the optimal perturbation radius in the applied scaling $n^{1/2}$ shows that in μP with naive perturbation scaling the optimal perturbation radius would grow as $n^{1/2}$.

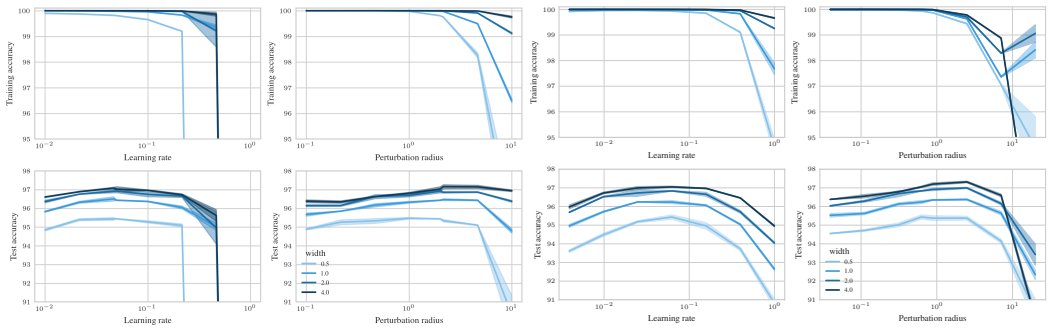
See the previous section, for a discussion of the remarkable stability of ResNets in SP. For the example of elementwise ASAM in SP, the optimal perturbation radius seems to grow.

For layerwise ASAM (Figure H.15), the optimal perturbation radius seems to grow in both SP and μP^2, suggesting that our scaling condition does not perfectly apply to this variant, although μP^2 ($97.09_{\pm 0.03}(+0.83)$) still outperforms SP ($96.86_{\pm 0.05}(+0.83)$) in terms of the optimal test accuracy. As Frobenius norms of weights are the only component that is not representable as a NE$\otimes$OR$\top$program, these Frobenius norms appear to scale differently than heuristically predicted over the course of training.



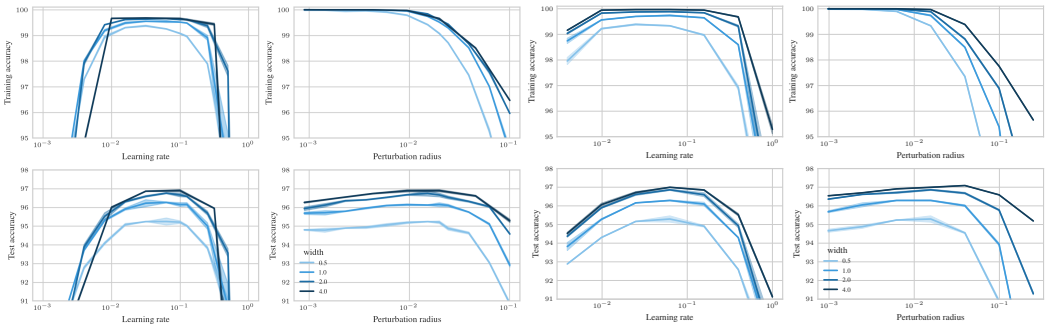
(a) SP, no ρ scaling (b) μP^2

Figure H.13: Same as Figure H.11 but for SAM-ON in SP without perturbation scaling (left) and in μP^2 (right). Both optimal learning rate and perturbation radius are remarkably stable in both μP^2 and SP. Since μP^2 for SAM-ON is just μP with global perturbation scaling $n^{1/2}$, transfer here implies that μP with width-independent scaling would not transfer.



(a) SP, no ρ scaling (b) μP^2

Figure H.14: Same as Figure H.11 but for elementwise ASAM in SP without perturbation scaling (left) and in μP^2 (right). Observe a consistent HP landscape in μP^2 but growing optimal perturbation radius in SP without perturbation scaling.



(a) SP, no ρ scaling (b) μP^2

Figure H.15: Same as Figure H.11 but for layerwise ASAM in SP without perturbation scaling (left) and in μP^2 (right). For layerwise ASAM, both μP^2 and SP seem to transfer the optimal learning rate as well as perturbation radius.

H.4 Gradient norm contributions have negligible effects on generalization performance

In this section we provide ablations concerning the question which layers should contribute non-vanishingly to the gradient norm in the denominator of the layerwise SAM perturbation rule (LP).

For MLPs, in Figure H.16 we scale all contributions to $\Theta(1)$, and then set the contribution of individual layers to zero, one by one. We observe no significant effect on the optimal test loss or hyperparameter transfer for MLPs. Any layer's contribution to the gradient normalization in the denominator of the SAM update rule can be set to 0 without a significant effect on the test loss. This raises the question which effect the gradient normalization has in μP. Does it contribute a scaling correction in SP, but may be dropped entirely in μP?

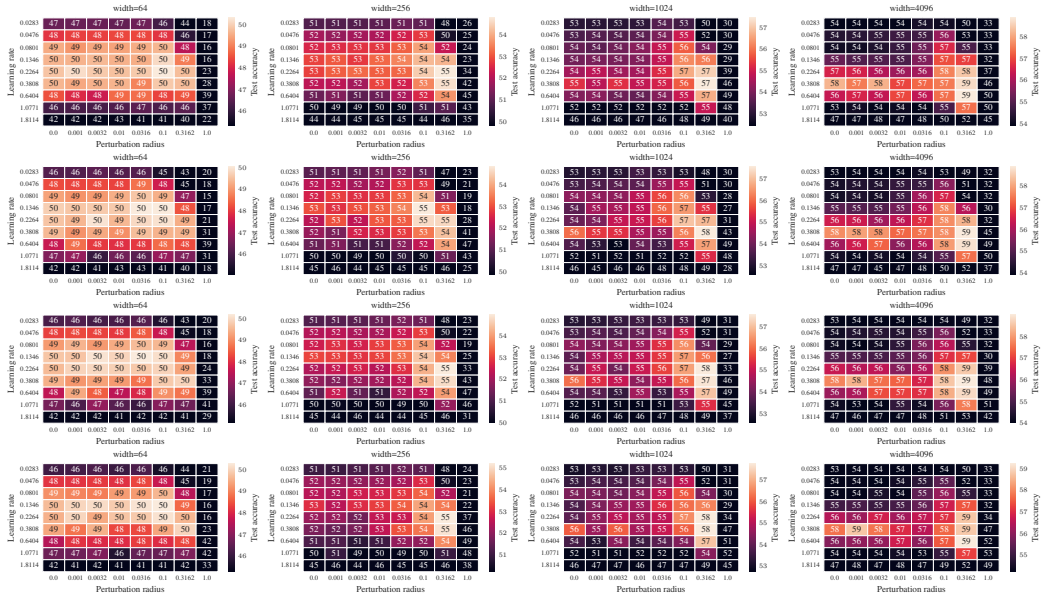
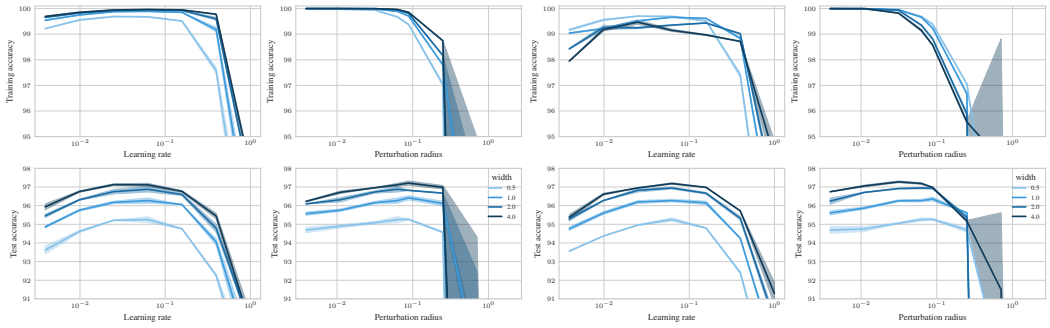


Figure H.16: Scaling the gradient norm contributions of all layers to $\Theta(1)$ (first row) and then setting the first layer gradient norm to 0 (2nd row), respectively the hidden layer (3rd row), last-layer (4th row). Each individual layer seems to have vanishing contribution to the optimal test error.

For ResNets, Figure H.17(a) shows accuracies when rescaling all layers' gradient norms to $\Theta(1)$, and Figure H.17(b) shows the results when using the original global gradient norm rescaled to $\Theta(1)$. Again, both methods achieve similar optimal test accuracy. The first variant shows cleaner hyperparameter transfer and monotonous improvement with width. When comparing to our original definition (LP) in Figure H.11, optimal performance is similar but rescaling all layers' gradient norm

contributions to $\Theta(1)$ may even produce a slightly more stable hyperparameter-loss landscape for ResNets.



(a) Rescaling all of SAM's denominator terms to $\Theta(1)$ (b) Global $\|\nabla L\|$ scaling

Figure H.17: Same as Figure H.11 but with scaling of the gradient norms in the SAM perturbation (LP) denominator that scales all terms to $\Theta(1)$ (left) and only global denominator scaling $\frac{\|\nabla L\|}{n_L}$ (right). All denominator scalings achieve similar optimal accuracy, show HP transfer in learning rate and monotonic test accuracy improvement with width. In global denominator scaling, the optimal ρ shifts with width.

H.5 SAM with layerwise gradient normalization

Here we consider SAM without the gradient normalization over all layers jointly. Instead we apply the layerwise perturbation rule presented in Appendix F.7,

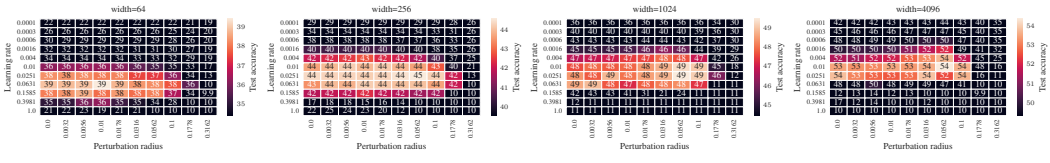
$$\boldsymbol{\epsilon}_t^l = \rho_l \cdot \nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t) / \|\nabla_{W^l} \mathcal{L}(f(\xi_t; W_t), y_t)\|_F.$$

In SP, we consider a global constant $\rho_l = \rho$, whereas for μP^2 the spectral condition $(*)$ requires $\rho_l = \rho \cdot \sqrt{\texttt{fan\_out}/\texttt{fan\_in}}$.

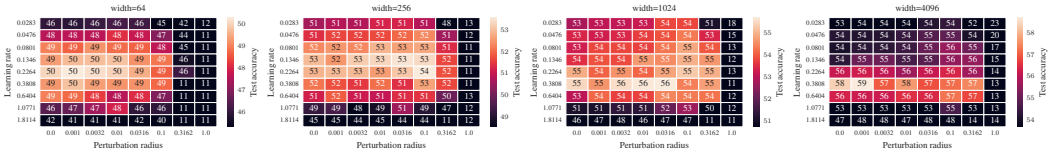
Overall, SAM without layer coupling performs decently, but is outperformed by the original SAM in particular in ResNets, in μP^2 and at large width. But note that for ResNets we adopt the hyperparameters tuned for the original SAM with layer coupling, so that these ablations only serve as preliminary experiments.

MLPs. In SP, we observe very similar performance as under the original SAM with layer coupling (Figure H.6). This may be because the last layer dominates the perturbation in both versions of SAM.

SAM without layer coupling achieves similar optimal generalization in μP^2 at each width compared to Figure H.9. The regime of stable (η, ρ) stays width-independent, but does not transfer the optimum consistently. This suggests complex or noisy dependence of the training dynamics on ρ.



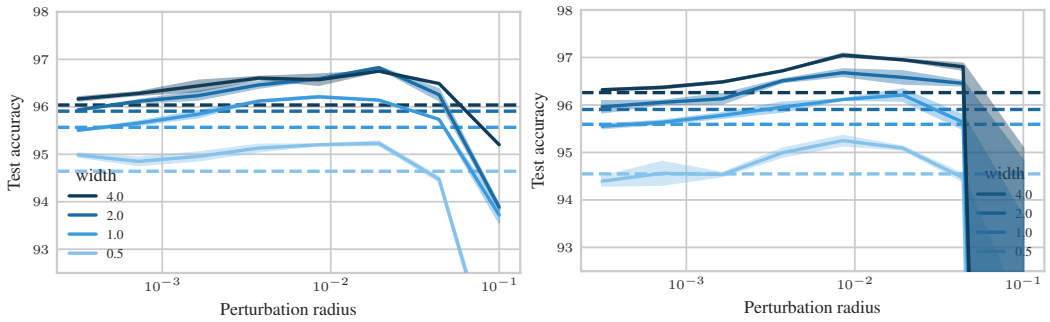
(a) SP



(b) μP^2

Figure H.18: **(SAM with layerwise normalization in MLPs)** Test accuracy as a function of learning rate η and perturbation radius ρ for an optimally-stopped MLP trained with SAM with layerwise normalization.

ResNets. Figure H.19 shows that decoupled SAM has decent performance, but is worse than original SAM with global normalization (Figure H.12) in both SP and μP^2, in particular at large width. As expected, ρ transfers in μP^2.



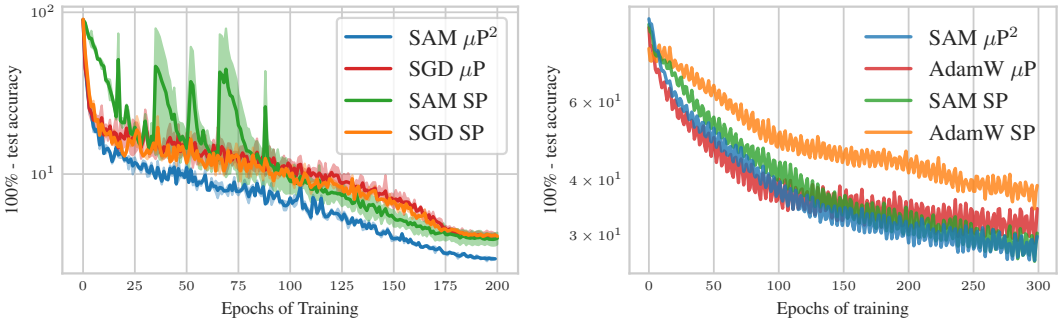
(a) SP (b) μP^2

Figure H.19: **(SAM with layerwise normalization in ResNets)** Test accuracy as a function of perturbation radius ρ for ResNets trained with SAM with layerwise normalization.

H.6 Test error over the course of training

Figure H.20 shows the test error of ResNets and ViTs over the course of training. μP^2 always achieves the best final test accuracy. In ResNets it also achieves a decent test accuracy the fastest and removes training instabilities of SAM in SP. While SGD in μP alone cannot compete with SAM in SP, SAM in μP^2 uniformly dominates over the entire course of training. Our theory suggests that in μP^2 the gradients are scaled correctly from the beginning, whereas in SP they have to self-stabilize first, which slows down convergence. We plan a closer analysis in an upcoming work.

In ViTs, μP generally achieves decent accuracy faster than SP, since gradient norms are already scaled correctly at initialization. SAM converges slower than the base optimizer AdamW in favor of drifting towards a better generalizing local minimum or saddle point. For ViTs at this moderate scale, SAM in SP catches up to SAM in μP^2 at the end of training.



(a) ResNet on CIFAR10 (b) ViT on CIFAR100

Figure H.20: Training a ResNet-18 with width multiplier 2 on CIFAR10 (left) and a ViT with width multiplier 2 on CIFAR100 (right). SGD and AdamW are the respective base optimizers.

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: In the abstract and introduction we state our main contributions while acknowledging related work. All main claims are theoretically proven and/or empirically verified.
   Guidelines:
   - The answer NA 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 NA 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: Limitations are discussed in the future work section as well as in the section in the appendix that is related to the respective limitation.
   Guidelines:
   - The answer NA 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: We state all assumptions in the main paper and Appendix C, and provide all formal proofs in Appendix E.
   Guidelines:
   - The answer NA 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: All experimental details are disclosed in Appendix G. Our perturbation scaling rules are clearly stated in the main paper. Their implementation with flexible `fan_in` and `fan_out` is explained in Appendix F.7, together with pseudocode for implementing our proposed scaling rule.

   Guidelines:

   - The answer NA 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: [No]

   Justification: We only propose width-dependent scaling of hyperparameters of an existing optimization algorithm. This can be easily implemented by following the scaling rules that we clearly specify in the main paper. In Appendix F.7 we even provide a code example that

contains the essential modifications.We are working on making Python code to reproduce all of our experiments publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental details are disclosed in the main paper or Appendix G.

   Guidelines:

   - The answer NA 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: As stated in Appendix G, we repeat all main experiments with multiple independent runs and report confidence bands within the empirical 2.5%- to 97.5%-quantiles. When we repeat experiments on Vision Transformers that we have also conducted on MLPs or ResNets, we do not use multiple runs due to limitations in computational resources.

   Guidelines:

   - The answer NA 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: We provide the type of GPU used and number of GPU seconds required for each experiment in Appendix G.
   Guidelines:
   - The answer NA 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 provide a theoretical analysis of a widely used optimization algorithm, point out the algorithm's limitations in large models and propose a correction. We do not foresee any ethical concerns.
   Guidelines:
   - The answer NA 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: [NA]
    Justification: This paper provides fundamental research toward understanding and improving existing optimization algorithms for neural networks. We do not release any model or data and do not consider generative models.
    Guidelines:
    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We only use standard vision architectures and vision datasets in our experiments and do not release any data or models.

    Guidelines:

    - The answer NA 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 cite the standard CIFAR10, CIFAR100 (Krizhevsky et al., 2009) and ImageNet1K (Deng et al., 2009) datasets following the standard practice. We also cite the Python assets `PyTorch` (Paszke et al., 2019), `mup` (Yang et al., 2022) and the GitHub repository implementing SAM (Samuel, 2022) that we use as a basis for our experiments.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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.