# Learning Rate Transfer and Feature Learning Across Depth for Constrained Spectral Optimizers: Complete Scion

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Hyperparameter transfer across model depth is instrumental in training deep models, as it enables learning rates calibrated on shallow networks to be effectively applied to deeper architectures. The recent CompleteP framework establishes a $1/L$ residual scaling rule that demonstrates this for Adam-family optimizers. We study whether similar transfer extends to constrained optimization methods, specifically Frank-Wolfe and projected non-Euclidean normalized steepest descent, where updates are determined by linear minimization oracles. We derive scaling rules for these methods which confirm that the $1/L$ residual scaling applies, although the arguments differ from the Adam-family optimizers. Our analysis supports depth-wise transfer and width-wise transfer of the learning rate for the Scion algorithm while ensuring feature learning and avoiding the kernel-regime. In experiments on NLP transformers (nanoGPT on FineWeb), vision transformer (DeiT on ImageNet), and vision MLPs (MLP-Mixer on CIFAR-10), we observe learning rate transfer across depth and feature learning for appropriately chosen residual scaling.

## 1 Introduction

Since their successful use in ImageNet (Krizhevsky et al., 2012) deep neural networks have been increasingly used in complex tasks such as image classification (Simonyan & Zisserman, 2014; He et al., 2016; Szegedy et al., 2015), speech recognition (Hinton et al., 2012; Graves et al., 2013; Amodei et al., 2015), and natural language processing (Sutskever et al., 2014; Vaswani et al., 2017; Devlin et al., 2019), where they have led to significant performance gains across benchmarks and applications. They are deployed for both difficult and critical applications, including medicine (Esteva et al., 2017; Topol, 2019), aeronautics (Julian & Kochenderfer, 2019; Sun & Wang, 2019), and nuclear engineering (Tang et al., 2022; Hu et al., 2021). Performance on complex tasks has been observed to increase with model scale, motivating the use of larger and heavier neural architectures, involving a high number of parameters.

There are many ways to scale deep neural networks, with the most common being increase its width or depth. However, increased performance is not the only consequence of scale. Indeed, as the scale increases so does the cost of training and tuning the models, making the ability to reuse hyperparameters across scales a pressing practical concern.

The Maximal Update Parametrization ($\mu$P) framework (Yang & Hu, 2021; Yang & Littwin, 2023) provides techniques to ensure that optimal hyperparameters remain invariant when the width of the model increases. This allows optimal hyperparameters found by tuning small scale models to be reliably transferred to larger ones. Figure 1 provides practitioners a step-by-step framework for implementing and using the learning rate transfer. A well-designed parameterization guarantees that the optimal hyperparameter remains invariant across scales. Additionally, $\mu$P prescribes parameter and learning-rate scalings that preserve non-degenerate *feature learning* dynamics, in which the internal representations evolve to fit the data (Yang & Hu, 2021). This avoids the *lazy learning* regime where the model behaves like its linearization (Chizat et al., 2019). However, the optimizer landscape has evolved significantly beyond the Stochastic Gradient Descent (SGD) and Adam (Kingma, 2014) updates originally considered in $\mu$P, leaving a gap.

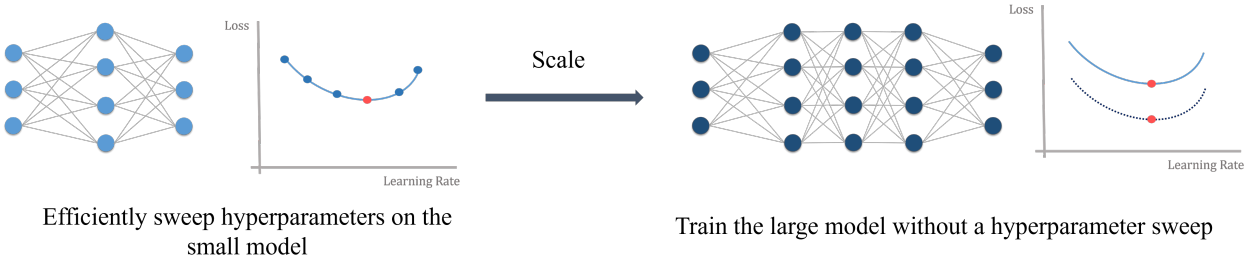



Figure 1: Practical learning rate transfer (Yang & Hu, 2021). Define a small, computationally inexpensive version of the model (i.e. with small dimensions) and run a hyperparameter sweep to find the optimal learning rate $\gamma^*$. Then, train the large-scale target model directly using the optimal $\gamma^*$, bypassing expensive tuning at scale.

Despite being introduced more than ten years ago (Carlson et al., 2015a;b;c), optimizers based on the spectral norm have recently reemerged as strong alternatives to standard coordinate-wise methods. In particular, Scion (Pethick et al., 2025b) (which also has an unconstrained variant) is a Conditional-Gradient-based algorithm Frank & Wolfe (1956) that relies on a Linear Minimization Oracle (LMO) over norm-ball constraints and explores the choice of a norm across layers. Varying the norm at each layer induces layer-wise update magnitudes, which control the relative update scales between layers. Updates based on the spectral norm have also attracted attention through their use in the Muon optimizer (Jordan et al., 2024b), which has gained popularity due to its performance in speed-running benchmarks (Jordan et al., 2024a) and training industrial-scale large language models like the Kimi, Deepseek, and the Trinity series models (Kimi Team et al., 2025; 2026; DeepSeek-AI, 2026; Singh et al., 2026). Unlike SGD or Adam, these methods precondition the update to weight matrices in a layer-wise manner using matrix-level operation such as the matrix sign, rather than using coordinate-wise scaling. As a result, the update structure is fundamentally different, meaning that the standard $\mu$P parametrization and the Tensor Programs (TP) analytic framework (Yang & Hu, 2021; Yang et al., 2023b) do not directly apply.

While $\mu$P provides a principled framework for width-wise hyperparameter transfer, it does not directly address the effect of increasing depth. Several extensions have explored depth-wise hyperparameter transfer (Yang et al., 2023b; Bordelon et al., 2024b; Dey et al., 2026; Qiu et al., 2025). Among them, CompleteP (Dey et al., 2026) identifies three desiderata (stable initialization, stable feature updates, and a non-lazy learning regime, also referred to as the Neural Tangent Kernel regime, where features do not evolve during training (Yang et al., 2021; Dey et al., 2026)) that enable joint width-and-depth-wise transfer. However, existing instantiations of CompleteP focus on SGD-like optimizers and do not cover spectral-norm-based update rules.

In this work, we study the compatibility of the CompleteP parameterization with Scion, referring to their combination as Complete Scion, and show that it satisfies all the desiderata outlined in CompleteP. More precisely, we show that the learning rate tuned at small scale transfer to larger models without additional tuning, assuming a residual scaling of the form $L^{-\alpha}$ with $\alpha \in [1/2, 1]$ and a learning rate scaled as $\gamma \propto L^{\alpha-1}$, for both Scion and Unconstrained Scion.

Our theoretical analysis establishes both width-wise and depth-wise hyperparameter transfer. For depth scaling, we confirm that the CompleteP desiderata (stable initialization and stable feature updates) hold for Scion and Unconstrained Scion, establishing depth-wise learning rate transfer. We show that stable feature updates are achieved in the $\alpha = 1$ regime. We emphasize that, while our final optimization objectives align with the CompleteP framework, our derivations and proofs differ. Existing analyses in CompleteP focus on well-normalized coordinate-wise optimization methods such as Adam, whose updates are applied independently across parameters. In contrast, we focus on spectral optimizers like Scion, which compute geometry-aware updates that are not regularized coordinate-wise, instead using linear minimization oracles over matrix-norm-ball constraints. As a consequence, the analytical techniques developed for coordinate-wise

optimizers do not directly apply, and establishing depth-wise hyperparameter transfer requires a different derivation adapted to constrained and unconstrained spectral optimization.

For width scaling, we prove that the optimal learning rate is independent of hidden-layer width for Scion (with weight decay), extending prior width-transfer results (Pethick et al., 2025b; Yang et al., 2023a) to the constrained spectral setting. Finally, we develop a variant of Scion, which proposes a new way to constrain the weights to be in a spectral ball or Stiefel manifold by using spectral clipping. We demonstrate that this projected version benefits from width scaling.

We empirically validate these theoretical depth-wise and width-wise hyperparameter transfer results and the feature learning multiple experiments across both vision and language tasks. Learning rate transfer experiments on transformers (nanoGPT (Karpathy, 2023) trained on FineWeb, and DeiT (Touvron et al., 2021) on ImageNet) and on MLPs (MLP-Mixer (Tolstikhin et al., 2021) trained on CIFAR-10) show that hyperparameters tuned at small width and depth successfully transfer to larger models.

From a practical standpoint, by enabling principled width and depth-wise hyperparameter transfer, our results make these spectral-norm-based optimizers significantly cheaper to tune, in the same way that $\mu$P enabled practical scaling for Adam and SGD related optimization methods.

We summarize our contributions as follows:

- We confirm that the CompleteP desiderata (stable initialization and stable feature updates) hold for Scion and Unconstrained Scion, under the $L^{-\alpha}$ residual scaling ($\alpha \in \{1/2, 1\}$) with an appropriately scaled learning rate $\gamma \propto L^{\alpha-1}$, establishing depth-wise learning rate transfer. Our analysis is tailored to updates that take matrix structure into account, which are not addressed by the coordinate-wise analyses used in existing CompleteP formulations.
- We prove width-wise hyperparameter transfer for Scion (with weight decay) and for Spectral Clipping, extending the width-transfer results of Pethick et al. (2025b); Yang et al. (2023a) to the constrained setting.
- We give practitioners a guide to tune layer-wise scale, and empirically validate the theoretical depth-wise and width-wise hyperparameter transfer results, as well as complete feature learning ($\alpha = 1$ vs. $\alpha = 1/2$), on both transformers (modded nanoGPT on FineWeb, and DeiT on ImageNet) and MLPs (MLP-Mixer on CIFAR-10) across vision and language datasets.

## 2 Related Work

**Width-wise hyperparameter transfer** The Maximal Update Parametrization ($\mu$P) (Yang & Hu, 2021; Yang & Littwin, 2023) initiated the Tensor Programs framework for hyperparameter transfer that determines how hyperparameters such as the learning rate or the initialization should vary with the model width going to infinity, ensuring that each layer exhibits stable learning and meaningful feature (maximal feature learning). $\mu$P enables optimal hyperparameters tuned on a narrow model to transfer to wider models. This line of work established that width-wise hyperparameter transfer is well understood for coordinate-wise optimizers such as SGD and Adam, and inspired further analyses for width scaling (Yang et al., 2021; 2023a; Bordelon & Pehlevan, 2022; Vankadara et al., 2024).

Kosson et al. (2025) study learning-rate transfer for Adam under $\mu$P, showing that the geometric alignment assumptions underlying $\mu$P hold only briefly at the start of training in practical large-scale settings. They demonstrate that for Adam, it is instead independent weight decay that stabilizes representation updates across widths for most of training, reframing $\mu$P's effect as an implicit learning-rate warmup.

Despite these refinements, works building on the $\mu$P framework focus on coordinate-wise updates structures and do not directly cover spectral-norm-based optimization methods.

**Depth-wise hyperparameter transfer** This idea has been extended, both in the Tensor Programs framework and beyond, to depth scaling for residual networks (Yang et al., 2023b; Bordelon et al., 2024b;

Dey et al., 2026; Qiu et al., 2025). The insight behind these works is to scale the residual branch by a factor $L^{-\alpha}$ that depends on the depth $L$ of the network, with $\alpha \in [1/2, 1]$, and to scale the learning rate accordingly.

More recently, CompleteP (Dey et al., 2026) formalized depth-wise hyperparameter transfer by introducing three desiderata: stable initialization, stable feature updates, and a non-lazy training regime. Within this framework, CompleteP shows that $\alpha = 1$ ensures complete feature learning and improved performance compared to $\alpha = 1/2$ for Adam-based optimizers.[1] While CompleteP provides a general conceptual framework, its concrete instantiations and proofs focus on SGD-like update rules, leaving open the question of whether these desiderata can be satisfied by optimizers with fundamentally different update structures.

Shigida et al. (2026) focus on the nGPT framework, restricting the model to a narrow class of architectures, and demonstrate learning rate transfer across both model dimension and token horizon. Ren et al. (2026), constrains the model by design, and implements a variant of $\mu$P using Frobenius norms for hypersphere parametrization. Under this Frobenius-sphere constraint and utilizing the Muon optimizer, they achieve learning rate transfer across model width, depth, and training tokens. In contrast, we operate in a more general setting, with constraints guarantee through the optimization.

Our work builds on the CompleteP framework and instantiates it for a new family of optimizers. We adopt the same architectural setting and depth-dependent residual scaling, but study a different update rule based on Scion. We show that, under appropriate scaling, both Scion and Unconstrained Scion satisfy the CompleteP desiderata, and we provide new proofs tailored to Scion 's update dynamics. In this sense, our contribution extends CompleteP to a broader class of optimization methods.

**Spectral-norm and geometry-based optimizers.** A parallel line of work studies optimization methods that incorporate geometric or spectral information into their updates. Shampoo (Gupta et al., 2018) introduces a tensor-wise preconditioning method, based on Kronecker-factored matrices. Shampoo uses these preconditioning matrices to whiten features along the left and right dimensions of the weight matrices. This significantly reduces the memory and computational cost compared to the computational cost of storing the full Hessian matrices at large scale. SOAP (Vyas et al., 2025) extends Shampoo by combining tensor-structured preconditioning with Adam updates. It applies on-the-fly adaptive updates in the eigen-basis of the Kronecker preconditioner matrices (Pethick et al., 2025a), improving the stability and reducing the computational cost of the training while preserving the scaling ability of Shampoo. In this work, we focus specifically on instant versions via LMO, noting that methods like SCION operate precisely as instant Shampoo.

Spectral-norm-based optimizers constitute a distinct and increasingly popular class. Muon (Jordan et al., 2024b) combines SGD with momentum and matrix normalization using Newton–Schulz iterations to produce orthogonal updates. The orthogonalization prevents updates from being dominated by a few directions, resulting in more balanced and robust training dynamics. Muon demonstrates rapid convergence and strong performances on large-scale deep learning tasks, including a speed record on CIFAR-10.

Scion (Pethick et al., 2025b) is a Conditional Gradient algorithm that utilizes the LMO subject to norm-ball constraints, allowing for choice of layer-wise norms. From this perspective, Scion plays a role for sign-based optimization (Riedmiller & Braun, 1993) analogously to that of Shampoo for Adam. Indeed, as Shampoo enriches Adam with structured matrix preconditioning, Scion augments sign updates with explicit and more general geometric constraints. Scion provides a unified framework on existing methods such as Muon (Jordan et al., 2024b), NormalizedSGD (Hazan et al., 2015), and SignSGD (Bernstein et al., 2018), that arise as special cases of unconstrained Stochastic Conditional Gradient under different norms (the spectral, the max-, and the Euclidean $L_2$ norms, respectively). Thus, Scion covers a broad family of geometric-aware methods, and has been implemented in the NVIDIA NeMo Emerging Optimizers library (NVIDIA NeMo Team, 2025).

Closest to this setting is the Modular Norm framework (Large et al., 2024), which establishes width-wise and depth-wise hyperparameter transfer using layer-wise normalized SGD. This direction was further developed through linear minimization oracles in Pethick et al. (2025b), which proves width-wise transfer for Uncon-

[1] The authors of Dey et al. (2026) argue that their results extend to any "well-normed" optimizer, and provide explicit derivations for SignSGD.

STRAINED SCION and serves as the primary foundation for our analysis. Relatedly, Qiu et al. (2025) study the width and depth scaling of the learning rate and the weight decay for matrix-preconditioned optimizers such as Shampoo, SOAP, and Muon+AdamW. Additionally, Islamov et al. (2026) analyze hyperparameter scaling from a data-budget perspective, establishing principled rules for scaling the batch size, sequence length, and step size specifically for stochastic conditional gradient methods like SCION.

## 3 Complete SCION

We first restate the desiderata required for effective feature learning and hyperparameter transfer. We then show that these desiderata are satisfied by SCION, both in its constrained and unconstrained forms.

### 3.1 Optimization problem

We consider training a neural network with $L$ hidden layers. We denote the full set of model parameters as $\mathbf{\Theta} = \{\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(L+1)}\}$, where each $\theta^{(\ell)}, \ell \in \{0, \ldots, L+1\}$ represents the parameters specific to layer $\ell$. Each $\theta^{(\ell)} = (\theta^{(\ell,k)})_k$ collects all the parameters of the layer $\ell$. The training goal is to minimize the loss $\mathcal{L}(\mathbf{\Theta})$, formulated as

$$\min_{\mathbf{\Theta} \in \mathcal{X}} \mathcal{L}(\mathbf{\Theta}) \tag{1}$$

where $\mathcal{X}$ represents the parameter set, that is either unconstrained (full parameter space) or constrained to $\mathcal{X} = \mathcal{D}$. We denote the partial gradient with respect to the $\ell$-th layer parameters as $\nabla_{\theta^{(\ell)}} \mathcal{L}(\mathbf{\Theta})$. The dependence on the full family of parameters $\mathbf{\Theta}$ will be implicit in the remainder of the paper, and we will simply write $\nabla_{\theta^{(\ell)}} \mathcal{L}$ and omit the dependence on $\mathbf{\Theta}$.

In contrast to many prior works on hyperparameter transfer, we will not always assume $\mathcal{X}$ is the full parameter space, instead allowing for some constraint set $\mathcal{D}$ which is defined as the Cartesian product of layer-wise norm-balls with radii $\{\rho^{(0)}, \ldots, \rho^{(L+1)}\}$, each constraining all parameter blocks of the corresponding layer.

$$\mathcal{D} = \prod_{\ell=0}^{L+1} \{\theta^{(\ell)} : \|\theta^{(\ell,k)}\| \leq \rho^{(\ell,k)}, \forall k\}. \tag{2}$$

There are two main ways to enforce these constraints: projection-based methods and Conditional-Gradient–type updates (Frank & Wolfe, 1956). In Conditional Gradient methods, each update is constructed as a convex combination of the current iterate and a point on the constraint set, ensuring that iterates remain within the feasible region. In contrast, projected gradient methods first take an unconstrained update step and then project the result back onto the constraint set at each iteration. An illustration of these two types of iterations can be found in Figure 2. In the next part, we detail the Stochastic version of Conditional Gradient, and the associated optimizer SCION.

### 3.2 SCION and UNCONSTRAINED SCION

SCION is a stochastic optimization framework based on the Stochastic Conditional Gradient (SCG), and its unconstrained version (uSCG), presented in Algorithms 1 and 2, respectively. In uSCG, the parameters $\theta$ are optimized over an unconstrained parameter space denoted $\mathcal{X}$, as opposed to the constrained feasible set denoted $\mathcal{D}$ used in SCION. Importantly, the use of weight decay can be interpreted within this framework as instances of constrained version, SCION, where the decay implicitly enforces norm-ball constraints on the parameters, as discussed in Pethick et al. (2025b;a).

SCG is classically used for constrained optimization problems as it ensures the iterates remain in a ball of constant norm. These algorithms adapt the geometry to the problem by using norm-constrained Linear Minimization Oracles (LMOs). The lmo is applied on the gradient, and its output minimizes a linear approximation to the function $\mathcal{L}$ we want to minimize under the norm-ball constraint set $\mathcal{D}$

$$\mathrm{lmo}(\nabla_{\theta^{(\ell)}} \mathcal{L}) \in \arg\min_{\nu \in \mathcal{D}} \langle \nabla_{\theta^{(\ell)}} \mathcal{L}, \nu \rangle. \tag{3}$$

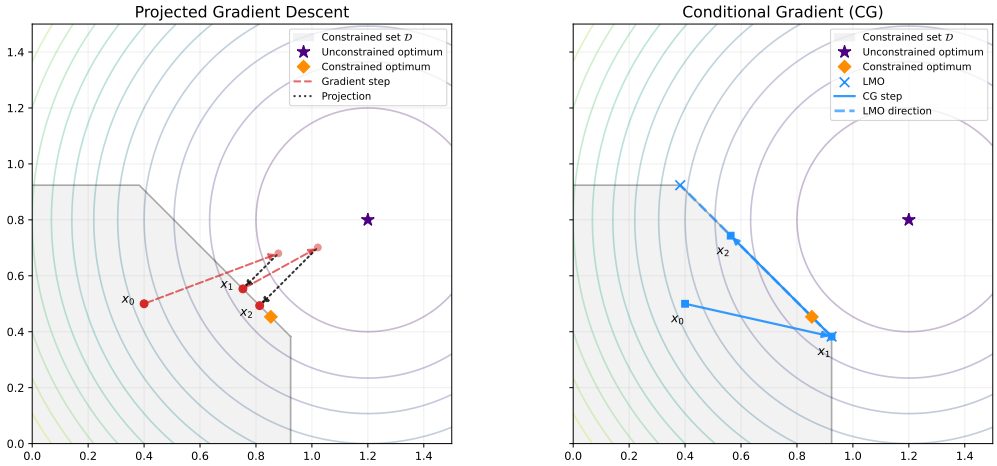



Figure 2: Projected gradient methods step outside the feasible set and then project back to the border of the constraint set, whereas Conditional Gradient methods preserve feasibility by updating through convex combinations with decreasing step sizes.

**Algorithm 1** SCG

**Input:** Horizon $n$, initialization $\boldsymbol{\Theta}_1 \in \mathcal{D}$, $M_0 = 0$, momentum $\beta_t \in [0, 1)$, and stepsize $\gamma_t \in (0, 1)$

1: **for** $t = 1, \ldots, n$ **do**
2: Sample data $x \sim \mathcal{P}$
3: $M_t \leftarrow (1 - \beta_t)\nabla_{\boldsymbol{\Theta}_t}\mathcal{L} + \beta_t M_{t-1}$
4: $\boldsymbol{\Theta}_{t+1} \leftarrow (1 - \gamma_t)\boldsymbol{\Theta}_t + \gamma_t \operatorname{lmo}(M_t)$
5: **end for**

**Algorithm 2** Unconstrained SCG (uSCG)

**Input:** Horizon $n$, initialization $\boldsymbol{\Theta}_1 \in \mathcal{X}$, $M_0 = 0$, momentum $\beta_t \in [0, 1)$, and stepsize $\gamma_t \in (0, 1)$

1: **for** $t = 1, \ldots, n$ **do**
2: Sample data $x \sim \mathcal{P}$
3: $M_t \leftarrow (1 - \beta_t)\nabla_{\boldsymbol{\Theta}_t}\mathcal{L} + \beta_t M_{t-1}$
4: $\boldsymbol{\Theta}_{t+1} \leftarrow \boldsymbol{\Theta}_t + \gamma_t \operatorname{lmo}(M_t)$
5: **end for**

We note that the lmo provides only the *direction* of the constrained steepest gradient descent. It does not give a step size or define the next iterate. Also, it is scale invariant: $\operatorname{lmo}(\mu s) = \operatorname{lmo}(s)$ for any $\mu > 0$.

Scion instantiates SCG and uSCG using layer-wise operator norms, resulting in the constrained variant (Scion) and the unconstrained variant (Unconstrained Scion), respectively. By choosing the norm based on the architecture to be optimized, Scion aligns the optimization direction with the model's structure, leading to a faster and more efficient training, especially for large neural networks. Update rules, associated LMOs, and per-layer learning rate scales are provided in Table 1 for typical configurations. Specific examples of the Unconstrained Scion step and its layer-wise learning rate scales can be found in Equation (4) and Equation (5).

For instance, considering a one-step Unconstrained Scion update with a `Sign` $\rightarrow$ `Spectral` $\rightarrow$ `Sign` configuration for a 1-layer MLP with parameters $\{W^{(0)}, W^{(1)}, W^{(2)}\}$, the iterates at the iteration $t + 1$ are given by:

$$
\begin{aligned}
\begin{bmatrix} W_{t+1}^{(0)} \\ W_{t+1}^{(1)} \\ W_{t+1}^{(2)} \end{bmatrix} &= \begin{bmatrix} W_t^{(0)} \\ W_t^{(1)} \\ W_t^{(2)} \end{bmatrix} + \gamma_t \operatorname{lmo}\left( \begin{bmatrix} M_t^{(0)} \\ M_t^{(1)} \\ M_t^{(2)} \end{bmatrix} \right) && \text{(by definition of Unconstrained Scion)} \\
&= \begin{bmatrix} W_t^{(0)} \\ W_t^{(1)} \\ W_t^{(2)} \end{bmatrix} + \gamma_t \begin{bmatrix} \operatorname{lmo}_{\|\cdot\|_{\text{Sign}}}\left(M_t^{(0)}\right) \\ \operatorname{lmo}_{\|\cdot\|_{\mathcal{S}_\infty}}\left(M_t^{(1)}\right) \\ \operatorname{lmo}_{\|\cdot\|_{\text{Sign}}}\left(M_t^{(2)}\right) \end{bmatrix} && \text{(the lmo is separable)} \\
&= \begin{bmatrix} W_t^{(0)} \\ W_t^{(1)} \\ W_t^{(2)} \end{bmatrix} - \gamma_t \begin{bmatrix} \rho_0 \left( \frac{1}{d_{\text{in}}} \operatorname{sign}\left(M_t^{(0)}\right) \right) \\ \rho_1 \left( \sqrt{\frac{d_1}{d_0}} \operatorname{orthog}\left(M_t^{(1)}\right) \right) \\ \rho_2 \left( \frac{1}{d_1} \operatorname{sign}\left(M_t^{(2)}\right) \right) \end{bmatrix} && \text{(compute the LMOs with the layerwise radii)}
\end{aligned}
\tag{4}
$$

where we used the separability of the lmo for the max norm $\|\mathbf{\Theta}\| = \max_\ell \{\|\theta^{(\ell)}\|_\ell\}$.

Then, the difference between iteration $t+1$ and $t$ for each layer $\ell$, denoted $\Delta W^{(\ell)}$ can be expressed as:

$$
\begin{aligned}
\Delta W^{(0)} &= \gamma_t \operatorname{lmo}_{\|\cdot\|_{\text{Sign}}}(M_t^{(0)}) = -\gamma_t \rho_0 \left( \frac{1}{d_{\text{in}}} \operatorname{sign}\left(M_t^{(0)}\right) \right) \\
\Delta W^{(1)} &= \gamma_t \operatorname{lmo}_{\|\cdot\|_{\mathcal{S}_\infty}}(M_t^{(1)}) = -\gamma_t \rho_1 \left( \sqrt{\frac{d_1}{d_0}} \operatorname{orthog}\left(M_t^{(1)}\right) \right) \\
\Delta W^{(2)} &= \gamma_t \operatorname{lmo}_{\|\cdot\|_{\text{Sign}}}(M_t^{(2)}) = -\gamma_t \rho_2 \left( \frac{1}{d_1} \operatorname{sign}\left(M_t^{(2)}\right) \right).
\end{aligned}
\tag{5}
$$

Table 1: Update rules and learning rate (LR) scaling for different architectural layers. This example utilizes the norm configuration `Sign` $\to$ `Spectral` $\to$ `Sign` for the embedding, hidden, and un-embedding layers, respectively. We provide the specific operator norms and associated lmo applied to the average gradient $M^{(\ell)}$, whose reduced Singular Value Decomposition (SVD) is denoted $M^{(\ell)} = U^{(\ell)}\Sigma^{(\ell)}V^{(\ell)\top}$ for all layers $\ell$. The parameters $\rho_e$, $\rho_h$, and $\rho_u$ designate the radii of the balls for the embedding, hidden, and un-embedding weights respectively. Following Recommendation 3.1 of Pethick et al. (2025b), typical configurations include `Spectral` $\to$ `Spectral` $\to$ `Sign` for image data, or `ColNorm` $\to$ `Spectral` $\to$ `Sign` for 1−hot inputs. For an exhaustive list of norms and associated LMOs, one can refer to Pethick et al. (2025b;a).

| | **Embedding** ($\ell = 0$) | $\to$ | **Hidden Layers** ($\ell \in [1, L]$) | $\to$ | **Un-embedding** ($\ell = L+1$) |
|---|---|---|---|---|---|
| **Norm** $\|\cdot\|$ | $\|\cdot\|_{\text{Sign}}$ | | $\|\cdot\|_{\mathcal{S}_\infty}$ | | $\|\cdot\|_{\text{Sign}}$ |
| **LMO** $\operatorname{lmo}(M^{(\ell)})$ | $-\rho_e \frac{1}{d_{\text{in}}} \operatorname{sign}(M^{(0)})$ | | $-\rho_h \sqrt{\frac{d_\ell}{d_{\ell-1}}} U^{(\ell)} V^{(\ell)\top}$ | | $-\rho_u \frac{1}{d_L} \operatorname{sign}(M^{(L+1)})$ |
| **Scaled LR** $\gamma_\ell$ | $\gamma_{\text{base}}$ | | $L^{\alpha-1}\gamma_{\text{base}}$ | | $\gamma_{\text{base}}$ |
| **Unconstrained Scion Update** $\Delta W^{(\ell)} = \gamma_\ell \operatorname{lmo}(M^{(\ell)})$ | $-\gamma_0 \rho_e \frac{1}{d_{\text{in}}} \operatorname{sign}(M^{(0)})$ | | $-\gamma_\ell \rho_h \sqrt{\frac{d_\ell}{d_{\ell-1}}} U^{(\ell)} V^{(\ell)\top}$ | | $-\gamma_{L+1} \rho_u \frac{1}{d_L} \operatorname{sign}(M^{(L+1)})$ |
| **Scion Update** $\Delta W^{(\ell)} = \gamma_\ell (\operatorname{lmo}(M^{(\ell)}) - W^{(\ell)})$ | $-\gamma_0 (\rho_e \frac{1}{d_{\text{in}}} \operatorname{sign}(M^{(0)}) + W^{(0)})$ | | $-\gamma_\ell (\rho_h \sqrt{\frac{d_\ell}{d_{\ell-1}}} U^{(\ell)} V^{(\ell)\top} + W^{(\ell)})$ | | $-\gamma_{L+1} (\rho_u \frac{1}{d_L} \operatorname{sign}(M^{(L+1)}) + W^{(L+1)})$ |

### 3.3 Conditions for Depth-wise Transfer

**Setup** In this section, we consider residual architecture of depth $L$, defined as

$$\begin{aligned} h^{(0)} &= W^{(0)}x \\ h^{(\ell)} &= Z^{(\ell)}h^{(\ell-1)} + L^{-\alpha}\mathcal{F}_\ell(h^{(\ell-1)};\theta_{\mathcal{F}}^{(\ell)}) \quad (\forall \ell \in \{1,\dots,L\}) \\ f &= W^{(L+1)}\phi(h^{(L)}), \end{aligned} \tag{6}$$

where $x \in \mathbb{R}^{d_{\text{in}}}$ are the input data, $h^{(\ell)} \in \mathbb{R}^{d_\ell}, \ell \in \{0,\dots,L\}$ are the hidden states, $Z^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, and $f \in \mathbb{R}^{d_{\text{out}}}$ is the output after nonlinear activation $\phi$. The embedding and un-embedding matrices are respectively $W^{(0)} \in \mathbb{R}^{d_0 \times d_{\text{in}}}$ and $W^{(L+1)} \in \mathbb{R}^{d_{\text{out}} \times d_L}$. The residual block $\mathcal{F}_\ell(h^{(\ell-1)};\theta_{\mathcal{F}}^{(\ell)}), \ell \in \{1,\dots,L\}$ can be an attention block, or an MLP (cf. Figure 3), and $\theta_{\mathcal{F}}^{(\ell)}$ is its trainable parameter. Following the notation introduced in 3.1, the parameters for the hidden layer $\ell$ are represented by the set $\theta^{(\ell)} = \{Z^{(\ell)}, \theta_{\mathcal{F}}^{(\ell)}\}$, and the parameters for the embedding and un-embedding layers are represented by $\theta^{(0)} = \{W^{(0)}\}$ and $\theta^{(L+1)} = \{W^{(L+1)}\}$ respectively.

Essentially, the residual branch is scaled with a depth-dependent factor $L^{-\alpha}$, where the exponent $\alpha$ dictates how the contribution of each residual block scales as the total depth $L$ grows. This parameterization follows the framework of Tensor Programs and infinite-depth limits (Yang et al., 2023b; Bordelon et al., 2024b).

The choice $\alpha = 1/2$ corresponds to the $\mu$P regime studied in Tensor Programs, where the cumulative contribution of the residual stream remains non-trivial in the infinite-depth limit and feature learning is preserved (Yang et al., 2023b; 2021).

Conversely, recent work suggests advantages for linear scaling. Indeed, Bordelon et al. (2024a) find $\alpha = 1$ optimizes learning at infinite depth, while Dey et al. (2026) argue that $\alpha = 1$ is the only value that enables stable training, maximal updates and feature learning at infinite dimensions.

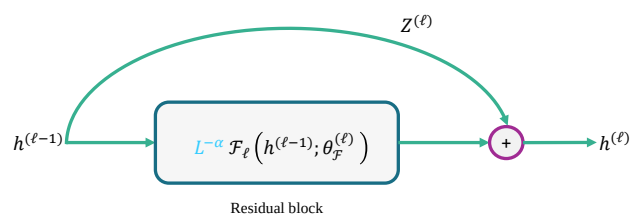



Figure 3: Residual block mechanism with depth-wise scaling $L^{-\alpha}$.

CompleteP (Dey et al., 2026) defines three desiderata, that provide a principled framework for understanding why this parametrization works, and what criteria the parametrization should respect in order to observe hyper-parameter transfer both in width and depth, as well as non-lazy learning.

Following the literature (Dey et al., 2026; Qiu et al., 2025; Zheng et al., 2026), we enumerate the desiderata for depth-scaling below.

**Notation** In the analysis, we will use the following notation:

$$\begin{aligned} f(n) = \Theta(g(n)) &\iff \left(\exists\,(c,C) \in (0,+\infty)^2\right) \quad cg(n) \le f(n) \le Cg(n) \quad \text{as } n \to \infty, \\ f(n) = \mathcal{O}(g(n)) &\iff \left(\exists\, C \in (0,+\infty)\right) \quad f(n) \le Cg(n) \quad \text{as } n \to \infty, \\ f(n) = \Omega(g(n)) &\iff \left(\exists\, c \in (0,+\infty)\right) \quad cg(n) \le f(n) \quad \text{as } n \to \infty. \end{aligned}$$

Note that the notation $\Theta()$ is different from the full parameter set mentioned in the previous subsections and written in bold as $\boldsymbol{\Theta}$.

We will also use $\|\cdot\|_2$ and $\|\cdot\|_{\mathcal{S}_\infty}$ to denote the Euclidean norm and the spectral norm, respectively.

**Desideratum 1** (Stable Initialization )**.** *The hidden layers and the output remain stable (preventing signals from exploding or vanishing as they pass through many layers) at the initialization when $L$ and $d_\ell$ become increasingly large:*

$$(\forall \ell \in \{1,\dots,L\}) \;\; \frac{1}{d_\ell}\|h^{(\ell)}\|_2^2 \in \Theta(1) \;\; \textit{and} \;\; \|f\|_2^2 \in \mathcal{O}(1), \quad \textit{as } d_\ell \to \infty, L \to \infty. \tag{7}$$

This property indicates how to adjust the variance of the initialization of the weights. In Dey et al. (2026), it is highlighted that this imposes $\alpha \geq \frac{1}{2}$, since otherwise the residual stream variance diverges when the depth is scaled.

**Desideratum 2** (Stable Feature Update ). [2] *The weights of each residual block should contribute with order $1/L$ to feature movements, and each non-residual block should contribute with constant order (e.g., each non-residual block has a meaningful impact on the residual stream):*

$$\begin{aligned} &\frac{1}{\sqrt{d_\ell}}\|\Delta_{\theta_{\mathcal{F}}^{(\ell)},Z^{(\ell)}} h^{(\ell)}\|_2 \in \Theta\left(\frac{1}{L}\right) \quad (\forall \ell \in \{1,\dots,L\}), \\ &\frac{1}{\sqrt{d_0}}\|\Delta W^{(0)} x\|_2 \in \Theta(1), \\ &\frac{1}{\sqrt{d_{\text{out}}}}\|\Delta W^{(L+1)} h^{(L)}\|_2 \in \Theta(1), \end{aligned} \tag{8}$$

*where $\Delta$ designates the variation of the parameters between two successive iterations.*

This criterion is the same as the maximal update condition when $L$ is a fixed constant, since we have $\Theta(1/L) = \Theta(1)$ with respect to width-scaling (Yang & Hu, 2021; Yang et al., 2023a). Moreover, Dey et al. (2026) shows that this criterion imposes $\alpha \leq 1$, a result also proven in Yang et al. (2023b).

As we shall see in the next subsections, the first and second desiderata can be guaranteed to be satisfied by Scion and Unconstrained Scion for any $\alpha \in [1/2, 1]$, under the condition that the learning rates, $\gamma$ and $\gamma_Z$, used to optimize the parameters $\{\theta_{\mathcal{F}}^{(\ell)}\}_{\ell\in\{1,\dots,L\}}$ and the operators $\{Z^{(\ell)}\}_{\ell\in\{1,\dots,L\}}$, respectively, are scaled as follows:

$$\gamma = \gamma_{\text{base}} L^{\alpha-1} \quad \text{and} \quad \gamma_Z = \gamma_{Z\,\text{base}} L^{-1}. \tag{9}$$

The learning rate relative to the embedding and un-embedding parameters do not need depth-wise scaling,

$$\gamma_{W^{(0)}} = \gamma_{W^{(0)}\,\text{base}} \quad \text{and} \quad \gamma_{W^{(L+1)}} = \gamma_{W^{(L+1)}\,\text{base}}. \tag{10}$$

We can interpret this in the unconstrained case as a *scale-dependent* operator norm that encodes the scaling geometry automatically, so that the optimal learning rate does not change and these scalings are encoded solely in the norm choice. Specifically, the fixed magnitude of the lmo induces a guaranteed bound on the maximum norm of the parameters, ensuring stable scaling.

### 3.4 Scion and Unconstrained Scion satisfy Desiderata 1 and 2

#### 3.4.1 Fundamental setup

As a simplified model, consider the block $\mathcal{F}_\ell(h^{(\ell-1)}; \theta_{\mathcal{F}}^{(\ell)})$ given by a one-layer MLP, for $\ell \in \{1,\dots,L\}$, that is

$$h^{(\ell)} = Z^{(\ell)} h^{(\ell-1)} + L^{-\alpha} W^{(\ell)} \phi(h^{(\ell-1)}) \tag{11}$$

where $\phi$ is a Lipschitz-continuous activation function. In this case, the trainable parameter of the residual block $\theta_{\mathcal{F}}^{(\ell)}$ is the weight matrix $W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ for $\ell \in \{1,\dots,L\}$. In our analysis, we consider that the operator matrices applied to the residual stream $Z^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ for $\ell \in \{1,\dots,L\}$, are learned and updated during training. This generalizes the standard identity shortcut to a learnable linear transformation, aligning with recent architectural paradigms that introduce parameterized transformations into the residual path. For instance, manifold-constrained hyper-connections (mHC) (Xie et al., 2025), which uses the lmo for Birkhoff polytope, utilizes learnable mixing to mix features in the residual stream, and Team et al. (2026) use an attention mechanism over the preceding layer output. In this case, following the notation defined in 3.1, the parameters for the hidden layer $\ell$ are simply the pair of matrices $\theta^{(\ell)} = \{Z^{(\ell)}, W^{(\ell)}\}$, and the parameters for the embedding and un-embedding layers are $\theta^{(0)} = \{W^{(0)}\}$ and $\theta^{(L+1)} = \{W^{(L+1)}\}$, respectively. In our analysis, we omit biases, but, in practice, when included they are scaled in the same way as the weight matrices, consistent with Dey et al. (2026).

[2] The corresponding Desideratum in Dey et al. (2026) has $\frac{1}{d_\ell}\|\Delta_{\theta^{(\ell)}} h^{(\ell)}\|_2^2 = \Theta(\frac{1}{L})$, which slightly differs from ours, as we discuss in Appendix E.

#### 3.4.2 Modeling assumptions

We focus on the worst-case analysis for a batch size of 1. In practice and in our experiments, large batches are used and the results remain in accordance with our theoretical conclusions.

**Initialization** As in Pethick et al. (2025b), we suppose we initialize with either a semi-orthogonal or Gaussian matrix, and apply the lmo with the spectral norm for every layer $\ell$. We note that using a scaled sign operator norm on the embedding or un-embedding layers is also viable, consistent with the analysis in Dey et al. (2026).

In particular, this leads (with high probability) to

$$\|W^{(\ell)}\|_{\mathcal{S}_\infty} = \Theta\left(\sqrt{\frac{d_\ell}{d_{\ell-1}}}\right) \quad \text{and} \quad \|\Delta W^{(\ell)}\|_{\mathcal{S}_\infty} = \Theta\left(\sqrt{\frac{d_\ell}{d_{\ell-1}}}\right). \tag{12}$$

We suppose the same assumptions apply for the matrices $Z^{(\ell)}, \ell \in \{1, \dots, L\}$.

For technical purposes, we will furthermore assume that the data x follows some distribution $\mathcal{P}$ such that $\|x\|_2 = \Theta(\sqrt{d_{in}})$. This is often the case, for images (Yang et al., 2023a) for instance.

**Alignment of initial weight matrices** Also, we assume that for all layers $\ell$, we have

$$\begin{aligned} \|W^{(\ell)}\phi(h^{(\ell-1)})\|_2^2 &= \Theta(\|W^{(\ell)}\|_{\mathcal{S}_\infty}^2 \|\phi(h^{(\ell-1)})\|_2^2), \\ \|W^{(\ell)}h^{(\ell-1)}\|_2^2 &= \Theta(\|W^{(\ell)}\|_{\mathcal{S}_\infty}^2 \|h^{(\ell-1)}\|_2^2). \end{aligned} \tag{13}$$

While this modeling assumption is not universally applicable, it has been empirically validated (Yang et al., 2023a) and consistently yields predictive results.

We suppose the same assumptions apply for the matrices $Z^{(\ell)}, \ell \in \{1, \dots, L\}$.

**Scale of the activation function $\phi$** We assume that the hidden features $\phi(h^{(\ell)})$ have the same scale before and after the application of the activation function $\phi$ for all the layers $\ell$:

$$\|\phi(h^{(\ell)})\|_2 = \Theta(\|h^{(\ell)}\|_2). \tag{14}$$

It is the same assumption as in the Assumption 2 of Yang et al. (2023a). This behavior is typical for most activation functions since they are designed to operate on inputs of order one and produce outputs that neither blow up nor collapse.

**Update expression of $W^{(\ell)}$** Since we consider a residual network, we need to compute, for every layer $\ell$

$$\Delta W^{(\ell)} = \begin{cases} \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}}\mathcal{L}) & \text{with Unconstrained Scion} \\ \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}}\mathcal{L}) - \gamma W^{(\ell)} & \text{with Scion.} \end{cases} \tag{15}$$

Further details about the updates and the LMOs expressions can be found in Table 1, in Equation (4) and in Equation (5).

The same update expressions apply for the matrices $Z^{(\ell)}, \ell \in \{1, \dots, L\}$.

To simplify the analysis, we consider the momentum $\beta^{(\ell)} = 0$ for each layer $\ell$. As established in our analysis, the desiderata can be ensured by imposing suitable conditions on the spectral norm of the updates. Since the LMO has a fixed spectral norm regardless of whether momentum is used, we expect its behavior to remain unchanged. This expectation is further supported by our experiments, in which we considered nonzero momentum values.

#### 3.4.3 Desideratum 1 is satisfied for Scion and Unconstrained Scion

Our next result considers semi-orthogonally initialized matrices with $\|W^{(\ell)}\|_{\mathcal{S}_\infty} = \Theta(\sqrt{\frac{d_\ell}{d_{\ell-1}}})$ and $\|Z^{(\ell)}\|_{\mathcal{S}_\infty} = \Theta(\sqrt{\frac{d_\ell}{d_{\ell-1}}}), \forall \ell \in \{1, \ldots, L\}$, and shows that Desideratum 1 holds, extending the result in Dey et al. (2026) for Gaussian variables initialization.

**Proposition 1** (Scion and Unconstrained Scion verify Desideratum 1)**.** *Under the assumptions defined in the subsection 3.4.2, the hidden layers and the output remain stable at the initialization when $L$ and $d_\ell$ become increasingly large.*

We provide a sketch of the demonstration for Scion and Unconstrained Scion, where the same arguments hold. Complete derivations are available in Appendix C.3.

*Proof sketch of Proposition 1.* Let $\ell \in \{1, \ldots, L\}$. By using the alignment of weight matrices in Equation (13), the scale of the activation function $\phi$ in Equation (14), and the weight initialization in Equation (12), we have

$$\begin{aligned}
\|h^{(\ell)}\|_2^2 &= \Theta(\|Z^{(\ell)} h^{(\ell-1)}\|_2^2 + \frac{1}{L^{2\alpha}} \|W^{(\ell)} \phi(h^{(\ell-1)})\|_2^2) \\
&= \Theta\left( \|Z^{(\ell)}\|_{\mathcal{S}_\infty}^2 \|h^{(\ell-1)}\|_2^2 + \frac{1}{L^{2\alpha}} \|W^{(\ell)}\|_{\mathcal{S}_\infty}^2 \|h^{(\ell-1)}\|_2^2 \right) \\
&= \Theta\left( \prod_{k=1}^{\ell} (\frac{d_k}{d_{k-1}} \times (1 + \frac{1}{L^{2\alpha}})) d_0 \right) \\
&= \Theta\left( d_\ell \prod_{k=1}^{\ell} (1 + \frac{1}{L^{2\alpha}}) \right).
\end{aligned}$$

Then, $\frac{1}{d_L}\|h^{(L)}\|_2^2 = \Theta\left((1 + \frac{1}{L^{2\alpha}})^L\right)$ converges when $L$ goes to infinity if and only if $\alpha \geq \frac{1}{2}$.

Moreover,

$$\frac{1}{d_\ell}\|h^{(\ell)}\|_2^2 = \Theta(1) \quad \forall \ell \in \{1, \ldots, L\} \text{ for } \alpha \geq \frac{1}{2}, \tag{16}$$

and, for $f = W^{(L+1)}\phi(h^{(L)})$, we have

$$\|f\|_2^2 = \Theta(\|W^{L+1}\|_{\mathcal{S}_\infty}^2 \|h^{(L)}\|_2^2) = \Theta(\frac{d_{\text{out}}}{d_L} \times d_L) = \mathcal{O}(1). \tag{17}$$

□

#### 3.4.4 Desideratum 2 is satisfied for Scion and Unconstrained Scion

Following the formulation of Desideratum 2, we study the change in $h^{(\ell)}$ when we update $\theta^{(\ell)}$ and $Z^{(\ell)}$ only. In our setting, we analyze the change in $h^{(\ell)}$ induced by the updates of $W^{(\ell)}$ and $Z^{(\ell)}$.

**Proposition 2** (Scion and Unconstrained Scion verify Desideratum 2)**.** *Under the assumptions defined in subsection 3.4.2, for* Scion *or* Unconstrained Scion *the weights of each block contribute with order $1/L$ to feature movements, and each non-residual block contributes with constant order when $L$ becomes increasingly large.*

We provide a sketch of the demonstrations for Scion. Complete derivations for Scion and Unconstrained Scion are available in Appendix C.4.2 and C.4.1.

*Proof sketch of Proposition 2 for* Scion . Let $\ell \in \{1, \dots, L\}$. We have

$$\begin{aligned}\|\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)}\|_2^2 &= \|\Delta Z^{(\ell)} h_t^{(\ell-1)} + L^{-\alpha} \Delta W^{(\ell)} \phi(h_t^{(\ell-1)})\|_2^2 \\ &= \Theta\Big(\|\gamma_Z Z^{(\ell)} h_t^{(\ell-1)}\|_2^2 + \|\gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}} \mathcal{L}) h_t^{(\ell-1)}\|_2^2 \\ &\quad + \|L^{-\alpha} \gamma W^{(\ell)} \phi(h_t^{(\ell-1)})\|_2^2 + \|L^{-\alpha} \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}} \mathcal{L}) \phi(h_t^{(\ell-1)})\|_2^2\Big).\end{aligned} \tag{18}$$

For the first term, using the alignment of the weights in Equation (13) and the weights initialization in Equation (12), we have

$$\begin{aligned}\|\gamma_Z Z^{(\ell)} h_t^{(\ell-1)}\|_2^2 &= \Theta\left(\gamma_Z^2 \|Z^{(\ell)}\|_2^2 \|h_t^{(\ell-1)}\|_2^2\right) \\ &= \Theta\left(\gamma_Z^2 d_\ell\right) \quad \text{from Desideratum 1.}\end{aligned} \tag{19}$$

For the third term, using the alignment of the weights in Equation (13) and the scale of the activation function $\phi$ in Equation (14) as well as the weights initialization in Equation (12), we have

$$\begin{aligned}\|L^{-\alpha} \gamma W^{(\ell)} \phi(h_t^{(\ell-1)})\|_2^2 &= \Theta\left(L^{-2\alpha} \gamma^2 \|W^{(\ell)}\|_2^2 \|h_t^{(\ell-1)})\|_2^2\right) \\ &= \Theta\left(L^{-2\alpha} \gamma^2 d_\ell\right) \quad \text{from Desideratum 1.}\end{aligned} \tag{20}$$

For the second and fourth term, plugging spectral lmo expressions leads to

$$\begin{aligned}&\|\gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}} \mathcal{L}) h_t^{(\ell-1)}\|_2^2 + \|L^{-\alpha} \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}} \mathcal{L}) \phi(h_t^{(\ell-1)})\|_2^2 \\ = \;& \Theta\left(\frac{d_\ell}{d_{\ell-1}} \gamma_Z^2 \|h_t^{(\ell-1)}\|_2^2 + L^{-2\alpha} \frac{d_\ell}{d_{\ell-1}} \gamma^2 \|\phi(h_t^{(\ell-1)})\|_2^2\right).\end{aligned}$$

Using the scale of the activation function assumption in Equation (14), and Desideratum 1, we have

$$\begin{aligned}&\|\gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}} \mathcal{L}) h_t^{(\ell-1)}\|_2^2 + \|L^{-\alpha} \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}} \mathcal{L}) \phi(h_t^{(\ell-1)})\|_2^2 \\ = \;& \Theta\left(d_\ell (\gamma_Z^2 + L^{-2\alpha} \gamma^2)\right).\end{aligned}$$

Finally, we have

$$\|\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)}\|_2^2 = \Theta\left(d_\ell (\gamma_Z^2 + L^{-2\alpha} \gamma^2)\right). \tag{21}$$

Therefore, to ensure that the entries of $\Delta h^{(\ell)}$ are $\Theta(1/L)$, we must set

$$\gamma = \gamma_{\text{base}} L^{\alpha-1}, \text{ and } \gamma_Z = \gamma_{Z\,\text{base}} L^{-1}. \tag{22}$$

□

**Remark**: If we do not learn the $Z^{(\ell)}$, then we have that $Z_{t+1}^{(\ell)} = Z_t^{(\ell)}$, and $\Delta Z^{(\ell)} = 0$, and we conclude by only requiring $\gamma = \gamma_{\text{base}} L^{\alpha-1}$.

In Figure 4, we empirically verify the feature learning condition by performing a coordinate check (Yang et al., 2023b). As we scale the depth $L$, we measure the magnitude changes at the final residual representation $h_L$ and at the output logits $f$. We observe that these magnitudes remain stable across scales, confirming successful feature learning.

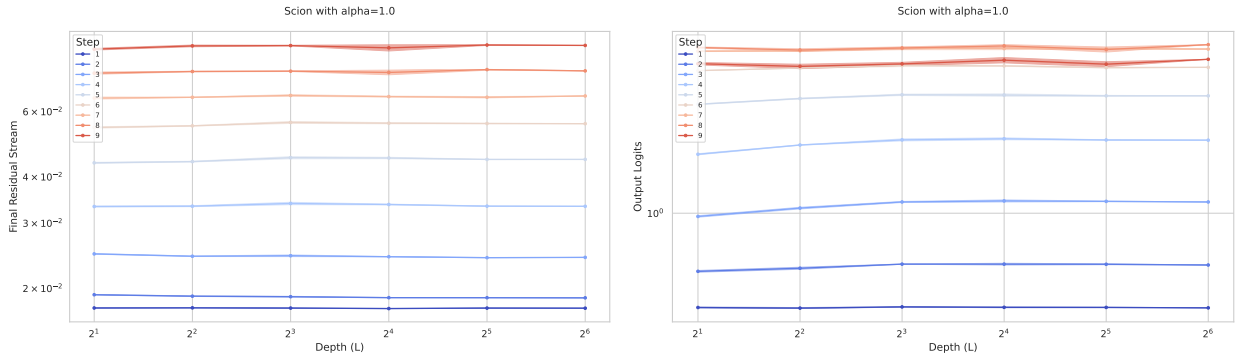



Figure 4: Coordinate checks for Scion, on nanoGPT trained on FineWeb, with $\alpha = 1.0$, and learning rate $\gamma_{\text{base}} = 2^{-11}$. We plot the change at the final residual representation $\|h_t^{(L)} - h_1^{(L)}\|_2$ (left plot) and at the output logits $\|f_t - f_1\|_2$ (right plot). More details on configuration can be found in Table 2 in Appendix B.

### 3.5 Complete Feature Learning

This criterion is not necessary for hyperparameter transfer, but it distinguishes $\alpha = 1$ from the other values in $[1/2, 1]$ as the only value for which the dynamics stay non-linear in its parameter, for every scale.

To present this criterion, we need two definitions described below.

**Linearization** The linearization $h^{\text{lin},\theta}$ of the function $h(\theta)$ with respect to $\theta$ about $\theta_0$ is

$$h^{\text{lin},\theta}(\theta, \theta_0) = h(\theta_0) + \langle \nabla_\theta h(\theta)|_{\theta_0}, \theta - \theta_0 \rangle. \tag{23}$$

We say $h$ is *linear* in $\theta$ if $h = h^{\text{lin},\theta}$.

**Lazy-regime** We say a layer $\phi(h^{(\ell)})$ is *lazy* with respect to a subset of parameters $\theta \subset \{\theta_j\}_{j<l}$ if at finite depth and width, $\phi(h^{(\ell)})$ is not linear in $\theta$ and the change $\Delta_\theta \phi(h^{(\ell)})$ at initialization from updating only $\theta$ (i.e. $\theta \mapsto \theta + \Delta\theta$) is asymptotically the same as the change to the linearization of $h$:

$$\frac{|\Delta_\theta h^{(\ell)} - \Delta_\theta h^{\text{lin},\theta}|}{|\Delta_\theta h^{\text{lin},\theta}|} = o(1) \quad d_\ell, L \to \infty. \tag{24}$$

Empirically, better generalization is observed when the network parameterization satisfies complete feature learning, i.e. when neither the hidden layers $\{\phi(h^{(\ell)})\}_{\ell \in \{1,\dots,L\}}$, nor the output model $f$ are lazy with respect to any subset of model parameters. We reproduce the experiment from Dey et al. (2026) for the Scion and Unconstrained Scion optimizers in Figure 5, which reports the best validation loss achieved across learning rates as a function of depth $L$, for $\alpha \in \{0, 1/2, 1\}$. Consistently with the complete feature learning criterion, $\alpha = 1$ is the only value that yields systematically lower loss as depth increases, for both Scion and Unconstrained Scion.

#### 3.5.1 Non-lazy regime for Scion

Another way to measure whether or not a parameterization results in the lazy regime is to measure the relative distance from initialization compared to the linearization. This gives a way to empirically verify which values of $\alpha$ enables a training in a non-lazy regime.

We reproduce the experiment of Dey et al. (2026) for Scion and Unconstrained Scion in Figure 6, where we plot:

$$\frac{|\Delta_\theta h^{(\ell)} - \Delta_\theta h^{\text{lin},\theta}|}{|\Delta_\theta h^{\text{lin},\theta}|} \tag{25}$$

for multiple depths and $\alpha \in \{0, 1/2, 1\}$, and we verify empirically that Scion and Unconstrained Scion are not lazy for $\alpha = 1$ and $\alpha = 0$.

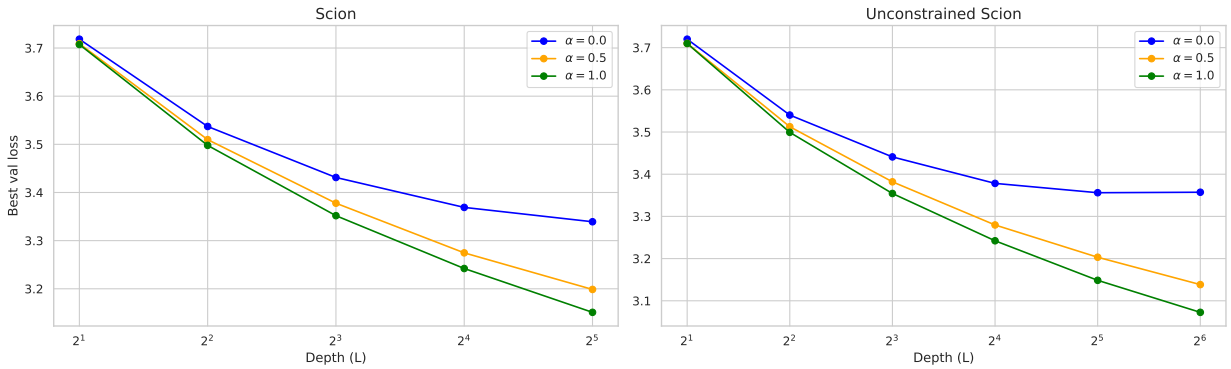



Figure 5: Best loss per learning rate in function of the depth $L$ for $\alpha \in \{0, 1/2, 1\}$, with Scion and Unconstrained Scion, on nanoGPT, trained on FineWeb. We observe that the choice $\alpha = 1$ provides the best performances overall. Note that the case $\alpha = 0$ is analogous to the SP/$\mu$P setup for Adam (Yang & Hu, 2021; Dey et al., 2026).

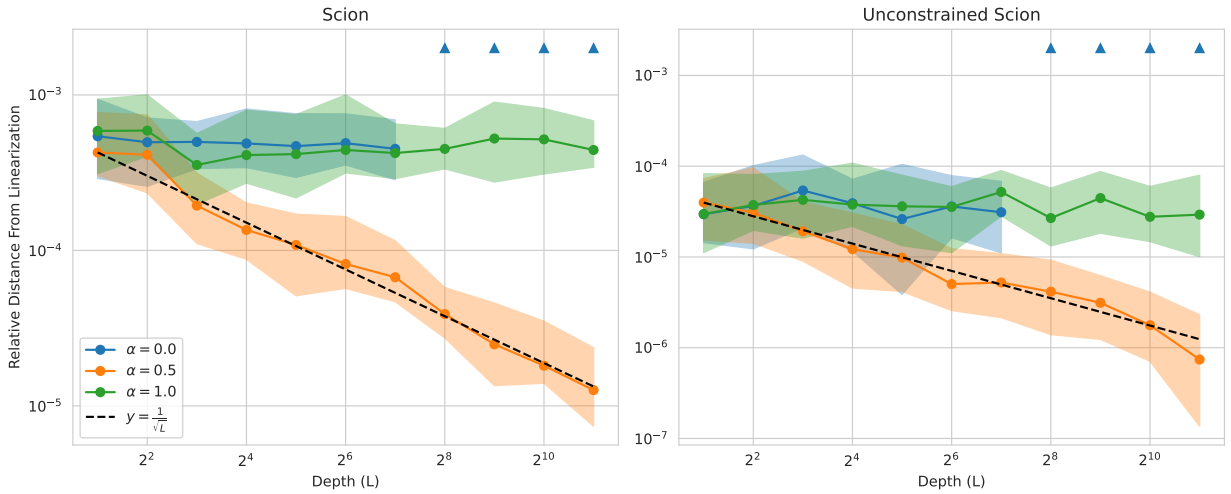



Figure 6: Relative distance from linearization for Scion and Unconstrained Scion on a 2-layer-per-block residual network, trained on MNIST. The y-axis represents Equation (25). The blue triangles represent the divergence of the case $\alpha = 0$ at large depth. Only $\alpha = 1$ results in complete and stable feature learning. $\alpha = 1/2$ approaches a linearized regime, at a rate of $\frac{1}{\sqrt{L}}$. Experimental details can be found in Table 6 in Appendix B.

### 3.6 Experiments

The architectures considered in our experiments have no learnable $Z^{(\ell)}$ matrices, which are fixed to the identity. The hidden-layer width is constant, and training is restricted to the $W^{(\ell)}$ operators.

#### 3.6.1 NanoGPT on FineWeb

We ran experiments on nanoGPT that we adapt from the code in Pethick et al. (2025a), which is based on modded-nanoGPT snapshot of Jordan et al. (2024b), itself based on the original nanoGPT of Karpathy (2023).

More experimental details can be found in Appendix B.

The architecture employs transformer blocks to implement the residual function $\mathcal{F}_\ell(h^{(\ell-1)}; \theta_{\mathcal{F}}^{(\ell)})$ for all hidden layers $\ell \in \{1, L\}$. We perform orthogonalization for the query, key, value $(Q, K, V)$, and MLP weights matrices separately, rather than applying the transformation to their concatenation.

Figures 7 and 8 show the depth-wise hyperparameter transfer for Scion and Unconstrained Scion, with the choice $\alpha = 1$ and $\alpha = 1/2$ respectively.

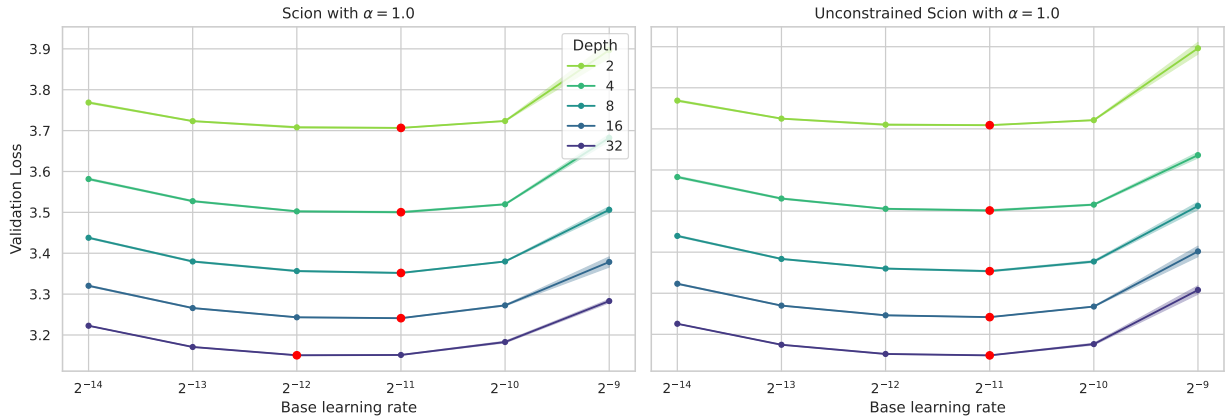



Figure 7: Scion and Unconstrained Scion on nanoGPT trained on FineWeb, with $\alpha = 1$, on 3 seeds. Depth-aligned and parallel loss curves indicate that the optimal base learning rate remain invariant with depth. When the minimizers of the curves do not exactly coincide, the loss values remain very close in this flat region. For the Scion case (left plot), the relative difference in loss value between base learning rate $2^{-12}$ and $2^{-11}$ for the deepest model is 0.024%.

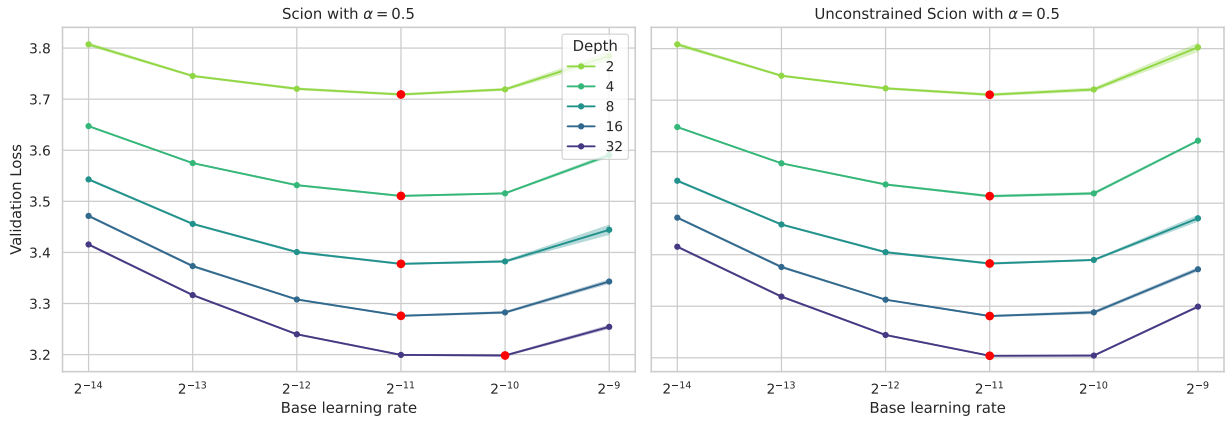



Figure 8: Scion and Unconstrained Scion on nanoGPT trained on FineWeb, with $\alpha = 0.5$, on 3 seeds. The value $\alpha = 0.5$ with the appropriate learning rate scaling also exhibits hyperparameter transfer with respect to the depth, as shown by the depth-aligned loss curves. For the Scion case (left plot), the minimizers of the curves do not exactly coincide in the flat region, and the relative difference in loss value between base learning rate $2^{-11}$ and $2^{-10}$ for the deepest model is 0.037%.

### 3.6.2 MLP-Mixer on CIFAR-10

We also used Scion on the MLP-Mixer architecture for image classification tasks (Tolstikhin et al., 2021), which offers the advantage of relying solely on MLP layers rather than convolutions, closely matching the modeling assumptions we made. Experimental details can be found in Appendix B.

The Figures 9 and 10 show the depth-wise hyperparameter transfer for Scion and Unconstrained Scion, with the choice $\alpha = 1$ and $\alpha = 1/2$ respectively.

Again, we observe on Figures 9 and 10 that the minimizers of the curves are mostly aligned, and, when small deviations occur, the minimal values remain very close.

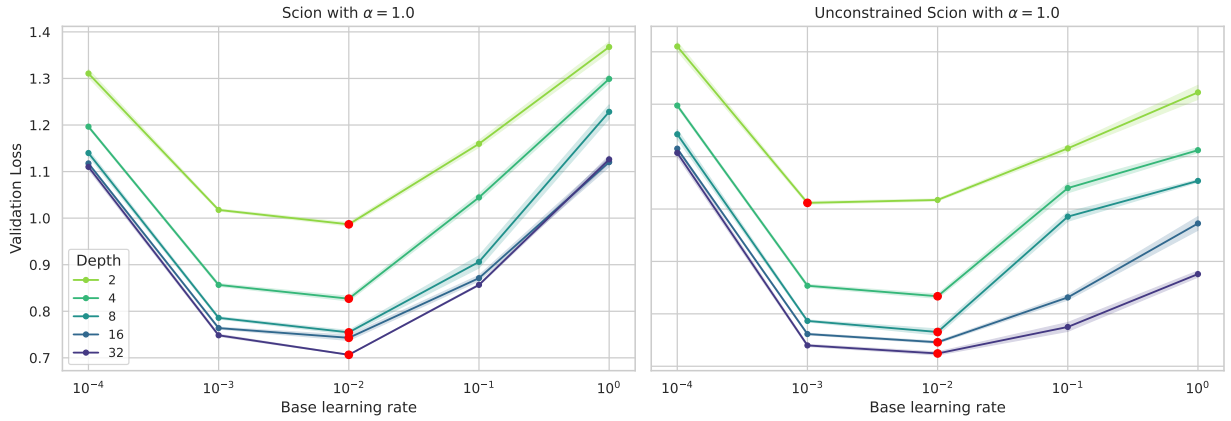



Figure 9: Scion and Unconstrained Scion on MLP-Mixer trained on CIFAR-10, with $\alpha = 1$, on 3 seeds. The curves exhibit different shapes and are not perfectly parallel, in contrast to the behavior observed in Figures 7 and 8. Nevertheless, the locations of the minimizers are largely aligned, and when deviations occur, the attained minimal values remain very close. For Unconstrained Scion (right plot), the relative difference in loss values for base learning rates $10^{-3}$ and $10^{-2}$ for the deepest model is approximately 3.2%.

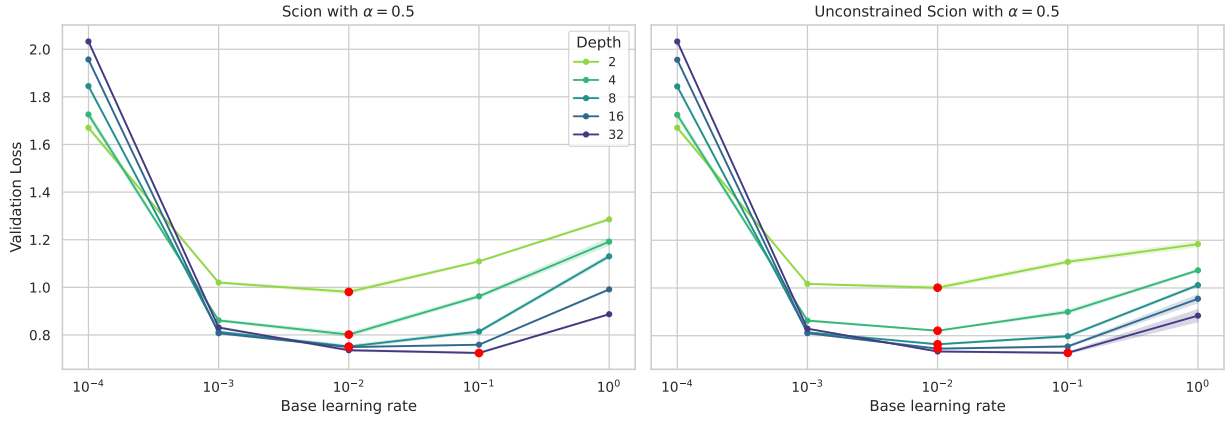



Figure 10: Scion and Unconstrained Scion on MLP-Mixer trained on CIFAR-10, with $\alpha = 0.5$, on 3 seeds. The curves display different behaviors from the nearly parallel trends observed in Figures 7 and 8. Despite these differences, the minimizers occur at similar locations, and the corresponding minimal values remain closely matched. For Scion (left plot), the relative difference in loss values for base learning rates $10^{-2}$ and $10^{-1}$ for the deepest model is approximately 1.5%. For Unconstrained Scion (right plot), the relative difference is approximately 0.75%.

### 3.6.3 DeiT on ImageNet

We trained a DeiT-base model adapted from Pethick et al. (2025a), which builds upon the original DeiT codebase of Touvron et al. (2021). For reference, the 12-layer baseline architecture by Touvron et al. (2021)

achieves a top-1 test accuracy of 81.8% on ImageNet when trained using AdamW for 300 epochs. In comparison, we achieve an accuracy of 82.3% in only 200 epochs, benefiting from the geometry-aware approach of Scion that significantly accelerates convergence and improves sample efficiency.

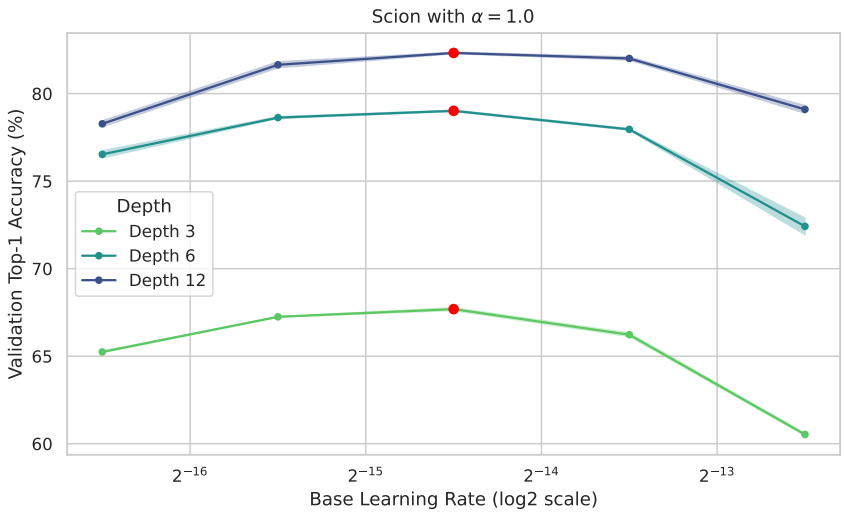



Figure 11: Scion on DeiT trained on ImageNet, with $\alpha = 1$, on 2 seeds. The similar trends and aligned optima of the top-1 accuracy curves across depths shows that the optimal base learning rate is invariant with respect to depth. The best accuracy, achieved with depth $L = 12$, is 82.3%.

## 4 Width-wise Transfer with Spectral Constraints

### 4.1 Spectral conditions for feature learning

For this section, we will simplify the setup and focus our derivations on MLPs. In particular, we desire that the features and the updates have $\Theta(1)$ element size with respect to the width.

#### 4.1.1 Problem setup

We consider a MLP with a fixed depth $L = \Theta(1)$, defined as

$$\begin{aligned} h^{(0)} &= W^{(0)}x \\ h^{(\ell)} &= W^{(\ell)}\phi(h^{(\ell-1)}) \quad (\forall \ell \in \{1, \dots, L\}) \\ f &= W^{(L+1)}\phi(h^{(L)}), \end{aligned} \tag{26}$$

where $x \in \mathbb{R}^{d_{\text{in}}}$ are the input data, $h^{(\ell)} \in \mathbb{R}^{d_\ell}, \ell \in \{0, \dots, L\}$ are the hidden states, and $f \in \mathbb{R}^{d_{\text{out}}}$ is the output after nonlinear activation $\phi$. The embedding and un-embedding matrices are respectively $W^{(0)} \in \mathbb{R}^{d_0 \times d_{\text{in}}}$ and $W^{(L+1)} \in \mathbb{R}^{d_{\text{out}} \times d_L}$.

#### 4.1.2 Spectral Scaling Condition

To ensure width-wise transfer, we adopt the conditions established by Yang et al. (2023a) for MLPs, which generalize to a broader range of architectures, including residual networks.

**Desideratum 3** (Feature Learning)**.** *The hidden layers and their updates should be stable with the width:*

$$\|h^{(\ell)}\|_2^2 = \Theta(d_\ell) \text{ and } \|\Delta h^{(\ell)}\|_2^2 = \Theta(d_\ell) \quad \forall \ell = 1, \dots, L. \tag{27}$$

Feature learning can be ensured by the spectral scaling condition on the weight matrices and their updates:

$$\|W^{(\ell)}\|_{S_\infty}^2 = \Theta(1) \text{ and } \|\Delta W^{(\ell)}\|_{S_\infty}^2 = \Theta(1) \quad \forall \ell = 1, \dots, L. \tag{28}$$

### 4.2 Width Transfer for Scion

We provide an analysis of the width-transfer property of Scion, which was only demonstrated for Unconstrained Scion in Pethick et al. (2025b).

**Proposition 3** (Scion and Unconstrained Scion verify Desideratum 3)**.** *Under the assumptions defined in the subsection 3.4.2, the updates of the features after a step of* Scion *or* Unconstrained Scion*, remain stable when the width $d_\ell$ become increasingly large.*

A demonstration of Proposition 3 is available in Appendix D.1, and an illustration of the width-wise transfer of the learning rate is given in Figure 12.

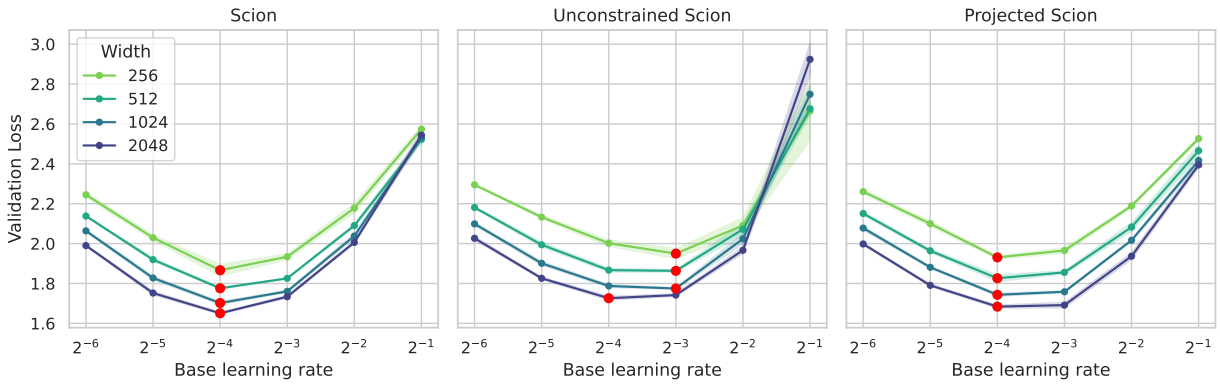



Figure 12: Width Transfer with Scion, Unconstrained Scion and Projected Scion on Shallow nanoGPT trained on Shakespeare-char dataset, with depth 3, on 3 seeds. We observe that the minimizers of the curves, highlighted by the red dots, are generally well aligned. In the case where they differ slightly, the minimal values are nearly identical, indicating a flat region around the optimum. For Unconstrained Scion (middle plot), the relative difference in loss values for base learning rates $2^{-4}$ and $2^{-3}$ for the largest model is approximately 2.9%. Experimental details can be found in Table 4 in Appendix B.

### 4.3 Projected Scion

In this section, we consider an algorithm that enforces the constraints through projections, by directly clipping the singular values of the weight matrices to lie within a certain range. Doing so provides control of the singular values of the weights and their updates, which can help stabilize training (Newhouse et al., 2025). We provide the definition of spectral clipping below.

**Definition 1** (Spectral Clipping)**.** *Let $W \in \mathbb{R}^{d\times n}$, and $W = U\Sigma V^\top$ be its singular value decomposition, where $\Sigma = diag(\sigma_1, \dots, \sigma_{min(d,n)})$ is diagonal matrix composed of the singular values of $W$, and $U, V$ are orthogonal matrices. Let $a, b \in \mathbb{R}_+^\star$ with $a < b$. The spectral clipping* $\text{spectral_clip}_{a,b} : R^{d\times n} \to R^{d\times n}$ *is defined as:*

$$\text{spectral_clip}_{[a,b]}(W) = U\,\text{clip}_{[a,b]}(\Sigma)V^\top \tag{29}$$

*where* $\text{clip}_{[a,b]}(\Sigma)$ *is the projection mapping of the singular values on* $[a, b]$.

Since computing the SVD is expensive in practice, especially for deep learning models with large weight matrices, we can reconstruct the clipping functions using the matrix sign function (Cesista, 2025) which can be approximated via the Newton-Schulz iterations (Kovarik, 1970; Björck & Bowie, 1971).

We can therefore study a variant of Scion, given in Algorithm 3, based on a projected version of SCG, where the update can be written as a composition of the spectral_clip function and the uSCG update.

Algorithm 3 also exhibits width-wise hyperparameter-transfer, as detailed in the next subsection.

**Algorithm 3** Projected SCG

**Input:** Horizon $n$, initialization $\mathbf{\Theta}_1 \in \mathcal{D}$, $M_0 = 0$, momentum $\beta_t \in [0, 1)$, spectral constraints $a$ and $b$ with $a < b$, and stepsize $\gamma_t \in (0, 1)$

**for** $t = 1, \ldots, n$ **do**
  Sample data $x \sim \mathcal{P}$
  **for** $\theta^{(\ell)} \in \mathbf{\Theta}$ **do**
    $M_t^{(\ell)} \leftarrow (1 - \beta_t)\nabla_{\theta_t^{(\ell)}}\mathcal{L} + \beta_t M_{t-1}^{(\ell)}$
    $\theta_{t+1}^{(\ell)} \leftarrow \text{spectral_clip}_{[a,b]}(\theta_t^{(\ell)} + \gamma_t \,\text{lmo}(M_t^{(\ell)}))$
  **end for**
**end for**

#### 4.3.1 Width transfer for Projected Scion

**Proposition 4** (Width invariance of the maximal update learning rate for Projected Scion)**.** *Under the assumptions defined in the subsection 3.4.2, the updates of the features after a step of* Projected Scion*, are* $\Theta(1)$ *with respect to the width* $d_\ell$*.*

In the following, we provide elements of analysis to support this result, where we consider the weights matrices constrained to lie on the Stiefel manifold. In other words, we conduct our derivations in the simplified setup where we enforce all singular values to be equal to $\sigma$. The derivations are available in Appendix D.2.

Beyond these constraints, the experiments reported in Figure 12 empirically suggest that the conclusions drawn from our theoretical analysis in the Stiefel manifold setting may extend to the more general case of clipping the singular values in an interval of the form $[0, \sigma]$.

Figure 12 illustrates the width-wise transfer property of Projected Scion.

## 5 Radius Transfer

Besides the learning rate, a remaining challenge lies in tuning the *scales* $\{\rho^{(\ell)}\}_{\ell \in \{0,\ldots,L+1\}}$ of Scion and Unconstrained Scion. These radii appear explicitly in the lmo expressions scaling the layer-wise magnitude of the updates, and therefore play a crucial role in the optimization process.

In this section, we provide empirical evidence for radius transfer for Scion and Unconstrained Scion across both width and depth. We formalize the radius transfer pipeline into the reproducible Protocol below.

**Protocol: Practical Learning Rate and Radius Transfer**

1. **Apply Scaling Rules:** Apply the parameter and learning-rate scaling rules defined in Section 3 and parametrize the radii. For instance, for an *(embedding → hidden → un-embedding)* architecture with tied embedding and un-embedding weights, parameterize the layer-wise radii configuration as $(a, 1, a)$, using a shared scalar factor $a > 0$.

2. **Low-Dimensional Sweep:** Define a small, computationally inexpensive version of the model (i.e. with small dimensions) and run a hyperparameter sweep to find the optimal learning rate and optimal $a$: $(\gamma^*, a^*)$.

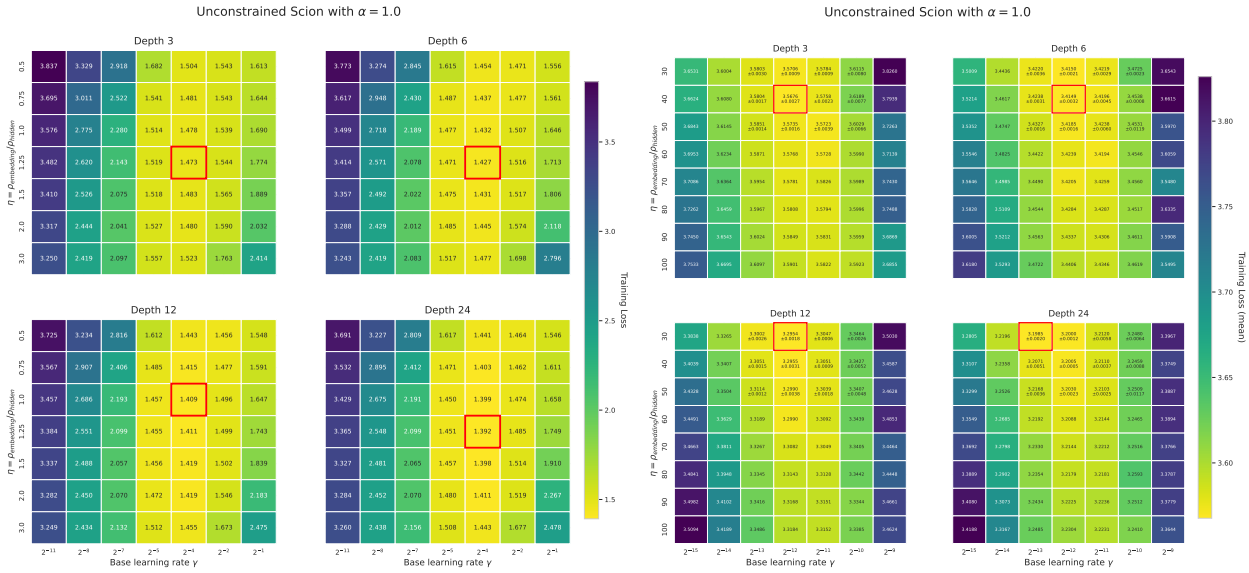
3. **Target Scale Deployment:** Train the large-scale target model directly using the optimal $(\gamma^*, a^*)$, bypassing expensive tuning at scale.

### 5.1 Radius and Depth Transfer

Large et al. (2024) defines the modular norm recursively based on the model's architecture, which in particular allows normalization within this norm using a user-specified mass. Under this modular norm framework, the authors explore mass allocation and show that both optimal mass ratio between the hidden layers and the extreme layers, and learning rate transfer from small model to larger ones, both in depth and width, when training with Adam.

We conducted a similar experiment on Shallow nanoGPT on Shakespeare-Char dataset, and on nanoGPT on FineWeb. Details of the experiments can be found in Table 4 and Table 2 respectively, in Appendix B. We trained multiple models of several depths, learning rates and ratios of radii $\eta = \frac{\rho_{\text{embedding}}}{\rho_{\text{hidden}}}$ where $\rho_{\text{embedding}}$ is the radius of the embedding and un-embedding layers and $\rho_{\text{hidden}}$ is the radius of the hidden layers. The results are displayed in Figure 13a and Figure 13b.

From this preliminary result, we observe that the transfer of both the optimal learning rate and radii occurs.




(a) Shakespeare-Char dataset. (b) FineWeb dataset.

Figure 13: Transfer of the ratio of the radii with Unconstrained Scion ($\alpha = 1$). The optimal combination of radii ratio and base learning rate is nearly identical across depths or lies within a flat region of the loss landscape, demonstrating that this landscape remains invariant with respect to depth. *Left* (a): Trained on Shallow nanoGPT, for 1 seed. For depth 12, the relative difference in loss between the cell $(\eta, \text{base learning rate}) = (1.25, 2^{-4})$ and the cell $(1.0, 2^{-4})$ is 0.1%. Experimental details are in Table 4 in Appendix B. *Right* (b): Trained on nanoGPT. The cells with std values were trained on 3 seeds. For depth 24, the relative difference in loss between the cell $(\eta, \text{base learning rate}) = (40, 2^{-12})$ and the cell $(30, 2^{-13})$ is 0.06%. Experimental details are in Table 2 in Appendix B.

### 5.2 Radius and Width & Depth Transfer

In this section, we provide empirical evidences on the learning rate transfer of Scion across both the width and the depth of the model. Figure 14 illustrates the joint width-and-depth-wise transfer of Scion.

The three models were trained for $5,100$ iterations. According to the Chinchilla scaling laws (Hoffmann et al., 2022), a compute-optimal training run requires approximately 20 tokens per parameter. Given our training configuration, where we consider a batch size of 512 and a sequence length of 1024, each training step processes $512 \times 1024 = 524,288$ tokens.

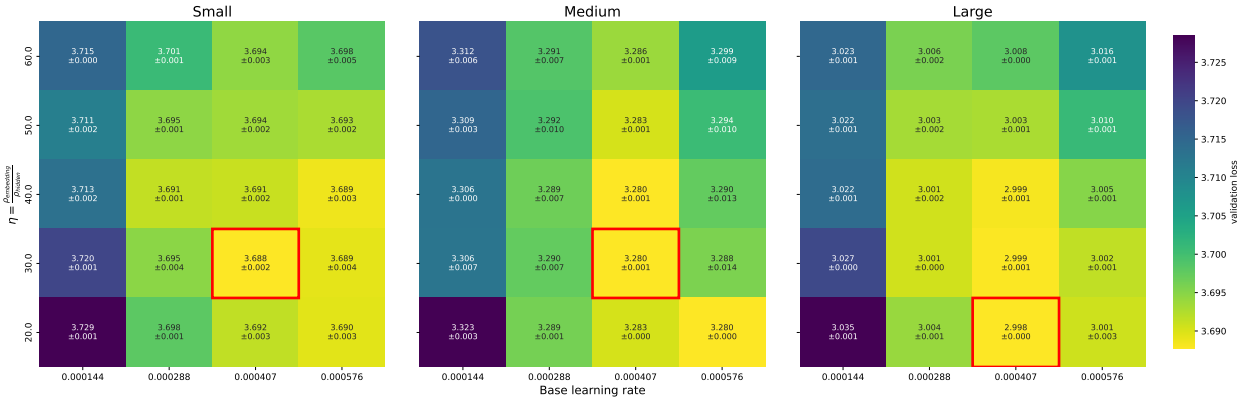



Figure 14: Width and depth transfer with Scion, on nanoGPT, trained on FineWeb, with $\alpha = 1$, on 3 seeds. The Small model has depth 6 and width 384, the Medium one has depth 12 and width 768 and the Large one has depth 24 and width 1536. For the Large model, the relative difference in loss between the cell $(\eta, \text{base learning rate}) = (30, 0.000407)$ and the cell $(20, 0.000407)$ is 0.03%. Experimental details are in Table 2 in Appendix B.

## 6 Conclusion

In this work, we investigated the hyperparameter-transfer behavior of Scion and its unconstrained variant, and we demonstrated that this optimizer supports both width and depth-wise transfer. Consistent with the findings of CompleteP (Dey et al., 2026), we observed that the setting $\alpha = 1$ yields the strongest performance and best transfer.

Our main contribution is to establish these transfer phenomena for the spectral optimizer Scion, in both its constrained and unconstrained variants, extending prior analyses of hyperparameter transfer.

Through a diverse set of experiments, we further showed that Scion can be parameterized to guarantee feature learning, operation in the non-lazy regime, and movement from initialization at the maximum stable learning rate.

Our experiments covered several deep neural network architectures and datasets, including nanoGPT trained on FineWeb, DeiT on ImageNet, and MLP-Mixer on CIFAR-10.

We finish by remarking that these results can also apply to Muon, provided that the components relying on Adam are scaled appropriately, and ensure that it remains compatible with the transfer regime.

### 6.1 Perspectives

Given the promising preliminary results observed in Section 5, it would be interesting to theoretically investigate the radius transfer property and derive a rigorous proof. Leveraging this hyperparameter property could prove highly beneficial for several downstream tasks. In particular, transferring the radius of a constrained set has direct implications for improving robustness against adversarial attacks (Goodfellow et al., 2015; Sokolić et al., 2017; Miyato et al., 2018), generalization bounds (Bartlett et al., 2017), or control on the Lipschitz continuity of neural networks (Large et al., 2024; Miyato et al., 2018). Extending to include other parameterizations of weight decay than the Frank-Wolfe step remains an interesting direction too (Chou, 2025).

## References

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jin Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Gregory Frederick Diamos, Erich Elsen, Jesse Engel, Linxi (Jim) Fan, Christopher Fougner, Awni Y. Hannun, Billy Jun, Tony Xiao Han, Patrick LeGresley, Xiangang Li, Libby Lin, Sharan Narang, A. Ng, Sherjil Ozair, Ryan J. Prenger, Sheng Qian, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Anuroop Sriram, Chong-Jun Wang, Yi Wang, Zhiqian Wang, Bo Xiao, Yan Xie, Dani Yogatama, Junni Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning*, 2015.

Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Åke Björck and Clazett Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *Advances in Neural Information Processing Systems*, 35:32240–32256, 2022.

Blake Bordelon, Hamza Chaudhry, and Cengiz Pehlevan. Infinite limits of multi-head transformer dynamics. *Advances in Neural Information Processing Systems*, 37:35824–35878, 2024a.

Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *The Twelfth International Conference on Learning Representations*, 2024b.

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted boltzmann machines. In *Artificial intelligence and statistics*, pp. 111–119. PMLR, 2015a.

David Carlson, Ya-Ping Hsieh, Edo Collins, Lawrence Carin, and Volkan Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):296–311, 2015b.

David E Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. *Advances in neural information processing systems*, 28, 2015c.

Franz Louis Cesista. Fast, numerically stable, and auto-differentiable spectral clipping via Newton-Schulz iteration, 2025. URL `http://leloykun.github.io/ponder/spectral-clipping/`.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

Jason Chuan-Chih Chou. Correction of decoupled weight decay. *arXiv preprint arXiv:2512.08217*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: CompleteP enables compute-efficient deep transformers, 2026.

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639): 115–118, 2017.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*, 2015.

Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, 2013.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. arXiv preprint, 2018.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in Neural Information Processing Systems*, 28, 2015.

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)*, pp. 770–778, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Phuongtrang Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29:82–97, 2012.

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 preprint arXiv:2203.15556*, 2022.

Guang Hu, Taotao Zhou, and Qianfeng Liu. Data-driven machine learning for fault detection and diagnosis in nuclear power plants: A review. *Frontiers in Energy Research*, 9:663296, 2021.

Rustem Islamov, Roman Machacek, Aurelien Lucchi, Antonio Silveti-Falls, Eduard Gorbunov, and Volkan Cevher. On the role of batch size in stochastic conditional gradient methods. *arXiv preprint arXiv:2603.21191*, 2026.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, `@fernbear.bsky.social`, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and `@Grad62304977`. Modded-NanoGPT: Speedrunning the NanoGPT baseline, 2024a. URL `https://github.com/KellerJordan/modded-nanogpt`.

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

Kyle D. Julian and Mykel J. Kochenderfer. Guaranteeing safety for neural network-based aircraft collision avoidance systems. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pp. 1–10. IEEE, 2019.

Andrej Karpathy. nanoGPT. `https://github.com/karpathy/nanoGPT`, 2023. Accessed: 2025-01-25.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, et al. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Diederik P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Atli Kosson, Jeremy Welborn, Yang Liu, Martin Jaggi, and Xi Chen. Weight decay may matter more than muP for learning rate transfer in practice. *arXiv preprint arXiv:2510.19093*, 2025.

Zdislav Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 7(3):386–389, 1970.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Tim Large, Yang Liu, Minyoung Huh, Hyojin Bahng, Phillip Isola, and Jeremy Bernstein. Scalable optimization in the modular norm. *Advances in Neural Information Processing Systems*, 37:73501–73548, 2024.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, April 30 – May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Laker Newhouse, R Preston Hess, Franz Cesista, Andrii Zahorodnii, Jeremy Bernstein, and Phillip Isola. Training transformers with enforced lipschitz constants. *arXiv preprint arXiv:2507.13338*, 2025.

NVIDIA NeMo Team. Emerging optimizers. `https://github.com/NVIDIA-NeMo/Emerging-Optimizers`, 2025. NVIDIA NeMo, `https://docs.nvidia.com/nemo/emerging-optimizers/`.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *The Thirty-Eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Thomas Pethick, Kimon Antonakopoulos, Antonio Silveti-Falls, Leena Chennuru Vankadara, and Volkan Cevher. Training neural networks at any scale. *arXiv preprint arXiv:2511.11163*, 2025a.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Forty-Second International Conference on Machine Learning (ICML 2025)*, 2025b.

Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. *arXiv preprint arXiv:2512.05620*, 2025.

Liliang Ren, Yang Liu, Yelong Shen, and Weizhu Chen. Rethinking language model scaling under transferable hypersphere optimization. *arXiv preprint arXiv:2603.28743*, 2026.

Martin A. Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, 1993. URL `https://api.semanticscholar.org/CorpusID:16848428`.

Boris Shigida, Boris Hanin, and Andrey Gromov. Learning rate transfer in normalized transformers. *arXiv preprint arXiv:2604.27077*, 2026.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Varun Singh, Lucas Krauss, Sami Jaghouar, Matej Sirovatka, Charles Goddard, Fares Obied, Jack Min Ong, Jannik Straube, Aria Harley, Conner Stewart, et al. Arcee trinity large technical report. *arXiv preprint arXiv:2602.17004*, 2026.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and M. R. D. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, PP:1–1, 2017.

Gang Sun and Shuyue Wang. A review of the artificial neural network surrogate modeling in aerodynamic design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16):5863–5872, 2019.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

Chenwei Tang, Caiyang Yu, Yi Gao, Jianming Chen, Jiaming Yang, Jiuling Lang, Chuan Liu, Ling Zhong, Zhenan He, and Jiancheng Lv. Deep learning in nuclear industry: A survey. *Big Data Mining and Analytics*, 5(2):140–160, 2022.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. Attention residuals. *arXiv preprint arXiv:2603.15031*, 2026.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision, 2021.

Eric J. Topol. High-performance medicine: The convergence of human and artificial intelligence. *Nature Medicine*, 25(1):44–56, 2019.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139, pp. 10347–10357, July 2021.

Leena Chennuru Vankadara, Jin Xu, Moritz Haas, and Volkan Cevher. On feature learning in structured state space models. *Advances in Neural Information Processing Systems*, 37:86145–86179, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

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

Greg Yang and Etai Littwin. Tensor programs IVb: Adaptive optimization in the infinite-width limit. In *International Conference on Learning Representations (ICLR)*, 2023.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs VI: Feature learning in infinite-depth neural networks. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023b.

Chenyu Zheng, Rongzhen Wang, Xinyu Zhang, and Chongxuan Li. Spectral condition for $\mu$P under width-depth scaling. *arXiv preprint arXiv:2603.00541*, 2026.

# A Additional figures

## A.1 Additional plots from the NanoGPT experiments

We trained multiple models where we vary the depth and the learning rate. Figure 15 illustrates the depth-wise hyperparameter transfer with $\alpha = 1.0$ and $\alpha = 0.5$, for Scion and Unconstrained Scion.

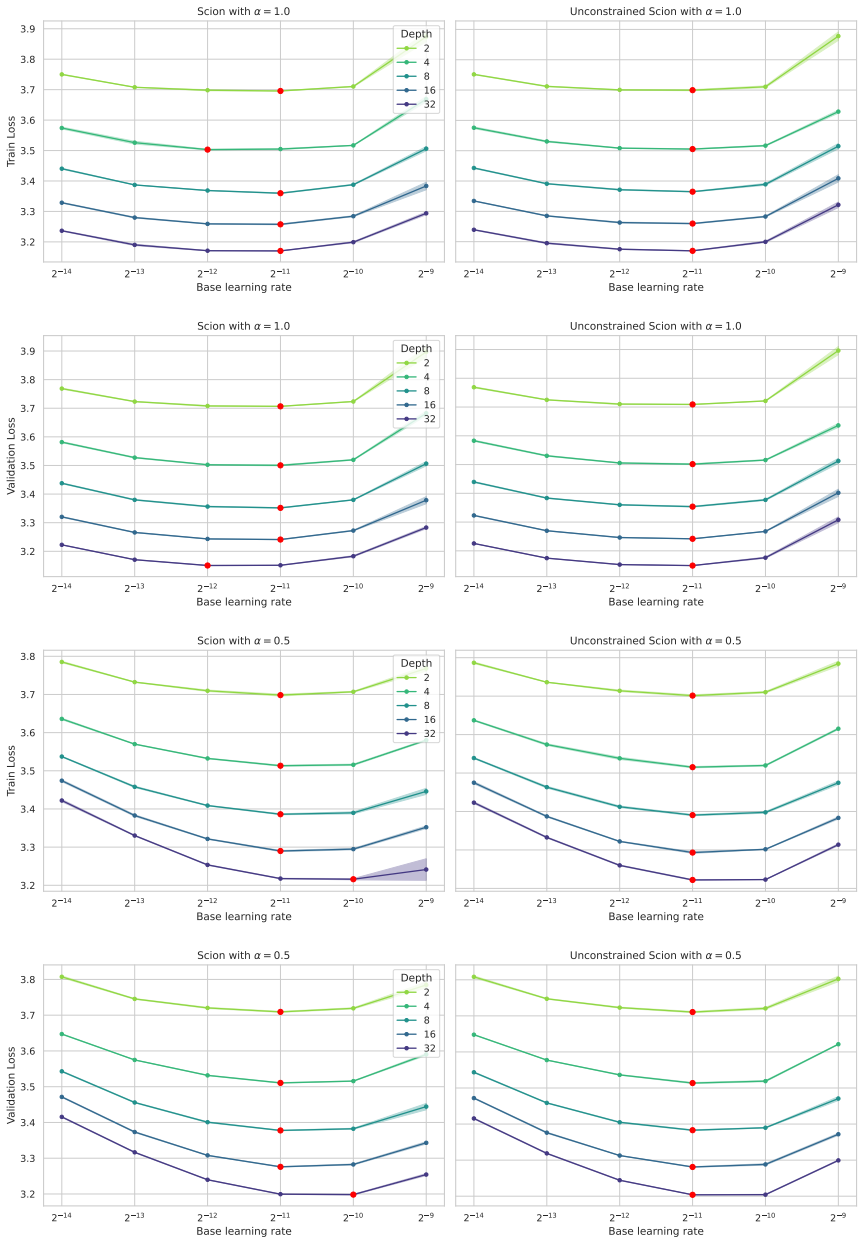



Figure 15: Scion (left columns) and Unconstrained Scion (right columns) on nanoGPT trained on FineWeb, on 3 seeds. The top rows corresponds to $\alpha = 1.0$ and the bottom rows to $\alpha = 0.5$. The minimizers of the different curves, indicated by red dots, are strongly aligned across depths. When they do not exactly coincide, the corresponding loss values remain very close, indicating a flat loss region around the optimum. The validation loss relative errors are reported in Figures 7 and 8. The training loss relative error with Scion is approximately 0.050% with $\alpha = 1$ and depth $L = 4$, and 0.052% with $\alpha = 0.5$ and depth $L = 32$.

### A.2 Additional plots from the MLP-Mixer experiments

Figure 16 illustrates the hyperparameter transfer in depth with $\alpha = 1.0$ and $\alpha = 0.5$ for Scion and Unconstrained Scion.

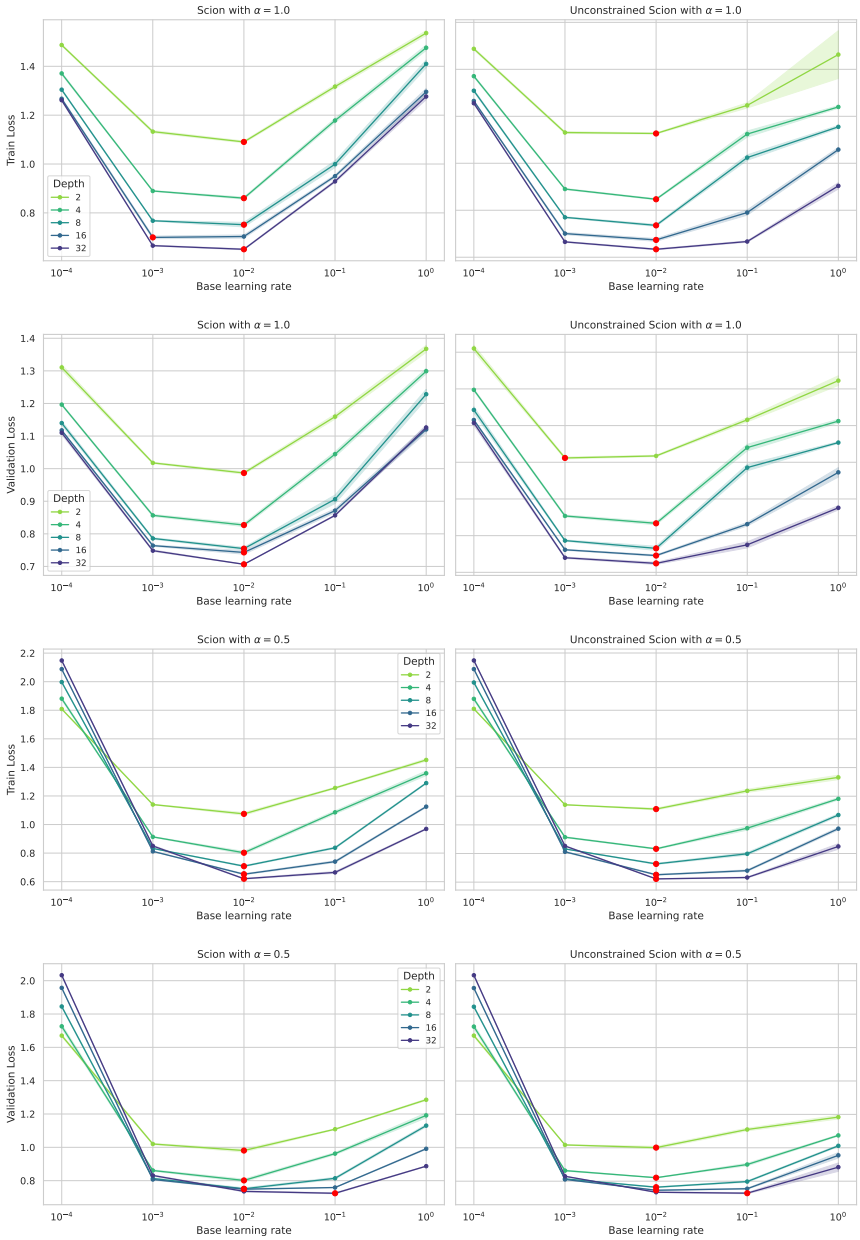



Figure 16: Scion (left columns) and Unconstrained Scion (right columns) on MLP-Mixer trained on CIFAR-10, on 3 seeds. The top row corresponds to $\alpha = 1.0$ and the bottom row to $\alpha = 0.5$. Compared to the FineWeb experiments, the loss curves are less parallel across depths, indicating greater heterogeneity in optimization dynamics. Nevertheless, the minimizers of the curves, indicated by red dots, remain largely aligned. When deviations occur, they lie within a flat region of the loss landscape, where the achieved loss values remain very close. The validation loss relative errors are reported in Figures 9 and 10. The training loss relative error with Scion with $\alpha = 1$ depth $L = 32$ is approximately 0.80%.

## B Experimental details

**NanoGPT** We ran experiments on a snapshot of the speed-running model modded-nanoGPT of Jordan et al. (2024a), which is an evolution of the original nanoGPT of Karpathy (2023). Specifically, it replaces positional embeddings with rotary embeddings, uses RMSnorm in place of LayerNorm, uses a linear learning-rate decay scheduler and employs a ScaledReLU$^2$ activation function. As this architecture ties the embedding and unembedding weights, and following the Recommendation 3.1 of Pethick et al. (2025b), we configured Scion and Unconstrained Scion using the Sign norm for the embedding and unembedding layers and the Spectral norm for the hidden layers: `Sign` $\rightarrow$ `Spectral` $\rightarrow$ `Sign`. The models are trained on the FineWeb dataset (Penedo et al., 2024), for 5100 iterations with a batch size of 512 sequences of length 1024, totaling approximately $2.67 \times 10^9$ tokens. According to Chinchilla scaling laws (Hoffmann et al., 2022), this data budget is compute optimal for a model with roughly $133 \times 10^6$ parameters, which corresponds to a depth of 19 layers. Our evaluation across depths 2, 4, 8, 16, and 32 thus allow us to analyze model performance in both the under-parameterized regime and the over-parameterized regime, with depth 16 serving as our closest near-optimal baseline.

The details on the hyperparameters can be found in Table 2. The experiments were conducted on four H100 GPUs.

**Shallow GPT** We also consider a nanoGPT model (Karpathy, 2023) where we additionally remove the weight sharing between the first and last layer so that the various input layers from can be investigated. Following Pethick et al. (2025b), we configured Scion and Unconstrained Scion with a `Spectral` $\rightarrow$ `Spectral` $\rightarrow$ `Sign` norm configuration across the embedding, hidden and un-embedding layers. The hyperparameters can be found in Table 4. The experiments were performed using four H100 GPUs.

**MLP-Mixer** MLP-Mixer is an alternative architecture to transformers and is based entirely on MLPs. It is reported to achieve performance comparable to that of Convolutional Neural Networks and Transformers. Each Mixer layer consists of a token-mixing MLP, and a channel-mixing MLP, with skip-connections. Each of these MLP has two layers and GELU activation function (Gaussian Error Linear Units function, defined as $GELU(x) = x\,\Phi(x)$where $\Phi$ is the Cumulative Distribution Function for Gaussian distribution). We replaced the Layer norm from Tolstikhin et al. (2021) with RMSnorm.

We used random cropping and data augmentation, by generating mixed samples on CIFAR10. The training was performed on 300 epochs, with a batch size of 2048. We used a cosine learning rate scheduler with a linear warm-up. The embedding dimension is set to 128, the hidden widths of the token-mixing MLP and the channel-mixing MLP are 64 and 512 respectively.

*Why not convolutions ?* We can note here that Scion does not provide spectral norm constraint guarantees to convolutional kernels, which also motivates the choice of this architecture. Indeed, the current Scion algorithm applies its constraints to the reshaped kernel tensor, rather than to the linear convolution operator, which corresponds to a large block-Toeplitz matrix. Since the spectral norm of the reshaped kernel is not the spectral norm of the induced convolution operator, the constraint enforced by the lmo is not applicable to the operator norm. Therefore, Scion cannot provide reliable spectral constraints for convolutional layers in practice.

Following the Recommendation 3.1 of Pethick et al. (2025b) for computer vision tasks, we used the Spectral norm for the first and hidden layers and Sign for the last layer: `Spectral` $\rightarrow$ `Spectral` $\rightarrow$ `Sign`.

The details on the hyperparameters can be found in Table 5. The experiments were conducted on four V100 GPUs.

**DeiT** The DeiT-base model utilized in this work was adapted from the codebase of Pethick et al. (2025a), which originates from the official DeiT implementation developed by Touvron et al. (2021). Following Pethick et al. (2025a), we used the norm configuration `Spectral` $\rightarrow$ `Spectral` $\rightarrow$ `Sign`. Additionally, the architecture substituted LayerNorm with RMSNorm, scaled the GELU activation function by a factor of $\sqrt{2}$, and use bias parameters, which slightly differs from the setup of our other experiments. See Table 3 for detail on the hyperparameters. The experiments were conducted on $4 \times 4$ H100 GPUs.

Table 2: NanoGPT hyperparameters.

| Hyperparameter | Unconstrained Scion and Scion |
|---|---|
| Width | 768 |
| Head dim | 128 |
| Activation function | ScaledReLU$^2$ |
| Vocabulary size | 50304 |
| Dataset | FineWeb |
| Batch size | 512 |
| Block size | 1024 |
| Iterations $n$ | 5100 |
| Warmdown | 28.5% |
| Stepsize schedule | Constant then linear decay |
| Averaging parameter $\beta$ | 0.9 |
| Radii | 3000/50/3000 |

Table 3: DeiT hyperparameters.

| Hyperparameter | Scion |
|---|---|
| Head dim | 64 |
| Activation function | $\sqrt{2}\cdot$ GELU |
| Normalization function | RMSNorm |
| Sequence length | 197 |
| Dataset | ImageNet-1k |
| Batch size | 4096 |
| Epochs | 200 |
| Stepsize schedule | Constant then cosine decay |
| Averaging parameter $\beta$ | 0.1 |
| Radii | 25/25/500 |

Table 4: Shallow GPT hyperparameters.

| Hyperparameter | Unconstrained Scion |
|---|---|
| Depth (in the width-transfer experiments) | 3 |
| Width (in the depth-transfer experiments) | 256 |
| Head dim | 64 |
| Activation function | $\sqrt{2}\cdot$ GELU |
| Vocabulary size | 64 |
| Dataset | Shakespeare Char |
| Batch size | 32 |
| Block size | 1024 |
| Iterations $n$ | 122 |
| Stepsize schedule | Linear decay |
| Averaging parameter $\beta$ | 0.9 |

Table 5: MLP-Mixer hyperparameters.

| Hyperparameter | Unconstrained Scion and Scion |
|---|---|
| Dataset | CIFAR10 |
| Embedding dimension | 128 |
| Token-mixing MLP hidden width | 64 |
| Channel-mixing MLP hidden width | 512 |
| Activation | ReLU |
| Batch size | 2048 |
| Learning rate schedule | Cosine decay with linear warm-up |
| Training epochs $n$ | 300 |
| Averaging parameter $\beta$ | 0.9 |
| Radii | 25/25/500 |

Table 6: Residual Network hyperparameters (for the relative distance from linearization analysis in Figure 6).

| Hyperparameter | Unconstrained Scion and Scion |
|---|---|
| Dataset | MNIST |
| Layers per branch | 2 |
| Activation | ReLU |
| Batch size | 128 |
| Iterations $n$ | 1 |
| Averaging parameter $\beta$ | 0.9 |

# C Proofs of Complete Scion from Section 3.4

We recall Desiderata 1 and 2 and demonstrate in the following that they hold for Scion and Unconstrained Scion.

## C.1 Depth transfer desiderata

**Desideratum 1** (Stable Initialization ). *The hidden layers and the output remain stable (preventing signals from exploding or vanishing as they pass through many layers) at the initialization when $L$ and $d_\ell$ become increasingly large:*

$$(\forall \ell \in \{1, \dots, L\}) \;\; \frac{1}{d_\ell}\|h^{(\ell)}\|_2^2 \in \Theta(1) \;\; \textit{and} \;\; \|f\|_2^2 \in \mathcal{O}(1), \quad \textit{as } d_\ell \to \infty, L \to \infty. \tag{7}$$

**Desideratum 2** (Stable Feature Update ). [3] *The weights of each residual block should contribute with order $1/L$ to feature movements, and each non-residual block should contribute with constant order (e.g., each non-residual block has a meaningful impact on the residual stream):*

$$\begin{aligned} &\frac{1}{\sqrt{d_\ell}}\|\Delta_{\theta_{\mathcal{F}}^{(\ell)}, Z^{(\ell)}} h^{(\ell)}\|_2 \in \Theta\left(\frac{1}{L}\right) \quad (\forall \ell \in \{1, \dots, L\}), \\ &\frac{1}{\sqrt{d_0}}\|\Delta W^{(0)} x\|_2 \in \Theta(1), \\ &\frac{1}{\sqrt{d_{\text{out}}}}\|\Delta W^{(L+1)} h^{(L)}\|_2 \in \Theta(1), \end{aligned} \tag{8}$$

*where $\Delta$ designates the variation of the parameters between two successive iterations.*

In the following, we will use the following argument: $\forall a, b > 0, (a+b)^2 = \Theta(a^2 + b^2)$. Indeed:

$$\begin{aligned} &(a+b)^2 = \mathcal{O}(a^2+b^2) \text{ since } (a+b)^2 = a^2 + 2ab + b^2 \leq 2(a^2+b^2). \\ \text{and} \quad &(a+b)^2 = \Omega(a^2+b^2) \text{ since } (a+b)^2 = a^2 + 2ab + b^2 \geq a^2 + b^2 \text{ with } a, b > 0, \\ \text{therefore :} \; &(a+b)^2 = \Theta(a^2+b^2), \forall a, b > 0. \end{aligned} \tag{30}$$

## C.2 Assumptions and tools

Throughout this section, we maintain the same assumptions introduced in the main body of the paper in Section 3.4.2. For brevity, these foundational assumptions are not repeated here.

**Update expression of $W^{(\ell)}$** Since we consider a residual network, we need to compute, for every layer $\ell$

$$\Delta W^{(\ell)} = \gamma \, \text{lmo}(\nabla_{W^{(\ell)}} \mathcal{L}) \qquad \text{with Unconstrained Scion} \tag{31}$$

$$\Delta W^{(\ell)} = -\gamma W^{(\ell)} + \gamma \, \text{lmo}(\nabla_{W^{(\ell)}} \mathcal{L}) \quad \text{with Scion} \tag{32}$$

We compute expression for $\text{lmo}(\nabla_{\mathrm{W}^{(\ell)}} \mathcal{L})$.

$$\nabla_{W^{(\ell)}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial h^{(\ell)}} \frac{\partial h^{(\ell)}}{\partial W^{(\ell)}} = \frac{\partial \mathcal{L}}{\partial h^{(\ell)}} \frac{1}{L^\alpha} \phi(h^{(\ell-1)})^\top.$$

According to the lemma C5 of Pethick et al. (2025b), the lmo of the rank one gradient with batch size one can be written as:

$$\text{lmo}\left(\frac{\partial \mathcal{L}}{\partial h^{(\ell)}} \phi(h^{(\ell-1)})^\top\right) = \rho^{(\ell)} \sqrt{\frac{d_\ell}{d_{\ell-1}}} \frac{\frac{\partial \mathcal{L}}{\partial h^{(\ell)}}}{\|\frac{\partial \mathcal{L}}{\partial h^{(\ell)}}\|_2} \frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2}. \tag{33}$$

[3] The corresponding Desideratum in Dey et al. (2026) has $\frac{1}{d_\ell}\|\Delta_{\theta^{(\ell)}} h^{(\ell)}\|_2^2 = \Theta(\frac{1}{L})$, which slightly differs from ours, as we discuss in Appendix E.

Therefore, in the case of our residual model, and by using the scale-invariance property of the lmo, we have

$$\mathrm{lmo}(\nabla_{W^{(\ell)}}\mathcal{L}) = \rho^{(\ell)}\sqrt{\frac{d_\ell}{d_{\ell-1}}}\frac{\frac{\partial\mathcal{L}}{\partial h^{(\ell)}}}{\|\frac{\partial\mathcal{L}}{\partial h^{(\ell)}}\|_2}\frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2} \quad (\forall \ell \in \{1,\dots,L\}), \tag{34}$$

$$\mathrm{lmo}(\nabla_{W^{(0)}}\mathcal{L}) = \rho^{(0)}\sqrt{\frac{d_0}{d_{\mathrm{in}}}}\frac{\frac{\partial\mathcal{L}}{\partial h^{(0)}}}{\|\frac{\partial\mathcal{L}}{\partial h^{(0)}}\|_2}\frac{x^\top}{\|x\|_2}, \tag{35}$$

$$\mathrm{lmo}(\nabla_{W^{(L+1)}}\mathcal{L}) = \rho^{(L+1)}\sqrt{\frac{d_{\mathrm{out}}}{d_L}}\frac{\frac{\partial\mathcal{L}}{\partial f}}{\|\frac{\partial\mathcal{L}}{\partial f}\|_2}\frac{\phi(h^{(L)})^\top}{\|\phi(h^{(L)})\|_2}. \tag{36}$$

**Update expression of $Z^{(\ell)}$** Similarly, let us suppose we learn the $\{Z^{(\ell)}\}_{\ell\in\{1,\dots,L\}}$ with the associated learning rate $\gamma_Z$ and radii $\{\rho_{Z\ell}\}_{\ell\in\{1,\dots,L\}}$. Then, we have

$$\nabla_{Z^{(\ell)}}\mathcal{L} = \frac{\partial\mathcal{L}}{\partial h^{(\ell)}}\frac{\partial h^{(\ell)}}{\partial Z^{(\ell)}} = \frac{\partial\mathcal{L}}{\partial h^{(\ell)}}h^{(\ell-1)^\top}.$$

Therefore,

$$\Delta Z^{(\ell)} = \gamma_Z\,\mathrm{lmo}(\nabla_{Z^{(\ell)}}\mathcal{L}) = \gamma_Z\rho_{Z\ell}\sqrt{\frac{d_\ell}{d_{\ell-1}}}\frac{\frac{\partial\mathcal{L}}{\partial h^{(\ell)}}}{\|\frac{\partial\mathcal{L}}{\partial h^{(\ell)}}\|_2}\frac{h^{(\ell-1)^\top}}{\|h^{(\ell-1)}\|_2} \quad (\forall \ell \in \{1,\dots,L\}). \tag{37}$$

### C.3 Proof that Desideratum 1 holds for Scion and Unconstrained Scion

**Proposition 1** (Scion and Unconstrained Scion verify Desideratum 1)**.** *Under the assumptions defined in the subsection 3.4.2, the hidden layers and the output remain stable at the initialization when $L$ and $d_\ell$ become increasingly large.*

We provide here the complete demonstration that the Desideratum 1 is satisfied for Scion and Unconstrained Scion.

*Proof.* Let $\ell \in \{1,\dots,L\}$. Using the alignment of the weights (Equation (13)) and the scale of the activation function $\phi$ assumptions (Equation (14)), we have

$$\begin{aligned}
\|h^{(\ell)}\|_2^2 &= \Theta(\|Z^{(\ell)}h^{(\ell-1)}\|_2^2 + \frac{1}{L^{2\alpha}}\|W^{(\ell)}\phi(h^{(\ell-1)})\|_2^2) \\
&= \Theta\left(\|Z^{(\ell)}\|_{\mathcal{S}_\infty}^2\|h^{(\ell-1)}\|_2^2 + \frac{1}{L^{2\alpha}}\|W^{(\ell)}\|_{\mathcal{S}_\infty}^2\|h^{(\ell-1)}\|_2^2\right) \qquad \text{from Equation (13) and Equation (14)} \\
&= \Theta\left((\|Z^{(\ell)}\|_{\mathcal{S}_\infty}^2 + \frac{1}{L^{2\alpha}}\|W^{(\ell)}\|_{\mathcal{S}_\infty}^2)\|h^{(\ell-1)}\|_2^2\right) \\
&= \Theta\left(\prod_{k=1}^{\ell}(\|Z^{(k)}\|_{\mathcal{S}_\infty}^2 + \frac{1}{L^{2\alpha}}\|W^{(k)}\|_{\mathcal{S}_\infty}^2)\|h^0\|_2^2\right).
\end{aligned}$$

From the alignment of the weights assumption (Equation (13)), combined with the initialization of the weights (Equation (12)), we have $\|h^0\|_2^2 = \Theta(\|W^0\|_{\mathcal{S}_\infty}^2 \|x\|_2^2) = \Theta\left(\frac{d_0}{d_{\text{in}}} d_{\text{in}}\right) = \Theta(d_0)$. Then,

$$\begin{aligned}
\|h^{(\ell)}\|_2^2 &= \Theta\left(\prod_{k=1}^{\ell} (\|Z^{(k)}\|_{\mathcal{S}_\infty}^2 + \frac{1}{L^{2\alpha}} \|W^{(k)}\|_{\mathcal{S}_\infty}^2) d_0\right) \\
&= \Theta\left(\prod_{k=1}^{\ell} (\frac{d_k}{d_{k-1}} + \frac{1}{L^{2\alpha}} \frac{d_k}{d_{k-1}}) d_0\right) \\
&= \Theta\left(\prod_{k=1}^{\ell} (\frac{d_k}{d_{k-1}} \times (1 + \frac{1}{L^{2\alpha}})) d_0\right) \\
&= \Theta\left(\frac{d_\ell}{d_0} \prod_{k=1}^{\ell} (1 + \frac{1}{L^{2\alpha}}) d_0\right) \\
&= \Theta\left(d_\ell \prod_{k=1}^{\ell} (1 + \frac{1}{L^{2\alpha}})\right).
\end{aligned}$$

Then, $\frac{1}{d_L}\|h^{(L)}\|_2^2 = \Theta\left((1 + \frac{1}{L^{2\alpha}})^L\right)$ converges when $L$ goes to infinity if and only if $\alpha \geq \frac{1}{2}$. Indeed, if we denote $y = \left(1 + \frac{1}{L^{2\alpha}}\right)^L$, then, by considering Taylor expansion, we have $\ln(y) = L \ln\left(1 + \frac{1}{L^{2\alpha}}\right) = L\left(\frac{1}{L^{2\alpha}} - \frac{1}{2L^{4\alpha}} + \mathcal{O}(\frac{1}{L^{6\alpha}})\right) = L^{1-2\alpha} - \frac{1}{2}L^{1-4\alpha} + \mathcal{O}(L^{1-6\alpha})$. Therefore,

$$\begin{cases} \ln(y) \to \infty \text{ and } y \to \infty & \text{if } \alpha < \frac{1}{2} \\ \ln(y) \to 1 \text{ and } y \to e & \text{if } \alpha = \frac{1}{2} \\ \ln(y) \to 0 \text{ and } y \to 1 & \text{if } \alpha > \frac{1}{2}. \end{cases}$$

Therefore,

$$\frac{1}{d_\ell}\|h^{(\ell)}\|_2^2 = \Theta(1) \quad \forall \ell \in \{1, \ldots, L\} \text{ for } \alpha \geq \frac{1}{2}. \tag{38}$$

For $f = W^{L+1}\phi(h^{(L)})$, using the alignment of the weights and the scale of the activation function $\phi$ assumptions, we have

$$\|f\|_2^2 = \Theta(\|W^{L+1}\|_{\mathcal{S}_\infty}^2 \|h^{(L)}\|_2^2) \qquad \text{from Equation (13) and Equation (14).}$$

Then, using the initialization of the weights and the previous derivations of $\|h^{(L)}\|_2^2$, we have

$$\begin{aligned}
\|f\|_2^2 &= \Theta(\frac{d_{\text{out}}}{d_L} \times d_L) \qquad \text{from Equation (12) and Equation (38)} \\
&= \Theta(1) \\
&= \mathcal{O}(1).
\end{aligned}$$

These derivations are the same for Scion, since they only relate to the initialization of the model. □

### C.4 Proof that Desideratum 2 holds for Scion and Unconstrained Scion

**Proposition 2** (Scion and Unconstrained Scion verify Desideratum 2)**.** *Under the assumptions defined in subsection 3.4.2, for* Scion *or* Unconstrained Scion *the weights of each block contribute with order $1/L$ to feature movements, and each non-residual block contributes with constant order when $L$ becomes increasingly large.*

We provide here the complete demonstration that the Desideratum 2 is satisfied for Unconstrained Scion and Scion.

#### C.4.1 Unconstrained Scion case

*Proof.* Let $\ell \in \{1, \dots, L\}$. For Unconstrained Scion, we use the expressions of the updates $\Delta W^{(\ell)}$ and $\Delta Z^{(\ell)}$ as in Equation (31), that are written as

$$\Delta W^{(\ell)} = \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}} \mathcal{L}) \tag{39}$$

$$\Delta Z^{(\ell)} = \gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}} \mathcal{L}). \tag{40}$$

Writing the expression of the layer $h^{(\ell)}$ in function of the previous $h^{(\ell-1)}$, we have

$$\begin{aligned}
\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)} &= h_{t+1}^{(\ell)} - h_t^{(\ell)} \\
&= Z_{t+1}^{(\ell)} h_{t+1}^{(\ell-1)} + L^{-\alpha} W_{t+1}^{(\ell)} \phi(h_{t+1}^{(\ell-1)}) - Z_t^{(\ell)} h_t^{(\ell-1)} - L^{-\alpha} W_t^{(\ell)} \phi(h_t^{(\ell-1)}).
\end{aligned}$$

Since we are only looking at the change with the update of $W^{(\ell)}$ and $Z^{(\ell)}$, we have $h_{t+1}^{(\ell-1)} = h_t^{(\ell-1)}$. Therefore,

$$\begin{aligned}
\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)} &= Z_{t+1}^{(\ell)} h_t^{(\ell-1)} + L^{-\alpha} W_{t+1}^{(\ell)} \phi(h_t^{(\ell-1)}) - Z_t^{(\ell)} h_t^{(\ell-1)} - L^{-\alpha} W_t^{(\ell)} \phi(h_t^{(\ell-1)}) \\
&= \Delta Z^{(\ell)} h_t^{(\ell-1)} + L^{-\alpha} W_{t+1}^{(\ell)} \phi(h_t^{(\ell-1)}) - L^{-\alpha} W_t^{(\ell)} \phi(h_t^{(\ell-1)}) \\
&= \Delta Z^{(\ell)} h_t^{(\ell-1)} + L^{-\alpha} (W_t^{(\ell)} + \Delta W^{(\ell)}) \phi(h_t^{(\ell-1)}) - L^{-\alpha} W_t^{(\ell)} \phi(h_t^{(\ell-1)}) \\
&= \Delta Z^{(\ell)} h_t^{(\ell-1)} + L^{-\alpha} \Delta W^{(\ell)} \phi(h_t^{(\ell-1)}).
\end{aligned}$$

By using the expressions of $\Delta W^{(\ell)}$ and $\Delta Z^{(\ell)}$ from Equation (39) and Equation (40) combined with the expressions of the associated LMOs in Equation (34) and Equation (37), we have

$$\begin{aligned}
\|\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)}\|_2^2 &= \Theta\left( \|\Delta Z^{(\ell)} h_t^{(\ell-1)}\|_2^2 + \|L^{-\alpha} \Delta W^{(\ell)} \phi(h_t^{(\ell-1)})\|_2^2 \right) \\
&= \Theta\left( \left\| \sqrt{\frac{d_\ell}{d_{\ell-1}}} \gamma_Z \frac{\frac{\partial \mathcal{L}}{\partial h^{(\ell)}}}{\|\frac{\partial \mathcal{L}}{\partial h^{(\ell)}}\|_2} \frac{(h_t^{(\ell-1)})^\top}{\|(h_t^{(\ell-1)})\|_2^2} (h_t^{(\ell-1)}) \right\|_2^2 \right. \\
&\quad \left. + L^{-2\alpha} \left\| \sqrt{\frac{d_\ell}{d_{\ell-1}}} \gamma \frac{\frac{\partial \mathcal{L}}{\partial h^{(\ell)}}}{\|\frac{\partial \mathcal{L}}{\partial h^{(\ell)}}\|_2} \frac{\phi(h_t^{(\ell-1)})^\top}{\|\phi(h_t^{(\ell-1)})\|_2^2} \phi(h_t^{(\ell-1)}) \right\|_2^2 \right) \\
&= \Theta\left( \frac{d_\ell}{d_{\ell-1}} \gamma_Z^2 \|h_t^{(\ell-1)}\|_2^2 + L^{-2\alpha} \frac{d_\ell}{d_{\ell-1}} \gamma^2 \|\phi(h_t^{(\ell-1)})\|_2^2 \right).
\end{aligned}$$

Using the scale of the activation function assumption (Equation (14)), and the Desideratum 1 (Equation (38)), we have

$$\begin{aligned}
\|\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)}\|_2^2 &= \Theta\left( \frac{d_\ell}{d_{\ell-1}} \gamma_Z^2 \|h_t^{(\ell-1)}\|_2^2 + L^{-2\alpha} \frac{d_\ell}{d_{\ell-1}} \gamma^2 \|h_t^{(\ell-1)}\|_2^2 \right) \\
&= \Theta\left( d_\ell \gamma_Z^2 + L^{-2\alpha} d_\ell \gamma^2 \right) \\
&= \Theta\left( d_\ell (\gamma_Z^2 + L^{-2\alpha} \gamma^2) \right).
\end{aligned}$$

So, $\frac{1}{d_\ell} \|\Delta_{W^{(\ell)},Z^{(\ell)}} h^{(\ell)}\|_2^2 = \Theta(\gamma_Z^2 + L^{-2\alpha} \gamma^2)$.

Therefore, if we want the entries of $\Delta h^{(\ell)}$ to be in $\Theta(1/L)$, we need to have

$$\gamma = \gamma_{\text{base}} L^{\alpha-1}, \text{ and } \gamma_Z = \gamma_{Z\,base} L^{-1}. \tag{41}$$

Now, let us consider the embedding layer $h^{(0)} = W^{(0)} x$.

Using the update expression $\Delta W^{(0)}$, from Equation (39), with the expression of the associated lmo in Equation (35), we have

$$\|\Delta W^{(0)}x\|_2^2 = \Theta\left(\gamma^2 \frac{d_0}{d_{\text{in}}}\|x\|_2^2\right) = \Theta\left(\gamma^2 d_0\right) = \gamma^2\Theta(d_0).$$

Therefore, we do not need to scale the learning rate of the first weight $W^{(0)}$ in order to respect Desideratum 2, and

$$\gamma_{W^{(0)}} = \gamma_{W^{(0)}\text{base}}. \tag{42}$$

Finally, for the output layer $f = W^{(L+1)}\phi(h^{(L)})$, using the update expression $\Delta W^{(L+1)}$ from Equation (39) with the associated lmo expressions in Equation 36, and Desideratum 1 (Equation (38)) applied to $h^{(L)}$, we have

$$\begin{aligned}
\|\Delta W^{(L+1)}h^{(L)}\|_2^2 &= \Theta\left(\gamma^2 \frac{d_{\text{out}}}{d_L}\frac{\|\frac{\partial \mathcal{L}}{\partial f}\|_2^2}{\|\frac{\partial \mathcal{L}}{\partial f}\|_2^2}\frac{\|\phi(h^{(L)})\|_2^2}{\|\phi(h^{(L)})\|_2^2}\|h^{(L)}\|_2^2\right)\\
&= \Theta\left(\gamma^2 \frac{d_{\text{out}}}{d_L}\|h^{(L)}\|_2^2\right)\\
&= \gamma^2\Theta\left(d_{\text{out}}\right).
\end{aligned}$$

Therefore, we do not need to scale the learning rate of the last weight $W^{(L+1)}$ in order to respect Desideratum 2.

$$\gamma_{W^{(L+1)}} = \gamma_{W^{(L+1)}\text{base}}. \tag{43}$$

□

#### C.4.2 Scion case

*Proof.* Let $\ell \in \{1, \dots, L\}$. For Scion, we use the expressions of the updates $\Delta W^{(\ell)}$ and $\Delta Z^{(\ell)}$ as in Equation (32), that are written as

$$\Delta W^{(\ell)} = -\gamma W^{(\ell)} + \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}}\mathcal{L})\phi(h_t^{(\ell-1)}) \tag{44}$$

$$\Delta Z^{(\ell)} = -\gamma_Z Z^{(\ell)} + \gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}}\mathcal{L})h_t^{(\ell-1)}. \tag{45}$$

Then,

$$\begin{aligned}
\|\Delta_{W^{(\ell)},Z^{(\ell)}}h^{(\ell)}\|_2^2 &= \|\Delta Z^{(\ell)}h_t^{(\ell-1)} + L^{-\alpha}\Delta W^{(\ell)}\phi(h_t^{(\ell-1)})\|_2^2\\
&= \Theta\Big(\|(-\gamma_Z Z^{(\ell)} + \gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}}\mathcal{L}))h_t^{(\ell-1)}\|_2^2\\
&\quad + \|L^{-\alpha}(-\gamma W^{(\ell)} + \gamma \operatorname{lmo}(\nabla_{W^{(\ell)}}\mathcal{L}))\phi(h_t^{(\ell-1)})\|_2^2\Big)\\
&= \Theta\Big(\|\gamma_Z Z^{(\ell)}h_t^{(\ell-1)}\|_2^2 + \|\gamma_Z \operatorname{lmo}(\nabla_{Z^{(\ell)}}\mathcal{L})h_t^{(\ell-1)}\|_2^2\\
&\quad + \|L^{-\alpha}\gamma W^{(\ell)}\phi(h_t^{(\ell-1)})\|_2^2 + \|L^{-\alpha}\gamma \operatorname{lmo}(\nabla_{W^{(\ell)}}\mathcal{L})\phi(h_t^{(\ell-1)})\|_2^2\Big).
\end{aligned}$$

We saw in the unconstrained case that the second and fourth terms are in $\Theta(d_\ell(\gamma_Z^2 + L^{-2\alpha}\gamma^2))$.

For the third term, using the alignment of the weights (Equation (13)) and the scale of the activation function $\phi$ (Equation (14)) assumptions, we have

$$\begin{aligned}
\|L^{-\alpha}\gamma W^{(\ell)}\phi(h_t^{(\ell-1)})\|_2^2 &= \Theta\left(L^{-2\alpha}\gamma^2\|W^{(\ell)}\|_2^2\|\phi(h_t^{(\ell-1)})\|_2^2\right)\\
&= \Theta\left(L^{-2\alpha}\gamma^2\|W^{(\ell)}\|_2^2\|h_t^{(\ell-1)}\|_2^2\right).
\end{aligned}$$

Then, using the initialization of the weights in Equation (12) and the Desideratum 1 in Equation (38), we have

$$\begin{aligned}\|L^{-\alpha}\gamma W^{(\ell)}\phi(h_t^{(\ell-1)})\|_2^2 &= \Theta\left(L^{-2\alpha}\gamma^2\frac{d_\ell}{d_{\ell-1}}d_{\ell-1}\right)\\ &= \Theta\left(L^{-2\alpha}\gamma^2 d_\ell\right).\end{aligned}$$

Similarly, for the first term, we have

$$\begin{aligned}\|\gamma_Z Z^{(\ell)}h_t^{(\ell-1)}\|_2^2 &= \Theta\left(\gamma_Z^2\|Z^{(\ell)}\|_2^2\|h_t^{(\ell-1)}\|_2^2\right)\\ &= \Theta\left(\gamma_Z^2\frac{d_\ell}{d_{\ell-1}}d_{\ell-1}\right)\\ &= \Theta\left(\gamma_Z^2 d_\ell\right).\end{aligned}$$

Finally, we have

$$\|\Delta_{W^{(\ell)},Z^{(\ell)}}h^{(\ell)}\|_2^2 = \Theta\left(d_\ell(\gamma_Z^2 + L^{-2\alpha}\gamma^2)\right).$$

We can conclude in the same way as in the unconstrained case.

Now, let us consider the embedding layer $h^{(0)} = W^{(0)}x$. With Scion, using the update expression of $W^{(0)}$ in Equation (44), combined with the associated lmo expression in Equation (35), we have

$$\begin{aligned}\Delta W^{(0)} &= -\gamma W^{(0)} + \gamma\mathrm{lmo}(\nabla_{W^{(0)}}\mathcal{L})\\ &= -\gamma W^{(0)} + \gamma\sqrt{\frac{d_0}{d_{\text{in}}}}\frac{\frac{\partial\mathcal{L}}{\partial h^{(0)}}}{\|\frac{\partial\mathcal{L}}{\partial h^{(0)}}\|_2}\frac{x^T}{\|x\|_2}.\end{aligned}$$

Then, using the alignment of the weights (Equation (13)) and the initialization of $W^{(0)}$ (Equation (12)) assumptions, we have

$$\begin{aligned}\|\Delta W^{(0)}x\|_2^2 &= \Theta\left(\gamma^2\|W^{(0)}x\|_2^2 + \gamma^2\frac{d_0}{d_{\text{in}}}\|x\|_2^2\right)\\ &= \Theta\left(\gamma^2\|W^{(0)}\|_2^2\|x\|_2^2 + \gamma^2\frac{d_0}{d_{\text{in}}}\|x\|_2^2\right) \quad \text{(Claim 1 in Yang et al. (2023a))}\\ &= \Theta\left(\gamma^2\frac{d_0}{d_{\text{in}}}d_{\text{in}} + \gamma^2\frac{d_0}{d_{\text{in}}}d_{\text{in}}\right)\\ &= \Theta\left(2\gamma^2 d_0\right)\\ &= \gamma^2\Theta\left(d_0\right).\end{aligned}$$

We can conclude similarly to the unconstrained case.

Finally, for the output layer $f = W^{(L+1)}\phi(h^{(L)})$, if we use the update expression of $W^{(L+1)}$ in Equation (44), combined with the associated lmo expression in Equation 36,we have

$$\Delta W^{(L+1)} = -\gamma W^{(L+1)} + \gamma\sqrt{\frac{d_{\text{out}}}{d_L}}\frac{\frac{\partial\mathcal{L}}{\partial f}}{\|\frac{\partial\mathcal{L}}{\partial f}\|_2}\frac{\phi(h^{(L)})^T}{\|\phi(h^{(L)})\|_2}.$$

Then, using the alignment of the weights (Equation (13)), the initialization of $W^{(L+1)}$ (Equation (12)) and the Desideratum 1 (Equation (38)) applied to $h^{(L)}$, we have

$$\begin{aligned}\|\Delta W^{(L+1)}h^{(L)}\|_2^2 &= \Theta\left(\gamma^2\|W^{(L+1)}h^{(L)}\|_2^2 + \gamma^2\|\mathrm{lmo}(\nabla_{W^{(L+1)}}\mathcal{L})h^{(L)}\|_2^2\right) \quad \text{(from Equation (30))}\\ &= \Theta\left(\gamma^2\|W^{(L+1)}\|_2^2\|h^{(L)}\|_2^2 + \gamma^2 d_{\text{out}}\right)\\ &= \Theta\left(\gamma^2\frac{d_{\text{out}}}{d_L}d_L + \gamma^2 d_{\text{out}}\right)\\ &= \Theta\left(\gamma^2 d_{\text{out}}\right).\end{aligned}$$

And we can conclude the same way as in the unconstrained case.

□

# D Proofs of the Width Transfer from Section 4

We recall Desideratum 3 and demonstrate that it holds for Scion, extending the prior width-transfer results derived by Pethick et al. (2025a) for Unconstrained Scion.

**Desideratum 3** (Feature Learning)**.** *The hidden layers and their updates should be stable with the width:*

$$\|h^{(\ell)}\|_2^2 = \Theta(d_\ell) \text{ and } \|\Delta h^{(\ell)}\|_2^2 = \Theta(d_\ell) \quad \forall \ell = 1, \dots, L. \tag{27}$$

## D.1 Proof of the width transfer with Scion, from Section 4.2

**Proposition 3** (Scion and Unconstrained Scion verify Desideratum 3)**.** *Under the assumptions defined in the subsection 3.4.2, the updates of the features after a step of* Scion *or* Unconstrained Scion*, remain stable when the width $d_\ell$ become increasingly large.*

*Proof.* Let $\ell \in \{1, \dots, L\}$. With Scion, the update of the weight matrix $W^{(\ell)}$ can be written as:

$$\Delta W^{(\ell)} = -\gamma W^{(\ell)} + \gamma \operatorname{lmo}(\nabla_{W_t^{(\ell)}} \mathcal{L}).$$

If we decompose the expression of the $i$-th entry of hidden layer $h^{(\ell)}$, the update of $h^{(\ell)}$ between the iteration $t$ and $t+1$ can be written as

$$\begin{aligned}
\Delta h_i^{(\ell)} &= (h_{t+1}^{(\ell)})_i - (h_t^{(\ell)})_i \\
&= (W_{t+1}^{(\ell)} \phi(h_{t+1}^{(\ell-1)}))_i - (W_t^{(\ell)} \phi(h_t^{(\ell-1)}))_i \\
&= (W_t^{(\ell)} \phi(h_{t+1}^{(\ell-1)}))_i + (\Delta W^{(\ell)} \phi(h_{t+1}^{(\ell-1)}))_i - (W_t^{(\ell)} \phi(h_t^{(\ell-1)}))_i \\
&= (W_t^{(\ell)} \phi(h_t^{(\ell-1)}))_i + (W_t^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i + (\Delta W^{(\ell)} \phi(h_t^{(\ell-1)}))_i \\
&\quad + (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i - (W_t^{(\ell)} \phi(h_t^{(\ell-1)}))_i \\
&= (W_t^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i + (\Delta W^{(\ell)} \phi(h_t^{(\ell-1)}))_i + (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i.
\end{aligned}$$

By rearranging the terms and removing the iteration $t$ for more concision, and by incorporating the expression of $\Delta W^{(\ell)}$, with the lmo expression of Equation (34), we have

$$\begin{aligned}
\Delta h_i^{(\ell)} &= (\Delta W^{(\ell)} \phi(h^{(\ell-1)}))_i + (W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i + (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i \\
&= -(\gamma W^{(\ell)} \phi(h^{(\ell-1)}))_i + \gamma \sum_{j=1}^{d_{l-1}} [\operatorname{lmo}(\nabla_{W^{(\ell)}} \mathcal{L})]_{i,j} h_j^{(\ell-1)} \\
&\quad + (W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i + (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i \\
&= -(\gamma W^{(\ell)} \phi(h^{(\ell-1)}))_i + \gamma \sum_{j=1}^{d_{l-1}} \sqrt{\frac{d_\ell}{d_{\ell-1}}} \frac{(\frac{d\mathcal{L}}{dh^{(\ell)}})_i}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2} \frac{\phi(h_j^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2} h_j^{(\ell-1)} \\
&\quad + (W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i + (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i \\
&= -(\gamma W^{(\ell)} \phi(h^{(\ell-1)}))_i + \gamma \sqrt{\frac{d_\ell}{d_{\ell-1}}} \frac{(\frac{d\mathcal{L}}{dh^{(\ell)}})_i}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2} \|\phi(h^{(\ell-1)})\|_2 \\
&\quad + (W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i + (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i.
\end{aligned}$$

We can compute the norm of the hidden state vector $h^{(\ell)}$

$$\begin{aligned}
\|\Delta h^{(\ell)}\|_2^2 &= \sum_{i=1}^{d_l} (\Delta h_i^{(\ell)})^2 \\
&= \Theta\left(\sum_{i=1}^{d_l} ((W_t^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i)^2 + ((\Delta W^{(\ell)} \phi(h^{(\ell-1)})_t)_i)^2 \right. \\
&\qquad \left. + ((\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i)^2\right) \\
&= \Theta\left(\gamma^2 \sum_{i=1}^{d_l} (W^{(\ell)} \phi(h^{(\ell-1)}))_i^2 + \gamma^2 \frac{d_\ell}{d_{\ell-1}} \sum_{i=1}^{d_l} \frac{\left(\frac{d\mathcal{L}}{dh^{(\ell)}}\right)_i^2}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2^2} \|\phi(h^{(\ell-1)})\|_2^2 \right. \\
&\qquad \left. + \sum_{i=1}^{d_l} (W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i^2 + \sum_{i=1}^{d_l} (\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}))_i^2 \right).
\end{aligned}$$

Then,

$$\begin{aligned}
\|\Delta h^{(\ell)}\|_2^2 = \Theta\left(\gamma^2 \|W^{(\ell)} \phi(h^{(\ell-1)})\|_2^2 + \gamma^2 \frac{d_\ell}{d_{\ell-1}} \|\phi(h^{(\ell-1)})\|_2^2 \right. \\
\left. + \|W^{(\ell)} \Delta\phi(h^{(\ell-1)})\|_2^2 + \|\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)})\|_2^2\right).
\end{aligned} \tag{46}$$

By using the alignment of the weights (Equation (13), the initialization of the weights $W^{(\ell)}$ (Equation (12)), and the scale of the activation function $\phi$ assumption in Equation (14), we have

$$\|h^{(\ell)}\|_2 = \Theta(\|W^{(\ell)}\|_{\mathcal{S}_\infty} \|h^{(\ell-1)}\|_2) = \Theta(\sqrt{d_\ell}).$$

With the scale of the activation function $\phi$ assumption in Equation (14), we also have:

$$\|\phi(h^{(\ell)})\|_2^2 = \Theta(d_\ell). \tag{47}$$

Therefore, by using the weight alignment assumption in Equation (13) and the weight initialization as in Equation (12), we have

$$\begin{aligned}
\|W^{(\ell)} \phi(h^{(\ell-1)})\|_2^2 &= \Theta(\|W^{(\ell)}\|_{S_\infty}^2 \|\phi(h^{(\ell-1)})\|_2^2) \\
&= \Theta\left(\frac{d_\ell}{d_{\ell-1}} \|h^{(\ell-1)}\|_2^2\right) \\
&= \Theta\left(\frac{d_\ell}{d_{\ell-1}} d_{\ell-1}\right) \\
&= \Theta\left(d_\ell\right).
\end{aligned} \tag{48}$$

Likewise,

$$\begin{aligned}
\|W^{(\ell)} \Delta\phi(h^{(\ell-1)})\|_2^2 &= \Theta\left(\frac{d_\ell}{d_{\ell-1}} \|\Delta\phi(h^{(\ell-1)})\|_2^2\right) \\
&= \mathcal{O}\left(\frac{d_\ell}{d_{\ell-1}} \|\Delta h^{(\ell-1)}\|_2^2\right) \\
&\qquad \text{(Lipschitz activation function).}
\end{aligned} \tag{49}$$

To write the term, $\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)})$, we can decompose the weight update $\Delta W^{(\ell)}$ and use the lmo expression of Equation (34), to get

$$\begin{aligned}
\Delta W^{(\ell)} \Delta\phi(h^{(\ell-1)}) = \gamma \sqrt{\frac{d_\ell}{d_{\ell-1}}} \frac{\left(\frac{d\mathcal{L}}{dh^{(\ell)}}\right)}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2} \frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2} \Delta\phi(h^{(\ell-1)}) \\
- \gamma W^{(\ell)} \Delta\phi(h^{(\ell-1)}).
\end{aligned}$$

So,

$$\begin{aligned}\|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 &= \Theta\left(\gamma^2\frac{d_\ell}{d_{\ell-1}}\left\|\frac{\left(\frac{d\mathcal{L}}{dh^{(\ell)}}\right)}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2}\frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2}\Delta\phi(h^{(\ell-1)})\right\|_2^2\right.\\ &\quad\left.+\gamma^2\|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2\right)\\ &= \Theta\left(\gamma^2\frac{d_\ell}{d_{\ell-1}}\left\|\frac{\left(\frac{d\mathcal{L}}{dh^{(\ell)}}\right)}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2}\frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2}\Delta\phi(h^{(\ell-1)})\right\|_2^2\right.\\ &\quad\left.+\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta\phi(h^{(\ell-1)})\|_2^2\right).\end{aligned}$$

By using the fact that the activation function $\phi$ is Lipschitz, we have

$$\begin{aligned}\|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 &= \mathcal{O}\left(\gamma^2\frac{d_\ell}{d_{\ell-1}}\left\|\frac{\left(\frac{d\mathcal{L}}{dh^{(\ell)}}\right)}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2}\frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2}\Delta\phi(h^{(\ell-1)})\right\|_2^2\right.\\ &\quad\left.+\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right).\end{aligned}$$

We rewrite the first term and use the Lipschitz continuity of the activation function $\phi$, to get

$$\begin{aligned}\gamma^2\frac{d_\ell}{d_{\ell-1}}\left\|\frac{\left(\frac{d\mathcal{L}}{dh^{(\ell)}}\right)}{\|\frac{d\mathcal{L}}{dh^{(\ell)}}\|_2}\frac{\phi(h^{(\ell-1)})^\top}{\|\phi(h^{(\ell-1)})\|_2}\Delta\phi(h^{(\ell-1)})\right\|_2^2 &= \Theta\left(\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta\phi(h^{(\ell-1)})\|_2^2\right)\\ &= \mathcal{O}\left(\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right).\end{aligned}$$

Therefore,

$$\begin{aligned}\|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 &= \mathcal{O}\left(\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2+\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right)\\ &= \mathcal{O}\left(2\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right).\end{aligned}$$

Finally, we can incorporate all these derivated terms into the expression of $\|\Delta h^{(\ell)}\|_2^2$ from Equation (46), and we have

$$\begin{aligned}\|\Delta h^{(\ell)}\|_2^2 &= \Theta\left(\gamma^2\|W^{(\ell)}\phi(h^{(\ell-1)})\|_2^2+\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\phi(h^{(\ell-1)})\|_2^2\right.\\ &\quad\left.+\|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2+\|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2\right)\\ &= \mathcal{O}\left(\gamma^2 d_\ell+\gamma^2 d_\ell+\|\Delta h^{(\ell-1)}\|_2^2\right.\\ &\quad\left.+2\gamma^2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right).\end{aligned}$$

Thus,

$$\|\Delta h^{(\ell)}\|_2^2 = \mathcal{O}\left(2\gamma^2 d_\ell+(1+2\gamma^2)\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right). \tag{50}$$

By induction, we can show that

$$\begin{aligned}\|\Delta h^{(\ell)}\|_2^2 &= \mathcal{O}\left(d_\ell \times (2\gamma^2)(1+\sum_{j=1}^{\ell-1}(1+2\gamma^2)^j)\right)\\ &= \mathcal{O}\left(d_\ell \times (2\gamma^2)(\sum_{j=0}^{\ell-1}(1+2\gamma^2)^j)\right).\end{aligned}$$

We provide here the details of the recursion:

• Initialization: For $\ell = 1$, we have:

$$\begin{aligned}\|\Delta h^{(1)}\|_2^2 &= \mathcal{O}\left(2\gamma^2 d_1 + (1+2\gamma^2)\frac{d_1}{d_{\text{in}}}\|\Delta x\|_2^2\right)\\ &= \mathcal{O}\left(2\gamma^2 d_1\right).\end{aligned}$$

• Heredity: Let us suppose that $\|\Delta h^{(\ell)}\|_2^2 = \mathcal{O}\left(d_\ell \times (2\gamma^2)(1+\sum_{j=1}^{\ell-1}(1+2\gamma^2)^j)\right)$.

Then:

$$\begin{aligned}\|\Delta h^{(\ell+1)}\|_2^2 &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} + (1+2\gamma^2)\frac{d_{\ell+1}}{d_\ell}\|\Delta h^{(\ell)}\|_2^2\right) \quad \text{(from Equation (50))}\\ &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} + (1+2\gamma^2)\frac{d_{\ell+1}}{d_\ell} \times d_\ell \times (2\gamma^2)(1+\sum_{j=1}^{\ell-1}(1+2\gamma^2)^j)\right)\\ &\quad \text{(heredity hypothesis)}\\ &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} + (1+2\gamma^2)d_{\ell+1} \times (2\gamma^2)(1+\sum_{j=1}^{\ell-1}(1+2\gamma^2)^j)\right)\\ &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} \times \left(1 + (1+2\gamma^2) \times (1+\sum_{j=1}^{\ell-1}(1+2\gamma^2)^j)\right)\right)\\ &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} \times \left(1 + (1+2\gamma^2) \times (\sum_{j=0}^{\ell-1}(1+2\gamma^2)^j)\right)\right)\\ &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} \times \left(1 + (\sum_{j=0}^{\ell-1}(1+2\gamma^2)^{j+1})\right)\right)\\ &= \mathcal{O}\left(2\gamma^2 d_{\ell+1} \times (1+\sum_{j=1}^{\ell}(1+2\gamma^2)^j)\right),\end{aligned}$$

which completes the proof by induction.

Also, since $\|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 + \|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 \geq 0$, we have

$$\|\Delta h^{(\ell)}\|_2^2 = \Omega\left(2\gamma^2 d_\ell\right).$$

Then, we have, by symmetry

$$\Delta h_i^{(\ell)^2} = \frac{1}{d_\ell}\|\Delta h^{(\ell)}\|_2^2 = \mathcal{O}\left(2\gamma^2 \sum_{j=0}^{\ell-1}(1+2\gamma^2)^j\right).$$

Thus,

$$\Delta h_i^{(\ell)^2} = \frac{1}{d_\ell}\|\Delta h^{(\ell)}\|_2^2 = \Omega\left(2\gamma^2\right).$$

Therefore, we have the maximal update condition since $\Delta h_i^{(\ell)^2} = \Theta(1)$ with respect to the width of the layer $d_\ell$. Also, the optimal learning rate satisfying this condition is independent of the hidden-layer widths.

□

### D.2 Proof from Section 4.3.1

**Proposition 4** (Width invariance of the maximal update learning rate for Projected Scion)**.** *Under the assumptions defined in the subsection 3.4.2, the updates of the features after a step of* Projected Scion*, are* $\Theta(1)$ *with respect to the width* $d_\ell$*.*

*Proof.* With the projected version of Scion, the update can be written as

$$W_{t+1}^{(\ell)} = \text{spectral_clipping}(W_t^{(\ell)} + \gamma\,\text{lmo}[\nabla_{W_t^{(\ell)}}\mathcal{L}]). \tag{51}$$

Let us consider the case where we clip the singular values in the interval $[\sigma, \sigma]$. In this case, the spectral clipping function can be written as the product of $\sigma$ and the matrix sign function (Cesista, 2025):

$$\text{spectral_clipping}_{[\sigma,\sigma]}(A) = \sigma\,\text{orthog}(A). \tag{52}$$

Thus, we have

$$\|\text{orthog}(A)\|_{\mathcal{S}_\infty} = 1. \tag{53}$$

Then,

$$\begin{aligned}
\|\Delta W\|_{\mathcal{S}_\infty}^2 &= \|W_{t+1} - W_t\|_{\mathcal{S}_\infty}^2 \\
&= \|\text{spectral_clipping}\left(W_t + \gamma\,\text{lmo}(\nabla_{W_t}\mathcal{L})\right) - W_t\|_{\mathcal{S}_\infty}^2 \\
&= \Theta\left(\|\text{spectral_clipping}\left(W_t + \gamma\,\text{lmo}(\nabla_{W_t}\mathcal{L})\right)\|_{\mathcal{S}_\infty}^2 + \|W_t\|_{\mathcal{S}_\infty}^2\right) \\
&= \Theta\left(\|\sigma\,\text{orthog}\left(W_t + \gamma\,\text{lmo}(\nabla_{W_t}\mathcal{L})\right)\|_{\mathcal{S}_\infty}^2 + \|W_t\|_{\mathcal{S}_\infty}^2\right) \\
&= \Theta\left(\sigma^2 + \|W_t\|_{\mathcal{S}_\infty}^2\right) \\
&= \Theta\left(\|W_t\|_{\mathcal{S}_\infty}^2\right) \\
&\quad (\sigma \text{ is a constant with respect to the width}) \\
&= \Theta\left(\frac{d_\ell}{d_{\ell-1}}\right).
\end{aligned} \tag{54}$$

As in the derivations with the Scion update from Appendix D.1, we want to evaluate

$$\|\Delta h^{(\ell)}\|_2^2 = \Theta\left(\|\Delta W^{(\ell)}\phi(h^{(\ell-1)})\|_2^2 + \|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 + \|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2\right). \tag{55}$$

For the first term, if we write the update $\Delta W^{(\ell)}$ and reuse the fact that $\|\phi(h^{(\ell)})\|_2^2 = \Theta(d_\ell)$ as in Equation (47), and $\|W_t^{(\ell)}\phi(h^{(\ell-1)})\|_2^2 = \Theta(d_\ell)$ from Equation (48), we have

$$\begin{aligned}
\left\|\Delta W^{(\ell)}\phi(h^{(\ell-1)})\right\|_2^2 &= \left\|\sigma\,\mathrm{orthog}\left(W_t^{(\ell)} + \gamma\,\mathrm{lmo}(\nabla_{W_t^{(\ell)}}\mathcal{L})\right)\phi(h^{(\ell-1)}) - W_t^{(\ell)}\phi(h^{(\ell-1)})\right\|_2^2 \\
&= \Theta\left(\left\|\sigma\,\mathrm{orthog}\left(W_t^{(\ell)} + \gamma\,\mathrm{lmo}(\nabla_{W_t^{(\ell)}}\mathcal{L})\right)\phi(h^{(\ell-1)})\right\|_2^2 + \left\|W_t^{(\ell)}\phi(h^{(\ell-1)})\right\|_2^2\right) \\
&= \Theta\left(\sigma^2\left\|\phi(h^{(\ell-1)})\right\|_2^2 + \left\|W_t^{(\ell)}\phi(h^{(\ell-1)})\right\|_2^2\right) \quad \text{(orthogonal matrix)} \\
&= \Theta\left(\sigma^2 d_{\ell-1} + d_\ell\right).
\end{aligned} \tag{56}$$

For the second term, as in the Equation (49) of Appendix D.1, we can use the alignment of the weights assumption (Equation (13)), and we have

$$\begin{aligned}
\|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 &= \Theta\left(\frac{d_\ell}{d_{\ell-1}}\|\Delta\phi(h^{(\ell-1)})\|_2^2\right) \\
&= \mathcal{O}\left(\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right) \\
&\qquad \text{(Lipschitz activation function).}
\end{aligned} \tag{57}$$

For the third term, we can decompose the update $\Delta W^{(\ell)}$, to get

$$\begin{aligned}
\left\|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\right\|_2^2 &= \left\|\sigma\,\mathrm{orthog}\left(W_t^{(\ell)} + \gamma\,\mathrm{lmo}(\nabla_{W_t^{(\ell)}}\mathcal{L})\right)\Delta\phi(h^{(\ell-1)}) - W_t^{(\ell)}\Delta\phi(h^{(\ell-1)})\right\|_2^2 \\
&= \Theta\left(\left\|\sigma\,\mathrm{orthog}\left(W_t^{(\ell)} + \gamma\,\mathrm{lmo}(\nabla_{W_t^{(\ell)}}\mathcal{L})\right)\Delta\phi(h^{(\ell-1)})\right\|_2^2 + \left\|W_t^{(\ell)}\Delta\phi(h^{(\ell-1)})\right\|_2^2\right) \\
&= \Theta\left(\sigma^2\left\|\Delta\phi(h^{(\ell-1)})\right\|_2^2 + \left\|W_t^{(\ell)}\Delta\phi(h^{(\ell-1)})\right\|_2^2\right) \\
&\quad \text{(orthogonal).}
\end{aligned} \tag{58}$$

Then, by using the Lipschitz continuity of $\phi$ and re-using the expression from Equation (57), we have

$$\left\|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\right\|_2^2 = \mathcal{O}\left(\sigma^2 d_{\ell-1} + \frac{d_\ell}{d_{\ell-1}}\left\|\Delta h^{(\ell-1)}\right\|_2^2\right). \tag{59}$$

Finally, by incorporating these terms into the expression of $\|\Delta h^{(\ell)}\|_2^2$ in Equation (55), we have

$$\begin{aligned}
\|\Delta h^{(\ell)}\|_2^2 &= \Theta\Big(\|\Delta W^{(\ell)}\phi(h^{(\ell-1)})\|_2^2 + \|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 \\
&\qquad + \|\Delta W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2\Big) \\
&= \mathcal{O}\left(d_\ell + 2\sigma^2 d_{\ell-1} + 2\frac{d_\ell}{d_{\ell-1}}\|\Delta h^{(\ell-1)}\|_2^2\right).
\end{aligned} \tag{60}$$

Also, since $\|W^{(\ell)}\Delta\phi(h^{(\ell-1)})\|_2^2 > 0$, we have that

$$\|\Delta h^{(\ell)}\|_2^2 = \Omega(d_\ell + 2\sigma^2 d_{\ell-1}) = \Omega(d_\ell + d_{\ell-1}) \quad \forall \ell = 1, \ldots, L. \tag{61}$$

Thus, if we proceed by induction and consider the width $d_\ell$ and $d_{\ell-1}$ to be of the same order and going to infinity, we have

$$\|\Delta h^{(\ell)}\|_2^2 = \Theta(d_\ell) \quad (\forall \ell = 1, \ldots, L), \tag{62}$$

and

$$\Delta h_i^{(\ell)^2} = \Theta(1) \quad (\forall \ell = 1, \ldots, L). \tag{63}$$

□

## E CompleteP style analysis

In this part, we analyze the scaling behavior of the network's hidden states using the heuristics found in Dey et al. (2026). Hence, the $Z^{(\ell)}$ matrices are not learned and are fixed as identity. The hidden-layer width is constant, and training is restricted to the $W^{(\ell)}$ operators.

**Proposition 5.** *For all iteration $t > 0$, under the CompleteP correlation assumptions, the normalized squared $\ell_2$-norm change of the final layer hidden state satisfies:*

$$\frac{1}{d_L}\|\Delta h_t^{(L)}\|_2^2 = \Theta(1).$$

*Proof.* Let $t > 0$ and write the definition of $h^{(L)}(t)$ to get

$$h^{(L)}(t) = h_t^{(0)} + L^{-\alpha}\sum_{\ell=0}^{L-1} W_t^{(\ell)}\phi(h_t^{(\ell)}).$$

Inserting $W_t^{(\ell)} = W_0^{(\ell)} + \sum_{t<t'} \Delta W_{t'}^{(\ell)}$ (unrolling the recursion) then gives

$$h_t^{(L)} = h_t^{(0)} + L^{-\alpha}\sum_{\ell=0}^{L-1} W_0^{(\ell)}\phi(h_t^{(\ell)}) + \underbrace{L^{-\alpha}\sum_{\ell=0}^{L-1}\sum_{t'<t} \Delta W_{t'}^{(\ell)}\phi(h_t^{(\ell)})}_{\Delta h_t^{(L)}}.$$

In CompleteP Desideratum 2 (Dey et al., 2026), they aim to show that $\|\Delta h_t^{(L)}\|_{RMS}^2 = \Theta(1)$ for all $t > 0$. Using Unconstrained Scion, the update has the form

$$\Delta W_{t'}^{(\ell)} = W_{t'+1}^{(\ell)} - W_{t'}^{(\ell)} = \gamma \frac{g_{t'}^{(\ell+1)}}{\|g_{t'}^{(\ell+1)}\|_2}\left[\frac{\phi(h_{t'}^{(\ell)})}{\|\phi(h_{t'}^{(\ell)})\|_2}\right]^\top.$$

Plugging this into the expression for $\Delta h_t^{(L)}$, we find

$$\Delta h_t^{(L)} = L^{-\alpha}\sum_{\ell=0}^{L-1}\sum_{t'<t} \gamma \frac{g_{t'}^{(\ell+1)}}{\|g_{t'}^{(\ell+1)}\|_2}\left[\frac{\phi(h_{t'}^{(\ell)})}{\|\phi(h_{t'}^{(\ell)})\|_2}\right]^\top \phi(h_t^{(\ell)}).$$

Note that, for each $\ell$, CompleteP claims that the coordinates of $\phi(h_t^{(\ell)})$ and $\phi(h_{t'}^{(\ell)})$ have $\Theta(1)$ correlation, so that $\phi(h_{t'}^{(\ell)})^\top\phi(h_t^{(\ell)}) = \Theta(d_\ell)$, while $\|\phi(h^\ell(t'))\|_2 = \Theta(\sqrt{d_\ell})$. We also make this correlation assumption here, and, by taking the norm squared and applying this yields

$$\begin{aligned}
\|\Delta h_t^{(L)}\|_2^2 &= L^{-2\alpha}\left\|\sum_{\ell=0}^{L-1}\sum_{t'<t} \gamma \frac{g_{t'}^{(\ell+1)}}{\|g_{t'}^{(\ell+1)}\|_2}\left[\frac{\phi(h_{t'}^{(\ell)})}{\|\phi(h_{t'}^{(\ell)})\|_2}\right]^\top \phi(h_t^{(\ell)})\right\|_2^2 \\
&= L^{-2\alpha}\gamma^2 d_L \left\|\sum_{\ell=0}^{L-1}\sum_{t'<t} \frac{g_{t'}^{(\ell+1)}}{\|g_{t'}^{(\ell+1)}\|_2}\right\|_2^2 \\
&= L^{-2\alpha}\gamma^2 d_L \Theta(L^2)
\end{aligned}$$

where the last equality follows because the layer gradients are strongly correlated across layers but uncorrelated across time steps.

For a sign update with the CompleteP scaling (Dey et al., 2026), we would get $1/L^2$ for the RMS norm squared of the update.

□

## F Coordinate Checks on nanoGPT

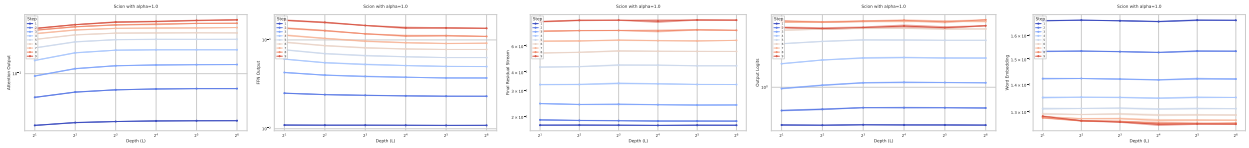


Figure 17: Coordinate check for Scion, on FineWeb, with $\alpha = 1.0$, and learning rate $\gamma = 2^{-11}$.

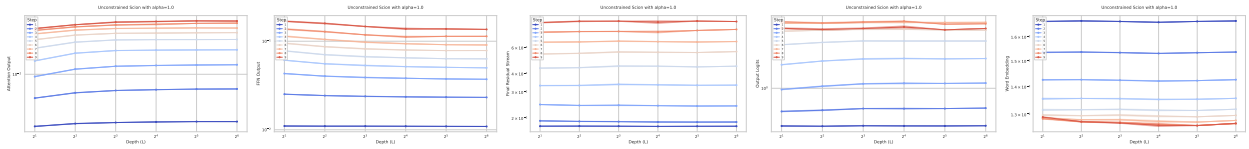


Figure 18: Coordinate check for Unconstrained Scion, on FineWeb, with $\alpha = 1.0$, and learning rate $\gamma = 2^{-11}$.

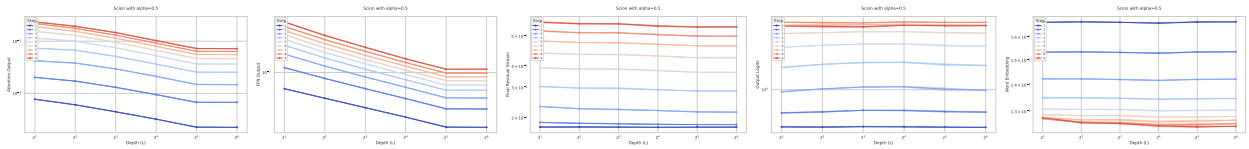


Figure 19: Coordinate check for Scion, on FineWeb, with $\alpha = 0.5$, and learning rate $\gamma = 2^{-11}$.

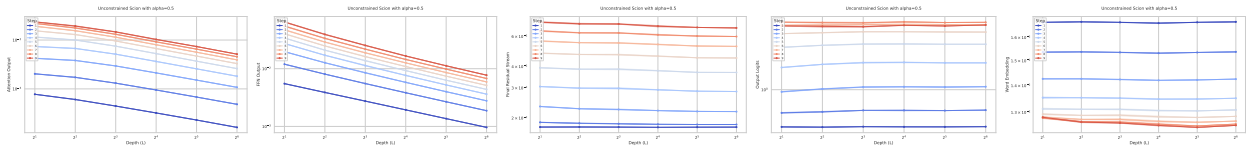


Figure 20: Coordinate check for Unconstrained Scion, on FineWeb, with $\alpha = 0.5$, and learning rate $\gamma = 2^{-11}$.