TO USE OR NOT TO USE MUON: HOW SIMPLICITY BIAS IN OPTIMIZERS MATTERS

Anonymous authors
Paper under double-blind review

ABSTRACT

While Adam has long been the ubiquitous default optimizer for deep neural networks, Muon has recently seen rapid adoption due to its superior training speed. Although much of the literature focuses on validating the benefits of Muon, our work investigates the potential downsides of the mechanism driving this speedup. On the theoretical front, we analyze the learning dynamics of simplified Muon on deep linear networks and multi-head linear attention. Our analysis reveals that Muon gains speed by avoiding saddle points, but at the expense of the simplicity bias exhibited by gradient descent, under which the complexity of the learned functional solution increases progressively. On the empirical front, we show experiments that demonstrate the consequences of losing the simplicity bias. Muon struggles to uncover common underlying structure across tasks and may be prone to fitting spurious features. More broadly, this paper serves as a reminder that faster optimization is rarely a free lunch; improvements in optimization can come at the cost of changes in the inductive biases that shape generalization.

1 INTRODUCTION AND BACKGROUND

Employing deep learning in practice usually involves careful decisions on the model architecture and data selection, while the choice of the optimizer typically relies on established defaults such as Adam, or more recently, Muon. This paper aims to spotlight the impact of optimizer choice: different optimizers take different optimization paths in the loss landscape, each with their own inductive biases. Despite this, prior works primarily focus on the benefit of faster optimization (Wen et al., 2026). There remains a lack of theory or intuition on how the learning trajectory taken impacts functional properties of the learned solution.

Specifically, we focus on the Muon optimizer (Jordan et al., 2024b) which has attracted a large audience due to its performance on the NanoGPT Speedrun (Jordan et al., 2024a), learning faster than other optimizers in wallclock time. It has since been adopted as one of the new defaults, e.g. in NanoChat (Karpathy, 2025), DeepSeek-V4 (DeepSeek-AI, 2026) and Kimi K3 (Team et al., 2026). However, we still lack an understanding of Muon's inductive biases, trajectories it takes to be fast, and the implications of these dynamics on the solution Muon-optimized model converges to.

Prior work has shown that Muon *does* learn all the features, e.g. much faster than gradient descent (GD) in a setting where their presence is largely imbalanced (Vasudeva et al., 2026; Wang et al., 2026). In our work, we focus on the question *how* Muon learns all these features. We find that Muon learns all features simultaneously at an equal rate, while GD learns them one by one. In other words, Muon does *not* possess this dynamical simplicity bias observed in GD (Saxe et al., 2014; 2019; Gidel et al., 2019; Kalimeris et al., 2019; Huh et al., 2023; Refinetti et al., 2023; Abbe et al., 2023; Zhang et al., 2026). Investigating the consequences, we find that Muon's equal learning of features can result in failure to capture generalizable structures. Here we summarize our contributions:

- We show the theory of GD- (Saxe et al., 2014) and simplified Muon- (Vasudeva et al., 2026) optimized deep linear networks (Section 2) provides a good mental model for feature learning, even in non-linear settings (Section 3). Importantly, both theory and empirics indicate that Muon loses the simplicity bias of GD, and learns all features equally, at the same time.
- Contrary to prior beliefs (Shah et al., 2020; Vasudeva et al., 2025; 2026), we show that optimizers with simplicity bias can surpass the ones without in a setting with spurious features. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

bigger picture is that the winner is determined by the data properties as well as the optimizer used (Section 3.1).

- We investigate potential downsides of Muon's equal feature learning: on a simple multimodal task, we show that Muon tends to *not* learn representations that generalize (Section 4).
- Shifting to more modern architectures, we theoretically analyze the learning dynamics of GD and simplified Muon on multi-head linear attention. Theory states that while GD learns the heads one-by-one, simplified Muon learns all the heads simultaneously (Section 5).
- Learning all heads at the same time could impact model's generalization, as related concepts may be learned by separate heads instead of having a common representation. We demonstrate this with a two-layer softmax transformer, which, when trained with Muon, does not extrapolate to unseen (but related) data, unlike its SGD-trained counterpart (Section 6).

The takeaway from our theory is that dynamical simplicity bias acts as an implicit curriculum (Liu et al., 2026a; Huang et al., 2026), forcing learning one by one, with the order of learning given by geometric properties of the model and data. Muon abandons this curriculum to optimize across multiple axes at once. However, in our theoretical settings, GD and Muon learn the same final solution. So does this matter in practice?

We design a sequence of experiments that demonstrate the answer is positive: learning a simpler feature first can unlock useful new features whose utility was not clear at initialization. While Muon's lack of curricula rapidly minimizes error, it prevents the model from stepping through simpler solutions that would have revealed the latent structures required for generalization—and this does happen in practice!

Muon Optimizer and Spectral GD. We briefly introduce the optimization step of **Muon** (MomentUm Orthogonalized by Newton-Schulz), as introduced in Jordan et al. (2024b). Unlike SGD and Adam, Muon takes advantage of the fact that most of the parameters optimized in deep learning are organized in weight matrices. Let $\boldsymbol{W}^{(t)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be a weight matrix after optimization step t, and $\boldsymbol{G}^{(t)}$ the usual GD update of the weights with momentum μ, i.e. $\boldsymbol{G}^{(t)} = \mu \boldsymbol{G}^{(t-1)} + \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}^{(t)})$. The update of Muon at time $t+1$ and learning rate η aims to be

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} - \eta \boldsymbol{U}^{(t)} \boldsymbol{V}^{(t)\top},$$

where $\boldsymbol{G}^{(t)} = \boldsymbol{U}^{(t)} \boldsymbol{S}^{(t)} \boldsymbol{V}^{(t)\top}$ is the Singular Value Decomposition (SVD) of $\boldsymbol{G}^{(t)}$.[1] In words, Muon aims to set all non-zero singular values of the update matrix to be equal to 1. This process is termed *orthogonalization* of $\boldsymbol{G}^{(t)}$. However, computing the SVD at every step is computationally prohibitive for deep learning. To address this, Muon employs Newton-Schulz iterations to efficiently approximate orthogonalization (Kovarik, 1970; Björck & Bowie, 1971).

We also introduce **Spectral GD** (Carlson et al., 2015; 2016), a simplification of Muon. Compared to Muon, Spectral GD (i) orthogonalizes the update using exact SVD, instead of approximate SVD with Newton-Schulz iterations; (ii) orthogonalizes $\nabla_{\boldsymbol{W}} L(\boldsymbol{W}^{(t)})$ directly and doesn't use momentum. Spectral GD retains important properties of Muon (orthogonalization) and is more mathematically tractable. We study Spectral GD for the theory and switch back to the full Muon for experiments.

2 Deep Linear Networks: Spectral GD Loses Simplicity Bias

Our journey towards understanding Muon's biases starts with theoretical analysis on deep linear networks trained with mean squared error loss. We informally present it here, attribute the results to Vasudeva et al. (2026), and compare to classical results for GD (Saxe et al., 2014). A simulation is provided in Figure 1, and formal statements and proofs are provided in Appendix B.

Theorem 2.1 (Gradient flow dynamics—Informal)**.** *Gradient flow dynamics in two-layer deep linear network exhibits gradual increase in complexity: starting from small initialization, principal components are learned one by one, from larger singular values s to smaller, sequentially increasing the rank of the learned solution. For each principal component to be learned, a saddle point needs to be escaped, taking $\propto 1/s$ time.*

[1] The SVD performed is compact, meaning the diagonal matrix $\boldsymbol{S}^{(t)}$ is of dimension $r \times r$ (r being the rank of $\boldsymbol{G}^{(t)}$) and contains only positive singular values.

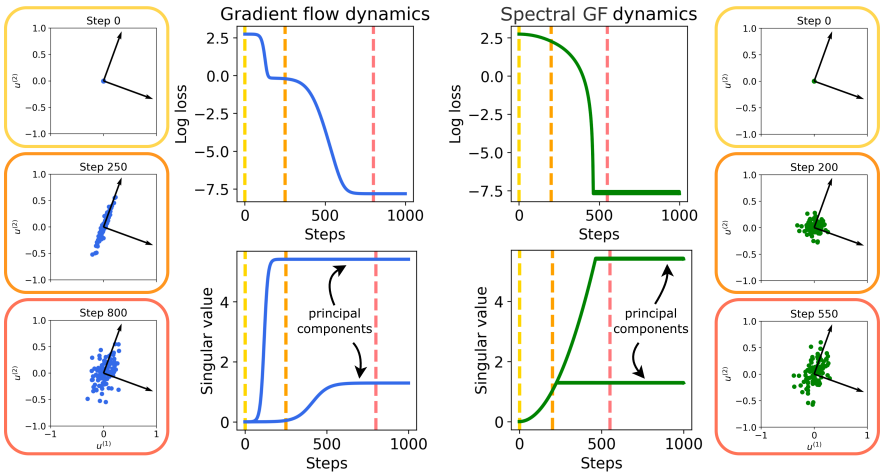

Figure 1: Illustration of the theory presented in Section 2, for gradient flow (*Left*) and Spectral GF (*Right*). The top row depicts the loss curve, the bottom one the evolution of singular values of the neural network. We also show the evolution of each row $u_i \in \mathbb{R}^d_{in}$ ($d_{in} = 2$) of the first layer matrix $U \in \mathbb{R}^{H \times d_{in}}$ for $H = 100$ hidden neurons (dots in the boxed plots). Black arrows are right singular vectors of the input-output data correlation matrix (see Appendix B for more details). Under GF, the first singular vector is learned to completion before learning of the second begins. In contrast, Spectral GF learns both simultaneously at the same speed; the smaller singular vector completes learning first, after which the u_is evolve only in the direction of the larger one. These simulations closely follow theorems 2.1, 2.2 and proofs in Appendix B, also containing simulation details.

Theorem 2.2 (Spectral gradient flow dynamics—Informal)**.** *Spectral gradient flow learns all principal components with the same speed, at the same time. Hence the ones with the smaller singular values s are fully learned first, in time $\propto \sqrt{s}$.*

The geometrical reason GD is slower is that its optimization trajectory passes near many saddle points on the loss landscape. In contrast, Spectral GD doesn't pass through these (or other) saddles, but also it doesn't learn the solution progressively—*it loses a simplicity bias*. While beneficial when learning features do not interact with each other (e.g. memorizing independent facts with different frequencies (Su, 2025; Vasudeva et al., 2026; Wang et al., 2026; Kim et al., 2026)), Spectral GD's learning is more *greedy*, in the sense that it learns all the different singular vectors simultaneously.

3 Qualitative Extension to Non-Linear Setting

Equipped with theoretical understanding of simplified Muon in a linear sandbox, we examine how the theory can help us understand more complex settings. To this end, we employ the full Muon to optimize the cross-entropy loss of a non-linear CNN with stochastic batches.

Task. We are interested in model's feature learning through training, analogous to principal components learning in the theory above. Each image in our training data has a **digit feature** (handwritten MNIST digit (Deng, 2012)) , and a **pixel feature**. All handwritten images of the digit i have the pixel feature—a pixel with intensity 1—at the same location, different one than digits $j \neq i$. This makes both the digit and pixel features predictive of the label. To evaluate the models, we use four different evaluation sets, shown in Figure 2a:

- *Test*: images with both the handwritten digit feature and the pixel feature corresponding to the desired label.
- *Digit*: images with only the digit feature, namely images from the original MNIST test set.
- *Misaligned, label=digit*: images with digit i and the pixel feature of $(i - 1) \pmod{10}$. The desired label corresponds to the digit feature.
- *Misaligned, label=pixel*: same misaligned images, but labels correspond to the pixel feature.

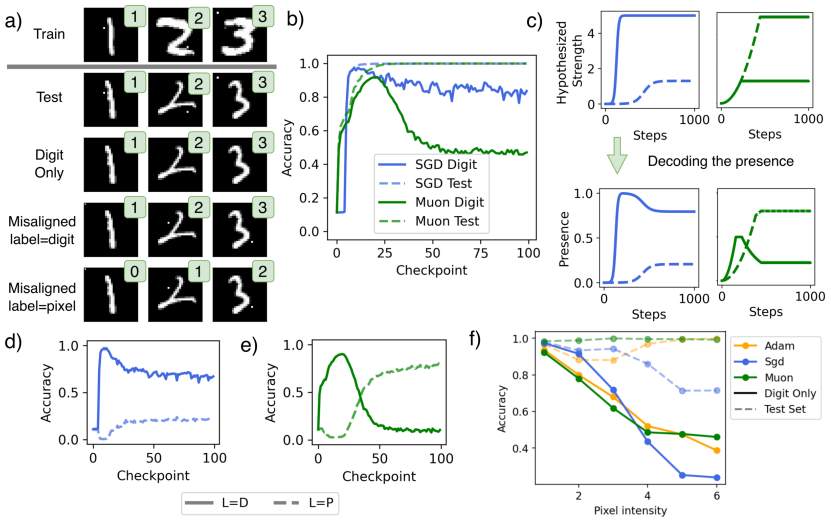

Figure 2: a) Training samples and different evaluation sets in the two feature learning setup. b) Accuracy on Test and Digit only evaluators. c) Converting hypothesized strengths to presence of each feature with (*Left*) GD and (*Right*) Muon. d) Accuracy of SGD on Misaligned evaluators with label=digit (L=D) and label=pixel (L=P). e) Accuracy of Muon on the same sets. We notice how these competing accuracies qualitatively follow the decoded hypothesized presence of the two features. f) Peak accuracy on the Digit only evaluator and Test accuracy (dashed) at the point that peak is achieved. The accuracies are plotted as we increase the intensity of the spurious pixel. The result illustrates that there is not a fixed optimizer better with spurious features; data properties also play a key role in determining the winner.

Experiments. Taking a look at the accuracies on Test and Digit evaluators in Figure 2b, we see that SGD pursues a solution relying only on the digit feature for much longer: the two curves separate only at the peak accuracy on Digit set. Muon's curves however start separating earlier, indicating that at the peak accuracy on Digit set, the model is already non-negligibly relying on the pixel feature as well. This is reminiscent of the theory: early in training, GD has learned only the largest principal component, while simplified Muon has learned all components in the same time. For all the details and further experiments, see Appendix D.2, and for Q&A Appendix E.

To connect with the theory more explicitly, we need an analogue of singular values. For this purpose, we define a concept of **strength** of each of the two features, imitating the two singular values. Imagine that the strength of each feature through training can be modeled by curves similar to singular value evolution for each optimizer. Empirically it's not clear how to extract feature strength from the model, but we can measure which feature the model is relying on more—we call the relevant quantity feature **presence**. We can extract feature presence by measuring the accuracy on Misaligned, label=digit and Misaligned, label=pixel evaluators—a model cannot give the correct answer to both, so there is competition.

In theory, to go from hypothesized strength $s_i(t)$ of feature i to its hypothesized presence $p_i(t)$ at time t, we use the following formula: $p_i(t) = s_i(t)\min(\frac{1}{\max_t s_i(t)}, \frac{1}{s_1(t)+s_2(t)})$. The first term in the minimum just serves as normalization, not letting the feature be present more than 100%. The second term is responsible for competition, making sure that $p_1(t) + p_2(t) \leq 1$. This means that once both features have large strength, their presence will be proportional to their strength. We plot the decoded feature presence from our hypothesized feature strengths in Figure 2c.

Comparing hypothesized presence with its empirical counterpart on the Misaligned evaluators for both SGD and Muon (Figure 2d,e reveals a striking qualitative similarity. This suggests that **evolution of the singular values is a good mental model for feature learning with both optimizers**.

Importantly, we also see that for SGD the digit feature has larger strength: the final accuracy on label=digit is larger than on label=pixel (see Appendix E.1 for a longer run). The contrary holds for Muon model, which tells us that pixel feature is more dominant in the final solution learned by

Muon. Crucially, this demonstrates that **the relative strength of learned features is intrinsically tied to the choice of optimizer, even when the models are exposed to identical training data**. This may be highly relevant when dealing with spurious features, as we investigate next.

3.1 How simplicity bias interacts with spurious features

Spurious features are defined as 'simpler' features in the training data that could be predictive, but do not generalize to other settings (Puli et al., 2022). An example of a spurious feature could be the scanner imprint on a medical scan, an image background in classification of birds, or the pixel feature in our MNIST setup (Puli et al., 2024). Prior work has claimed that simplicity bias of SGD hurts in these settings compared to Adam and Muon, as it leads to simpler solutions that rely on spurious features (Shah et al., 2020; Vasudeva et al., 2025; 2026).

Our results in Figure 2b indicate differently. First, they show that SGD can outperform Muon (and Adam, see Appendix E.1), measured by the peak accuracy on Digit or Misaligned, label=digit sets. Furthermore, the situation can easily change: when varying the intensity of the spurious pixel in Figure 2f, we observe the transition from SGD being better to Muon as the intensity increases. The hypothesis from the theory explains this: while the intensity is low enough, the digit feature is more dominant and SGD's simplicity bias is helpful. The situation changes for larger intensities, when the simplicity bias focuses on the more dominant, spurious feature, hurting the performance of SGD; Muon learns both features at the same time which in this situation helps. This shows that **optimizing with Muon is a double-edged sword: while learning is more uniform across the features, task and the data determine whether that is a good thing or not**.

4 Spectral GD does not Learn Generalizable Representations

The better understanding of Muon's equal feature learning leads us to question the potential downsides compared to GD. Our testbed for that purpose is a toy task which can be solved in multiple ways, but only one—the one exploiting the common underlying structure—achieves good out of distribution generalization. In that case, we say that the model has learned the *generalizable representations*.

Setup. We use the routing task setup from Saxe et al. (2022), an instantiation of a multimodal task. The architecture and task are presented in Figure 3a,b. While all different input and output sources use their own linear encoders and decoders respectively, the hidden layer is used by all. During training, each sample (x, y) will have its input source (i.e. which domain x came from) and output source (domain of y), and only corresponding layers are used and trained. One could imagine this setting being a simplification for multi-modal networks, where the different sources may be text, images, and sound. The goal of the task is to learn the mapping from 4 numbers $\{1, 2, 3, 4\}$ to 4 vectors, as shown in Figure 3b. However, each input source has different vector representations for $\{1, 2, 3, 4\}$: for each input source, we generate four 4-dimensional orthonormal vectors, encoding the four numbers. For simplicity, we don't transform the output vectors based on output domain (but note that still, each output domain has its own linear decoder). We train the model with mean squared error loss over the predicted output vector and the true one (details in Appendix D.3).

The key challenge of the task is that not all input-output pairs of sources are seen in training. Specifically, we only see samples (x, y) where x came from input source j, and y is from output domain j or $j + 1$ $(M + 1 := 1)$, for all $j = 1, ..., M$. The shared linear layer is of dimension 64×64, large enough to memorize all the pairings seen in training, even without realizing that the task has a simple underlying structure to exploit.

Experiments. The empirical results are captured in Figure 3c,d, where we show the function learned across all input-output pairs of sources. Training with SGD results in learning the shared representations across all inputs, because even the input-output pairs that have never been seen during training do end up producing the correct underlying function. This generalization, however, doesn't happen when optimizing with Spectral GD. While it does learn to solve the task for input-output pairs seen during training, it does so by memorizing each input-output pair seen, as can be inferred from the poor generalization on input-output pairs not present in training. It is important to note that both optimizers achieve perfect convergence (Figure 3e), thus the hope of Spectral GD recovering the underlying shared representations with more data is gone. Further confirmation comes from the hidden

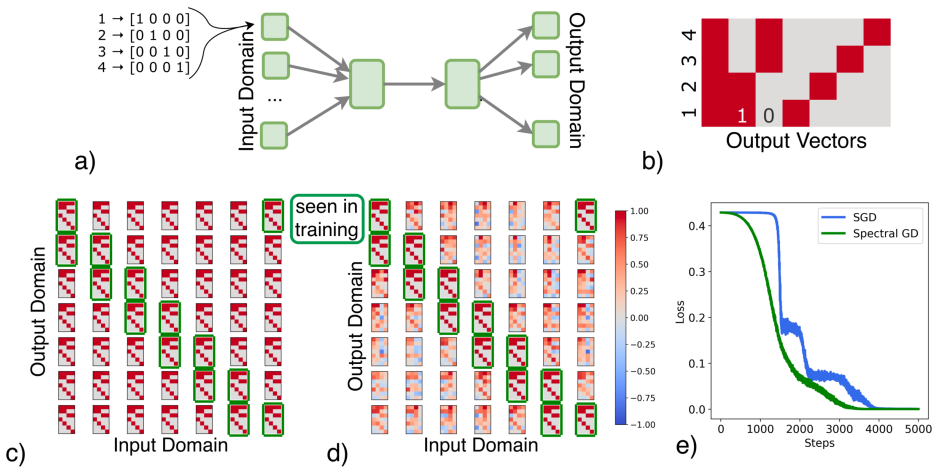

Figure 3: a) The neural network used to solve the task, where each of the gray arrows is a linear layer, with no nonlinearities in between. There is $M = 7$ input and output domains. Each input domain has its own, fixed 4 orthonormal vectors to represent $\{1, 2, 3, 4\}$. b) The underlying task we're learning: mapping each number in $\{1, 2, 3, 4\}$ to the output vector shown. Results after training in the 'routing' setup with c) SGD and d) Spectral GD, together with e) training loss curves. We plot the function the models learned (4 different column vectors represent the image of $\{1, 2, 3, 4\}$) for all the different input-output pairs of sources. Circled in green are the pairs seen during training.

layer's spectral properties: whereas the SGD solution converges to an approximate low-rank structure (rank $= 4$, equaling the dimension of the underlying task), the Spectral GD solution exhibits a significantly higher effective rank with a heavy-tailed spectrum—a hallmark of memorization rather than learning shared structure.

This example demonstrates that the removal of simplicity bias, while yielding faster convergence, taxes the structural quality of the learned solution. **By prioritizing rapid loss reduction, the optimizer bypasses the discovery of common latent structures**, converging instead to a less generalizable solution. Next, we show this behavior can be found in attention.

5 Theory on Spectral GD with Multi-Head Linear Attention

Intrigued by the results in the simpler architectures, we question whether these effects persist in more modern architectures, particularly in transformers. Although recent work has studied the saddle points and simplicity bias in GD-optimized transformers (Boix-Adsera et al., 2023; Edelman et al., 2024; Rende et al., 2024; Geshkovski et al., 2024; Zhang et al., 2025; Varre et al., 2025; Yüksel et al., 2025; Saha et al., 2026), no such analysis exists for Muon. To fill in this gap, we theoretically study the linear self-attention model (Von Oswald et al., 2023) trained on an in-context linear regression task (Garg et al., 2022) and build on the training dynamics analysis of Zhang et al. (2025). We choose to study in-context linear regression over other transformer tasks because its scalar-valued output makes the spectral GF and GF dynamics more tractable.

Task. Each training sample $\mu = 1, \ldots, P$ is generated by drawing a D-dimensional task vector $\boldsymbol{w}_\mu \sim \mathcal{N}(0, \boldsymbol{I}_D)$, and drawing $(N+1)$ D-dimensional $\boldsymbol{x}$ tokens, $\boldsymbol{x}_{\mu,n}, \boldsymbol{x}_{\mu,q} \in \mathbb{R}^D$ $(n = 1, \ldots, N)$, independently from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Lambda})$. The input sequence is constructed as

$$\boldsymbol{X}_\mu = \begin{bmatrix} \boldsymbol{x}_{\mu,1} & \cdots & \boldsymbol{x}_{\mu,N} & \boldsymbol{x}_{\mu,q} \\ y_{\mu,1} & \cdots & y_{\mu,N} & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{\mu,1} & \cdots & \boldsymbol{x}_{\mu,N} & \boldsymbol{x}_{\mu,q} \\ \boldsymbol{w}_\mu^\top \boldsymbol{x}_{\mu,1} & \cdots & \boldsymbol{w}_\mu^\top \boldsymbol{x}_{\mu,N} & 0 \end{bmatrix} \in \mathbb{R}^{(D+1)\times(N+1)}.$$

The first N columns of $\boldsymbol{X}_\mu$ are the context and $\boldsymbol{x}_{\mu,q}$ is the query. The target output is $y_q = \boldsymbol{w}_\mu^\top \boldsymbol{x}_{\mu,q}$, which is the task vector of this sequence applied to the query. This task is called in-context linear regression because the model needs to linearly regress the task vector in context using N pairs of $(\boldsymbol{x}_{\mu,n}, y_{\mu,n})$ and then apply it to the query.

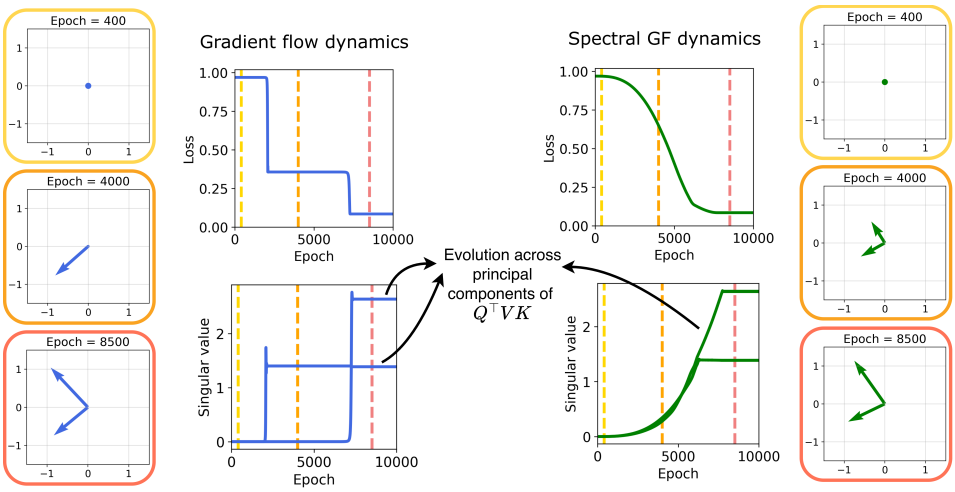

Figure 4: Learning dynamics of multi-head linear attention trained with GF (Left) and Spectral GF (Right). Top row shows the loss curves, and bottom one the growth of singular values of $\boldsymbol{Q}^\top \boldsymbol{V}\boldsymbol{K}$ for each of the $D = 2$ principal components. Framed in orange are the snapshots of $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ through training. This simulation follows the theory, illustrating the head-by-head learning with GF and all heads in the same time with Spectral GF.

Model. We analyze one-layer linear attention with H heads and rank-one key and query weights in each head. Following prior work (Zhang et al., 2024; 2025; Lu et al., 2025), we study linear attention in which weight entries irrelevant to this task are initialized to zero. In this regime, the prediction of the model for $y_{\mu,q}$ given input $\boldsymbol{X}_\mu$ is $\hat{y}_{\mu,q} = \sum_{i=1}^{H} v_i \left(\frac{1}{N}\sum_{n=1}^{N} y_{\mu,n}\boldsymbol{x}_{\mu,n}^\top\right)\boldsymbol{k}_i\boldsymbol{q}_i^\top\boldsymbol{x}_{\mu,q}$, where the value, key, and query weights in the i-th head are denoted as $v_i \in \mathbb{R}$, and $\boldsymbol{k}_i, \boldsymbol{q}_i \in \mathbb{R}^D$. Writing these in matrix form gives $\boldsymbol{K}, \boldsymbol{Q} \in \mathbb{R}^{H\times D}$ and a diagonal matrix $\boldsymbol{V} \in \mathbb{R}^{H\times H}$ with i-th entry on the diagonal v_i. We train the model with squared loss, $L = \mathbb{E}[(y_{\mu,q} - \hat{y}_{\mu,q})^2]/2$. We denote the in-context covariance of $\boldsymbol{x}_{\mu,n}$ as $\hat{\boldsymbol{\Lambda}}_\mu = \sum_{n=1}^{N} \boldsymbol{x}_{\mu,n}\boldsymbol{x}_{\mu,n}^\top/N$. To facilitate comparison between GD and Spectral GD, we rewrite the gradient flow dynamics from Zhang et al. (2025) in matrix form:

$$\frac{d\boldsymbol{V}}{dt} = \texttt{diag}\left(\boldsymbol{K}\left(\boldsymbol{\Lambda}^2 - \mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]\boldsymbol{K}^\top\boldsymbol{V}\boldsymbol{Q}\boldsymbol{\Lambda}\right)\boldsymbol{Q}^\top\right) \tag{1a}$$

$$\frac{d\boldsymbol{K}}{dt} = \boldsymbol{V}\boldsymbol{Q}\left(\boldsymbol{\Lambda}^2 - \boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{V}\boldsymbol{K}\mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]\right) \tag{1b}$$

$$\frac{d\boldsymbol{Q}}{dt} = \boldsymbol{V}\boldsymbol{K}\left(\boldsymbol{\Lambda}^2 - \mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]\boldsymbol{K}^\top\boldsymbol{V}\boldsymbol{Q}\boldsymbol{\Lambda}\right). \tag{1c}$$

Under small initialization and assumptions stated below, gradient flow exhibits saddle-to-saddle dynamics Zhang et al. (2025), learning one head at a time (Figure 4 Left). Under the same condition, we show that Spectral GF dynamics is characterized by Theorem 5.1.

Assumptions. We assume the attention is parameterized with weight matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, and Spectral GD updates are applied to those matrices. We also assume joint diagonalizability of $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ and $\boldsymbol{\Lambda}$ at small initialization, meaning the left principal components of $\boldsymbol{K}, \boldsymbol{Q}$ are the canonical basis vectors in $\mathbb{R}^H$, and the right ones are the eigenvectors of $\boldsymbol{\Lambda}$. We assume $H = D$, and for $H > D$ simulations are in Appendix D.5.

Theorem 5.1 (Spectral gradient flow on linear attention—Informal)**.** *Unlike gradient flow, in the setting described, Spectral gradient flow doesn't pass through saddle points and hence doesn't learn one head at a time. Instead, all heads are being learned at the same time and rate, linearly per component $\boldsymbol{Q}, \boldsymbol{K}$ and $\boldsymbol{V}$. This leads to each head i learning the eigenspace of $\boldsymbol{\Lambda}$ with eigenvalue λ_i in time $\propto \lambda_i^{-1/3}$.*[2] *For a simulation, see Figure 4 Right.*

The proof and formal statement for GF and spectral GF are provided in Appendices C.1 and C.2.

[2]This is for the case $N \to \infty$, for $N < \infty$ see the formal version.

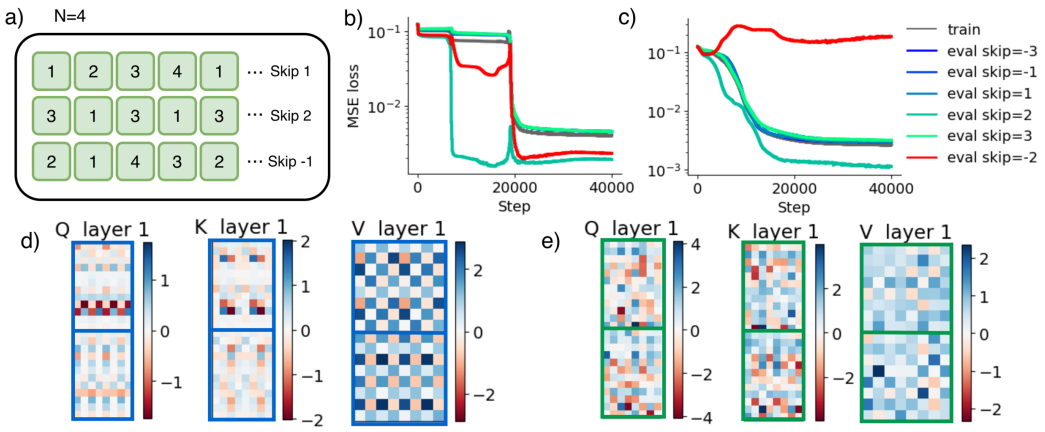

Figure 5: a) Three example sequences with different skips from our task, for $N = 4$. Loss on the in- and out-of-distribution data in our transformer setup, when optimized by **b) SGD** and **c) Muon**. The model has been trained on all skips shown, apart from skip=-2 (red). d) Final weights of the SGD trained transformer. e) Muon trained instead. The two heads are separated by the framing. Notice how the SGD algorithm picked up the underlying cyclic structure and generalized to unseen skip, where Muon learned all skips more separately, thus not generalizing.

The dynamical simplicity biases in linear attention and linear networks are driven by two slightly different mechanisms. In GF optimized linear attention, weights in one head grow at a time, while the others remain near zero. In deep linear networks, all hidden neurons evolve, but they grow along one principal component direction at a time. We have shown that simplified Muon removes both types of simplicity bias. In the case of GD, Zhang et al. (2025) showed that with heads of higher rank $R > 1$, once a head is activated (meaning its weights reach $O(1)$), learning $\boldsymbol{Q}$ and $\boldsymbol{K}$ empirically follows the dynamics of a deep linear network, as indeed these two matrices are multiplied in the attention. For Spectral GF, this would imply that transformer architecture is prone to losing both types of simplicity biases. Further analysis of learning dynamics of attention matrices with Muon seems promising, and can help us justify major decisions in large scale models. One example is applying Muon to all heads concatenated, as is usually done, or to each separately, as found empirically beneficial by Zeng et al. (2026); Team et al. (2026).

6 A Potential Downside of Using Muon on Transformers

Thinking about generalization, it is important that (i) a single concept is learned by a single head and (ii) closely related concepts are learned by the same head. Muon learning all heads simultaneously may take away the implicit curriculum of SGD, resulting in ungeneralizable representations. In this section, we provide empirical evidence for this.

Model. We train a two-layer causal transformer on MSE loss, with 2 softmax heads in each layer. Our inputs are one hot vectors and we don't use token embedding and unembedding. We use RoPE positional encoding. For more details, see Appendix D.4, and for ablations see Q&A in E.3.

Task. The task itself should be simple, featuring shared underlying structure and easy evaluation to test whether the model has learned it. Satisfying all this, we introduce the *cycle task*: the model is trained on next token prediction to learn the numbers from 1 to N in cyclical order, with different skips. Namely, to draw a sample, first draw its skip s from a subset of $\{1, 2, ..., N-1\}$ uniformly, then the first number in the sequence i also uniformly, independently of the skip. The sample is given by the sequence $(i, (i+s) \pmod N, (i+2s) \pmod N, ...)$ with the length $10N$, and $0 \equiv N$. Figure 5a contains a couple of examples for the setting $N = 4$. The catch this time is that the model sees only a strict subset of skips during training, and the goal is to see whether it learns to generalize through evaluating on unseen skips.

Experiments. We evaluate the SGD and Muon in the $N = 8$ setting, training on the skip set $\{1, 2, 3, -1, -3\}$. Although the models do not encounter skip -2 during training, they are exposed to two forward-backward pairs: $(1, -1)$ and $(3, -3)$. We investigate whether observing these symmetries, alongside the presence of skip 2, provides sufficient signal for extrapolating to the unseen skip -2. The results, presented in Figure 5b,c, demonstrate that the SGD-optimized transformer learns skip -2 nearly as well as skip 2. In contrast, the Muon-optimized model exhibits poor performance on this extrapolated task, suggesting that **Muon struggles to learn unified representations that generalize** across different skips. The heads both optimizers learned, shown in Figure 5d,e, exhibit the theory-suggested differences: while SGD's solution is effectively low-rank, with higher contribution from one head than the other, Muon's solution uses both heads more equally. This failure to abstract shared rules poses a risk for tasks like mathematics, where the goal is not to memorize specific derivation steps, but to acquire reusable techniques applicable to unseen problems.

7 Discussion

Optimizers are usually compared in terms of how quickly they reduce the loss; however, ultimately what we care about is the functional solution they learned. We focused on the Muon optimizer (Jordan et al., 2024b), and explored the consequences of its implementation choices. Theoretically, we explain why it is faster than GD alternatives, but has the trade-off of losing a simplicity bias. Motivated by the theory, we constructed a couple of examples where SGD outperforms Muon, an outcome rarely seen in prior work. First, in an example with a spurious feature, we showed how the theory applies, and witnessed Muon being worse due to learning all features simultaneously. Contextualizing this within prior work on the pitfalls of simplicity bias, we demonstrated that there is no universally optimal optimizer with spurious features; instead, the interaction between the optimization algorithm and data properties dictates performance. We also evaluated Muon on underlying tasks with shared structure across settings and noticed how Muon fails to learn unified representations. This failure also happened in transformers, leading to Muon generalizing worse than SGD.

Bridging training dynamics theory closer to practice. Orthogonal to exploring the downsides of Muon, this work contributes to narrowing the gap between theoretical understanding of learning dynamics and practical optimization. Prior theoretical work on training dynamics spans a relatively broad range of architectures (Hu et al., 2020; Zhang et al., 2026; Ziyin et al., 2026) and tasks (Saxe et al., 2019; Shah et al., 2020; Refinetti et al., 2023), but almost always assumes gradient descent as the optimization procedure. We show how the training dynamics change when switching from gradient descent to simplified Muon, revealing the loss of simplicity in its trajectory.

Showing the consequences exist. In our theory, the features were *decoupled*, meaning learning one did not impact learning the other. Indeed, both GD and Muon ended up at the same solution. However, in practice, the order of learning *can* matter for generalization: e.g. in our cycle task, it was important to first learn that the tokens have an underlying cyclical structure, and only after specialize to each skip—the features were *coupled*. We showed that when the data has coupled features, optimizer used is highly relevant: the same loss does *not* mean the same solution is learned.

Limitations and Future Work. Our work, trying to keep closer to theoretically tractable settings, did not answer the questions about Muon optimizer at larger scale. Concurrent work of Liu et al. (2026b) empirically shows that Muon-trained GPT-2 memorizes more than Adam and SGD counterparts. Specifically, they found that Muon-trained models perform worse than Adam after SFT on a math dataset. We reached a similar conclusion from a different perspective: our theory explains *why* memorization happens. We expect coupled features are also present in math, but not in memorization tasks, hence Muon hurts on math question answering in Liu et al. (2026b), but helps with associative memory in Wang et al. (2026). The next important step would be quantifying the dependence between features within a natural language corpus (how much are they coupled), then investigating optimizers behaviors on more or less coupled subsets.

Altogether, our paper explains the importance of analyzing inductive biases and future work should take these biases into account when developing an optimizer, rather than focusing purely on speed. Deciding on what makes a good optimizer is a hard question, and once we have the answer (or an idea for a high quality answer), perhaps it will become easier to act upon it.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods (coding) and refining the text of the paper. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Emmanuel Abbe, Enric Boix Adserà, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. In Gergely Neu and Lorenzo Rosasco (eds.), *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pp. 2552–2623. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/abbe23a.html.

Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.08.022. URL https://www.sciencedirect.com/science/article/pii/S0893608020303117.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c0c783b5fc0d7d808f1d14a6e9c8280d-Paper.pdf.

Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1NvflqAdoom.

Raphaël Berthier. Incremental learning in diagonal linear networks. *Journal of Machine Learning Research*, 24(171):1–26, 2023. URL http://jmlr.org/papers/v24/22-1395.html.

Å. Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971. ISSN 00361429. URL http://www.jstor.org/stable/2949484.

Enric Boix-Adsera, Etai Littwin, Emmanuel Abbe, Samy Bengio, and Joshua Susskind. Transformers learn through gradual rank increase. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 24519–24551. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4d69c1c057a8bd570ba4a7b71aae8331-Paper-Conference.pdf.

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic Spectral Descent for Restricted Boltzmann Machines. In Guy Lebanon and S. V. N. Vishwanathan (eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 111–119, San Diego, California, USA, 09–12 May 2015. PMLR. URL https://proceedings.mlr.press/v38/carlson15.html.

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, 2016. doi: 10.1109/JSTSP.2015.2505684.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

Ezra Edelman, Nikolaos Tsilivis, Benjamin L. Edelman, Eran Malach, and Surbhi Goel. The evolution of statistical induction heads: In-context learning markov chains. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 64273–64311. Curran Associates, Inc., 2024. doi: 10.52202/079017-2050. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/75b0edb869e2cd509d64d0e8ff446bc1-Paper-Conference.pdf.

Kenji Fukumizu. Effect of batch learning in multilayer neural networks. *Gen*, 1(04):1E–03, 1998.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf.

Borjan Geshkovski, Hugo Koubbi, Yury Polyanskiy, and Philippe Rigollet. Dynamic metastability in the self-attention model, 2024. URL https://arxiv.org/abs/2410.06833.

Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f39ae9ff3a81f499230c4126e01f421b-Paper.pdf.

Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *Advances in Neural Information Processing Systems*, 33:17116–17128, 2020.

Yu Huang, Zixin Wen, Yuejie Chi, Yuting Wei, Aarti Singh, Yingbin Liang, and Yuxin Chen. On the emergence of implicit curriculum in RLVR learning dynamics. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=KxYCE98u1d.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=bCiNWDmlY2.

Tom Jacobs, Chao Zhou, and Rebekka Burkholz. Never saddle: Reparameterized steepest descent as mirror flow. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YgudIlQ9nC.

Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJflg30qKX.

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. URL https://kellerjordan.github.io/posts/muon/.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/b432f34c5a997c8e7c806a895ecc5e25-Paper.pdf.

Andrej Karpathy. nanochat: The best chatgpt that $100 can buy, 2025. URL https://github.com/karpathy/nanochat.

Juno Kim, Eshaan Nichani, Denny Wu, Alberto Bietti, and Jason D. Lee. Sharp capacity scaling of spectral optimizers in learning associative memory, 2026. URL https://arxiv.org/abs/2603.26554.

Zdislav Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 7(3):386–389, 1970. doi: 10.1137/0707031. URL https://doi.org/10.1137/0707031.

Andrew K. Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryfMLoCqtQ.

Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AHOs7Sm5H7R.

Emmy Liu, Kaiser Sun, Millicent Li, Isabelle Lee, Lindia Tjuatja, Jen tse Huang, and Graham Neubig. What do language models learn and when? the implicit curriculum hypothesis, 2026a. URL https://arxiv.org/abs/2604.08510.

Yuxing Liu, Jianyu Wang, and Tong Zhang. Optimizer-model consistency: Full finetuning with the same optimizer as pretraining forgets less. *arXiv preprint arXiv:2605.06654*, 2026b.

Yue M. Lu, Mary Letey, Jacob A. Zavatone-Veth, Anindita Maiti, and Cengiz Pehlevan. Asymptotic theory of in-context learning by linear attention. *Proceedings of the National Academy of Sciences*, 122(28):e2502599122, 2025. doi: 10.1073/pnas.2502599122. URL https://www.pnas.org/doi/abs/10.1073/pnas.2502599122.

Hartmut Maennel, Olivier Bousquet, and Sylvain Gelly. Gradient descent quantizes relu network features. *arXiv preprint arXiv:1803.08367*, 2018.

Aahlad Puli, Lily Zhang, Yoav Wald, and Rajesh Ranganath. Don't blame dataset shift! shortcut learning due to gradients and cross entropy. *Advances in Neural Information Processing Systems*, 36:71874–71910, 2023.

Aahlad Manas Puli, Lily H Zhang, Eric Karl Oermann, and Rajesh Ranganath. Out-of-distribution generalization in the presence of nuisance-induced spurious correlations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=12RoR2o32T.

Aahlad Manas Puli, Nitish Joshi, Yoav Wald, He He, and Rajesh Ranganath. Nuisances via negativa: Adjusting for spurious correlations via data augmentation. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=RIFJsSzwKY.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

Maria Refinetti, Alessandro Ingrosso, and Sebastian Goldt. Neural networks trained with SGD learn distributions of increasing complexity. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28843–28863. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/refinetti23a.html.

Riccardo Rende, Federica Gerace, Alessandro Laio, and Sebastian Goldt. A distributional simplicity bias in the learning dynamics of transformers. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 96207–96228. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ae6c81a39079ddeb88b034b6ef18c7fe-Paper-Conference.pdf.

Rohan Saha, Farzane Aminmansour, and Alona Fyshe. Understanding the staged dynamics of transformers in learning latent structure, 2026. URL https://arxiv.org/abs/2511.19328.

Andrew Saxe, Shagun Sodhani, and Sam Jay Lewallen. The neural race reduction: Dynamics of abstraction in gated networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19287–19309. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/saxe22a.html.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6120.

Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116 (23):11537–11546, 2019.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9573–9585. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6cfe0e6127fa25df2a0ef2ae1067d915-Paper.pdf.

Wei Shen, Ruichuan Huang, Minhui Huang, Cong Shen, and Jiawei Zhang. On the convergence analysis of muon. *arXiv preprint arXiv:2505.23737*, 2025.

Weijie Su. Isotropic curvature model for understanding deep learning optimization: Is gradient orthogonalization optimal? *arXiv preprint arXiv:2511.00674*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton Van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16761–16772, 2022.

Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye4g3AqFm.

Aditya Varre, Gizem Yüce, and Nicolas Flammarion. Learning in-context n-grams with transformers: Sub-n-grams are near-stationary points. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 60924–60963. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/varre25a.html.

Bhavya Vasudeva, Jung Whan Lee, Vatsal Sharan, and Mahdi Soltanolkotabi. The rich and the simple: On the implicit bias of adam and sgd. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Bhavya Vasudeva, Puneesh Deora, Yize Zhao, Vatsal Sharan, and Christos Thrampoulidis. How muon's spectral design benefits generalization: A study on imbalanced data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YzjS4jcfmS.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/von-oswald23a.html.

Shuche Wang, Fengzhuo Zhang, Jiaxiang Li, Cunxiao Du, Chao Du, Tianyu Pang, Zhuoran Yang, Mingyi Hong, and Vincent Y. F. Tan. Muon outperforms adam in tail-end associative memory learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=twbMFL0DMp.

Taishi Watanabe, Ryo Karakida, and Jun-nosuke Teramae. The impact of anisotropic covariance structure on the training dynamics and generalization error of linear networks. *arXiv preprint arXiv:2601.06961*, 2026.

Kaiyue Wen, David Leo Wright Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2J51qUZ0iG.

Oğuz Kaan Yüksel, Rodrigo Alvarez Lucendo, and Nicolas Flammarion. Incremental learning of sparse attention patterns in transformers. In *EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM)*, 2025. URL https://openreview.net/forum?id=sZb2e9hM58.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024. URL http://jmlr.org/papers/v25/23-1042.html.

Yedi Zhang, Aaditya K Singh, Peter E. Latham, and Andrew M Saxe. Training dynamics of in-context learning in linear attention. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76047–76087. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhang25br.html.

Yedi Zhang, Andrew M Saxe, and Peter E. Latham. Saddle-to-saddle dynamics explains a simplicity bias across neural network architectures. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Vit5M0G5Gb.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham M. Kakade. Deconstructing what makes a good optimizer for autoregressive language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zfeso8ceqr.

Liu Ziyin, Botao Li, and Xiangming Meng. Exact solutions of a deep linear network. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24446–24458. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9a940e858b17f01c402e164835140c4a-Paper-Conference.pdf.

Liu Ziyin, Yizhou Xu, Tomaso Poggio, and Isaac Chuang. Neural quadratic forms: A unified minimal model for sudden learning and scaling laws, 2026. URL https://arxiv.org/abs/2608.13335.

A Extended Related Work

We mostly base our theory in Section 2 for GD on Zhang et al. (2026) and for Spectral GD on Vasudeva et al. (2026). The theory for GD in Section 5 is based on Zhang et al. (2025).

Saddle-to-saddle dynamics and simplicity bias. Saddle-to-saddle dynamics (Jacot et al., 2021; Berthier, 2023; Abbe et al., 2023; Zhang et al., 2026) is a phenomenon observed across architectures (Maennel et al., 2018; Teney et al., 2022; Boix-Adsera et al., 2023), causing the learning to happen by gradually increasing the complexity of the solution. This gives rise to a particular type of simplicity bias (Arpit et al., 2017; Gidel et al., 2019; Rahaman et al., 2019; Hu et al., 2020). While complex tasks may ultimately require complex representations, Occam's razor remains a vital inductive principle for robust learning and generalization. Because of this, the question of simplicity bias in deep architectures has been broadly studied, both theoretically (Valle-Perez et al., 2019; Arora et al., 2019) and empirically (Kalimeris et al., 2019; Shah et al., 2020; Teney et al., 2022). Most of the theoretical papers used gradient descent for the optimizer (Kalimeris et al., 2019; Gidel et al., 2019; Refinetti et al., 2023). This motivated us to think about the biases of other optimizers, especially Muon, as we can show it avoids these saddles.

Understanding Muon. From the theory side, Su (2025) analyses Muon optimizer on isotropic curvature model. Shen et al. (2025) provide a convergence rate analysis of (simplified) Muon, in comparison with GD.

Most of the empirical papers focus on the benefits of Muon. In Vasudeva et al. (2026) they compare optimizers on imbalanced classes setting using MNIST-CIFAR dataset and Tiny Stories for language generation. Wang et al. (2026) show that in a transformer, MLP and OV-circuit benefit the most from Muon (versus Adam), and also, motivated by associative memory model, show that Muon outperforms Adam on a memorization task (QA dataset about biographical information) where each dataset entry appears with different frequency. They state '*Muon consistently yields more isotropic weight matrices with broadly distributed spectral energy than Adam, both throughout training and across random initializations, thereby supporting richer feature representations*'. In Section 4, we gave an example where 'richer' in their sense (i.e. learning everything there is to learn) is actually bad. Both of these works emphasize how Muon is better in the setting of imbalanced data, learning all modalities more evenly.

Vasudeva et al. (2026) also provides a comparison of the optimizers in a setting with spurious correlations, however, different than ours. In the data setting where 99%) of samples contains a dominant spurious feature, they show that while all optimizers learn the task perfectly, Muon learns it the fastest. They conclude: '*This suggests that Muon's spectral design promotes balanced learning of these components, leading to superior generalization.*'. While the premise is true, we showed that whether Muon generalizes better depends on the task—in the setting where we want to learn the more dominant features of the data, SGD wins.

Existing literature has largely focused on validating the efficiency benefits of Muon. An orthogonal example from Puli et al. (2023) suggests that faster learning can trade-off for the price of the solution relying more on spurious features. To complement these findings, we investigated specific scenarios where Muon's inductive biases lead to suboptimal performance. By exploring these trade-offs, we aim to foster a more comprehensive understanding of the optimizer and motivate future algorithmic improvements.

B THEORY ON DEEP LINEAR NETWORKS

Here we formalize the theory on training dynamics for deep linear networks for both GD and Spectral GD. The well-studied framework of deep linear networks (Fukumizu, 1998; Saxe et al., 2014; 2019; Lampinen & Ganguli, 2019; Ji & Telgarsky, 2019; Jacot et al., 2021; Ziyin et al., 2022) is particularly interesting for understanding the non-linear learning dynamics in deep models. Gradient flow results can be found in the prior work studying the simplicity bias of GD (Saxe et al., 2014; Kalimeris et al., 2019; Gidel et al., 2019; Jacot et al., 2021; Zhang et al., 2026). For Spectral GD, we largely focus on the results in Vasudeva et al. (2026). The theory explains why Spectral GD learns faster, but also reveals the cost of it: loss of the simplicity bias of GD.

Setup. Let a forward pass of a two-layer deep linear network be $\hat{y} = \boldsymbol{V}\boldsymbol{U}x$, where $\boldsymbol{V} \in \mathbb{R}^{d_{\text{out}} \times H}, \boldsymbol{U} \in \mathbb{R}^{H \times d_{\text{in}}}$, $x \in \mathbb{R}^{d_{\text{in}}}$, and H is the number of hidden neurons. Let the rows of $\boldsymbol{U}$ be $u_i^\top$ and the columns of $\boldsymbol{V}$ be v_i. We are given n input-output pairs $\{(x_i, y_i)\}_{i=1}^n$. The goal is to minimize the MSE loss $L(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2n}\sum_i \|\hat{y}_i - y_i\|^2$. Here we use full-batch gradient flow, starting from infinitesimal initializations $\boldsymbol{U}_0, \boldsymbol{V}_0$. Dynamics dependence on the dataset is fully captured by these statistics:

$$\Sigma_{xx} = \frac{1}{n}\sum_{i=1}^n x_i x_i^\top, \quad \Sigma_{yx} = \frac{1}{n}\sum_{i=1}^n y_i x_i^\top. \tag{2}$$

Assumption B.1. We assume $\Sigma_{xx} = \boldsymbol{I}$ and the joint diagonalizability of Σ_{yx}, $\boldsymbol{U}_0$ and $\boldsymbol{V}_0$.

The identity covariance assumption is required for solvable gradient flow dynamics (Fukumizu, 1998; Saxe et al., 2014). For relaxing $\Sigma_{xx} = \boldsymbol{I}$, we refer to Gidel et al. (2019); Advani et al. (2020); Vasudeva et al. (2026); Watanabe et al. (2026). The joint diagonalizability assumption is justified by the fact that under small initialization, the principal components of U, V do align with the ones of Σ_{yx} early on in the training, a phenomenon known as the silent alignment (Atanasov et al., 2022).

Gradient flow solution. The gradients of L with respect to $\boldsymbol{U}$ and $\boldsymbol{V}$ are given by

$$\nabla_{\boldsymbol{U}} L = \boldsymbol{V}^\top(\boldsymbol{V}\boldsymbol{U}\Sigma_{xx} - \Sigma_{yx}), \nabla_{\boldsymbol{V}} L = (\boldsymbol{V}\boldsymbol{U}\Sigma_{xx} - \Sigma_{yx})\boldsymbol{U}^\top. \tag{3}$$

Let $\Sigma_{yx} = \sum_{k=1}^D s_k q_k r_k^\top$ be the SVD of Σ_{yx} where s_k are positive singular values. Let them be ordered $s_1 \geq ... \geq s_D$.

Theorem B.2 (Gradient Flow Dynamics)**.** *Consider the gradient flow dynamics $\dot{\boldsymbol{U}} = -\nabla_{\boldsymbol{U}} L$ and $\dot{\boldsymbol{V}} = -\nabla_{\boldsymbol{V}} L$ on the loss $L(\boldsymbol{U}, \boldsymbol{V})$ starting from infinitesimal initialization. Then:*

1. *Fixed Points: Any point $(\boldsymbol{U}, \boldsymbol{V})$ such that $\boldsymbol{V}\boldsymbol{U} = \sum_{k=1}^r s_k q_k r_k^\top$ (for $r \leq D$) with $u_i \in$ span$\{r_k\}_{k=1}^r$, $v_i \in$ span$\{q_k\}_{k=1}^r$ is a fixed point of the dynamics.*
2. *Sequential Learning: The k-th singular value $\sigma_k(t)$ of the product matrix $\boldsymbol{V}\boldsymbol{U}$ evolves according to a sigmoid function, with a convergence time $\propto s_k^{-1}$. Consequently, for any k such that $s_k > s_{k+1}$, there exists a time interval when the k-th mode is fully learned ($\sigma_k \approx s_k$) while the $(k+1)$-th mode is negligible ($\sigma_{k+1} \approx 0$).*

The proof, although standard, can be found in Appendix B.1. Combining the two results leads to the explanation of saddle-to-saddle dynamics from small initialization. At the start of training, all singular values of the solution $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{U}$ are effectively 0. The rank of $\boldsymbol{W}$ then increases gradually, first learning the highest singular values s_k. At the point rank has gone up to r (for all $r \leq D$), $\boldsymbol{V}\boldsymbol{U}$ effectively satisfies the conditions of 1, and the learning trajectory passes close to the corresponding saddle point. This is reflected in a loss plateau, and 2. predicts the time of escaping this saddle. In case of a singular value s_k having multiplicity $m > 1$, the rank at that point will increase by m and the whole singular subspace will be learned at the same time.

Spectral gradient flow solution. Here we use the same notation and assumptions as in the previous case. For spectral gradient descent, the dynamics is different. This time, all principal components are learned simultaneously, with the same speed, until each one saturates.

Theorem B.3 (Spectral Gradient Flow Dynamics)**.** *Observe the spectral gradient flow $\dot{\boldsymbol{U}} = -\text{orth}(\nabla_{\boldsymbol{U}} L)$ and $\dot{\boldsymbol{V}} = -\text{orth}(\nabla_{\boldsymbol{V}} L)$ where $\text{orth}(\cdot)$ is the orthogonalization of a matrix. Starting from infinitesimal initialization, we have:*

1. *Solution Trajectory: The learning trajectory of $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{U}$ will sequentially pass through all $\boldsymbol{W}_r := \sum_{k=1}^{r} s_{r+1} q_k r_k^\top + \sum_{k=r+1}^{D} s_k q_k r_k^\top$ for $r \leq D$ (from higher to lower) such that $s_r > s_{r+1}$.*
2. *Equal Learning: To get from one $\boldsymbol{W}_r$ to the next, $\boldsymbol{W}$ learns singular spaces of Σ_{yx} in the same time, at the same rate, until saturation. Furthermore, the evolution of k-th singular value $\sigma_k(t)$ of $\boldsymbol{W}$ follows a quadratic curve, leading to convergence time $\propto \sqrt{s_k}$.*

We provide the proof in Appendix B.2, or refer to Vasudeva et al. (2026), with slightly different assumptions. Consequently, training with Spectral GD also happens in phases: in each phase, singular values $\sigma_{r+1}(t) = s_{r+1}, ..., \sigma_D(t) = s_D$ are fully learned, while $\sigma_1(t), ..., \sigma_r(t)$ make their transition from s_{r+1} to s_r, the next smallest singular value. The contrast with GD is the order of learning the singular values: while GD fully learns the higher ones first, Spectral GD learns all of them in the same time, leading to smaller ones being fully learned first.

Experiments. We both validate and illustrate the theory in Figure 1, where we perform full dataset (Spectral) GD on standard Gaussian data $x_i \in \mathbb{R}^{d_{\text{in}}}$, and $y_i \in \mathbb{R}^{d_{\text{out}}}$ computed as a noised linear function of x_i. In this usual linear regression setting, we sample the regression weights also as a standard normal. The weights $\boldsymbol{U}, \boldsymbol{V}$ are initialized also as Gaussian, but with small variance (0.01). We see that the evolution closely follows the theory.

Empirically, because we're performing a discretization of the dynamics of 'Spectral gradient flow', and due to using a spectral method, it does happen that after the last $D - r$ principal components have been learned, the update matrix in the dynamics above is not exactly rank r but also have some noise, and is usually full rank. In that case, Spectral GD does fit this noise, causing a highly oscillatory path in the loss landscape (see Figure 7), and the oscillations don't go away even if we include the momentum, as per Muon optimization (Figure 8). Although in a simple setting of deep linear networks the model always converges to the correct solution (as it is a stable minimum), this suggest that in more complicated settings Spectral GD and Muon may be less robust when using higher learning rate. Potentially, this observation could motivate future improvements of spectral optimizers.

B.1 Proof of Theorem B.2

Theorem. *Consider the gradient flow dynamics $\dot{\boldsymbol{U}} = -\nabla_{\boldsymbol{U}} L$ and $\dot{\boldsymbol{V}} = -\nabla_{\boldsymbol{V}} L$ on the loss $L(\boldsymbol{U}, \boldsymbol{V})$ starting from infinitesimal initialization. Then:*

1. *Fixed Points: Any point $(\boldsymbol{U}, \boldsymbol{V})$ such that $\boldsymbol{V}\boldsymbol{U} = \sum_{k=1}^{r} s_k q_k r_k^\top$ (for $r \leq D$) with $u_i \in \text{span}\{r_k\}_{k=1}^{r}$, $v_i \in \text{span}\{q_k\}_{k=1}^{r}$ is a fixed point of the dynamics.*
2. *Sequential Learning: The k-th singular value $\sigma_k(t)$ of the product matrix $\boldsymbol{V}\boldsymbol{U}$ evolves according to a sigmoid function, with a convergence time $\propto s_k^{-1}$. Consequently, for any k such that $s_k > s_{k+1}$, there exists a time interval when the k-th mode is fully learned ($\sigma_k \approx s_k$) while the $(k+1)$-th mode is negligible ($\sigma_{k+1} \approx 0$).*

Proof. **Fixed Points.** The gradients for the deep linear network under $\Sigma_{xx} = \boldsymbol{I}$ are:

$$\nabla_{\boldsymbol{U}} L = \boldsymbol{V}^\top(\boldsymbol{V}\boldsymbol{U} - \Sigma_{yx}),$$
$$\nabla_{\boldsymbol{V}} L = (\boldsymbol{V}\boldsymbol{U} - \Sigma_{yx})\boldsymbol{U}^\top.$$

Let the candidate solution be $\boldsymbol{W}_r = \boldsymbol{V}\boldsymbol{U} = \sum_{k=1}^{r} s_k q_k r_k^\top$. Substituting this into the residual term $(\boldsymbol{V}\boldsymbol{U} - \Sigma_{yx})$ yields:

$$\boldsymbol{W}_r - \Sigma_{yx} = \sum_{k=1}^{r} s_k q_k r_k^\top - \sum_{k=1}^{D} s_k q_k r_k^\top = -\sum_{k=r+1}^{D} s_k q_k r_k^\top.$$

By the theorem assumption, $\text{row}(U) \in \text{span}\{r_k\}_{k=1}^r$ and $\text{col}(V) \in \text{span}\{q_k\}_{k=1}^r$, hence:

$$\nabla_{\boldsymbol{U}} L = -\boldsymbol{V}^\top \left(\sum_{k=r+1}^{D} s_k q_k r_k^\top \right) = 0,$$

$$\nabla_{\boldsymbol{V}} L = -\left(\sum_{k=r+1}^{D} s_k q_k r_k^\top \right) \boldsymbol{U}^\top = 0.$$

Thus, $(\boldsymbol{U}, \boldsymbol{V})$ is a fixed point.

Sequential Learning. This is a standard result found in many deep linear networks work (Saxe et al., 2014; Gidel et al., 2019; Li et al., 2021; Jacot et al., 2021; Zhang et al., 2026). For completeness, we provide a proof here. We analyze the evolution of the product matrix $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{U}$. We assume the standard "balanced" condition $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{U}\boldsymbol{U}^\top$, which is an invariant/conservation law of gradient flow if initialized infinitesimally, as we assume. The dynamics of the product matrix $\boldsymbol{W}$ are given by:

$$\dot{\boldsymbol{W}} = \dot{\boldsymbol{V}}\boldsymbol{U} + \boldsymbol{V}\dot{\boldsymbol{U}} = (\Sigma_{yx} - \boldsymbol{W})\boldsymbol{U}^\top \boldsymbol{U} + \boldsymbol{V}\boldsymbol{V}^\top (\Sigma_{yx} - \boldsymbol{W}).$$

Under the balanced condition, we have $\boldsymbol{U}^\top \boldsymbol{U} = (\boldsymbol{W}^\top \boldsymbol{W})^{1/2}$ and $\boldsymbol{V}\boldsymbol{V}^\top = (\boldsymbol{W}\boldsymbol{W}^\top)^{1/2}$. Substituting this into the dynamics yields:

$$\dot{\boldsymbol{W}} = -(\boldsymbol{W} - \Sigma_{yx})(\boldsymbol{W}^\top \boldsymbol{W})^{1/2} - (\boldsymbol{W}\boldsymbol{W}^\top)^{1/2}(\boldsymbol{W} - \Sigma_{yx}).$$

The fact that $\boldsymbol{U}(0), \boldsymbol{V}(0)$ and Σ_{yx} are jointly diagonalizable implies that $\boldsymbol{W}_0 = \boldsymbol{V}(0)\boldsymbol{U}(0)$ and Σ_{yx} are as well. If $\Sigma_{yx} = QSR^T$ is an SVD decomposition, then SVDs of matrices involved are: $\boldsymbol{W}_0 \boldsymbol{W}_0^\top = QD_1Q^T$, $\boldsymbol{W}_0^\top \boldsymbol{W}_0 = RD_2R^\top$ and $(\Sigma_{yx} - \boldsymbol{W}_0) = QD_3R^\top$. As taking the square root doesn't change left and right principal components, we infer that the full gradient update in Equation B.1 is also aligned with the principal components of Σ_{yx}. Hence the alignment of principal components of $\boldsymbol{W}$ and Σ_{yx} from the beginning stays true throughout training.

Let $\boldsymbol{W}(t) = \sum_{k=1}^{D} \sigma_k(t) q_k r_k^\top$. Substituting this spectral decomposition into the matrix ODE decouples the system into independent scalar differential equations for each singular value $\sigma_k(t)$:

$$\dot{\sigma}_k(t) = -(\sigma_k - s_k)\sigma_k - \sigma_k(\sigma_k - s_k) = 2\sigma_k(t)(s_k - \sigma_k(t)).$$

This is the logistic differential equation. The solution is:

$$\sigma_k(t) = \frac{s_k}{1 + \left(\frac{s_k}{\sigma_k(0)} - 1\right) e^{-2s_k t}}.$$

This solution shows that $\sigma_k(t)$ follows a sigmoidal trajectory, saturating to s_k. Consequently, the time taken for s_k to be learned is of order $1/s_k$, implying that modes with larger singular values are learned significantly faster, establishing the sequential learning property. □

B.2 Proof of Theorem B.3

Theorem. *Observe the spectral gradient flow $\dot{\boldsymbol{U}} = -\text{orth}(\nabla_{\boldsymbol{U}} L)$ and $\dot{\boldsymbol{V}} = -\text{orth}(\nabla_{\boldsymbol{V}} L)$ where $\text{orth}(\cdot)$ is the orthogonalization of a matrix. Starting from infinitesimal initialization, we have:*

1. *Solution Trajectory: The learning trajectory of $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{U}$ will sequentially pass through all $\boldsymbol{W}_r := \sum_{k=1}^{r} s_{r+1} q_k r_k^\top + \sum_{k=r+1}^{D} s_k q_k r_k^\top$ for $r \leq D$ (from higher to lower) such that $s_r > s_{r+1}$.*
2. *Equal Learning: To get from one $\boldsymbol{W}_r$ to the next, $\boldsymbol{W}$ learns singular spaces of Σ_{yx} in the same time, at the same rate, until saturation. Furthermore, the evolution of k-th singular value $\sigma_k(t)$ of $\boldsymbol{W}$ follows a quadratic curve, leading to convergence time $\propto \sqrt{s_k}$.*

Proof. **Solution Trajectory.** The proof, with slightly different assumptions, can be found in Vasudeva et al. (2026). Nonetheless, we follow our setup and provide it here for completeness.

We assumed joint diagonalizibility of $U(0) = \sum_{k=1}^{D} \sigma_k^U(0) z_k r_k^\top$, $V(0) = \sum_{k=1}^{D} \sigma_k^V(0) q_k z_k^\top$ and $\Sigma_{yx} = \sum_{k=1}^{D} s_k q_k r_k^\top$, which is justified by the infinitesimal initialization. As we will see, once

$U(t), V(t)$ are jointly diagonalizable in this way, they stay so for all $t' \geq t$). Suppose for some $r \leq D, t_0 \geq 0$, $W(t_0) = \sum_{k=1}^{r} s_{r+1} q_k r_k^\top + \sum_{k=r+1}^{D} s_k q_k r_k^\top$. Indeed, this is satisfied at initialization $t_0 = 0$, with $r = D$, once we denote $s_{D+1} = 0$. If it holds at t_0, then

$$\begin{aligned}
\nabla_{\boldsymbol{U}} L(\boldsymbol{U}(t_0)) &= \boldsymbol{V}(t_0)^\top (\boldsymbol{W}(t) - \Sigma_{yx}) \\
&= \left(\sum_{k=1}^{D} \sigma_k^V(t_0) q_k z_k^\top \right)^\top \left(\sum_{k=1}^{r} (\sigma_{r+1} - \sigma_k) q_k r_k^\top \right) \\
&= \sum_{k=1}^{r} \sigma_k^V(t_0)(\sigma_{r+1} - \sigma_k) z_k r_k^\top \\
\nabla_{\boldsymbol{V}} L(\boldsymbol{V}(t_0)) &= (\boldsymbol{W}(t_0) - \Sigma_{yx}) \boldsymbol{U}(t_0)^\top \\
&= \left(\sum_{k=1}^{r} (\sigma_{r+1} - \sigma_k) q_k r_k^\top \right) \left(\sum_{k=1}^{D} \sigma_k^U(t_0) z_k r_k^\top \right)^\top \\
&= \sum_{k=1}^{r} \sigma_k^U(t_0)(\sigma_{r+1} - \sigma_k) q_k z_k^\top
\end{aligned}$$

From this, the orthogonalizations of the gradients are $\text{orth}(\nabla_{\boldsymbol{U}} L(t_0)) = -\sum_{k=1}^{r} z_k r_k^\top$, $\text{orth}(\nabla_{\boldsymbol{V}} L(t_0)) = -\sum_{k=1}^{r} q_k z_k^\top$. This means that change in $\boldsymbol{U}(t), \boldsymbol{V}(t)$ happens only in the first r singular vectors, and the dynamics will look as a gradient flow $\dot{\boldsymbol{U}}(t) = \sum_{i=1}^{r} z_k r_k^\top$, $\dot{\boldsymbol{V}}(t) = \sum_{k=1}^{r} q_k z_k^\top$. This keeps the matrices jointly diagonalizable, and therefore we can observe the dynamics on the singular values, from initial conditions at t_0:

$$\begin{aligned}
\dot{\sigma_k^V}(t) &= \mathbb{1}_{\{\sigma_k^V(t)\sigma_k^U(t) < s_k\}} \\
\dot{\sigma_k^U}(t) &= \mathbb{1}_{\{\sigma_k^V(t)\sigma_k^U(t) < s_k\}},
\end{aligned}$$

noting that $\sigma_k(t) = \sigma_k^V(t)\sigma_k^U(t)$. By symmetry at infinitesimal initialization, all $\sigma_k^V(t)$ and $\sigma_k^U(t)$ will be the same for $k \leq r$, and hence for all $k \leq r$, $\sigma_k(t)$ will evolve the same, from $\sigma_k(t_0) = s_{r+1}$ to $\sigma_k(t_1) = s_r$. At time t_1, for all $m < k \leq r$ with m largest s.t. $s_{m+1} = s_r$, the growth of $\sigma_k(t)$ stops. Hence the solution W passes through W_m.

Equal learning. Continuing with the notation above, if we start from $\sigma_k^V(0) = \sigma_k^U(0) = 0$, the solution of the system is $\sigma_k^U(t) = \sigma_k^V(t) = \min(t, \sqrt{s_k})$, and hence $\sigma_k(t) = \min(t^2, s_k)$. This shows both the equal growth and the quadratic curve for learning singular values of Σ_{yx}. Consequently, the time to learn $\sigma_k(t) = s_k$ is $t \propto \sqrt{s_k}$.

□

C Theory on Linear Attention

Here we formalize the theory on linear attention from Section 5. Under the setup described in Section 5, Zhang et al. (2025) showed the training dynamics for each head $i = 1, \cdots, H$ is given by

$$\begin{aligned}
\frac{dv_i}{dt} &= \boldsymbol{k}_i^\top \left(\boldsymbol{\Lambda}^2 - \mathbb{E}\left(\hat{\boldsymbol{\Lambda}}^2\right) \sum_{i'=1}^{H} v_{i'} \boldsymbol{k}_{i'} \boldsymbol{q}_{i'}^\top \boldsymbol{\Lambda} \right) \boldsymbol{q}_i, \\
\frac{d\boldsymbol{k}_i}{dt} &= v_i \left(\boldsymbol{\Lambda}^2 - \mathbb{E}\left(\hat{\boldsymbol{\Lambda}}^2\right) \sum_{i'=1}^{H} v_{i'} \boldsymbol{k}_{i'} \boldsymbol{q}_{i'}^\top \boldsymbol{\Lambda} \right) \boldsymbol{q}_i, \\
\frac{d\boldsymbol{q}_i}{dt} &= v_i \left(\boldsymbol{\Lambda}^2 - \boldsymbol{\Lambda} \sum_{i'=1}^{H} v_{i'} \boldsymbol{q}_{i'} \boldsymbol{k}_{i'}^\top \mathbb{E}\left(\hat{\boldsymbol{\Lambda}}^2\right) \right) \boldsymbol{k}_i.
\end{aligned}$$

where $\hat{\boldsymbol{\Lambda}}$ is the in-context covariance of $\boldsymbol{x}$ tokens and the expectation of $\hat{\boldsymbol{\Lambda}}^2$ is given by

$$\mathbb{E}\left(\hat{\boldsymbol{\Lambda}}^2\right) = \boldsymbol{\Lambda}^2 + \frac{\boldsymbol{\Lambda} + \text{tr}(\boldsymbol{\Lambda})\boldsymbol{I}}{N} \boldsymbol{\Lambda}.$$

The reason we are able to write the dynamics compactly is that the loss is only computed for a scalar entry in the output sequence $\hat{y}_q$. This is made possible in the in-context linear regression setting, as value matrix of the $\boldsymbol{x}$-entry is irrelevant for our prediction, and we only care about the y-entry v_i, a scalar that commutes with other data statistics matrices.

As mentioned in the main text, we collect the relevant key and query vectors for each head $\boldsymbol{k}_i, \boldsymbol{q}_i \in \mathbb{R}^D$ in matrices $\boldsymbol{K}, \boldsymbol{Q} \in \mathbb{R}^{H\times D}$. Value scalars of each head v_i are collected in the diagonal matrix $\boldsymbol{V} \in \mathbb{R}^{H\times H}$. Using this notation allows us to rewrite the dynamics in matrix form:

$$\begin{aligned}
\frac{d\boldsymbol{V}}{dt} &= \texttt{diag}\left(\boldsymbol{K}\left(\boldsymbol{\Lambda}^2 - \mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]\boldsymbol{K}^\top\boldsymbol{V}\boldsymbol{Q}\boldsymbol{\Lambda}\right)\boldsymbol{Q}^\top\right),\\
\frac{d\boldsymbol{K}}{dt} &= \boldsymbol{V}\boldsymbol{Q}\left(\boldsymbol{\Lambda}^2 - \boldsymbol{\Lambda}\boldsymbol{Q}^\top\boldsymbol{V}\boldsymbol{K}\mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]\right),\\
\frac{d\boldsymbol{Q}}{dt} &= \boldsymbol{V}\boldsymbol{K}\left(\boldsymbol{\Lambda}^2 - \mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]\boldsymbol{K}^\top\boldsymbol{V}\boldsymbol{Q}\boldsymbol{\Lambda}\right),
\end{aligned}$$

where $\texttt{diag}(\mathbf{A})$ denotes a diagonal matrix with entries on the diagonal being the same as the ones in $\mathbf{A}$. Writing the dynamics like this doesn't change the the trajectory of gradient flow solutions compared to using the full matrices $\boldsymbol{W}_K, \boldsymbol{W}_Q \in \mathbb{R}^{H\times(D+1)}$ and $\boldsymbol{W}_V \in \mathbb{R}^{H\times(D+1)\times(D+1)}$. However, in case of Spectral GF, the reparameterization from $\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V$ to $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ does change the dynamics. Nonetheless, we use the reparameterization as it makes the Spectral GF solution mathematically tractable. The key phenomena of all heads being learned at the same time obtained from the theory is still present without the reparameterization, as shown in Figure 13 in Appendix D.5.

We denote the eigenvectors of $\boldsymbol{\Lambda}$ as $\{\boldsymbol{g}_j\}_{j=1}^D$ and the corresponding eigenvalues as $\{\lambda_j\}_{j=1}^D$

$$\boldsymbol{\Lambda} = \sum_{j=1}^{D} \lambda_j \boldsymbol{g}_j \boldsymbol{g}_j^\top.$$

The matrix $\mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right]$ has the same eigenvectors as $\boldsymbol{\Lambda}$ and eigenvalues $\{\hat{\lambda}_j\}_{j=1}^D$

$$\mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right] = \sum_{j=1}^{D} \hat{\lambda}_j \boldsymbol{g}_j \boldsymbol{g}_j^\top, \quad \text{where } \hat{\lambda}_j = \lambda_j^2\left(1 + \frac{1 + \text{tr}(\boldsymbol{\Lambda})/\lambda_j}{N}\right). \tag{6}$$

Assumption C.1. For both GF and Spectral GF, we assume infinitesimal initialization and joint diagonalizablity on the initialization of $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ and $\boldsymbol{\Lambda}$. That is

$$\boldsymbol{V}(0) = \sum_{i=1}^{H} v_i(0)\boldsymbol{e}_i\boldsymbol{e}_i^\top, \quad \boldsymbol{K}(0) = \sum_{j=1}^{D} \sigma_j^K(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top, \quad \boldsymbol{Q}(0) = \sum_{j=1}^{D} \sigma_j^Q(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top, \tag{7}$$

with $v_i(0) = \sigma_i^K(0) = \sigma_i^Q(0) = \epsilon_i$ for all $i = 1, ..., H$, where ϵ_i is infinitesimal.

In the case of gradient flow, Zhang et al. (2025) explained with their ansatz that from arbitrary but small initialization, the head vectors $\boldsymbol{k}_i, \boldsymbol{q}_i$ will align with the eigenvectors $\boldsymbol{g}_i$, hinting that the assumption may be redundant. Intuitively, this is because with gradient flow, the growth of $\boldsymbol{k}_i$ along each of eigenvector $\boldsymbol{g}_i$ will approximately follow the dynamics $\frac{d(\boldsymbol{k}_i^\top \boldsymbol{g}_i)}{dt} = \lambda_i^2(\boldsymbol{k}_i^\top \boldsymbol{g}_i)^2$, exhibiting the fastest grow of $\boldsymbol{k}_i$ along the eigenvector associated with largest eigenvalue λ_i (and similarly for $\boldsymbol{q}_i$). In case of Spectral GD, this doesn't hold, as growth along each eigenvalue is better modeled by $\frac{d(\boldsymbol{k}_i^\top \boldsymbol{g}_i)}{dt} = \mathbb{1}_{\boldsymbol{k}_i^\top \boldsymbol{g}_i < (\lambda_i)^{-1/3}}$. This makes the growth along each $\boldsymbol{g}_i$ equal, and thus silent alignment is not happening with Spectral GD. If one was to fully formally analyze the dynamics of Spectral GF without the above assumption, decomposing each $\boldsymbol{k}_i, \boldsymbol{q}_i$ along all of $\boldsymbol{g}_i$ may be crucial. We leave this formality for future work, and show the simulations without this assumption in Figures 4a, 11 and 12.

Under this assumption at initialization, following from the dynamics in Equation 5, the matrices $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ and $\boldsymbol{\Lambda}$ stay jointly diagonalizable through training, and the $\texttt{diag}$ operator in Equation 5a

can be removed. Indeed, if we observe the dynamics at initialization, we have:

$$\begin{aligned}
\boldsymbol{C}(0) &:= \boldsymbol{\Lambda}^2 - \boldsymbol{\Lambda}\boldsymbol{Q}^\top(0)\boldsymbol{V}(0)\boldsymbol{K}(0)\mathbb{E}\left[\hat{\boldsymbol{\Lambda}}^2\right] = \left(\sum_{j=1}^{D}\lambda_j^2\boldsymbol{g}_j\boldsymbol{g}_j^\top\right) \\
&- \left(\sum_{j=1}^{D}\hat{\lambda}_j\boldsymbol{g}_j\boldsymbol{g}_j^\top\right)\left(\sum_{j=1}^{D}\sigma_j^K(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right)^\top\left(\sum_{i=1}^{H}v_i(0)\boldsymbol{e}_i\boldsymbol{e}_i^\top\right)\left(\sum_{j=1}^{D}\sigma_j^Q(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right)\left(\sum_{j=1}^{D}\lambda_j\boldsymbol{g}_j\boldsymbol{g}_j^\top\right) \\
&= \sum_{j=1}^{D}\lambda_j^2\boldsymbol{g}_j\boldsymbol{g}_j^\top - \sum_{j=1}^{D}\hat{\lambda}_j\sigma_j^K(0)v_j(0)\sigma_j^Q(0)\lambda_j\boldsymbol{g}_j\boldsymbol{g}_j^\top \\
&= \sum_{j=1}^{D}\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(0)\sigma_j^K(0)\sigma_j^Q(0)\right)\boldsymbol{g}_j\boldsymbol{g}_j^\top.
\end{aligned}$$

$$\begin{aligned}
\frac{d\boldsymbol{V}(0)}{dt} &= \texttt{diag}\left(\left(\sum_{j=1}^{D}\sigma_j^K(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right)\boldsymbol{C}(0)\left(\sum_{j=1}^{D}\sigma_j^Q(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right)^\top\right) \\
&= \sum_{j=1}^{H}\sigma_j^K(0)\sigma_j^Q(0)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(0)\sigma_j^K(0)\sigma_j^Q(0)\right)\boldsymbol{e}_j\boldsymbol{e}_j^\top,
\end{aligned}$$

$$\begin{aligned}
\frac{d\boldsymbol{K}(0)}{dt} &= \left(\sum_{i=1}^{H}v_i(0)\boldsymbol{e}_i\boldsymbol{e}_i^\top\right)\left(\sum_{j=1}^{D}\sigma_j^Q(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right)\boldsymbol{C}(0) \\
&= \sum_{j=1}^{D}v_j(0)\sigma_j^Q(0)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(0)\sigma_j^K(0)\sigma_j^Q(0)\right)\boldsymbol{e}_j\boldsymbol{g}_j^\top,
\end{aligned}$$

$$\begin{aligned}
\frac{d\boldsymbol{Q}(0)}{dt} &= \left(\sum_{i=1}^{H}v_i(0)\boldsymbol{e}_i\boldsymbol{e}_i^\top\right)\left(\sum_{j=1}^{D}\sigma_j^K(0)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right)\boldsymbol{C}(0) \\
&= \sum_{j=1}^{D}v_j(0)\sigma_j^K(0)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(0)\sigma_j^K(0)\sigma_j^Q(0)\right)\boldsymbol{e}_j\boldsymbol{g}_j^\top.
\end{aligned}$$

We observe that the dynamics completely decouple along the principal components at initialization ($t = 0$). Consequently, this decoupled structure is preserved for all time $t \geq 0$, regardless of whether standard or spectral gradient flow is employed. Let the solutions at time t be

$$\boldsymbol{V}(t) = \sum_{i=1}^{H}v_i(t)\boldsymbol{e}_i\boldsymbol{e}_i^\top, \quad \boldsymbol{K}(t) = \sum_{j=1}^{D}\sigma_j^K(t)\boldsymbol{e}_j\boldsymbol{g}_j^\top, \quad \boldsymbol{Q}(t) = \sum_{j=1}^{D}\sigma_j^Q(t)\boldsymbol{e}_j\boldsymbol{g}_j^\top. \tag{8}$$

C.1 Gradient Flow.

Theorem C.2 (Gradient flow on linear attention)**.** *Under Assumption C.1, when optimized by gradient flow, the total weights of linear attention evolve as*

$$\boldsymbol{Q}^\top\boldsymbol{V}\boldsymbol{K} = \sum_{j=1}^{D}v_j(t)^3\boldsymbol{g}_j\boldsymbol{g}_j^\top,$$

where the magnitude in each eigen-direction v_j ($j = 1, \cdots, D$) is given by

$$\frac{dv_j}{dt} = v_j^2\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j^3\right).$$

This gives rise to time separation between heads being learned, where head j is learned in time $\propto 1/(\lambda_j^2 \epsilon_j)$.

Proof. Substituting Equation (8) into Equation (5), we obtain

$$\frac{d}{dt}\sum_{i=1}^{H} v_i(t)\boldsymbol{e}_i\boldsymbol{e}_i^\top = \sum_{j=1}^{D} \sigma_j^K(t)\sigma_j^Q(t)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(t)\sigma_j^K(t)\sigma_j^Q(t)\right)\boldsymbol{e}_j\boldsymbol{e}_j^\top,$$
$$\frac{d}{dt}\sum_{j=1}^{D} \sigma_j^K(t)\boldsymbol{e}_j\boldsymbol{g}_j^\top = \sum_{j=1}^{D} v_j(t)\sigma_j^Q(t)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(t)\sigma_j^K(t)\sigma_j^Q(t)\right)\boldsymbol{e}_j\boldsymbol{g}_j^\top,$$
$$\frac{d}{dt}\sum_{j=1}^{D} \sigma_j^Q(t)\boldsymbol{e}_j\boldsymbol{g}_j^\top = \sum_{j=1}^{D} v_j(t)\sigma_j^K(t)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(t)\sigma_j^K(t)\sigma_j^Q(t)\right)\boldsymbol{e}_j\boldsymbol{g}_j^\top.$$

Equating both sides, we get for $j = 1, \cdots, D$

$$\frac{dv_j}{dt} = \sigma_j^K\sigma_j^Q\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j\sigma_j^K\sigma_j^Q\right),$$
$$\frac{d\sigma_j^K}{dt} = v_j\sigma_j^Q\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j\sigma_j^K\sigma_j^Q\right),$$
$$\frac{d\sigma_j^Q}{dt} = v_j\sigma_j^K\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j\sigma_j^K\sigma_j^Q\right).$$

Because $v_j, \sigma_j^K, \sigma_j^Q$ are initialized to be equal and will stay equal throughout training, the dynamics for each eigen-direction can be summarized by one separable, ordinary differential equation

$$\frac{dv_j}{dt} = v_j^2\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j^3\right), \quad j = 1, \cdots, D. \tag{10}$$

Equation (10) does not admit a closed-form solution of $v_j(t)$ in terms of t. However, its behavior is easy to predict from Equation (10). With small initialization $v_j(0) = \epsilon_j$, $v_j(t)$ undergoes a plateau of duration $T_j \approx 1/(\lambda_j^2\epsilon_j)$. This is because the initial dynamics behaves like $\dot{v}_j = v_j^2\lambda_j^2$, until $v_j^2\lambda_j^2 = \Theta(1)$. After the plateau, $v_j(t)$ grows rapidly and converges to $v_j(\infty) = \left(\frac{\lambda_j}{\hat{\lambda}_j}\right)^{1/3}$.

The overall weight thus evolves as

$$\boldsymbol{Q}^\top\boldsymbol{V}\boldsymbol{K} = \sum_{j=1}^{D} v_j(t)^3\boldsymbol{g}_j\boldsymbol{g}_j^\top,$$

where $v_j(t)$ solves the differential Equation (10). Since each eigen-direction grows after a plateau of duration $T_j \approx 1/(\lambda_j^2\epsilon_j)$, distinct eigenvalues λ_j or distinct initial conditions ϵ_j in each head will lead to stage-like dynamics, in which different eigen-directions are learned at different times. □

C.2 Spectral Gradient Flow.

Theorem C.3 (Spectral gradient flow on linear attention)**.** *Under Assumption C.1, when optimized by Spectral gradient flow, the total weights of linear attention evolve as*

$$\boldsymbol{Q}^\top\boldsymbol{V}\boldsymbol{K} = \sum_{j=1}^{D} \min\left[t^3, \frac{\lambda_j}{\hat{\lambda}_j}\right]\boldsymbol{g}_j\boldsymbol{g}_j^\top,$$

and the magnitude in each eigen-direction v_j ($j = 1, \cdots, D$) is given by

$$v_j(t) = \min\left[t, \left(\frac{\lambda_j}{\hat{\lambda}_j}\right)^{1/3}\right]$$

Contrary to time separation in the GF case, Spectral GF learns all heads at the same time, with the same rate, where head j is learned in time $\propto \left(\frac{\lambda_j}{\hat{\lambda}_j}\right)^{1/3}$.

Proof. Similar to the gradient flow case, we observe the dynamics in SVD decomposed form, but with orthogonalization, denoted by `orth`:

$$\frac{d}{dt}\sum_{i=1}^{H} v_i(t)\boldsymbol{e}_i\boldsymbol{e}_i^\top = \texttt{orth}\left(\sum_{j=1}^{D}\sigma_j^K(t)\sigma_j^Q(t)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(t)\sigma_j^K(t)\sigma_j^Q(t)\right)\boldsymbol{e}_j\boldsymbol{e}_j^\top\right),$$

$$\frac{d}{dt}\sum_{j=1}^{D}\sigma_j^K(t)\boldsymbol{e}_j\boldsymbol{g}_j^\top = \texttt{orth}\left(\sum_{j=1}^{D} v_j(t)\sigma_j^Q(t)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(t)\sigma_j^K(t)\sigma_j^Q(t)\right)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right),$$

$$\frac{d}{dt}\sum_{j=1}^{D}\sigma_j^Q(t)\boldsymbol{e}_j\boldsymbol{g}_j^\top = \texttt{orth}\left(\sum_{j=1}^{D} v_j(t)\sigma_j^K(t)\left(\lambda_j^2 - \lambda_j\hat{\lambda}_j v_j(t)\sigma_j^K(t)\sigma_j^Q(t)\right)\boldsymbol{e}_j\boldsymbol{g}_j^\top\right).$$

Dynamics again decouples, and for each head $j = 1, \cdots, D$ we have:

$$\frac{dv_j}{dt} = \mathbb{1}_{\{v_j\sigma_j^K\sigma_j^Q < \lambda_j/\hat{\lambda}_j\}},$$
$$\frac{d\sigma_j^K}{dt} = \mathbb{1}_{\{v_j\sigma_j^K\sigma_j^Q < \lambda_j/\hat{\lambda}_j\}},$$
$$\frac{d\sigma_j^Q}{dt} = \mathbb{1}_{\{v_j\sigma_j^K\sigma_j^Q < \lambda_j/\hat{\lambda}_j\}}.$$

Thus, using that $v_j(0) = \sigma_j^K(0) = \sigma_j^Q(0) = \epsilon_j$, their evolution is characterized by

$$\frac{dv_j}{dt} = \mathbb{1}_{\{v_j(t)^3 < \lambda_j/\hat{\lambda}_j\}}, \quad j = 1, \cdots, D.$$

This exactly results in

$$v_j(t) = \min\left[t, \left(\frac{\lambda_j}{\hat{\lambda}_j}\right)^{1/3}\right],$$

hence the head j is learned in time $T_j \propto \left(\frac{\lambda_j}{\hat{\lambda}_j}\right)^{1/3}$, and the full weights dynamics are

$$\boldsymbol{Q}^\top\boldsymbol{V}\boldsymbol{K} = \sum_{j=1}^{D}\min\left[t^3, \frac{\lambda_j}{\hat{\lambda}_j}\right]\boldsymbol{g}_j\boldsymbol{g}_j^\top.$$

When $N \to \infty$, $\hat{\lambda}_j \to \lambda_j^2$ as can be seen from Equation 6, and thus the time for learning the head j in this regime is $T_j \propto \lambda^{-1/3}$.

In the case of Spectral GF, all eigen-directions are being learned at the same time and with the same speed, and no stage-like dynamics is present. Notice also that, while the convergence time in the case of GF depends on the initial conditions ϵ_j, the dependence is no longer present in Spectral GF. □

D Further Experiments and Details

We note that all the experiments in the paper are small scale, they were ran on a single L40 GPU, requiring not more than 8GB of RAM. The experiments take minutes to finish individually.

D.1 Deep Linear Networks

More analogous experiments to the ones in Figure 1 can be found in Figure 6.

Additionally, we show the oscillations of the singular values of $\boldsymbol{W}$ during training in Figures 7,8, happening as a consequence of our discussion in Appendix B.

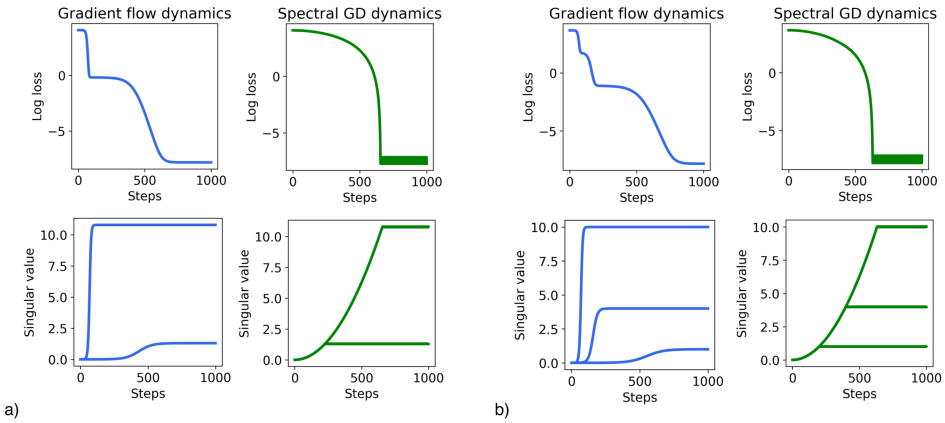

Figure 6: Additional figures supporting the theory from Section 2. a) $d_{\text{in}} = d_{\text{out}} = 2$; b) $d_{\text{in}} = d_{\text{out}} = 3$

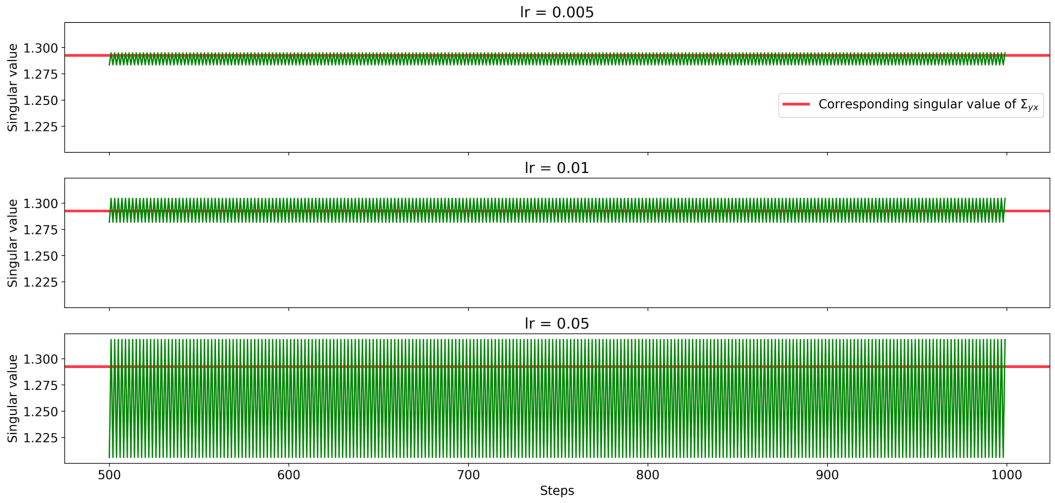

Figure 7: The oscillations of $\sigma_2(t)$ around s_2 in the setting from Figure 1, shown for different values of the learning rate. This phenomena happens after a principal component is effectively learned by Spectral GD, but not exactly. Then the small noise in the direction of that principal component is amplified by orthogonalization, and the step of order 1 is taken, independently of noise magnitude.

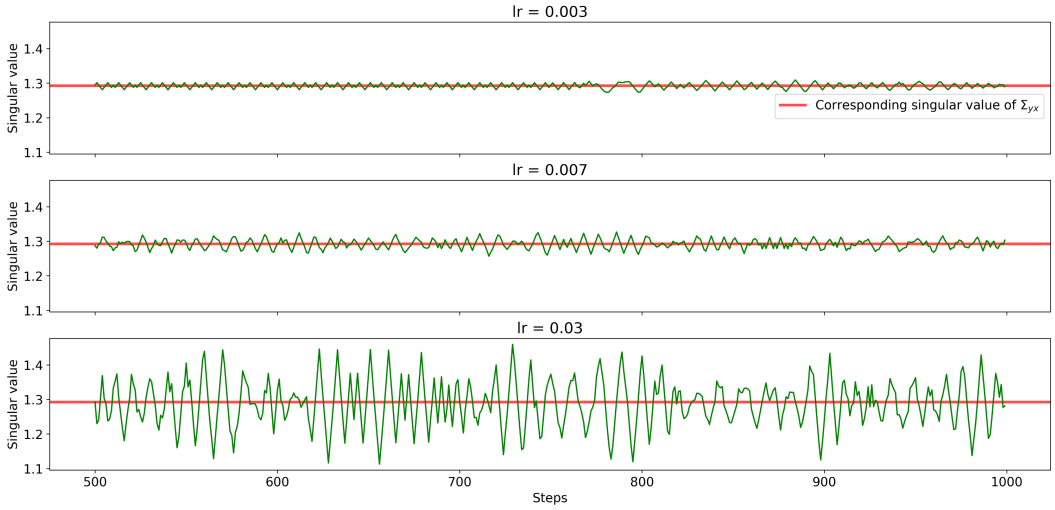

Figure 8: Same setup as Figure 7, but this time we're using Spectral GD with momentum, a step closer to recovering Muon optimization.

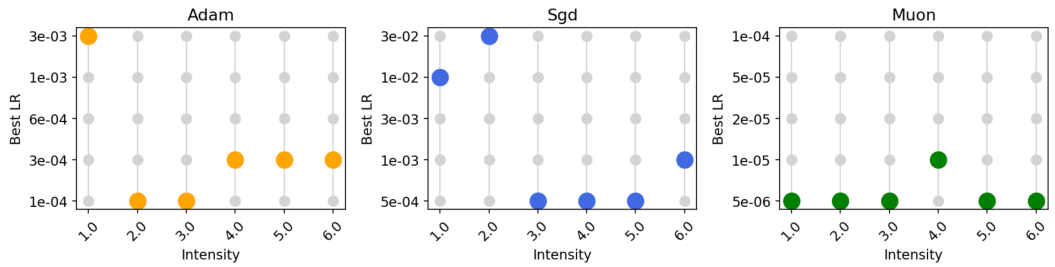

Figure 9: Learning rates used for experiments with pixel and digit feature in Section 3.

D.2 Digit and Pixel Feature Learning

Here we provide more implementation details on the results in Figures 2 and 16. For all three optimizers we picked the best performing learning rates across multiple runs. The learning rates sweep results are shown in Figure 9. We used Muon implementation `SingleDeviceMuonWithAuxAdam` from Jordan et al. (2024b), with the base learning rate applied to Adam optimized parameters, and for the Muon ones the learning rate was $80\times$ larger. Muon optimized parameters had weight decay set to 0.1, and momentum for both SGD and Muon was 0.95. Adam's beta parameters were set to $(0.9, 0.999)$. For training batch size was set to 64, and we were evaluating results with checkpoints every 100 batches. In total, the base run had 10 epochs. Default pixel feature has intensity 1. We used a simple CNN with 2 convolutional layers followed by 2 linear layers. Weights are initialized as Gaussian, with mean 0 and standard deviation 0.01. Convolution kernel sizes are 3, and the number of filters is 32 in the first and 64 in the second layer, with MaxPool of size 2 in between. The first fully connected layer has output dimension 128, and the second one 10 (the number of classes).

D.3 Routing Task and Shared Representations

Setup Details. Here we clarify the setup used for experiments in Section 4. We denote a sample $(x_i, y_i)_{j,o}$ to mean that x_i comes from input source j and y_i is going to the output source o. During training, we only see samples $(x_i, y_i)_{j,j}$ and $(x_i, y_i)_{j,j+1}$. Denote with $W_j^{in} \in \mathbb{R}^{64\times 4}$ the input

encoder for source j, $W_o^{out} \in \mathbb{R}^{7\times 64}$ the decoder for output source o, and $W^h \in \mathbb{R}^{64\times 64}$ the hidden linear layer. Then given $(x_i, y_i)_{j,o}$, the output of the network is $\hat{y}_i = W_o^{out} W^h W_j^{in} x_i$. We use MSE objective to train the network, i.e. minimize $\|\hat{y}_i - y_i\|^2$. Each gradient update involves a batch containing a single sample from each allowed input-output pair. Full batch sampling procedure can be seen in Algorithm 1.

Algorithm 1 Sampling a batch in the routing task

Require: Number of sources $m = 7$, allowed shifts $k = 2$, total numbers in the task $N = 4$, orthonormal set of N input vectors in $\mathbb{R}^N$ for each input source j: $\{(v_1^j, ..., v_N^j)\}_{j=0,...,m-1}$

Initialize batch $\mathcal{B} \leftarrow \emptyset$
for input source $j = 0$ **to** $m - 1$ **do**
  for shift $s = 0$ **to** $k - 1$ **do**
    Output source $o \leftarrow (j + s) \bmod m$
    Sample a random data index $i \sim \mathcal{U}(1, N)$
    $x_i \leftarrow v_i^j$
    $y_i \leftarrow$ output vector corresponding to i (Figure 3b)
    Add sample to batch: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x_i, y_i)_{j,o}\}$
  end for
end for
return Training batch $\mathcal{B}$

Analysis. In a similar setup, Saxe et al. (2022) theoretically demonstrate that GD has an implicit bias towards learning shared representations, with the core argument lying in Theorem B.2 2. Because of our gained intuition, we identified this setting as the one where Spectral GD may be disadvantaged by the absence of simplicity bias. Mathematically, each gating mechanism follows the dynamics derived by Saxe et al. (2022):

$$\frac{d}{dt}B_1 = \frac{\sqrt{P}}{M^2} B_2 B_1 \left[S - B_2 B_1^2 D\right]$$
$$\frac{d}{dt}B_2 = \frac{P}{M^2} B_1^2 \left[S - B_2 B_1^2 D\right]$$

Here, B_1 and B_2 represent the spectra/mode strengths of the gated components, while S and D denote certain data statistics (see Saxe et al. (2022)). The focus is on "pathway counting" argument: the gradient updates are scaled by pre-factors proportional to P, the number of pathways passing through a gated unit. In the initial phase of learning, where dynamics are dominated by exponential growth, these pre-factors act as rate constants; consequently, configurations with maximal P (i.e., $P = M^2$, corresponding to shared representations) are exponentially faster to learn and win the "neural race". In contrast, Spectral GD orthogonalizes each of the updates, effectively removing the dependence of learning speed on the pathway multiplicity P, and the order of speed of learning for each gating mechanism is the same. In that case, the winning gating strategy is dependent on other conditions, such as the random initialization.

D.4 Transformer in Cycle Task

Implementation details related to the transformer model are as follows. We use two layer softmax attention transformer, with 2 heads in each layer, with causal masking and RoPE positional embedding. The token dimension being equal to the embedding dimension was $N = 8$, and the inputs were one-hot encodings of each number. The rank of each head was 12. The transformer was attention-only, meaning it didn't have MLPs. The residual connections were present after the first layer, but not after the second (as it was the last). Every weight matrix of dimensions $(\texttt{in_dim}, \texttt{out_dim})$ was initialized as Gaussian with mean $\mathbf{0}$ and std $0.01/\sqrt{\texttt{in_dim} \times \texttt{out_dim}}$. For SGD, learning rate was set to 0.2 for the second layer and 5 times larger for the first (as otherwise the model didn't learn) and momentum to 0.9. As no weights used biases, for Muon run all weights were optimized with `SingleDeviceMuon` from Jordan et al. (2024b). The learning rate was 0.0003 with the default momentum of 0.95. Training was done by minimizing MSE loss for each next token prediction. We train for 40,000 steps, each one involving a single batch of size 512. We used $N = 8$ numbers in a cycle and sequence length 80.

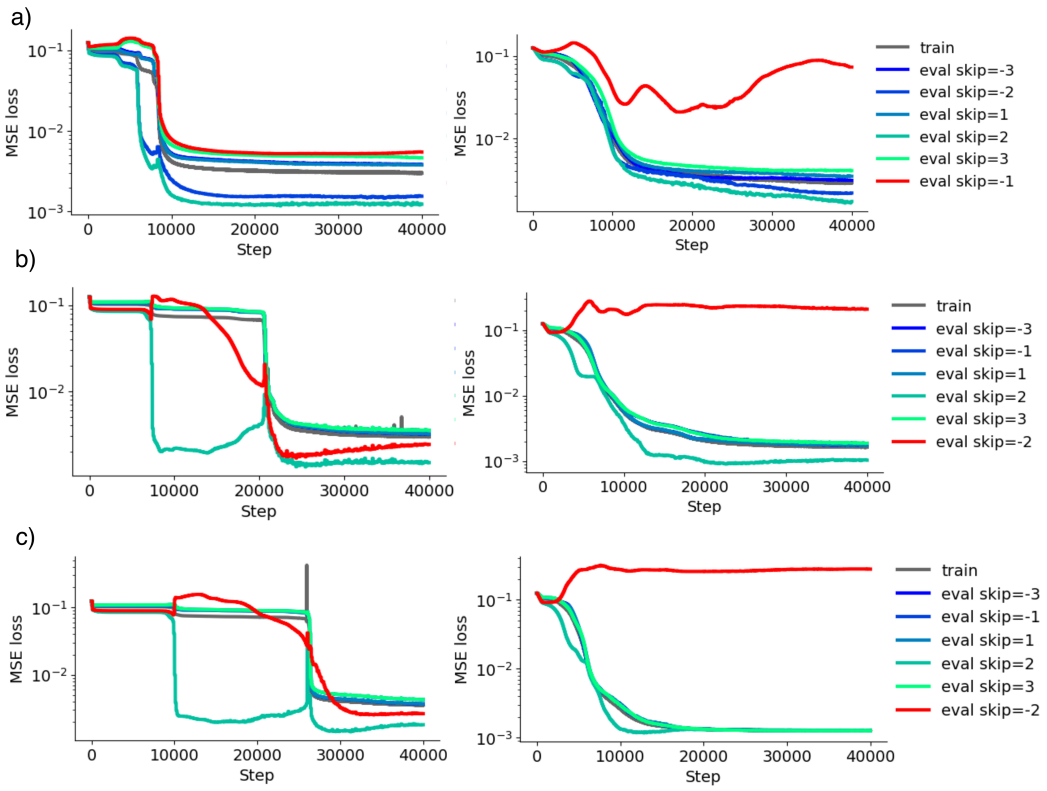

Figure 10: a) SGD (Left) and Muon (Right) optimized transformers, trained on skips $\{1, 2, 3, -2, -3\}$, and evaluated at skip -1. b) The same run as in the base setting, but with 3 heads per layer instead. c) 4 heads instead. All plots show the same phenomena observed in Section 6: SGD (Left) generalizes to the unseen skip, and Muon (Right) doesn't.

Further experiments. We reproduce the phenomenon across couple of different setups. In Figure 10a, we show the results when training on the skips $\{1, 2, 3, -2, -3\}$ and observe whether the transformers generalize to skip -1. In 10b we train with transformers with 3 heads, each rank 12, and in 10c with 4 heads.

D.5 Linear Attention Simulations Beyond the Assumptions

For the simulations in Figure 4, we used the task setup described in Section 5. While the figure is illustrating the learning dynamics of Spectral GD, for its generation we used Muon. The linear attention model consists of three matrices $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ that were optimized either by GD with learning rate 0.01 or Muon with learning rate 0.00023. The initialization of $\boldsymbol{K}, \boldsymbol{Q}$ is Gaussian with mean 0 and std $\texttt{init}/\sqrt{HD}$, where the $\texttt{init} = 0.01$ for GD and $\texttt{init} = 0.001$ for Muon. $\boldsymbol{V}$ is initialized as diagonal, with diagonal entries being Gaussian with mean 0 and std $\texttt{init}/H$. We do not further constrain it to stay diagonal through training, although empirically it seems to be the case. Note that this setting doesn't impose the joint diagonalizability assumption directly, but only through small initialization. For GD this is enough as the growth happens along one direction at a time, aligning the key and query vectors—a phenomenon called silent alignment. For Muon this is not the case, as the key and query vectors grow equally in all directions of eigenvectors of $\boldsymbol{\Lambda}$. Nonetheless, the general phenomenology of all heads being learned in the same time still holds.

We show more simulations for $D = H = 3$ in Figure 11, and for $D = 2, H = 20$ in Figure 12. For the simulations applying Spectral GD to full matrices $\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V$ instead of the reparameterized $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$, see Figure 13, showing the learning of all heads simultaneously still happens.

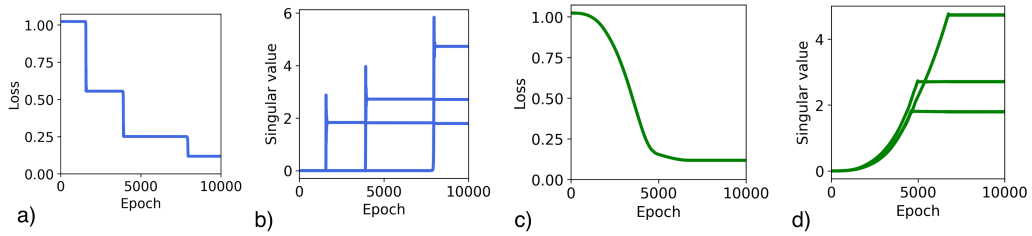

Figure 11: a) Loss for GD, b) evolution of singular values of $\boldsymbol{KVQ}^\top$ for GD optimized attention, c) loss for Spectral GD d) singular values for Spectral GD in the setting $D = H = 3, R = 1$.

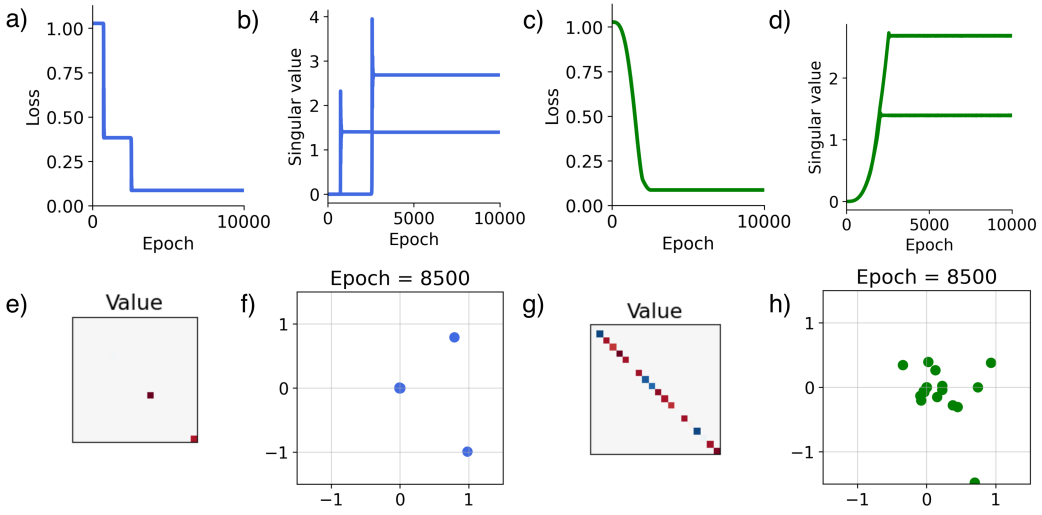

Figure 12: Linear attention learning when $H > D$, here $H = 20, D = 2$. a) GD loss; b) GD singular values; c) Spectral GD loss; d) Spectral GD singular values; e) GD final value matrix; f) GD key vectors at step 8500; g) Spectral GD final value matrix; h) Spectral GD key vectors at step 8500. All together, we notice how GD activates as many heads as the task requires, but not more. On the other hand, Spectral GD activates all the heads, showcasing its loss of simplicity bias.

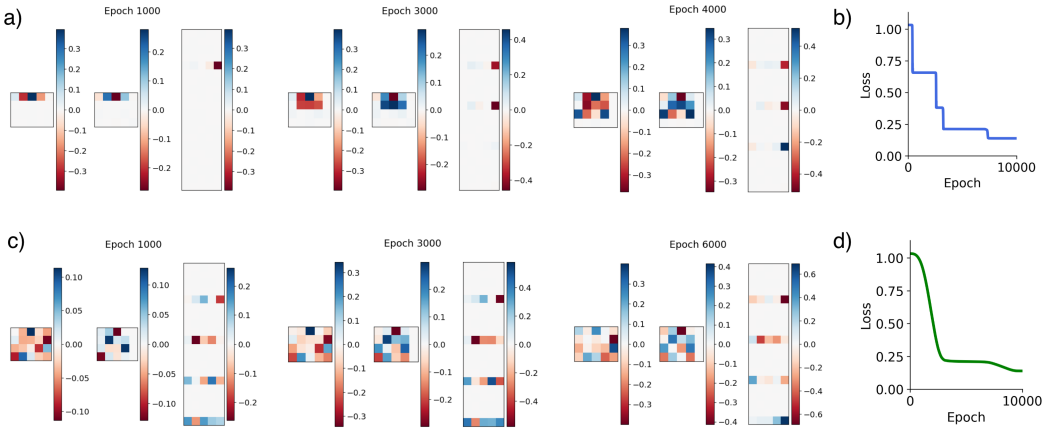

Figure 13: Empirical observations of weight matrices (a,c) $\boldsymbol{W}_K, \boldsymbol{W}_Q, \boldsymbol{W}_V$ in that order, through training and loss curves (b,d) when trained with GD (a,b) and Spectral GD (c,d).

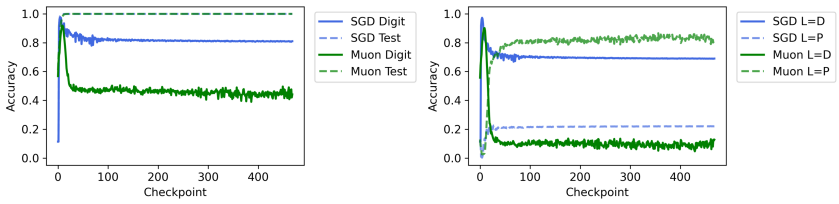

Figure 14: Accuracies of Muon and SGD algorithms ran for 100 epochs on (Left) Test and Digit only and (Right) Misaligned evaluators.

E Questions and Answers

Many sub-questions arose while we were writing the paper. For the flow of presentation, we didn't include them in the main text, but instead collect them here.

- **What about Adam optimizer?** For the theory part, Adam's per-parameter update rule makes it hard to mathematically derive the evolution of singular values of its solution matrix. There is no guarantee even that the principal components will be aligned through training, even if they are at initialization. Nonetheless, it is the case that Adam also doesn't follow the same trajectory as GD, escaping the saddles (Jacobs et al., 2026). In fact, an Adam simplification frequently used by theorists, named SignGD, employed on a simplified, diagonal deep linear network has the exact same dynamics (mathematically) as Spectral GD in the same setting. All that to say, Adam also loses the simplicity bias of GD. Below we show how it compares empirically in each setting we considered.

E.1 Digit and Pixel Feature Learning

- **What if the state where the digit is a dominant feature for SGD is meta-stable, and eventually the pixel becomes dominant?** With all the saddle point plateaus SGD has to escape, we were wondering if the accuracy on Misaligned; label=digit set will eventually drop. We show a longer experiment, this time with 100 epochs in Figure 14. Noticing how SGD accuracy on Misaligned datasets stabilized, we doubt that the situation will change after further training.
- **Have you plotted average probability of the correct token instead of accuracy?** Probability for each relevant label has a nice property of being continuous per sample, where accuracy on a sample contributes either with 0 or 1. For this reason, we also plot the probabilities on Misaligned datasets in Figure 15a,b, analogous to the plots in Figure 2d,e. The qualitative similarity with the hypothesized feature presence remains.
- **How does Adam perform?** We saw in Section 3.1 that in the presence of spurious features Adam can also be worse than SGD, contrary to the story in Vasudeva et al. (2025). In Figure 16 we show the accuracy curves of Adam in the two feature learning setup.
- **Have you tried the setting where spurious pixel is highly, but not fully correlated with the label?** To this end, we train on a set where α of all images contain the spurious pixel, for $\alpha = 0.95$ and $\alpha = 0.9$. Peak accuracy for all three optimizers on Digit only set is almost perfect. For that reason, we plot the peak accuracy on Misaligned; label=digit set instead. The results as we vary the intensity of the spurious pixel are shown in Figure 15c,d. Once again, SGD is better for low pixel intensities, but as the intensity of spurious pixel grows, the pixel feature becomes more dominant and hurts the performance of SGD more than Muon and Adam.

E.2 Gated Deep Linear Network on Routing Task

- **Does it help if you introduce momentum into Spectral GD run?** We experimented with adding the heavy-ball momentum of 0.95 to both SGD and Spectral GD runs. The result still holds: SGD learns unseen pairs, and Spectral GD does not (Figure 17a,b).

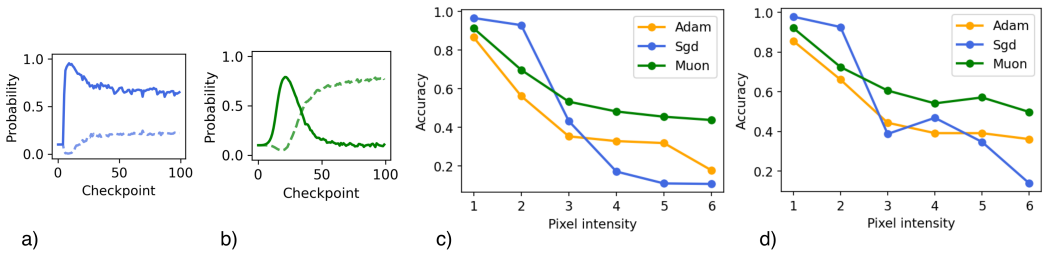

Figure 15: Probabilities of the label=digit (full line) and label=pixel outputs, on the base runs optimized by a) SGD and b) Muon. The curves behave similarly to the accuracy ones in Figure 2d,e. The peak accuracies on Misaligned; label=digit set when only a) $\alpha = 0.95$, b) $\alpha = 0.9$ fraction of training samples has the spurious pixel.

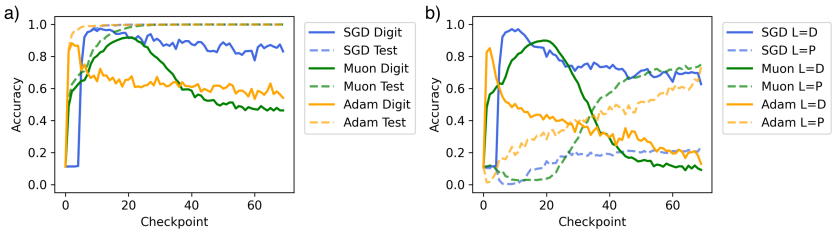

Figure 16: Same setup as in Figure 2b,d, but now together with Adam's performance.

- **How does full Muon do on this task?** The result doesn't change if we also employ Newton-Schulz iterations to recover the full Muon, as shown in Figure 17c.
- **Does it help if you apply weight decay for Spectral GD?** Weight decay also does not help either, see Figure 17d.

E.3 Transformer Learning Cycles

- **How do Adam and AdamW compare in your setting?** Adam and AdamW, while behaving differently than Muon, also seemingly fail to capture the underlying similarity between different cycle skips. In Figure 18, we provide their loss curves.
- **What happens if you switch to cross-entropy loss?** If we use cross-entropy loss instead of MSE, SGD again learns the unseen skip, and Muon does not (Figure 19). However, we note here that SGD result seems to be highly sensitive in the learning rate: 0.05 results in learning, but both 0.04 and 0.06 result in unstable learning where training loss explodes. Removing the instabilities is a big gap to close if one wants to compare Muon and SGD at larger scale (Zhao et al., 2025).
- **What happens if you add Layer Norm?** We train the transformers with a fixed (unlearnable, `elementwise_affine=False`) layer norm; for SGD we lowered the learning rate to 0.02. We see that both optimizers seemingly failed to capture the shared structure in Figure 20.
- **What happens if you add MLPs?** Results of adding the MLPs with hidden layer ratio being 4 are shown in Figure 21. We notice seemingly noisier training with both optimizers, and SGD also losing the plateaus; both runs didn't result in learning the unseen skip -2.
- **Maybe Muon needs more regularization, what happens if you increase the weight decay?** The original run is ran with the default setup of `SingleDeviceMuon` from Jordan et al. (2024b), which has weight decay 0. Here we present results if we increase the weight decay to 0.1 and 0.2 in Figure 22. We see that the weight decay doesn't help. Again this may be because weight decay constraints both heads equally, and to see the

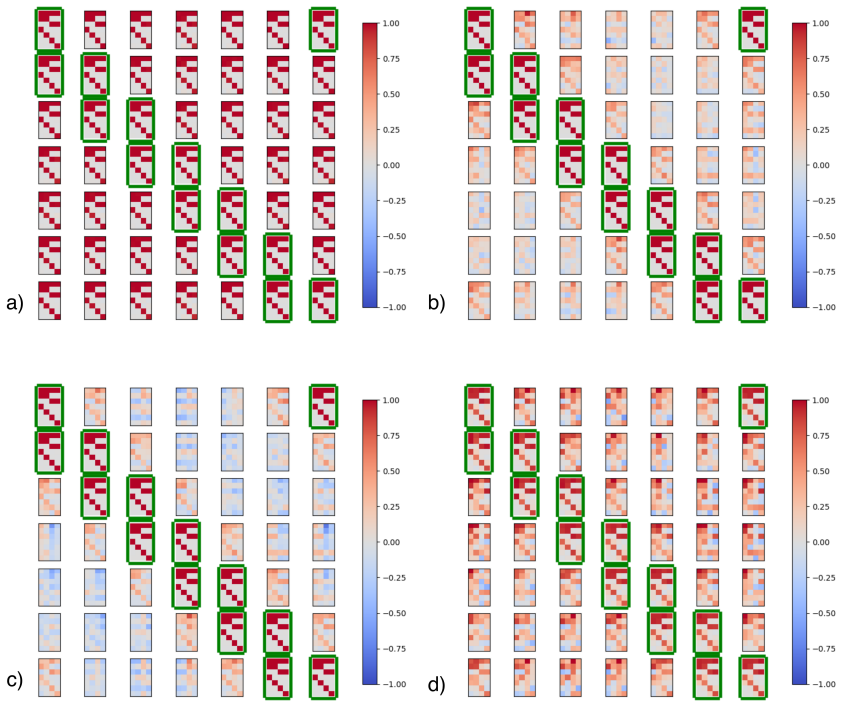

Figure 17: Ablations on the routing experiment from Section 4. First row shows what happens if we add the momentum of 0.95 to a) SGD and b) Spectral GD run. c) Full Muon run. d) Spectral GD run with added weight decay of 0.1.

regularization that could produce a similar behavior to SGD, one needs to constrain them unequally, perhaps with a weight decay schedule per head.

- **What happens if you use only one head?** We plot the results in Figure 23. With one head, SGD run was more unstable, so we lowered the learning rate to 0.1. Muon still doesn't generalize to unseen skip, and perhaps this time the reason lays more closely to the theory on deep linear networks, or it could be due to all the nonlinearities deviating from the theory setting.
- **What happens if the head rank R is smaller than N, is SGD still generalizing?** The results with ranks $R = 8, 4$ suggest that SGD struggles to find generalizable structures in these settings.
- **Can curriculum help Muon learn all skips with shared representations?** We were curious about this, and introduced couple of different curricula to see whether they can recover the effect of gradual learning with SGD. In Figure 25, we see that none of our curricula really helped with learning the skip -2 which is unseen in training. Our hypothesis is that Muon's problem with activating all heads at the same time is still causing the problem, where the information for different skips, though highly related, ends up being learned by different heads.
- **Can dynamic curriculum in weight activations help learn all skips with shared representations?** Another type of curriculum is allowing the weights to update gradually through training. The motivation behind this is the observed dynamic curricula in weight activation of GD. We implement weight activation "head-by-head", and within a head, QK circuit is activated "neuron-by-neuron". A part of weights being active means the optimizer update is applied to it from that point onwards. That means that the value matrix for the first head is activated first, and the Q, K matrices rows are activated one by one as training progresses. Once all the rows of the Q, K matrices are active, the second head actives. V is fully active immediately, while QK circuit activates neuron-by-neuron (row-by-row). We show the results when using AdamW optimizer with this dynamic weights curricula in Figure 26, observing it does better than the plain AdamW, learning the unseen skip -2. We

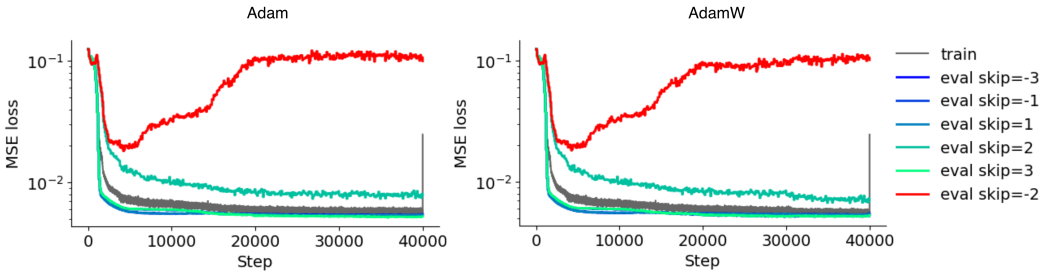

Figure 18: Results on transformer cyclic task for Adam and AdamW optimizers, trained in the same setup of Figure 5.

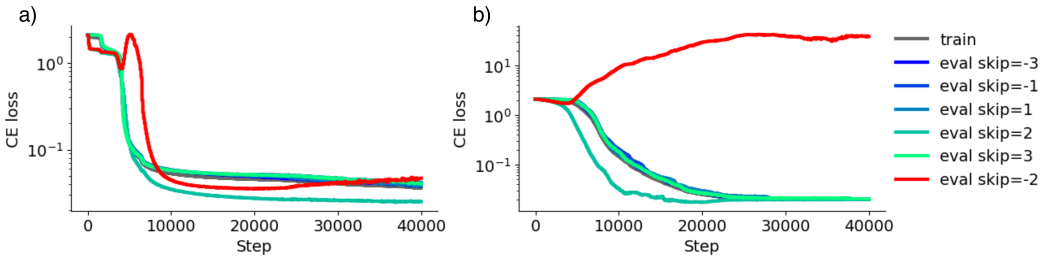

Figure 19: Cross-entropy loss curves for a) SGD and b) Muon in the otherwise same setting of the cycle task as in the main text.

note that the success of learning the unseen skip highly varied when we change the learning rate. The use of Muon with the dynamic curricula in weights is not possible because of orthogonalization.

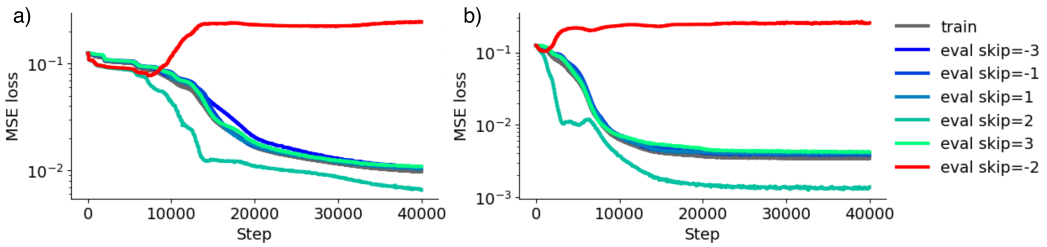

Figure 20: MSE loss curves on transformers with layer norm trained with a) SGD and b) Muon.

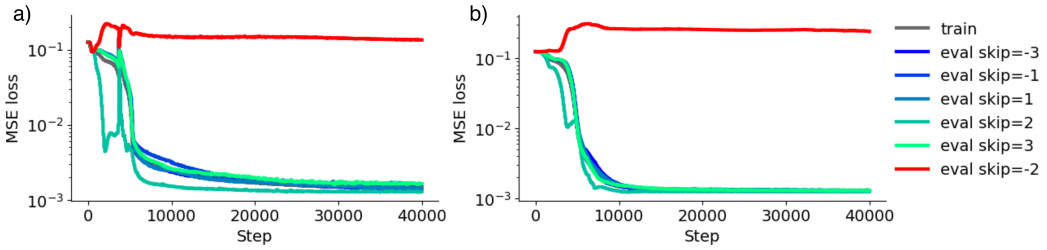

Figure 21: The loss curves shown for a) SGD and b) Muon, when the models now additionally have MLPs after attention layers.

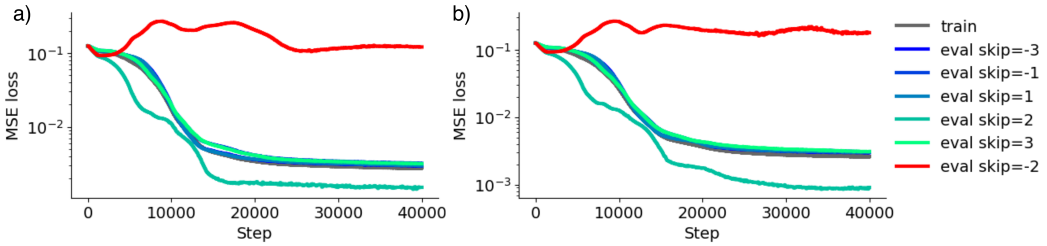

Figure 22: Muon trained transformer on our task with weight decay a) 0.1 and b) 0.2. We see that weight decay did not improve the generalization of the model.

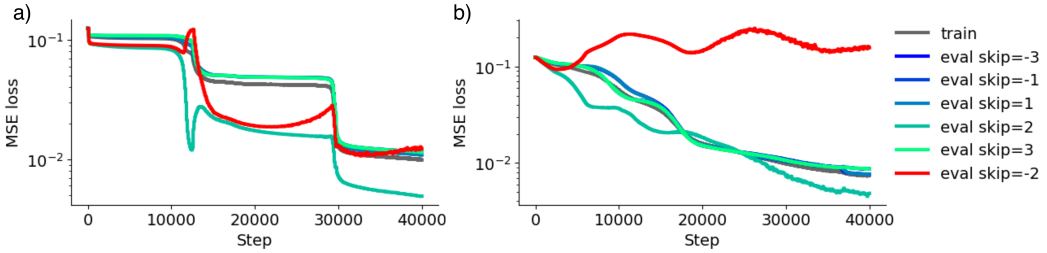

Figure 23: a) SGD and b) Muon trained transformers, now with a single head of dimension 12.

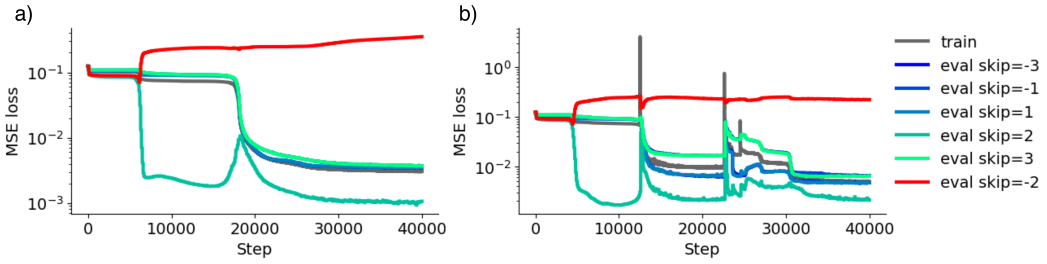

Figure 24: Base setting SGD trained transformer, but instead with ranks a) R=8 and b)R=4 per head. Having smaller rank took away the generalization. We also notice more training instabilities with $R = 4$.

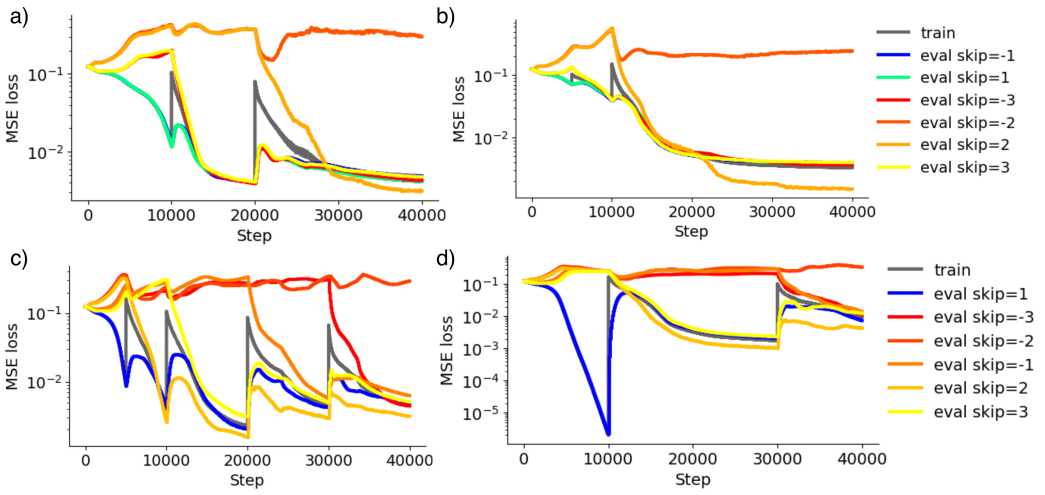

Figure 25: Introducing a couple of different curricula into the task of learning cycles with Muon. Curricula are a) at step 0: skips 1 and -1, at step 10000: skips 3 and -3, at step 20000: skip 2; b) at step 0: skips 1 and -1, at step 5000: skips 3 and -3, at step 10000: skip 2 c) at step 0: skip 1, at step 5000: skip 2, at step 10000: skip 3, at step 20000: skip -1, at step 30000: skip -3; d) at step 0: skip 1, at step 10000: skips 2, 3, at step 30000: skips -1 and -3.

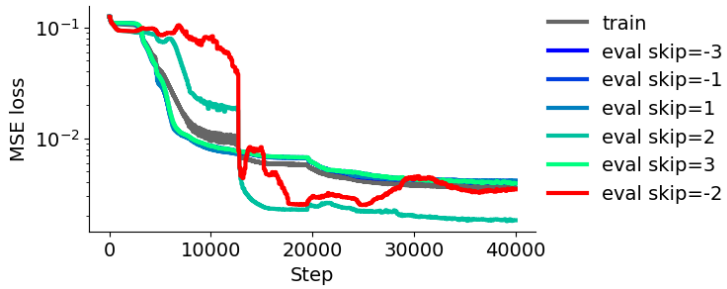

Figure 26: Employing the dynamic curriculum in weight activation, using AdamW optimizer with learning rate 0.0004. We see that learning the unseen skip is more successful now, compared to vanilla AdamW run.