Hallmarks of Optimization Trajectories in Neural Networks and LLMs: Directional Exploration and Redundancy

Anonymous Authors[1]

Abstract

We propose a fresh take on understanding the mechanisms of neural networks by analyzing the rich directional structure of optimization trajectories, represented by their pointwise parameters. Towards this end, we introduce a natural notion of the complexity of optimization trajectories which help hallmark the directional nature of optimization in neural networks: when is there redundancy, and when exploration. We utilize the trajectory perspective to showcase the effect of scale on regularizing the directional nature of trajectories. As a by-product, we also observe an intriguing heterogeneity of Q,K,V dynamics in the middle attention layers in LLMs which, however, is homogenized by scale. Importantly, we put the significant directional redundancy observed to the test by demonstrating that training only scalar batchnorm parameters some while into training matches the performance of training the entire network, and thus exhibiting the potential for hybrid optimization schemes geared towards efficiency.

1. Introduction

Given a network architecture and the training task, the loss landscape — which is the high-dimensional surface whose each point characterizes the fit of the parameters to the task objective — entails the possible trajectories that might be followed by an optimization algorithm, such as stochastic gradient descent (SGD). The particular sets of paths are, however, also determined by the particular optimization choices and hyperparameters, such as the learning rate, momentum, batch size, weight decay, and more. In fact, the regions and topographical features of the landscape that are never encountered or realized in typical optimization paths, might as well not be in the landscape at all. *Essentially*, the optimization trajectories are the probes through which the loss landscape is accessed.

Consequently, a significant body of literature builds around the principle of an inherently regular manner of traversing in the landscape, i.e., the implicit bias (Gunasekar et al., 2018; Li et al., 2019; 2020; Moroshko et al., 2020), facilitated by optimization algorithms. This preferential landscape access is in comparison with the actual surface level, possibly treacherous, non-convexity, and thus suggesting that the network stays clear of sub-optimal local minima. Therefore, despite the complexity of the neural landscapes, implicit bias lends a formal and reasonable support to the empirical success of massively over-parameterized neural networks. Hence, we might expect to see traits and hallmarks of regularity in the sequence of steps that make up the optimization trajectories of neural networks. Then, we can try to ask ourselves,

> *How are these trajectories structured? Do these paths have a lot of zigzags and bends, reaching the solution winding and coiling, or are they straight and direct? And does this depend upon the phase of optimization (early vs late)?*

This, in essence, is the key research theme of our present study. More precisely, we explore and develop indicators (hallmarks) about the complexity/regularity of the optimization trajectory. In particular, we analyze and compare multiple intermediate checkpoints amongst themselves, across different scenarios and large-scale case studies. Notably, a qualitative hallmark, which we call the *'Trajectory Map'*, conveys the directional (dis)similarity of the parameters and visually depicts the nature of optimization within and across various stages of training, i.e., at a pan-trajectory level.

The focus of our investigation is to study the properties of trajectories since it (a) brings in a level of architecture agnosticity and helps unlock shared insights onto features of optimization, (b) contains an intrinsic data-dependence that, for all intents and purposes, implies no explicit inference over additional data samples is needed (which anyway might not be possible due to resource or privacy constraints), (c) allows analyzing and prognosing the developing solution

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

strategy on-the-fly, over the course of training (instead of waiting until convergence), and (d) provides potential hints at the redundancies plaguing the optimization procedure.

Our contributions are: 1) We propose the novel perspective of trajectory maps and showcase its use for hallmarking the directional nature of optimization. 2) We show how scale provides a regularizing effect on the directional complexity of trajectories, and also find an interesting observation about Q,K,V heterogeneity in the middle attention layers. 3) We put the demonstrated directional redundancy to the test, by showing that only training the scalar parameters of (batch) normalization layers a short while into training suffices for matching the performance, thus laying the seeds for efficient optimization hybrids.

2. Methodology

Matrix representation of Trajectory. Let us assume the optimization trajectory consists of a set $\mathcal{T}$ of points $\{\boldsymbol{\theta}_t\}_{t=0}^{T}$, each denoting the (flattened) parameters of the network encountered at some step and which live in the parameter space $\boldsymbol{\theta} = \mathbb{R}^p$, i.e., $\mathcal{T} \subseteq \boldsymbol{\theta}$. This set of points need not contain the entire set of points visited in the course of optimization but instead can represent a subset of points, possibly sampled at an interval of k points. It will be convenient to organize this set of points, which define the trajectory, in the form of a matrix, $\boldsymbol{\Theta} \in \mathbb{R}^{(T+1)\times p}$, whose first dimension $T+1$ makes explicit the inclusion of the initialization $\boldsymbol{\theta}_0$.

Trajectory Map. Analyzing the matrix $\boldsymbol{\Theta}$, on its own, might get cumbersome as the size of modern networks ranges in millions and billions of parameters. Hence, we will resort to looking at functions of the kernel matrix $\mathbf{K} = \boldsymbol{\Theta}\boldsymbol{\Theta}^\top$ which would be a square matrix of shape $n = T+1$. Further, it will also be helpful to isolate and analyze the directional aspect of the trajectory, for which we will normalize the set of points by their norm, and in effect, consider the set $\widehat{\mathcal{T}} = \{\boldsymbol{\theta}_t/\|\boldsymbol{\theta}_t\|_2\}_{t=0}^{T}$ with the respective matrix $\widehat{\boldsymbol{\Theta}}$. As a result, the ensuing kernel matrix $\widehat{\boldsymbol{\Theta}}\widehat{\boldsymbol{\Theta}}^\top$, which we will refer to as $\mathbf{C}$, will contain the relative cosine similarities between every pair of points in the trajectory. So, $(\mathbf{C})_{ij}$ is, cos-sim$(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \langle\boldsymbol{\theta}_i, \boldsymbol{\theta}_j\rangle/\|\boldsymbol{\theta}_i\|_2\,\|\boldsymbol{\theta}_j\|_2$. Hereafter, we will refer to $\mathbf{C}$ as the *Trajectory Map* (TM). We would like to remark that, although not necessary, here we are essentially considering linear kernels, for as we will see they deliver a great mileage by themselves.

Mean Directional Similarity (MDS). Besides visualizing the TM as the qualitative hallmark, as a quantitative hallmark we will consider Mean Directional Similarity (MDS), where we take the cosine similarity averaged over the entire trajectory map, i.e., over every pair of points in the trajectory. This can be written as, $\omega := \frac{1}{n^2}\,\mathbb{1}_n^\top \cdot \mathbf{C} \cdot \mathbb{1}_n$, where $\mathbb{1}_n^\top = (1\cdots1)^\top \in \mathbb{R}^{1\times n}$ denotes the vector of all ones and $n = |\mathcal{T}|$ is the cardinality of the trajectory. By using the form of the matrix $\mathbf{C}$ discussed before, we can further rewrite MDS as, $\omega = \left\|\frac{1}{n}\,\widehat{\boldsymbol{\Theta}}^\top \mathbb{1}_n\right\|^2$. Now, it becomes apparent that MDS essentially projects all the trajectory points onto the unit sphere, computes their average and finally takes the squared norm.

To get a better sense of MDS, we can consider its two possible extremes: (a) all the parameter unit-vectors cancel out, yielding a value of $\omega = 0$. For instance, this would happen in the scenario when the points in the trajectory are exactly following a circular orbit around the origin; or, (b) when each of the parameters point in the same direction, implying that the trajectory is simply a linear path, with $\omega = 1$. Knowing the nature of these two extremes, we can expect neither to be desirable in an ideal trajectory.

3. Hauling Trajectory Hallmarks for LLMs

With all the recent interest in Large Language Models (LLMs) and their scaling, a natural question is what kind of structure is exhibited by the trajectories in the case of language modelling tasks and across models of different sizes. Besides, while we expect increasing parameter count to provide new directions for learning, does scale make the optimization trajectories complex or does it instead regularize them?

Thanks to Pythia's (Biderman et al., 2023) publically released model checkpoints over training, for GPT-NeoX (Black et al., 2022) models — ranging in sizes from 14 Million (M) to 12 Billion (B) — we can provide answers to the above questions. Given that processing all the available checkpoints would require several terabytes of cache, we select every fourth checkpoint, resulting in 39 checkpoints that we analyze for models of sizes: 14M, 70M, 160M, 410M, 1.4B, 2.8B, 6.9B, 12B. The results for a shortlist of these experiments can be found in Figure 1 (for more, see Figure 67).

First, we note that there is a tiny square grid in the upper-left corner, which delineates precisely the learning rate warmup phase. Next, the subsequent larger grid starts out, for 14M, with distinct subgrids but then with increasing model scale takes a funnel-like shape around 1.4B, before becoming rather homogeneous by 12B parameters. Even the horizontal and vertical slivers corresponding to warmup and the rest of the epochs start to assume a higher cosine similarity with scale. Overall, increasing scale lends an intense dark hue to the trajectory maps, which leads us to posit that the inductive bias of scale might be related to regularizing the trajectories. We would like to emphasize that considered models live in million and billion dimensional spaces, and rapidly reaching a cosine similarity of ~ 0.9 or more with the final solution is rather noteworthy.

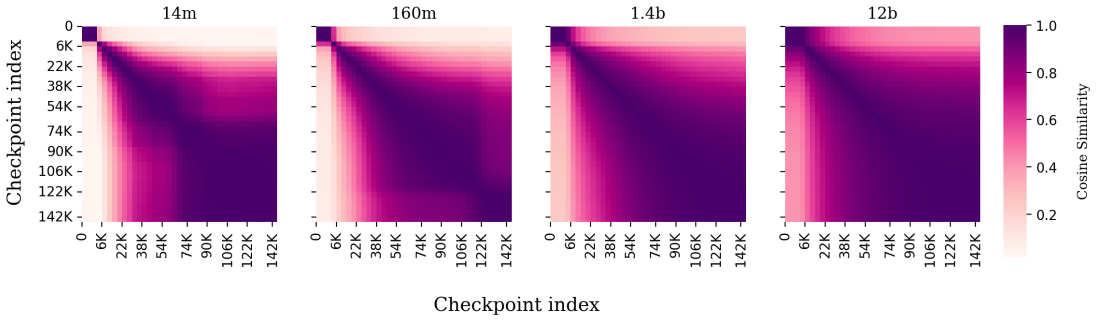


Figure 1. Trajectory Maps of Pythia GPT-NeoX models across three orders of scales trained on Pile. The corresponding MDS $\omega = 0.650, 0.678, 0.759, 0.815$.

Why do parameters become aligned with scale? A Theoretical Argument. We prove, in Appendix A, that this surprising finding about the progressive increase in cosine similarity with scale has a relatively simple explanation, at least in the case of the large-width limit of deep networks. The gist of our argument is that *in the large width limit, any parameter updates that lead to stable feature updates must necessarily yield updated parameters that are identically aligned with their initialisation.* This is a well-known fact for lazy learning regimes like the Neural Tangent Kernel (Jacot et al., 2020) or standard parameterisations, where no feature learning occurs. What may be surprising is that this is necessarily true for feature learning regimes like μP (Yang et al., 2022).

Layerwise Q,K,V dynamics homogenize over depth at $\sim$ 1B scale and over. Another interesting aspect of the structure of trajectory maps for the 14M and 160M cases is their relative heterogeneity (distinct subgrids), as compared to the billion parameter models. This heterogeneity cannot be explained away through the settings of a learning rate scheduler (which is just a cosine scheduler), so we inspect the layerwise trajectory maps (i.e., by building it over parameters of each layer separately). In particular, we find an intriguing heterogeneous structure for these models that is most starkly present in the query-key-value (Q,K,V) parameters of the attention layers, especially the middle layers, as shown in Figure 2. This indicates that the Q,K,V parameter dynamics converge at different timescales, with the middle layers converging the last directionally in contrast to earlier and later layers. Moreover, we find that *scale has the striking effect of homogenizing the Q,K,V dynamics*, as shown in Figure 74, with the layerwise trajectory maps essentially resembling the network-wide trajectory map structure in Figure 1.

4. Putting Directional Redundancy to the Test

Our trajectory map analysis has revealed a significant directional redundancy (see also Appendices D, F, G for further experiments) in the optimization trajectories of neural networks, which is especially prominent later in training. This raises the question of whether this redundancy can be leveraged for creating efficient hybrid optimization schemes or if it conceals minute but crucial directional changes. Therefore, to test the true nature of directional redundancy, we consider optimizing only some layerwise scalar parameters after a little while into training. Further, instead of attaching extra scalar parameters per neuron or channel and tuning them, we will repurpose the scalar weight and bias parameters present in the ubiquitous (batch/layer) normalization layers in modern network architectures. We are inspired by the prior work of Frankle et al. (2021) who demonstrated the remarking expressivity of training just the scalar parameters in batch-normalization (BN) layers in the entire network. Given our resource constraints, we will consider experiments on image classification datasets as a start and test the potential of this idea.

CIFAR10 experiments. We experiment with ResNet20 on CIFAR10 using SGD over 160 epochs, freezing non-BN layers at different points and training only the $1,376$ scalar parameters in the BN layers. Figure 3 presents the results alongside the trajectory map of the full network training.

When right from the initialization, just the BN layers are trained, as claimed in Frankle et al. (2021), the resulting network achieves a test accuracy of 54.8%, which although quite interesting falls nevertheless considerably short of the 91.2% test accuracy obtained by training the entire network. In a way, from the horizontal strip of trajectory map around 0, which pales as we move from the left to right, we see that the full network parameters at the end of training deviate significantly in their directionality as opposed to the parameters at initialization (the cosine similarity being ~ 0.3). However soon after, from around 40 epochs, where the trajectory map starts developing a dark hue (and the cosine similarity to the final parameters climbs to about ~ 0.9), training BN parameters alone brings us to within 2% of the full-network accuracy; and from around 80 epoch gets to within 1%, and completely matches thereafter. Notably, this feat is *remarkably achieved by training only* 0.5% *of the overall parameters,* and this fares even better than training all the parameters in the bulkier last layer as shown in Figure 76.

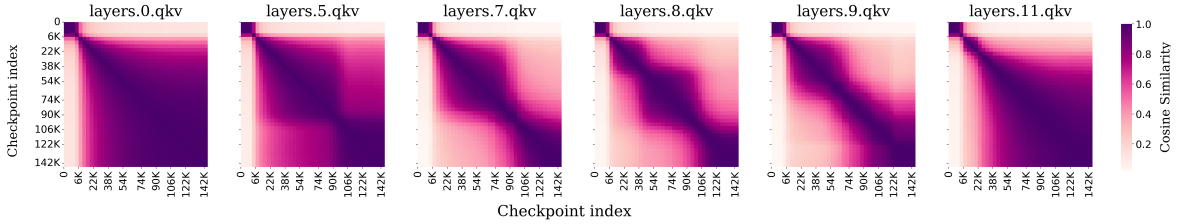


Figure 2. Layerwise Trajectory Maps for Q,K,V weights across depth fot GPT-NeoX trained on the Pile for the 160M model. This suggests that the Q,K,V parameters of the middle layers seem to be converging last, and differ from earlier and later layers in this regard.

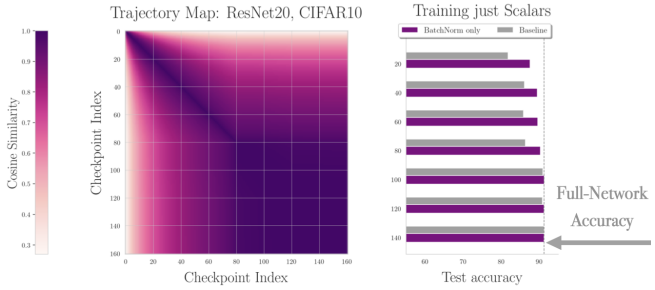


Figure 3. Directional Redundancy put to test by training only the BN scalar parameters. The trajectory map of the original, full network, training is shown on the left and, in the bar chart to the right, the bars denoting the performance achieved by training BN parameters from a particular epoch are horizontally aligned with the corresponding trajectory map rows. 'Baseline' is the accuracy just before switching to optimizing the BN parameters only.

ImageNet experiments. Likewise, for ResNet50 trained on ImageNet as in Figure 7, we found that although training BN layers alone (which are just 0.18% of the parameters) right from initialization gets us to an accuracy of just $\sim 6.4\%$ (a paltry amount, given the full-network's accuracy of $\sim 76\%$), but training it from epoch 30 gets to within $\sim 10\%$ of the full-network's accuracy and from epoch 60 to within about $\sim 2\%$ of it, i.e, to $\sim 73.2\%$ top-1 test accuracy.

It should be mentioned that in the above BN training procedure, although the runtime of forward pass ends up being similar and the backwards slightly faster than training the entire network, this leads to significant savings in the GPU memory consumption as the optimization buffers (like the momentum buffer or those used for preconditioning in adaptive methods like Adam) now need to be of much smaller size (typically $< 1\%$, i.e., 99% savings) and could, in turn, make larger batch sizes feasible. More broadly, we foresee that this idea can be readily adapted into a hybrid optimization scheme, where regular training can be interleaved with memory-light training of normalization layer parameters, and where the latter is done on cheaper GPU instances in the cloud. All in all, these findings pointedly show that the observed directional redundancy from the trajectory maps truly manifests during training and *can be potentially utilized to reap practical gains.*

5. Related Work

Directional Convergence. Prior work has theoretically noted a notion of directional convergence (Ji & Telgarsky, 2020), wherein the parameters of simple networks and classifiers converge quickly, in terms of their direction. Likewise Merrill et al. (2020) observed that the cosine similarity between subsequent parameter checkpoints during T5 (Raffel et al., 2023) pre-training rapidly approaches one. Our analysis of the trajectory maps can thus be seen as related to this aspect, however, we provide a more rounded and refined picture of the directional nature of optimization trajectory during training and utilize it to make training efficient, instead of just noting the rapid directional convergence.

Mechanistic Understanding of Neural Networks and LLMs. Besides, our analysis can also pave the way for a novel data-free mechanistic understanding of LLMs (like in Figure 2). This holistic optimization path perspective can complement the top-down approach of mechanistic understanding of LLMs via influence functions (Grosse et al., 2023) and the bottom-up circuit view of Transformers (Elhage et al., 2021).

6. Conclusion

Overall, we have merely scratched the surface of this trajectory perspective into understanding optimization behaviour in neural networks. We genuinely believe that there is a lot to be understood about the complex, intermingled behaviour of optimization in deep learning, and hopefully, this work will drive further research into the directional aspects of trajectories and contribute towards hybrid optimization schemes that can exploit the showcased directional redundancy.

Limitations and Future Work. While we have focused on the directional aspects of trajectories, the length of the trajectories also holds relevance, especially when the experiments under comparison may have unequal number of sampled checkpoints. We expect a version of MDS, which is weighted by the step lengths to be useful in such a scenario. Besides, our experiments with training scalar parameters are currently based on vision models, given the excessive resources involved in testing this for LLMs. However, we hope that the wider community can seize on this observation and exploit it for training LLMs more efficiently.

References

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming, 2020.

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability, 2022.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1, 2021.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization, 2021.

Frankle, J., Schwab, D. J., and Morcos, A. S. Training batchnorm and only batchnorm: On the expressive power of random features in cnns, 2021.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., Hubinger, E., Lukovsiūtė, K., Nguyen, K., Joseph, N., McCandlish, S., Kaplan, J., and Bowman, S. R. Studying large language model generalization with influence functions, 2023.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pp. 1832–1841. PMLR, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks, 2020.

Ji, Z. and Telgarsky, M. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33:17176–17186, 2020.

Kumar, T., Bordelon, B., Gershman, S. J., and Pehlevan, C. Grokking as the transition from lazy to rich training dynamics, 2024.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, Y., Ma, T., and Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations, 2019.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Merrill, W., Ramanujan, V., Goldberg, Y., Schwartz, R., and Smith, N. Effects of parameter norm growth during transformer training: Inductive bias from gradient descent. *arXiv preprint arXiv:2010.09697*, 2020.

Moroshko, E., Gunasekar, S., Woodworth, B., Lee, J. D., Srebro, N., and Soudry, D. Implicit bias in deep linear classification: Initialization scale vs training accuracy, 2020.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Singh, S. P., Bachmann, G., and Hofmann, T. Analytic insights into structure and rank of neural network hessian maps. *Advances in Neural Information Processing Systems*, 34:23914–23927, 2021.

Yang, G., Hu, E. J., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., and Gao, J. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022.

A. Why cosine similarities increase with scale?

Note, we assume that the majority of the parameter norm lies in the square hidden matrices, and not the input or output layers. Moreover, we use $o, O, \boldsymbol{\theta}$ to denote standard mathematical notation with regards to scaling in the limit width $n \to \infty$. For vectors, this notation is entry-wise.

Suppose we have a hidden layer with input $x_0 \in \mathbb{R}^n$ for width n, that is acted on by (without loss of generality) a square matrix $W_0 \in \mathbb{R}^{n \times n}$ to give:

$$h_0 = W_0 x_0$$

We suppose x has $\boldsymbol{\theta}(1)$ entries, as is the case with standard initialisations/parameterisations (He et al., 2015). We suppose W_0 has i.i.d. elements with initialisation that is $O(1/\sqrt{n})$ in order to ensure that each element of the features h has entries $\boldsymbol{\theta}(1)$.

Now, if we take a gradient update with learning rate η on some downstream loss L that depends on h (and not W or x), we get:

$$W_1 = W_0 - \eta \mathrm{d}h \cdot x_0^\top$$

where $\mathrm{d}h = \frac{\partial L}{\partial h} \in \mathbb{R}^{n \times 1}$ is our feature derivative.

Then if we have new input x_1 (wlog $x_1 = x_0$), we have new features:

$$h_1 = x_1 W_1 = h_0 - n\eta \mathrm{d}h \cdot \frac{x_0^\top x_0}{n}$$

For our features to be stable (i.e. $\boldsymbol{\theta}(1)$) after the update, we need $n\eta \mathrm{d}h$ to be $O(1)$, because $\frac{x_0^\top x_0}{n} = \boldsymbol{\theta}(1)$ by assumption on x. NB: if $n\eta \mathrm{d}h = o(1)$ we have no feature learning (ie NTK regime (Jacot et al., 2018)), and if $n\eta \mathrm{d}h = \boldsymbol{\theta}(1)$ we have feature learning (ie μP (Yang et al., 2022)).

In any case, $\eta \mathrm{d}h = O(1/n)$ entry-wise, which means that $W_1 - W_0 = -\eta \mathrm{d}h \cdot x_0^\top$ has $O(1/n)$ entries, again by assumption on the scale of elements of x_0.

But because $W_0 = \boldsymbol{\theta}(1/\sqrt{n})$, the initialisation will elementwise-dominate the $O(1/n)$ update for the first training step (and more training steps follows by induction). As a result, the update $W_T - W_0$ will always be an order of at least $\sqrt{n}$ smaller than the initialisation, and hence the new parameters W_T will be exactly aligned with the initialisation W_0 for all T in the large width limit, i.e. the cosine similarities will be 1.

B. Relative Trajectory Maps

Elsewhere it might be useful to analyze the trajectory relative to some point $\boldsymbol{\theta}_\tau$ as the origin, so there we will instead consider the set of points $\mathcal{T}_\tau = \{\boldsymbol{\theta}_t - \boldsymbol{\theta}_\tau\}_{t=0}^T$, and correspondingly organize it in the matrix $\boldsymbol{\Theta}_\tau \in \mathbb{R}^{(T+1)\times p}$. When τ is itself one of the points of the trajectory, then we will omit the row of zeros and shape the matrix as $\mathbb{R}^{T\times p}$. A natural point from where to contextualize the trajectory would be the initialization $\boldsymbol{\theta}_0$, and this relative trajectory will then be denoted as $\boldsymbol{\Theta}_0$ (where the subscript 0 is not to be confused for the usual origin O, namely, $\boldsymbol{\theta}_O = \mathbf{0}$).

C. Comparing Gradient Trajectories with Random Walks

The structure that we observe in trajectory maps following gradient trajectories raises the question of if we would observe similar structure in a random walk.

If we have T timesteps or epochs, with parameter space $\boldsymbol{\theta} \in \mathbb{R}^p$ and "learning rate" schedule $(\eta_t)_{t=1}^T$, we can consider a random walk with updates:

$$\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} \overset{\text{ind.}}{\sim} \mathcal{N}(0, \eta_t^2 I_p)$$

which is to say that at time step t, each parameter coordinate in the parameter vector is updated independently with a Gaussian of variance η_t^2, and the updates are independent across different time steps.

Then, if θ^i denotes a *single* parameter coordinate for a dimension $i \leqslant p$, for two time steps $s < t$, we have:

$$(\theta_s^i, \theta_t^i) \sim \mathcal{N}(0, \left(\begin{smallmatrix} H_s & H_s \\ H_s & H_t \end{smallmatrix}\right))$$

where $H_u = \sum_{t'=1}^u \eta_{t'}^2$ is the cumulative squared learning rate from $t' = 1$ to $t' = u$.

Then, by the strong law of large numbers we have for the large parameter space $p \to \infty$ limit:

$$\frac{1}{p}\|\boldsymbol{\theta}_s\|_2^2 = \frac{1}{p}\sum_{i=1}^p (\theta_s^i)^2 \overset{a.s.}{\to} H_s\,, \quad \frac{1}{p}\|\boldsymbol{\theta}_t\|_2^2 = \frac{1}{p}\sum_{i=1}^p (\theta_t^i)^2 \overset{a.s.}{\to} H_t$$

$$\frac{1}{p}\langle \boldsymbol{\theta}_s, \boldsymbol{\theta}_t \rangle = \frac{1}{p}\sum_{i=1}^p \theta_t^i \theta_s^i \overset{a.s.}{\to} H_s$$

and by the property of composing almost sure limits, we also have almost sure convergence in the cosine similarity:

$$\frac{\langle \boldsymbol{\theta}_s, \boldsymbol{\theta}_t \rangle}{\|\boldsymbol{\theta}_s\|_2 \|\boldsymbol{\theta}_t\|_2} \overset{a.s.}{\to} \frac{H_s}{\sqrt{H_s H_t}} = \sqrt{\frac{H_s}{H_t}}$$

in the large parameter space limit, which we use as an approximation to give analytic formulas for the trajectory map and MDS that we can compare to gradient trajectories.

One thing to note, is that this cosine similarity $\sqrt{\frac{H_s}{H_t}}$ becomes invariant to the *scale* of the learning rates η, and instead it is the relative rate of decay in the learning rate schedule that matters.

C.1. Experimental Results

The first thing to note is that our empirical simulation of the random walk matches the theoretical limit described in the section above, for a finite parameter count (such as $10,000$). Next, comparing the these relative trajectory maps with those for ResNet50 (Figure 5) we find that the latter reveal a much more directional redundancy component to their trajectories as opposed to random walks. This further lends support to the thesis that optimization trajectories ensued when training neural networks are highly structured and have significant directional redudndancy.

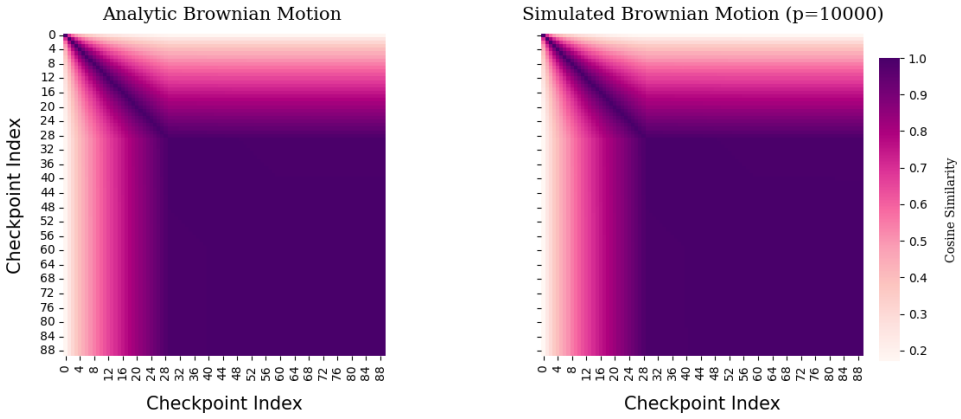


Figure 4. With step size decay: Relative Trajectory Map for a Random Walk/Brownian motion in both analytic and empirically simulated settings.

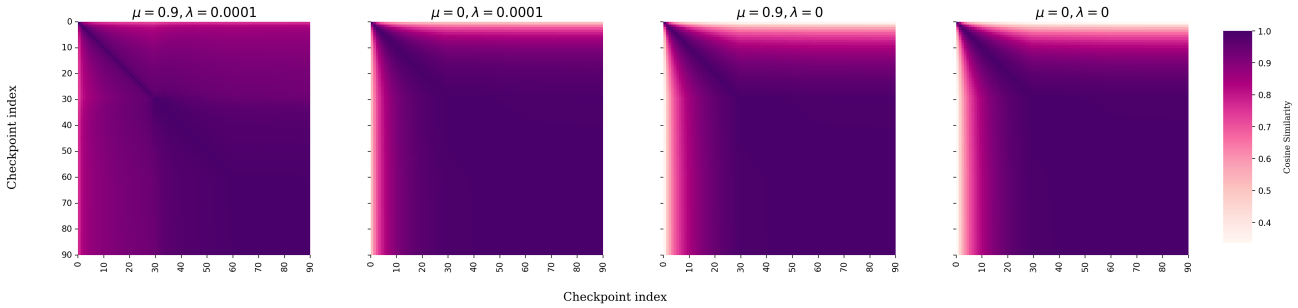


Figure 5. Relative Trajectory Maps, with respect to initialization, of ResNet50 models for different amounts of momentum and weight decay.

An additional thing to note is that the above relative trajectory map for random walks covers the setting of decreasing the step size to mirror how the optimization procedure is setup for ResNet50. In the case of no such step size decay, the analytic and empirical versions of the relative trajectory map are depicted in the Figure 6.

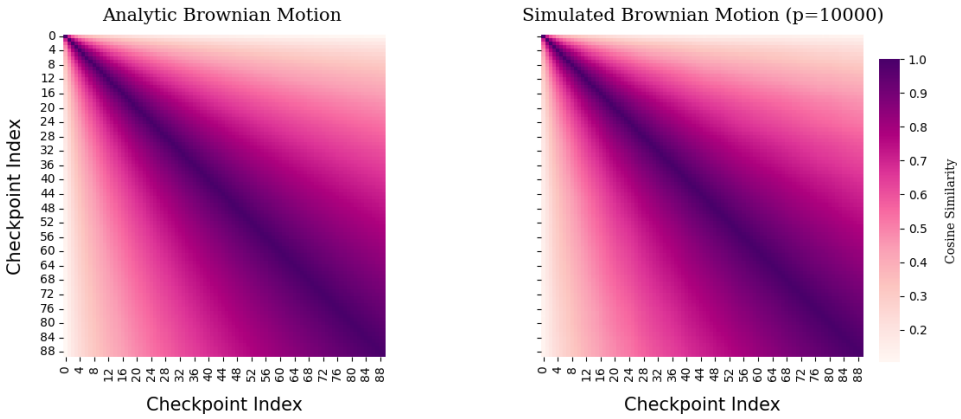


Figure 6. No Step size decay: Relative Trajectory Map for a Random Walk/Brownian motion in both analytic and empirically simulated settings.

D. A Tale of Hyperparameters

There is an emerging folk intuition that, over the past decade, networks have co-evolved hand-in-hand, amidst other things, with a certain particular set of optimizers and hyperparameter choices, deviating from which tends to produce significantly poor results. Let us see one such case in action, for a ResNet50 trained on ImageNet with SGD, achieving the familiar top-1 accuracy of $\sim 76\%$. Further, as is usual convention, this network was trained via a learning rate $\eta = 0.1$, momentum $\mu = 0.9$, batch size $B = 256$, weight decay $\lambda = 0.0001$ for 90 epochs, with a multiplicative decay by a factor of 0.1 at epochs 30 and 60.

Visual Tour of the Trajectory. Before we start the hyperparameter excursion, let us look at the nature of the optimization trajectory, as visualized via the proposed qualitative hallmark, namely the trajectory map $\mathbf{C}$. In particular, we save epochly checkpoints from the initialization until the final epoch, and thereby giving us a total of 91 checkpoints. We plot this 91×91 matrix in Figure 7.

For starters, we can easily make out three[1] distinct phases of optimization which are marked by an increased darkening of the pixels and their locations are precisely where learning rate decay was applied. The onset of these phases also seems to bring about an increased cosine similarity of the parameters contained within these phases, which seems to imply that following the learning rate decay the optimization is honing into a progressively confined subspace of the landscape.

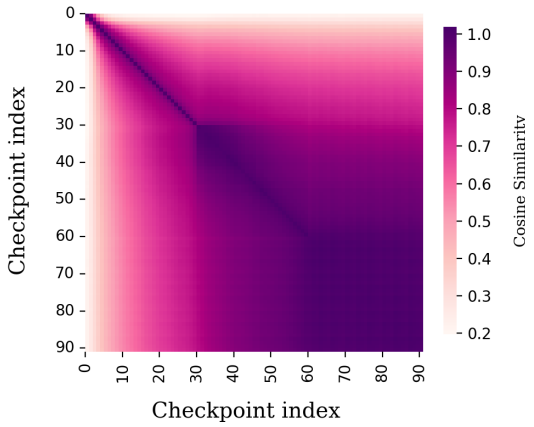


Figure 7. Trajectory Map of ResNet50 on ImageNet, $\omega = 0.764$.

Next, the mean directionality score (MDS) for this particular figure comes out to be $\omega = 0.764$. To contextualize this value, we should remind ourselves that we are working in a space of ~ 25.6 million dimensions. This should serve to emphasize that network optimization trajectories are highly structured, and not merely random points[2] in high-dimensions whose cosine similarity goes to zero. As a further note, the cosine similarity of two different instantiations of the ResNet50 parameters, from the same (and the usual) random initialization scheme, gives a value[3] of 0.374. However, besides these comparisons, it is still somewhat unclear what the value of MDS tells us about the trajectory. In particular, we need to ask ourselves (i) How does it rank in comparison to possibly other trajectories? (ii) Is it a high enough value, or should we aim for something much larger? (iii) But since the maximum value of ω can be 1 and since we have initialization in the mix here, obtaining such a value would not be so desirable as it would effectively mean the absence of any feature learning (Chizat et al., 2020), which seems to be a critical component[4] behind deep learning's success.

[1] If we look closely, there seems to be another phase transition neighbouring the initialization and the subsequent couple of epochs, giving rise to a thin horizontal and vertical sliver of relatively lighter colour in this figure.

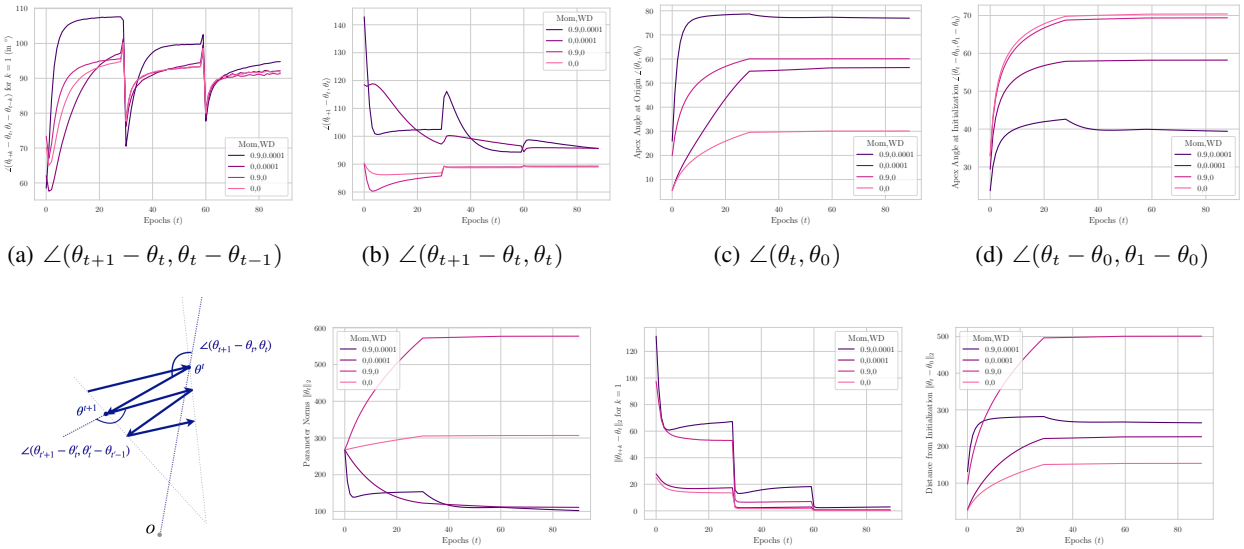
[2] In Section C, we analyze the relative trajectory map and MDS for a random walk/Brownian motion, and in comparison we find that (expectedly) the trajectory maps of neural networks are more directionally redundant.

[3] A reason for this high value of cosine similarity is the presence of BatchNorm or LayerNorm learnable scale parameters, which are typically initialized to all ones.

[4] Also, a certain amount of directional exploration is crucial as evident from our analysis of the grokking (Power et al., 2022) phenomenon discussed in Section J.

E. A Ride with Momentum & Decay

In order to obtain a more refined understanding of the above observation, we would like to inspect in detail the angular and norm-based measures of the trajectory. We will largely[5] focus on momentum and weight decay, given the especially intriguing directional nature of the trajectories caused by them. Now, let us start by taking a look at how aligned are the net or the aggregate updates[6] in the presence of these hyperparameters and otherwise.



(a) $\angle(\theta_{t+1}-\theta_t, \theta_t-\theta_{t-1})$ (b) $\angle(\theta_{t+1}-\theta_t, \theta_t)$ (c) $\angle(\theta_t, \theta_0)$ (d) $\angle(\theta_t-\theta_0, \theta_1-\theta_0)$

(e) Illustration of EoS with Momentum (f) $\|\theta_t\|_2$ (g) $\|\theta_{t+k}-\theta_t\|_2$ (h) $\|\theta_t-\theta_0\|_2$

Figure 8. Angular and Norm-based measures: ResNet50, ImageNet, Momentum and Weight Decay

E.1. Momentum and the Angle between Updates

In particular, in Figure 8(a), we plot the angle between consecutive epochs, i.e, $\angle(\theta_{t+1}-\theta_t, \theta_t-\theta_{t-1})$. Interestingly, we find that this angle becomes obtuse a short while into the training process, and further, this angle is larger when momentum is turned on versus when off. Moreover, rather visibly, this increase in angle is larger when weight decay is also enabled, suggesting that weight decay and momentum are closely intertwined. Taken as such, this observation would point as to how the MDS increases when these hyperparameters are switched off. To gain a somewhat better understanding of this mechanism, we turn to the simplest and oft-employed model of a quadratic problem.

Lemma 1. *Given a quadratic problem with ℓ_2 regularization of strength $\alpha > 0$, namely, $\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{2}\boldsymbol{\theta}^\top \mathbf{M}\boldsymbol{\theta} + \frac{1}{2}\alpha\|\boldsymbol{\theta}\|^2$, with $\mathbf{M} \in \mathbb{R}^{d\times d}$ symmetric with eigenvalues $\lambda_1 \geqslant \cdots \geqslant \lambda_d$, the angle between successive steps $\Delta_t = \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}, \Delta_{t+1} = \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t$, when using gradient descent with a one-step momentum ($\mu > 0$) and learning rates η_t, η_{t+1}, can be upper and lower bounded as follows:*

$$\langle \Delta_t, \Delta_{t+1}\rangle \leqslant \eta_t\eta_{t+1}(1-\eta_t(\mu+\alpha+\lambda_d))(\lambda_d+\alpha)^2\|\boldsymbol{\theta}_{t-1}\|^2$$
$$\langle \Delta_t, \Delta_{t+1}\rangle \geqslant \eta_t\eta_{t+1}(1-\eta_t(\mu+\alpha+\lambda_1))(\lambda_1+\alpha)^2\|\boldsymbol{\theta}_{t-1}\|^2$$

Proof. Given function $f(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top\mathbf{M}\boldsymbol{\theta} + \frac{1}{2}\alpha\|\boldsymbol{\theta}\|^2$, the gradient at $\boldsymbol{\theta}$ will be $\nabla f(\boldsymbol{\theta}) = (\mathbf{M}+\alpha\mathbf{I})\boldsymbol{\theta}$. Then at the first optimization step, we do

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta_t(\mathbf{M}+\alpha\mathbf{I})\boldsymbol{\theta}_{t-1}$$

[5] In the Appendix G, we also present a wider set of results, namely, the directional effects of hyper-parameters such as learning rate and batch size, recent regularizers like Sharpness-Aware Minimization (Foret et al., 2021) (SAM), as well as more datasets and architectures such as Vision Tranformers on ImageNet and VGG16 on CIFAR10, and for different amounts of label noise in the dataset.

[6] We qualify this by 'net' or 'aggregate' as we are working on somewhat coarser granularity (i.e., 1 epoch) than every update or step. But our current granularity is still rich enough to allow for the presented trends to persist even if go $2\times$ to $5\times$ more coarser.

The particular update being $\Delta_t := \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} = -\eta_t(\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_{t-1}$. The next update is similar, but now we also have to factor in the momentum,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{t+1}\left(\nabla f(\boldsymbol{\theta}_t) - \mu\eta_t(\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_{t-1}\right)$$

$$\begin{aligned}
\Delta_{t+1} := \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t &= -\eta_{t+1}\left((\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_t - \mu\eta_t(\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_{t-1}\right) \\
&= -\eta_{t+1}\left((\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_{t-1} - \eta_t(\mathbf{M} + \alpha\mathbf{I})^2\theta_{t-1} - \mu\eta_t(\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_{t-1}\right) \\
&= -\eta_{t+1}\left((1 - \mu\eta_t - \alpha\eta_t)\mathbf{I} - \eta_t\mathbf{M}\right)(\mathbf{M} + \alpha\mathbf{I})\boldsymbol{\theta}_{t-1}
\end{aligned}$$

Now, let us evaluate the inner-product $\langle \Delta_t, \Delta_{t+1} \rangle$,

$$\langle \Delta_t, \Delta_{t+1} \rangle = \eta_t\eta_{t+1}\boldsymbol{\theta}_{t-1}^\top \underbrace{(\mathbf{M} + \alpha\mathbf{I})\left((1 - \mu\eta_t - \eta_t\alpha)\mathbf{I} - \eta_t\mathbf{M}\right)(\mathbf{M} + \alpha\mathbf{I})}_{\mathbf{Z}}\boldsymbol{\theta}_{t-1}$$

Now without loss of generality we can consider $\mathbf{Z}$ to be a diagonal matrix, as $\mathbf{Z}$ is symmetric since $\mathbf{M}$ is symmetric, we can consider its spectral decomposition $\mathbf{Z} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ and project $\boldsymbol{\theta}_0$ onto its eigenvectors contained in $\mathbf{U}$. With this the matrices in the middle are diagonal and we can commute them, which yields us the following matrix:

$$\mathbf{Z} = \operatorname{diag}\begin{pmatrix} (1 - \mu\eta_t - \eta_t\alpha - \eta_t\lambda_1)(\lambda_1 + \alpha)^2 \\ \vdots \\ (1 - \mu\eta_t - \eta_t\alpha - \eta_t\lambda_d)(\lambda_d + \alpha)^2 \end{pmatrix}$$

where, we have denoted the eigenvalues of $\mathbf{M}$ as $\lambda_1 \geqslant \cdots \geqslant \lambda_d$.

Since the inner product of the updates is a quadratic form, we can upper and lower bound it based on the maximum and minimum eigenvalues of $\mathbf{Z}$, thus giving:

$$\eta_t\eta_{t+1}\lambda_{\min}(\mathbf{Z})\|\boldsymbol{\theta}_{t-1}\|^2 \leqslant \langle \Delta_t, \Delta_{t+1} \rangle \leqslant \eta_t\eta_{t+1}\lambda_{\max}(\mathbf{Z})\|\boldsymbol{\theta}_{t-1}\|^2$$

Because of the above form of eigenvalues of $\mathbf{Z}$ (diagonal matrices have their eigenvalues as their diagonal entries), we will have:

$\lambda_{\max}(\mathbf{Z}) = (1 - \mu\eta_t - \eta_t\alpha - \eta_t\lambda_d)(\lambda_d + \alpha)^2$ and $\lambda_{\min}(\mathbf{Z}) = (1 - \mu\eta_t - \eta_t\alpha - \eta_t\lambda_1)(\lambda_1 + \alpha)^2$ □

The proof inherently considers the solution at $\boldsymbol{\theta}^\star = \mathbf{0}$, but if that is not the case, we can substitute it in the objective and our derived bounds would scale in the squared distance to the solution, i.e. $\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^\star\|^2$. Besides, in the above proof, we consider a one-step momentum, which inherently means resetting the momentum after every 2 steps. This is done for convenience, as our main purpose is to anyways gain insights into the phenomenon and not provide its ultimate proof.

Turning to the bounds themselves, notice that if the learning rate $\eta_t \geqslant 1/(\lambda_1 + \mu + \alpha)$, the lower bound will turn negative and will be multiplied by a factor of $(\lambda_1 + \alpha)^2\|\boldsymbol{\theta}_{t-1}\|^2$. On the other hand, although the first term of the upper bound might still be positive, importantly, it is scaled by a factor of $(\lambda_d + \alpha)^2 \approx \alpha^2$ for matrices $\mathbf{M}$ which are close to degenerate ($\lambda_d \to 0$).

Low-rank Hessian and Edge of Stability. In our context, the Hessian of the loss with respect to the parameters will play the role of the matrix $\mathbf{M}$, since we can assume a second-order Taylor series will hold across the two steps. But it is also known through prior empirical work that the Hessian is significantly degenerate (Sagun et al., 2017), which has also been proven rigorously for deep linear fully-connected and convolutional networks (Singh et al., 2021). Furthermore, this requirement on the learning rate η_t is actually looser than the adaptivity of the largest eigenvalue of the Hessian to the learning rate $\lambda \approx \frac{2}{\eta}$, as shown in the recent work on Edge of Stability (EoS) (Cohen et al., 2022).

Explaining the Obtuse angles. Owing to these facts, we will have that $\lambda_d(\mathbf{M}) \approx 0$, and which further implies that the upper bound on the inner-product between the updates will be approximately zero, and the lower-bound will be large in absolute value but negative. Therefore, this explains how the angles between consecutive epochs can be obtuse. More broadly, the obtuse angle indeed implies that there are oscillations, especially along the direction of the largest Hessian eigenvector. Further, from Lemma 1, we see that the magnitude of the inner-product of the updates scales in proportion to η_{t+1}. Hence, a way to dampen the oscillations[7] is to decrease the learning rate, and as can be seen in Figure 8(a), the learning rate decay at epochs 30 and 60 is followed right after with the angles turning from obtuse to acute. Lastly, here in the constraint on the learning rate (the additive terms α and μ), we can also see momentum and weight decay go hand-in-hand, each accentuating the effect of the other.

Towards a holistic picture of Momentum. Besides, in Figure 8(c) and 8(d), we find that in the presence of momentum, a larger angle is traced at the origin by the trajectory, suggesting a more directional exploration, while the angle traced at initialization is smaller. The latter can also be seen from Figure 8(h), since with momentum, the trajectory moves further away from the initialization. Apart from this, in the absence of weight decay, the updates seem to be strengthening with momentum and the parameter norm rises 8(f) as well, giving rise to a mental picture of a trajectory similar to that left purple trajectory in Figure 9, at least until the training hits EoS.

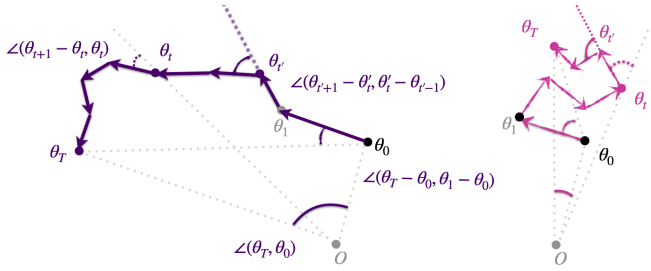


Figure 9. Illustration of two trajectories and angular measures.

With weight decay, as there is a decrease in parameter norm Figure 8(f) alongside the EoS process, as well as due to the presence of larger obtuse angles, we expect a reasonable affinity with our illustration in Figure 8(e), where we see the updates oscillating and slowly drifting towards the origin O below.

E.2. More than just Weight Decay

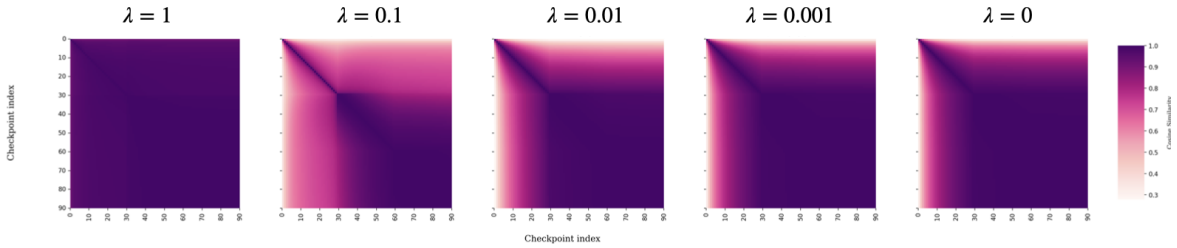


Figure 10. Relative Trajectory Maps (wrt. initialization) of ResNet50 models for different weight decay.

Now that we have gained a richer understanding of momentum and its interaction with weight decay, let us turn to weight decay alone and understand its directional effect. We have already noticed the increase in mean directional similarity (MDS) when weight decay is disabled for ResNet50 trained with SGD on ImageNet. In fact, we find a similar effect with an adaptive optimizer, like AdamW (Loshchilov & Hutter, 2017) — the trajectory maps for which are shown in Figure 10. Here, we used regularization constants from $\lambda = 0$ until the first value where we witness a decrease in test performance, which in this case was $\lambda = 1$. Specifically, we analyze the weight decay coefficients in $\lambda \in \{1, 0.1, 0.01, 0.001, 0\}$. The corresponding MDS come out to be, $\omega = 0.731, 0.679, 0.844, 0.882, 0.885$. We notice that, as before, increasing weight

[7]These oscillations need not necessarily translate into gross instabilities at the level of the loss, since as can be seen in Figure 8(g), the update norms progressively shrink in each of the three learning rate phases.

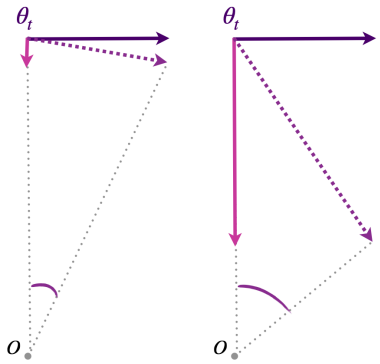


Figure 11. **Directional exploration effect of Weight Decay.** The downward vector represents the pull towards the origin (O) due to weight decay, while the rightward vector the force due to the loss .

decay leads to a heightened directional exploration, or lower MDS; except $\lambda = 1$ being the seeming anomaly.

But we find that this can be remedied simply by looking at the relative trajectory maps (Figure 10), and computing the relative MDS, i.e., ω_0 is $0.985, 0.807, 0.862, 0.897, 0.900$ for $\lambda = 1, 0.1, 0.01, 0.001, 0$ respectively. This occurs since such a high weight decay $\lambda = 1$, causes this particular network to underfit (train/test top-1 accuracy are $54.63\%, 50.52\%$). The performance for the rest of the networks improves, more or less, as expected with weight decay, and in particular, achieve accuracies of $75.45\%, 73.38\%, 71.03\%, 71.41\%$.

Having reaffirmed our results extensively about the directional exploration due to weight decay, we can understand it through a simple physics-based intuition, as shown in the Figure 11. In particular, we can think of the loss gradient pulling the network parameters rightwards, while the force exerted by weight decay tries to pull the network downwards. The relative strengths of these two 'forces' have been represented by the lengths of the two vector arrows. We notice that as the weight decay strength is increased, from the left subfigure to the right, the angle traced at the origin (O) also increases. This explains how weight decay can contribute towards directional exploration.

F. Detailed Experimental Results

F.1. ResNet50: Switching off the hyperparameters

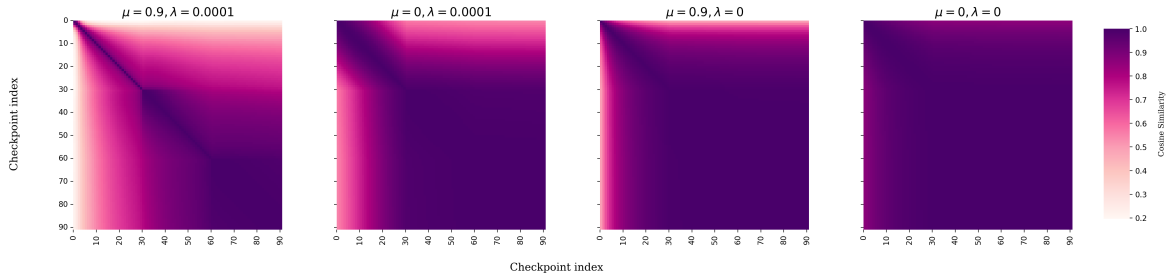
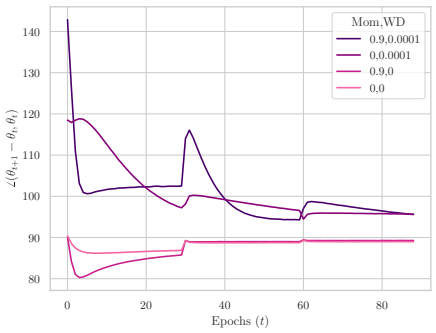
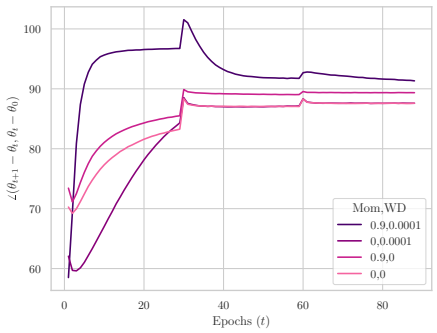


Figure 12. Trajectory Maps of ResNet50 models across different amounts of momentum and weight decay

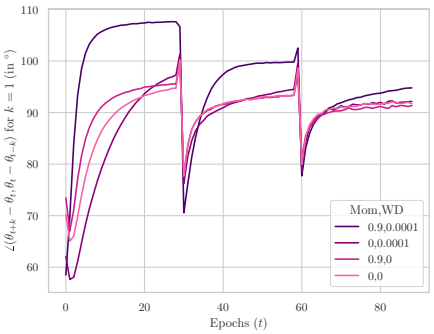
The relative trajectory maps can be found in Figure 5.



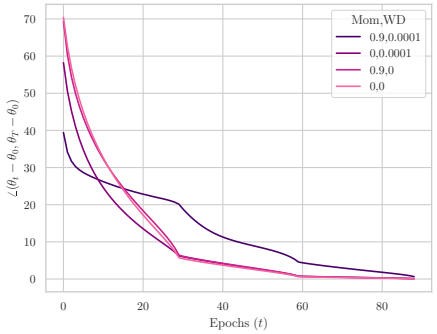
(a) $\angle(\theta_{t+1}-\theta_t,\theta_t)$



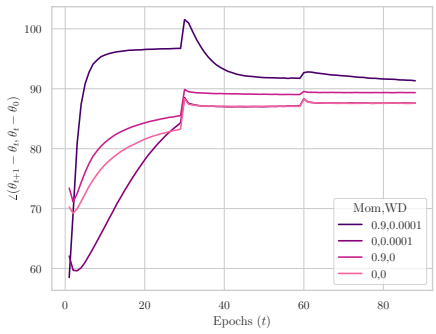
(b) $\angle(\theta_{t+1}-\theta_t,\theta_T-\theta_0)$



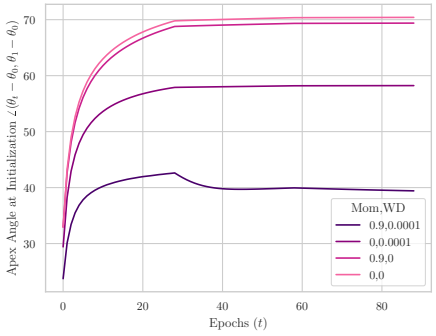
(c) $\angle(\theta_{t+k}-\theta_t,\theta_t-\theta_{t-k})$, for $k=1$



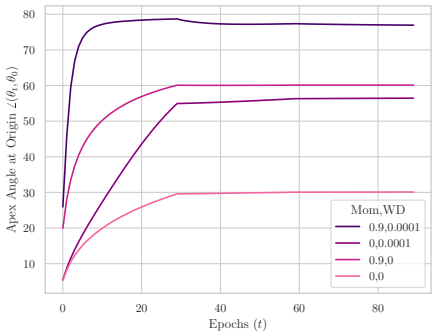
(d) $\angle(\theta_t-\theta_0,\theta_T-\theta_0)$



(e) $\angle(\theta_{t+1}-\theta_t,\theta_T-\theta_0)$

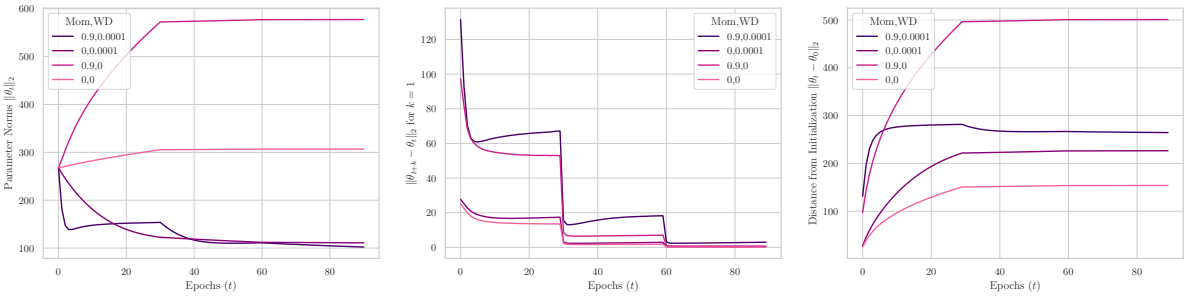


(f) Apex Angle at Initialization $\angle(\theta_t-\theta_0,\theta_1-\theta_0)$



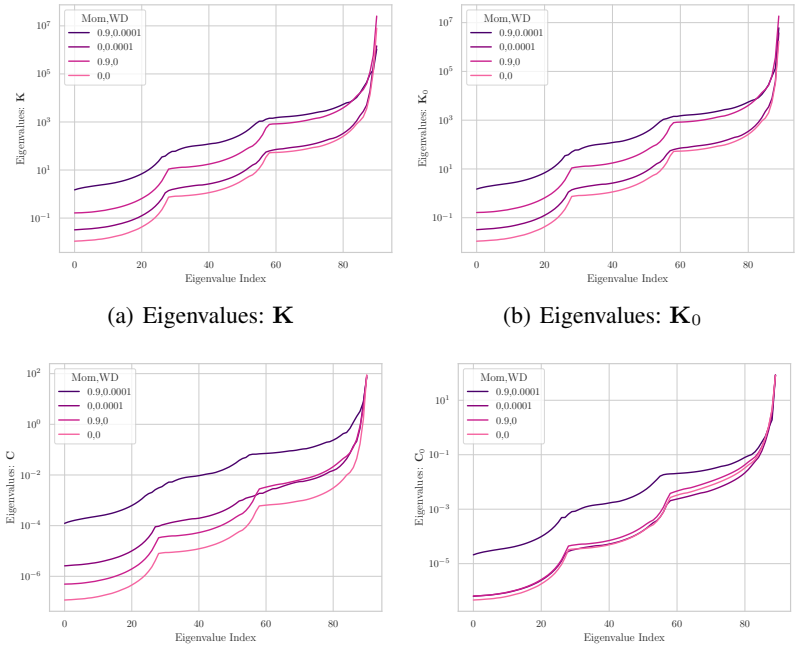
(g) Apex Angle at Origin $\angle(\theta_t,\theta_0)$

Figure 13. Angular measures of the Trajectory for ResNet50 trained on ImageNet



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 14. Norm-based measures of the Trajectory for ResNet50 trained on ImageNet



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 15. Spectral measures of the Trajectory for ResNet50 trained on ImageNet

F.2. ResNet50: Weight Decay, AdamW

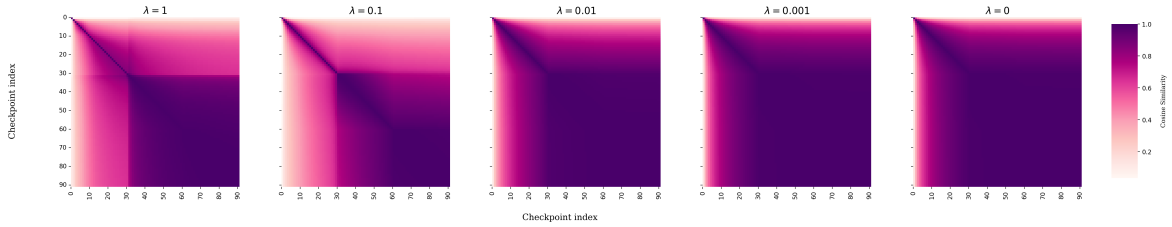


Figure 16. Trajectory Maps of ResNet50 models across different amounts of weight decay

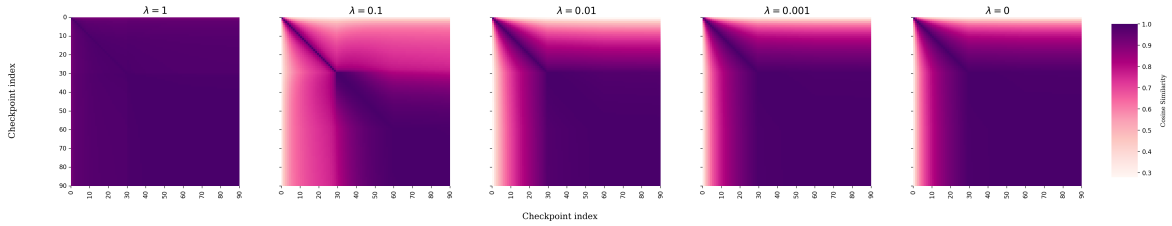
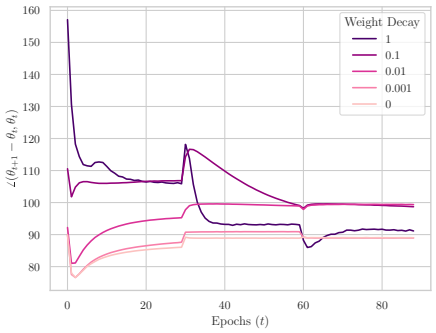
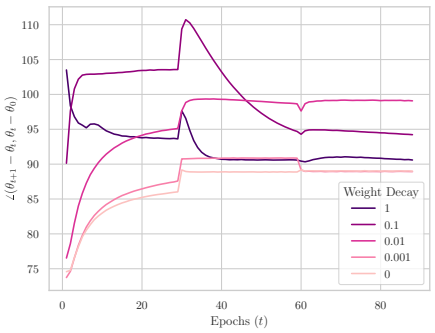


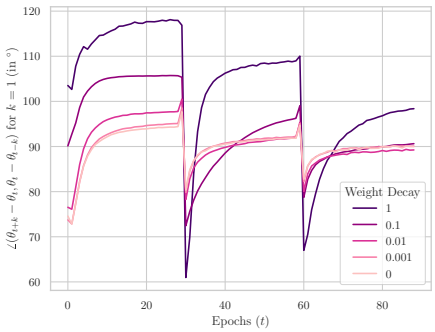
Figure 17. Relative Trajectory Maps, with respect to initialization, of ResNet50 models across different amounts of weight decay



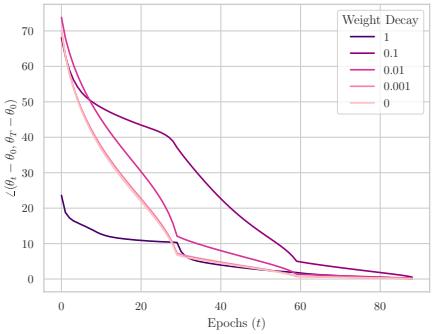
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



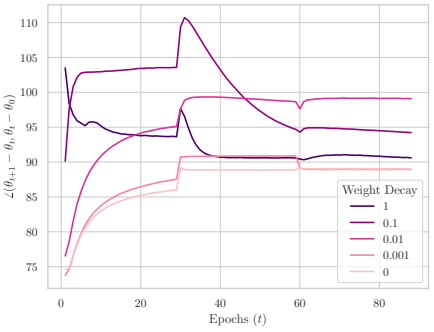
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



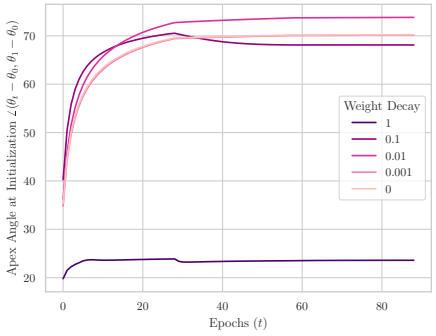
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



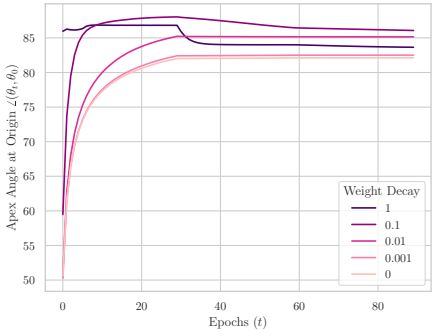
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

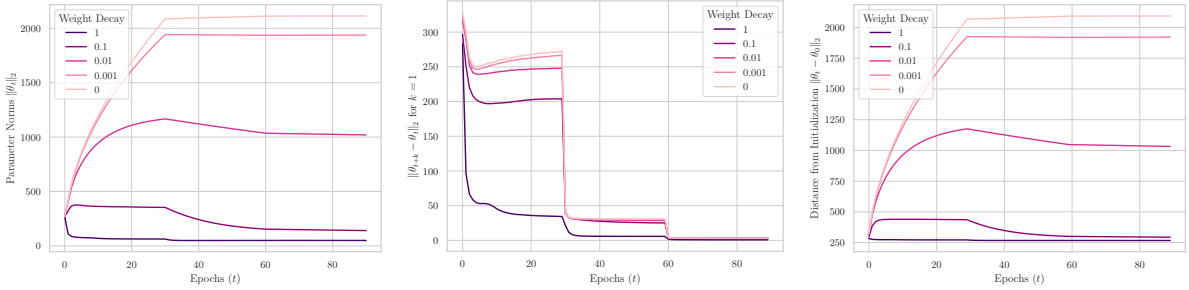


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



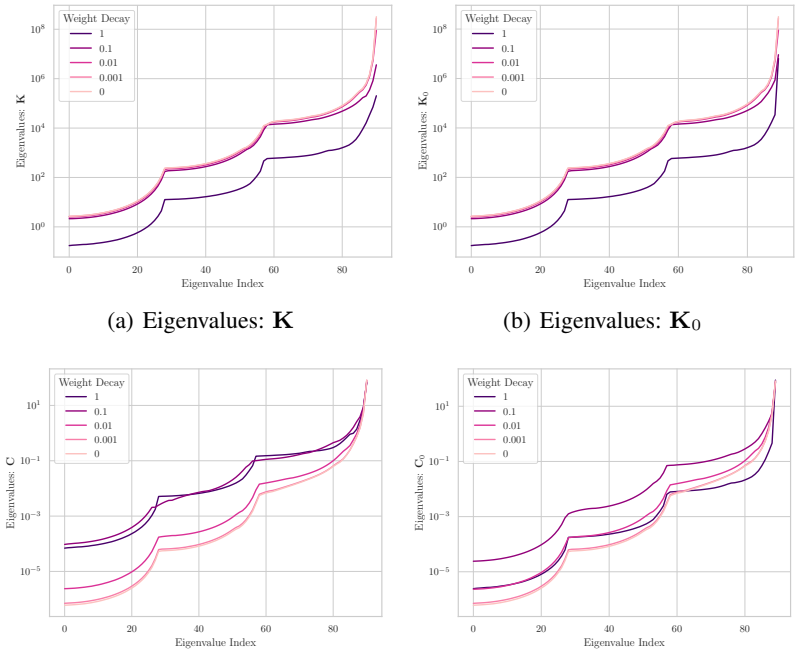
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 18. Angular measures of the Trajectory for ResNet50 trained on ImageNet



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 19. Norm-based measures of the Trajectory for ResNet50 trained on ImageNet



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 20. Spectral measures of the Trajectory for ResNet50 trained on ImageNet

G. Directional Effects in Other Key Settings

In addition to the momentum and weight decay, there are other crucial hyperparameters, such as learning rate and batch size, whose directional effects warrant a mention. We carry out additional experiments in these settings, and from where, the key findings are that the **learning rate**, as it would be easy to guess, indeed encourages directional exploration leading to low MDS scores. But, somewhat more interestingly, we find that increasing the **batch size** also helps further exploration and thereby decreases the MDS scores. The trajectory maps can be found in the Figure 61 While we encourage the curious reader to have a look at the Appendix G.9, we find that with increased batch size, the angle between the updates as well as the angle between the update and the current location become increasingly obtuse, and thus making room for a wider directional exploration. In contrast, for smaller batch sizes these angles are closer to 90°. We hypothesize that a similar mutual interaction, as observed with weight decay and momentum, also occurs with batch size is considered. A detailed analysis, however, remains outside the current scope.

Lastly, we also experimented with **Sharpness-Aware Minimization** (Foret et al., 2021) (SAM), where we found that a higher value of the SAM regularization coefficient leads to a slightly increased directional similarity, which could potentially be related to SAM directing optimisation to flatter basins wherein the individual points are more directionally alike and have higher cosine similarities. The detailed results can be found in the Appendix G.3.

Other Settings and Datasets. As a final remark for this section, we would like to emphasize that similar results for weight decay as well as momentum, can be found under different hyperparameter settings in the supplementary material. In particular, we analyze the qualitative and quantitative hallmarks for multiple values of learning rate, weight decay, and momentum for VGG16 on CIFAR10 as well as other values for momentum and weight decay in the case of ResNet50 trained with SGD, and even Vision Transformer trained with AdamW on ImageNet across varying weight decay, but these have to be omitted here due to space constraints.

G.1. ViT: Weight Decay, AdamW

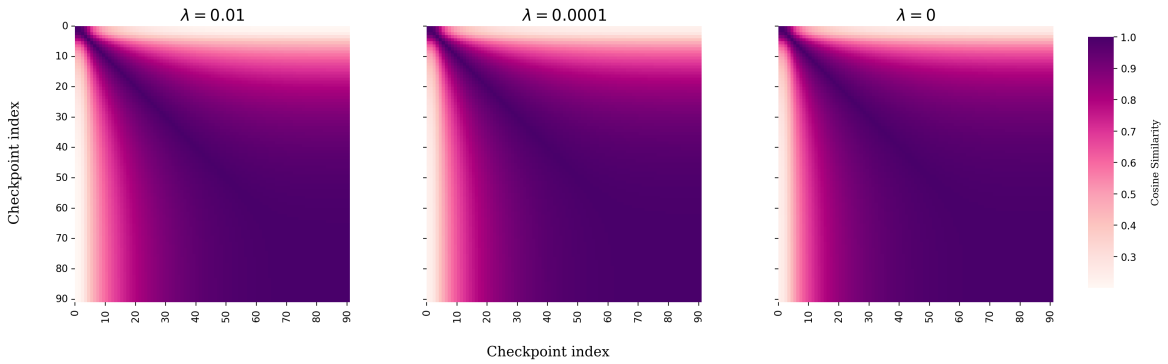


Figure 21. Trajectory Maps of ViT models across different amounts of weight decay

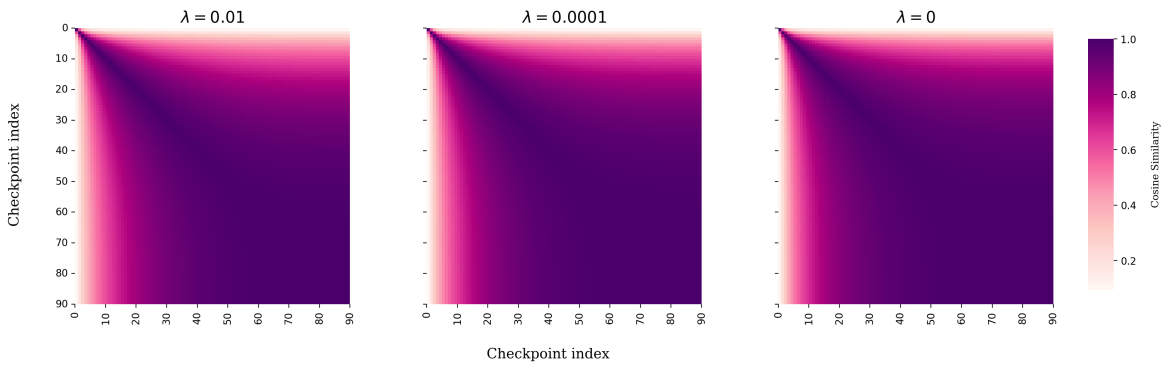
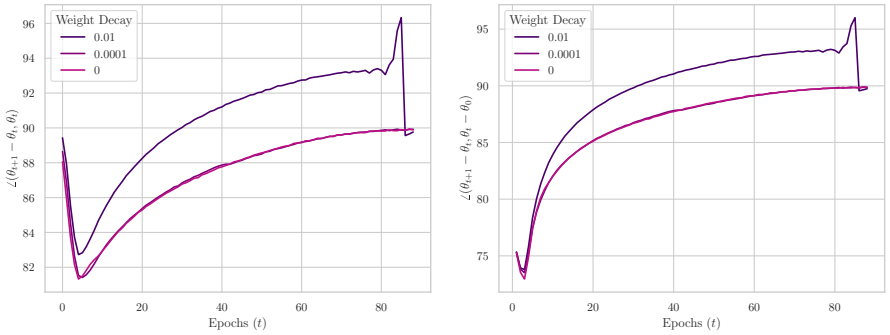
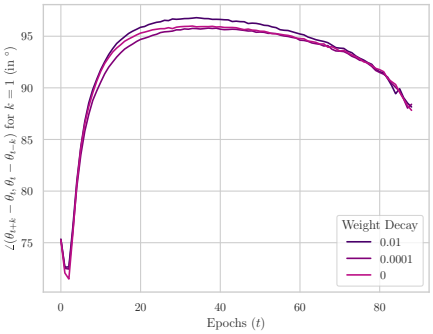


Figure 22. Relative Trajectory Maps, with respect to initialization, of ViT models across different amounts of weight decay

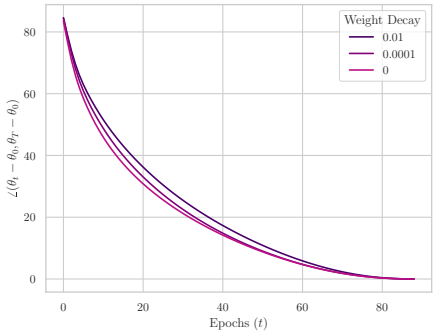


(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$

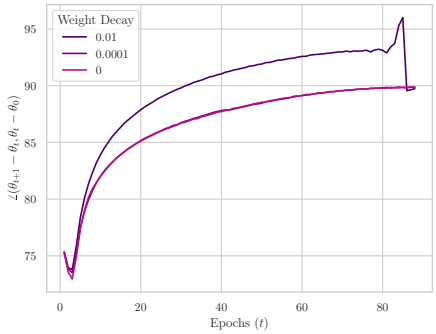
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



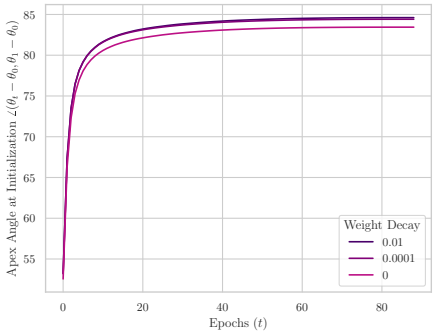
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



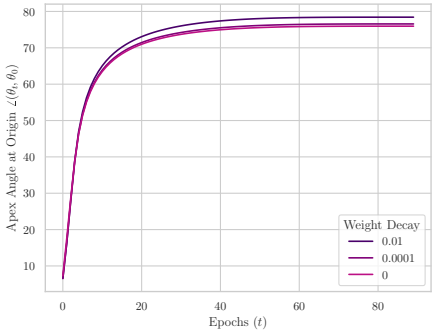
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

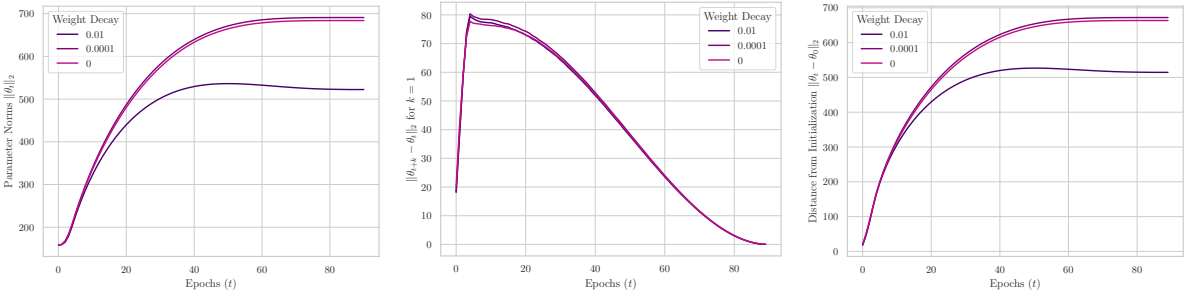


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



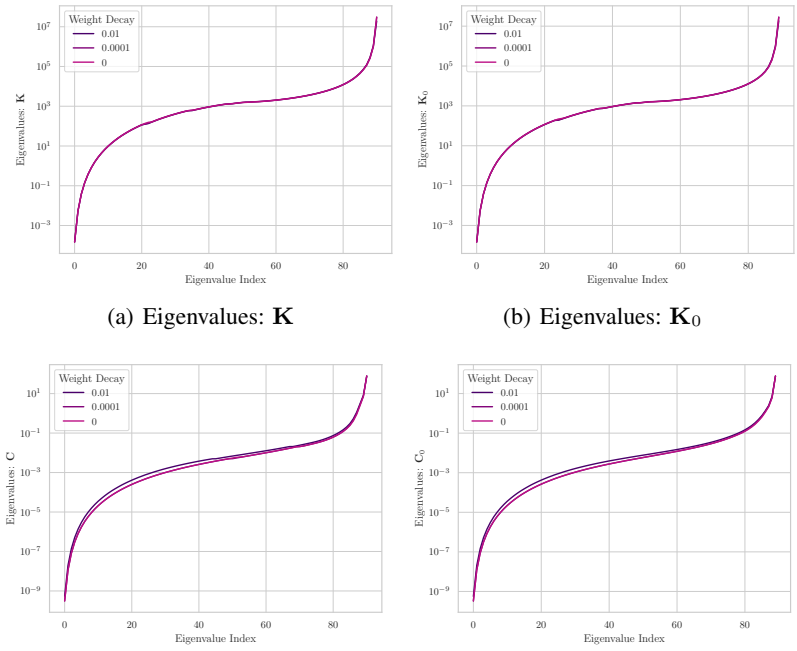
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 23. Angular measures of the Trajectory for ViT trained on the ImageNet dataset



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 24. Norm-based measures of the Trajectory for ViT trained on the ImageNet dataset



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 25. Spectral measures of the Trajectory for ViT trained on ImageNet

G.2. ResNet50: Weight Decay, SGD

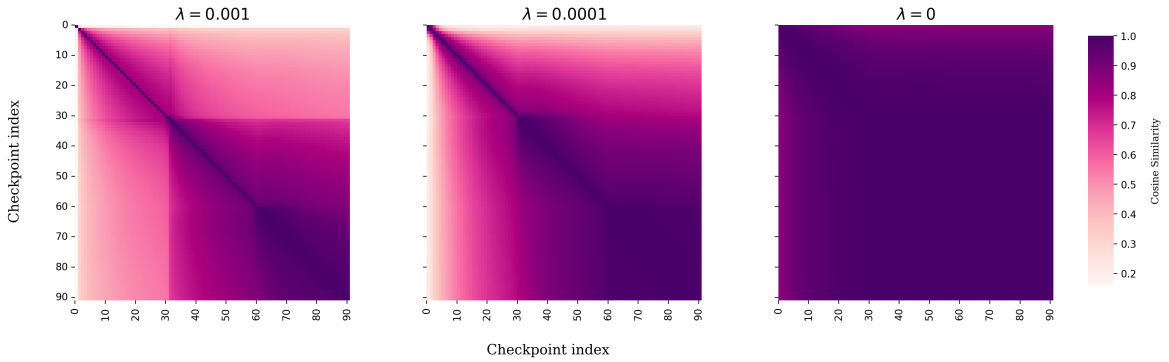


Figure 26. Trajectory Maps of ResNet50 models across different amounts of weight decay

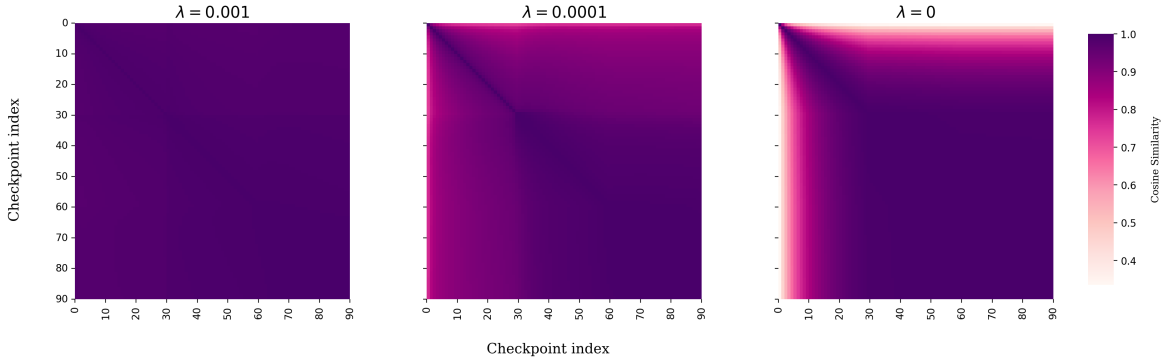
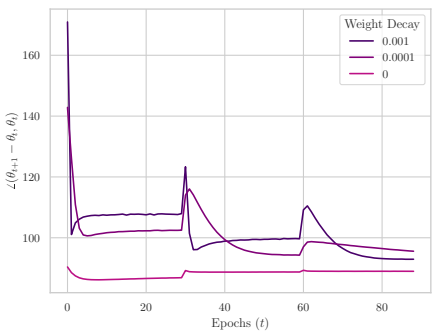
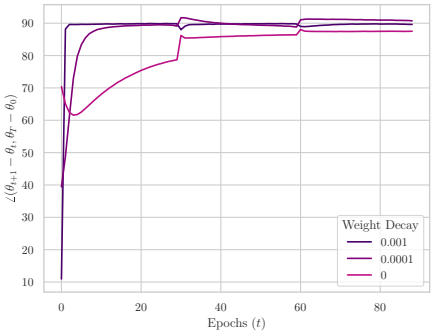


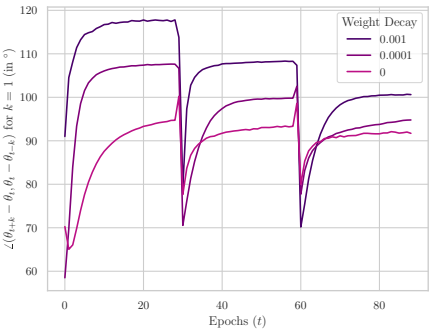
Figure 27. Relative Trajectory Maps, with respect to initialization, of ResNet50 models across different amounts of weight decay



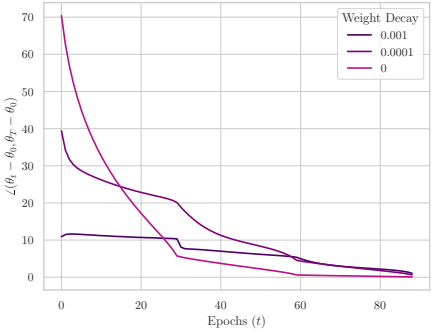
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



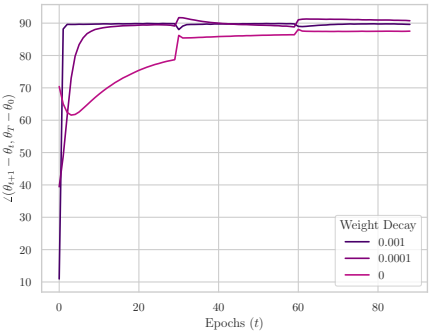
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



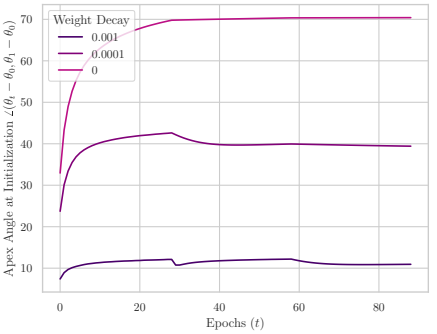
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



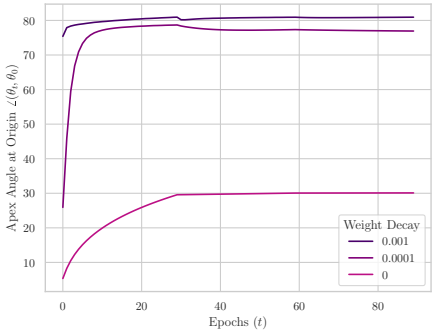
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

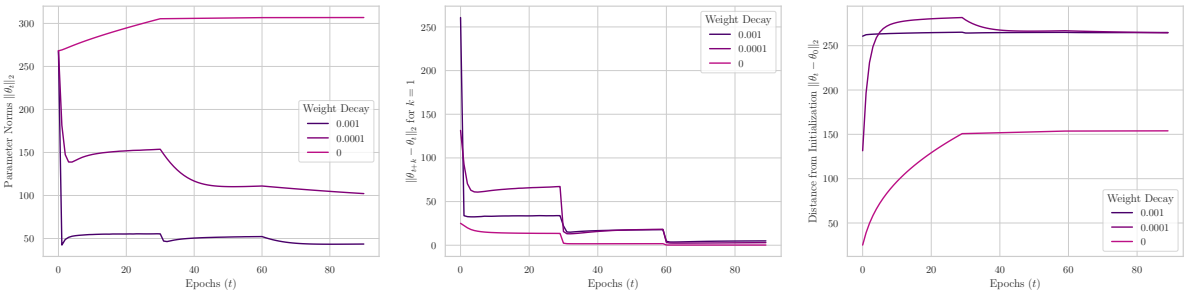


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



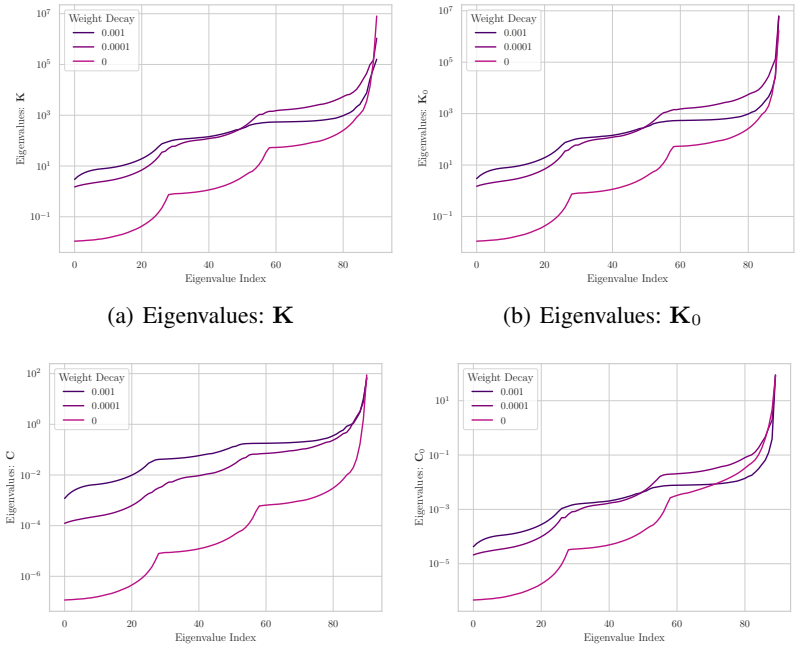
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 28. Angular measures of the Trajectory for ResNet50 trained on ImageNet



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 29. Norm-based measures of the Trajectory for ResNet50 trained on ImageNet



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 30. Spectral measures of the Trajectory for ResNet50 trained on ImageNet

G.3. ResNet50: Sharpness Aware Minimization analysis

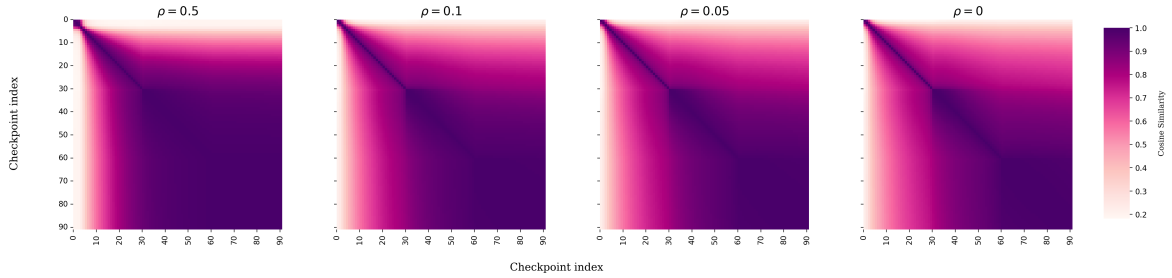


Figure 31. Trajectory Maps of ResNet50 models across different values of SAM regularization coefficient

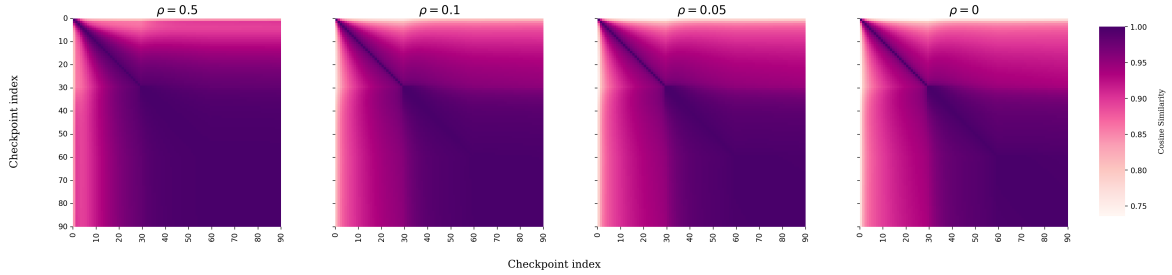
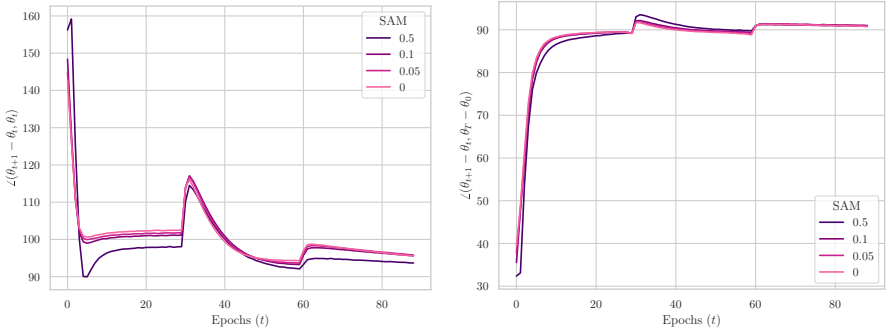
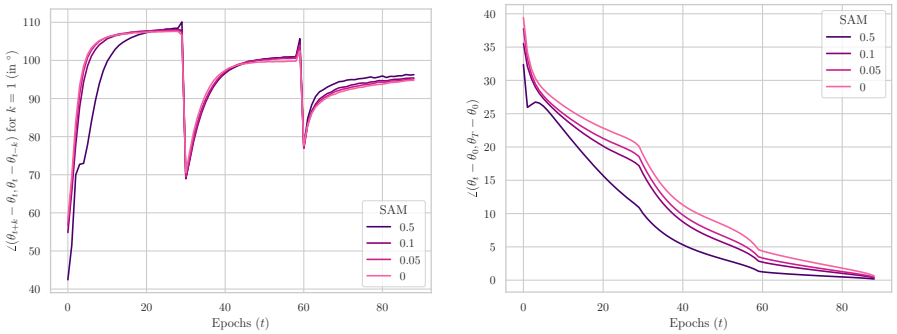


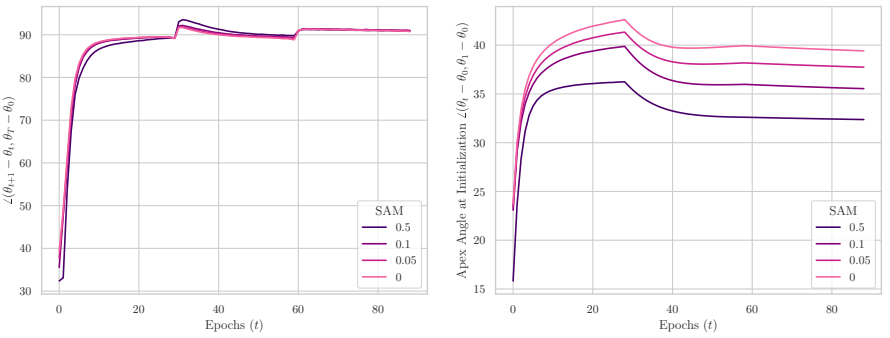
Figure 32. Relative Trajectory Maps, with respect to initialization, of ResNet50 models across different values of SAM regularization coefficient



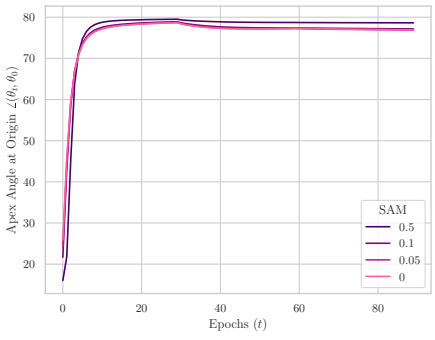
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$ (b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$ (d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$

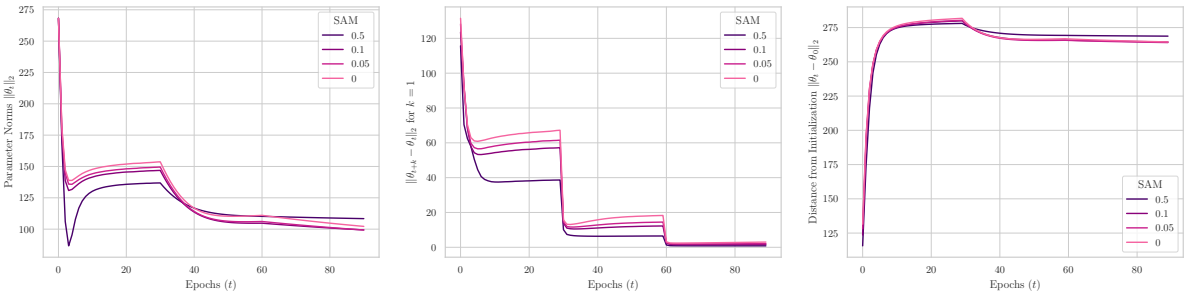


(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$ (f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



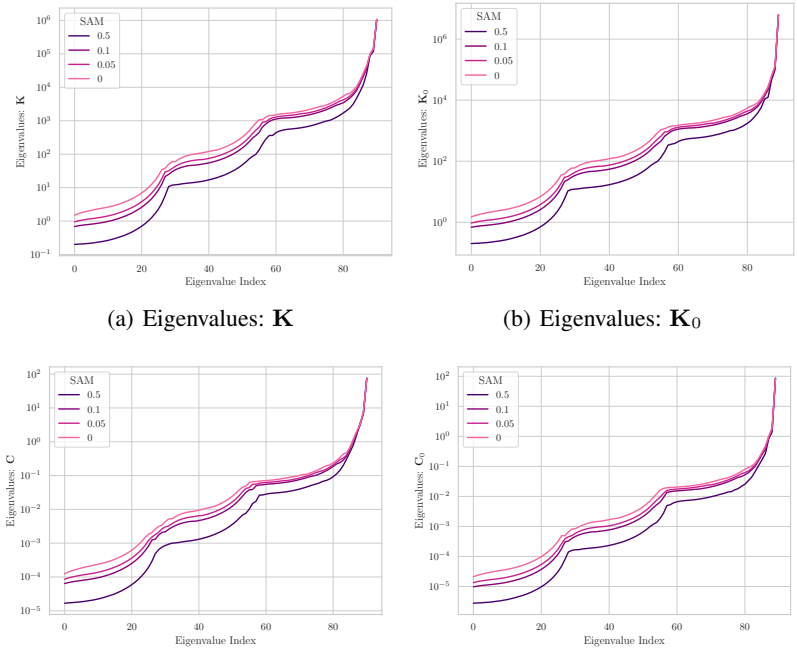
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 33. Angular measures of the Trajectory for ResNet50 trained on ImageNet



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 34. Norm-based measures of the Trajectory for ResNet50 trained on ImageNet



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 35. Spectral measures of the Trajectory for ResNet50 trained on ImageNet

G.4. ResNet50: Momentum Analysis, LR 0.1, WD 0.0001

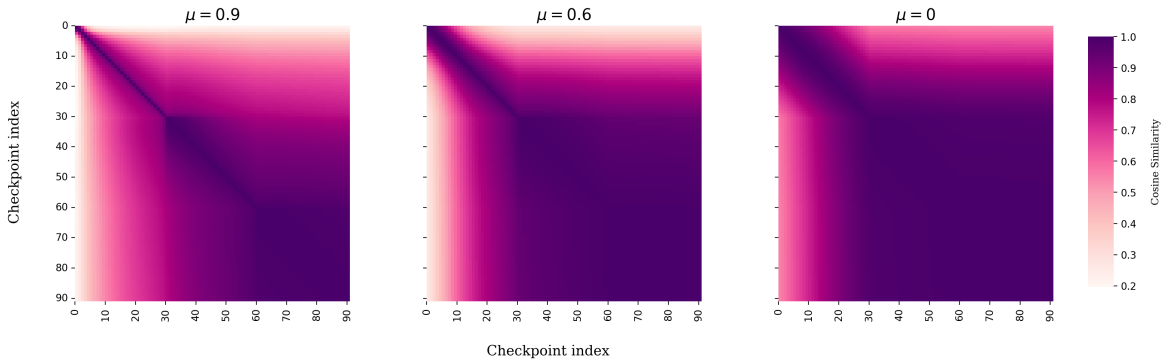


Figure 36. Trajectory Maps of ResNet50 models across different amounts of momentum

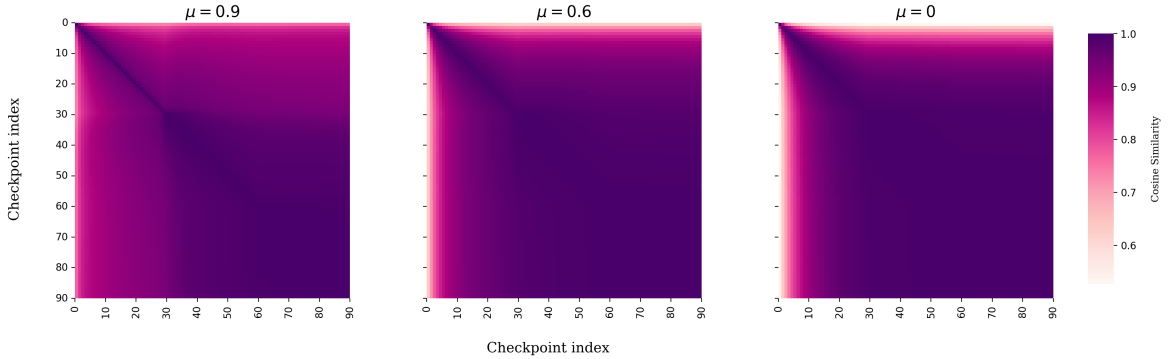
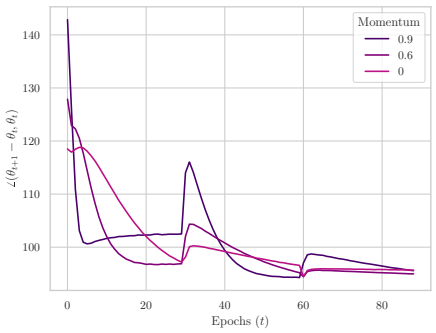
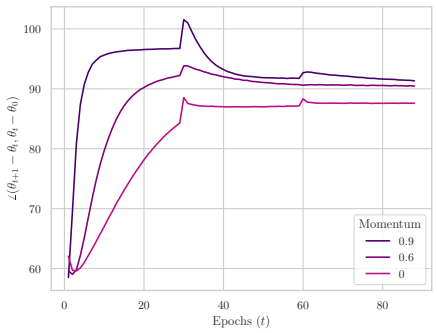


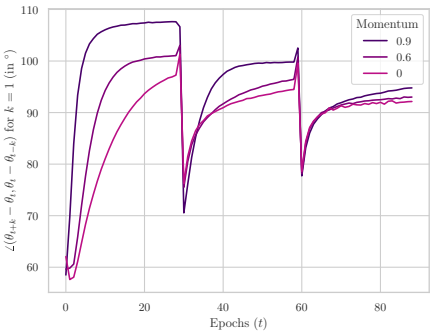
Figure 37. Relative Trajectory Maps, with respect to initialization, of ResNet50 models across different amounts of momentum



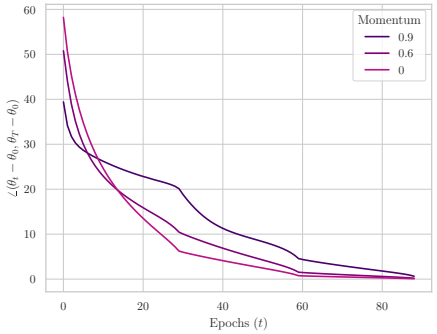
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



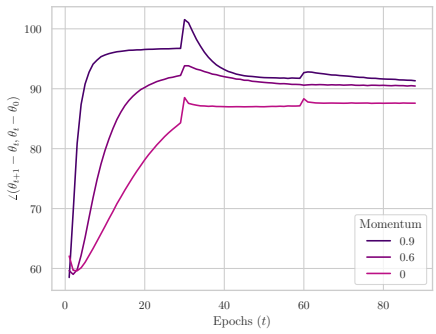
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



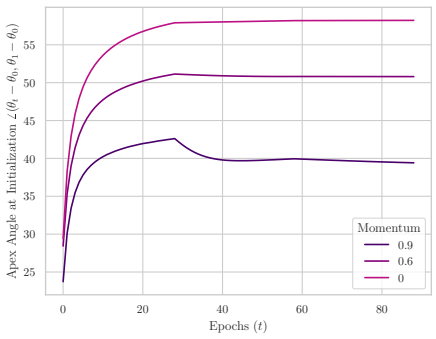
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



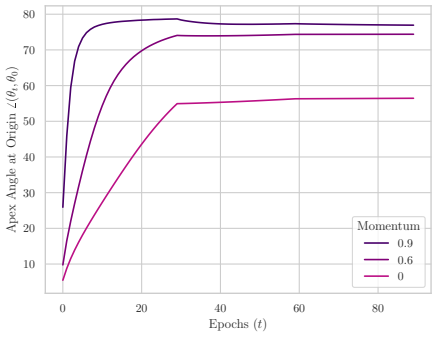
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

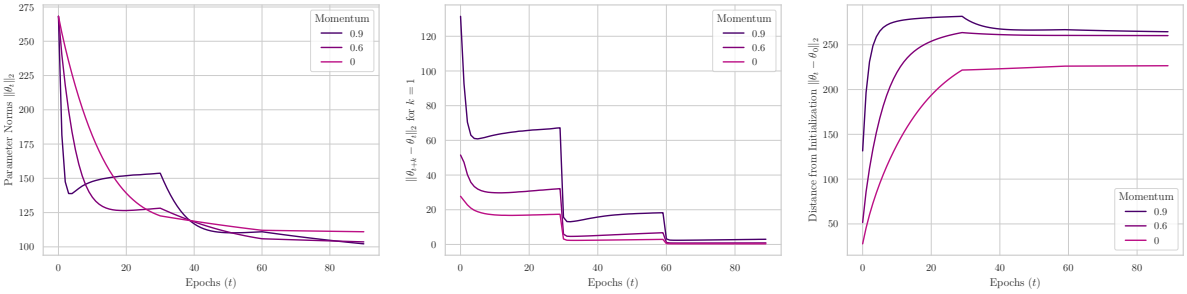


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



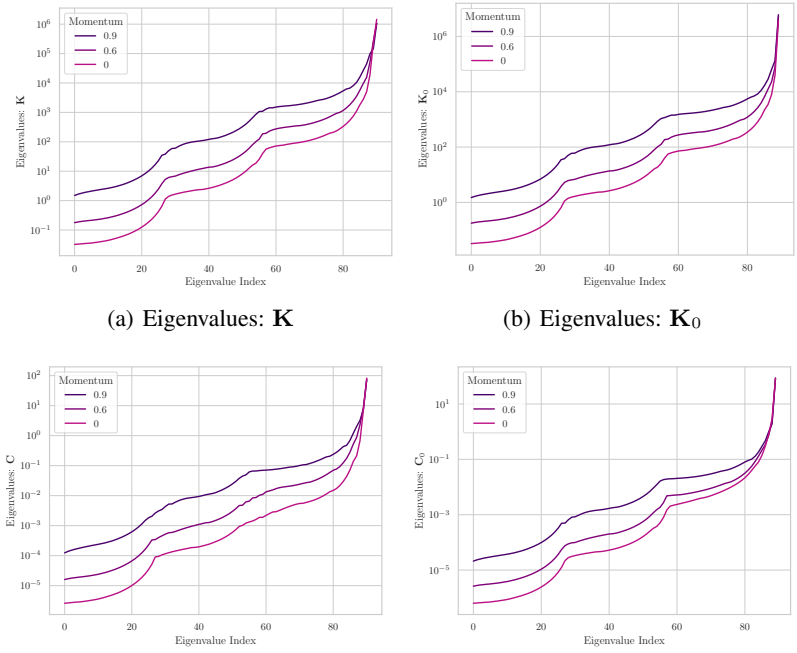
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 38. Angular measures of the Trajectory for ResNet50 trained on ImageNet



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 39. Norm-based measures of the Trajectory for ResNet50 trained on ImageNet



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 40. Spectral measures of the Trajectory for ResNet50 trained on ImageNet

G.5. VGG: Momentum Analysis, LR 0.1, WD 0.0001

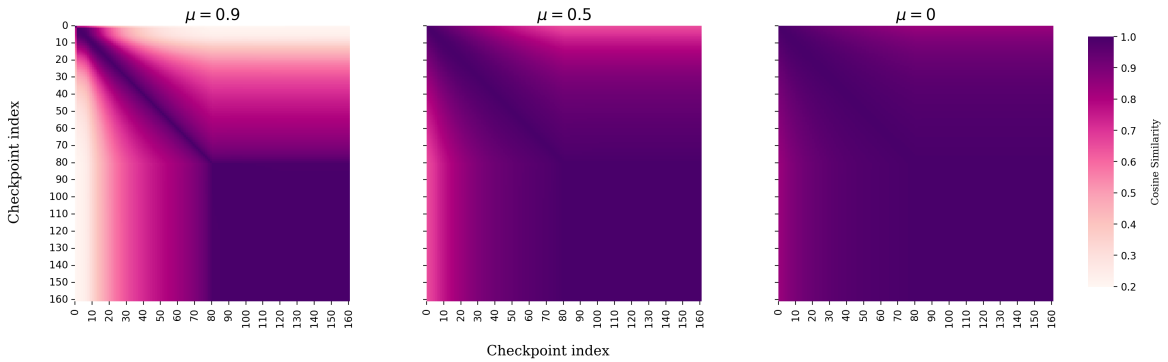


Figure 41. Trajectory Maps of VGG16 models across different amounts of momentum

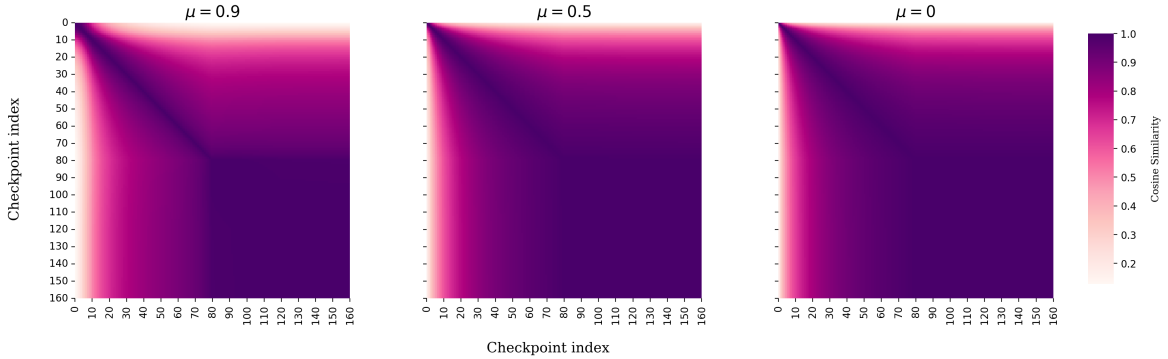
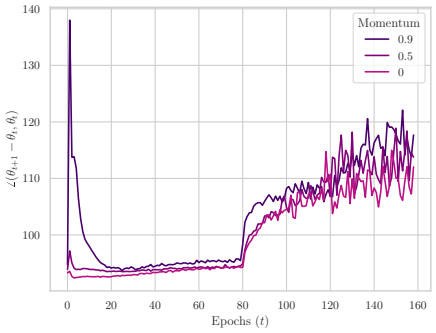
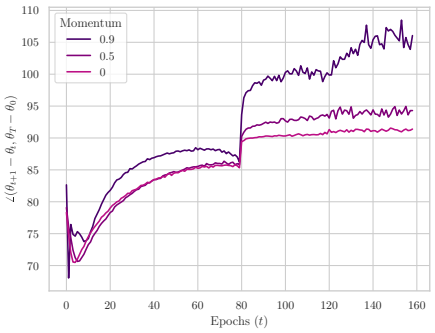


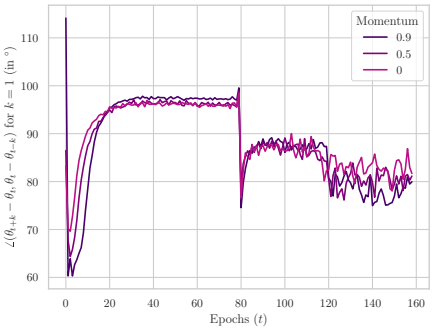
Figure 42. Relative Trajectory Maps, with respect to initialization, of VGG16 models across different amounts of momentum



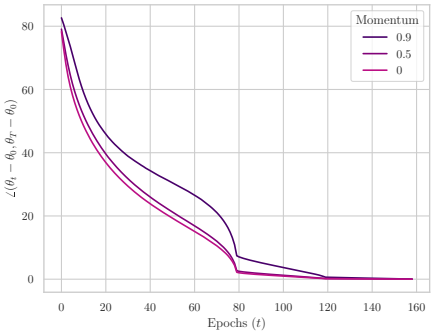
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



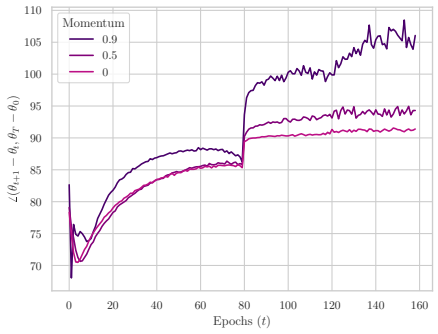
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



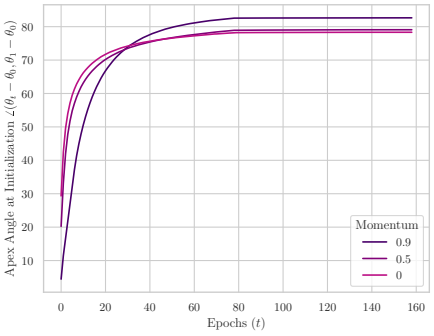
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



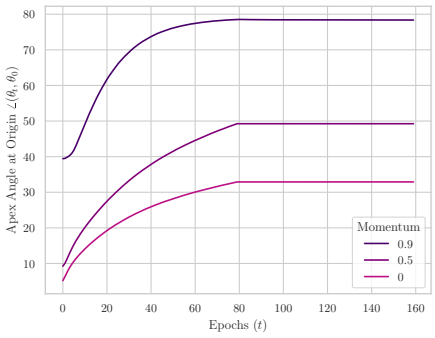
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

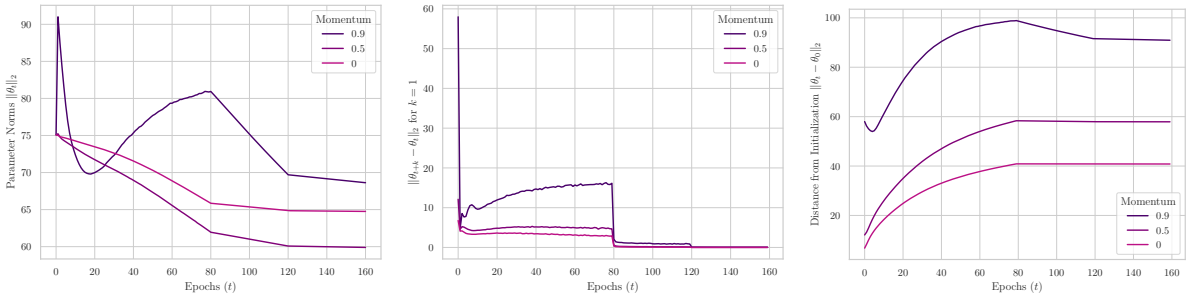


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



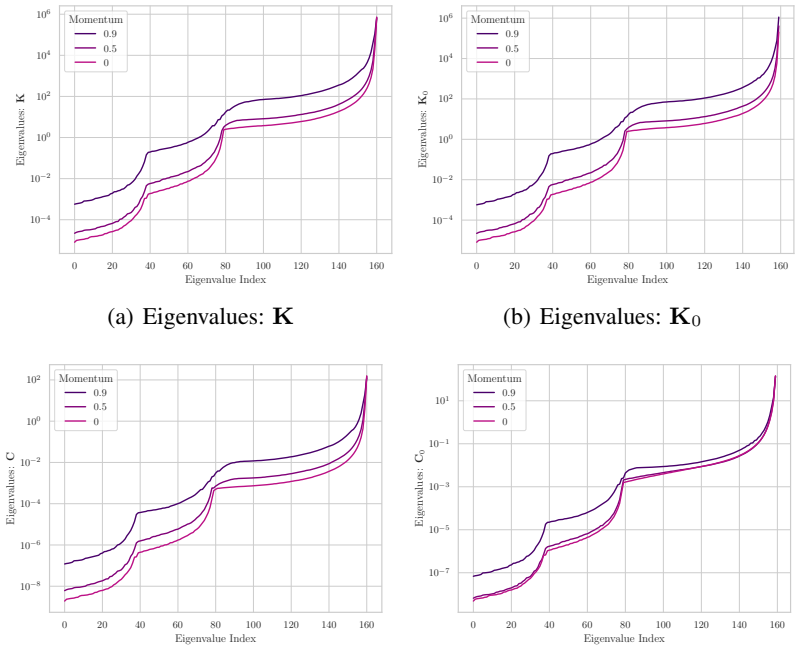
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 43. Angular measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 44. Norm-based measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 45. Spectral measures of the Trajectory for VGG16 models trained on CIFAR10.

G.6. VGG: Momentum Analysis, LR 0.1, WD 0

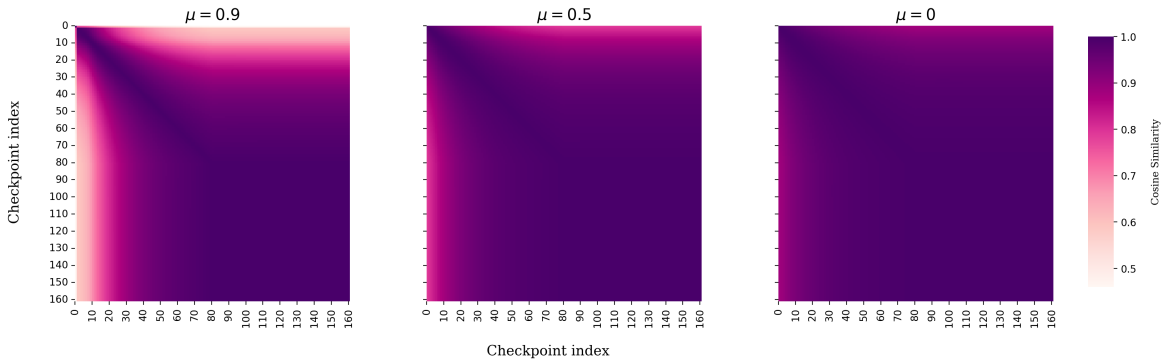


Figure 46. Trajectory Maps of VGG16 models across different amounts of momentum

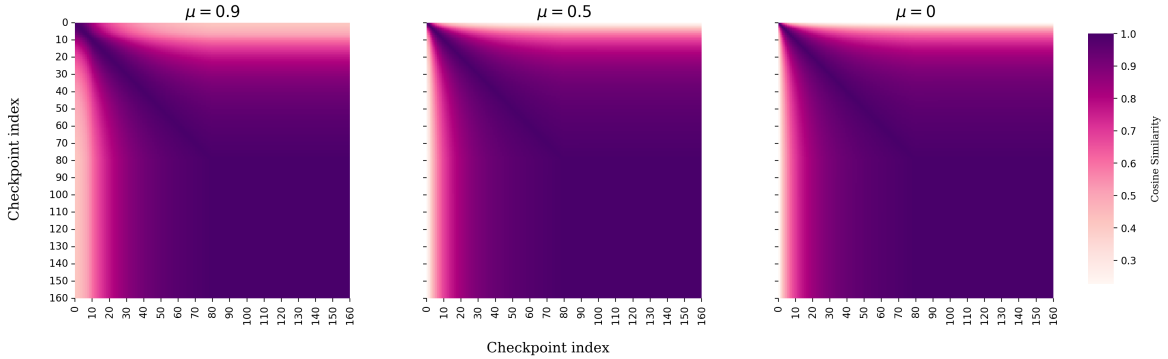
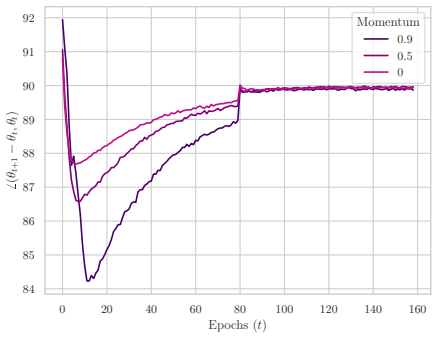
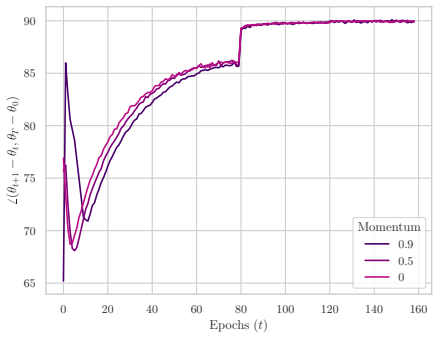


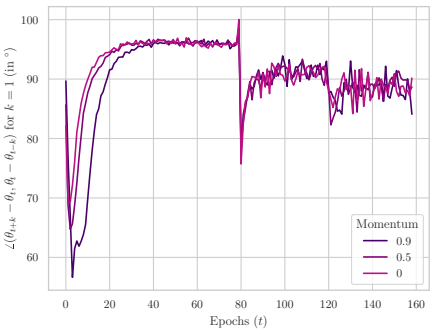
Figure 47. Relative Trajectory Maps, with respect to initialization, of VGG16 models across different amounts of momentum



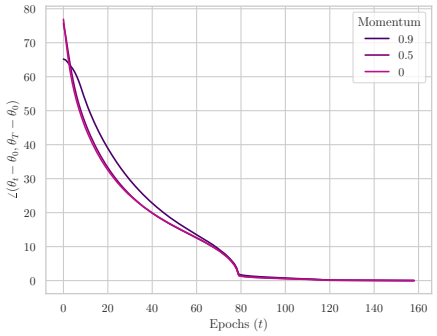
(a) $\angle(\theta_{t+1}-\theta_t,\theta_t)$



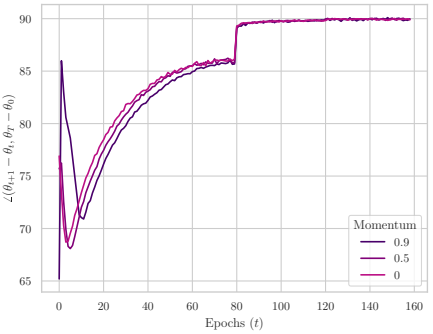
(b) $\angle(\theta_{t+1}-\theta_t,\theta_T-\theta_0)$



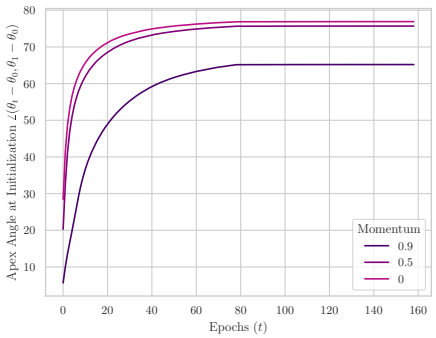
(c) $\angle(\theta_{t+k}-\theta_t,\theta_t-\theta_{t-k})$, for $k=1$



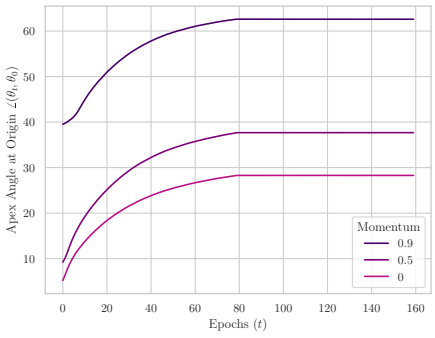
(d) $\angle(\theta_t-\theta_0,\theta_T-\theta_0)$



(e) $\angle(\theta_{t+1}-\theta_t,\theta_T-\theta_0)$

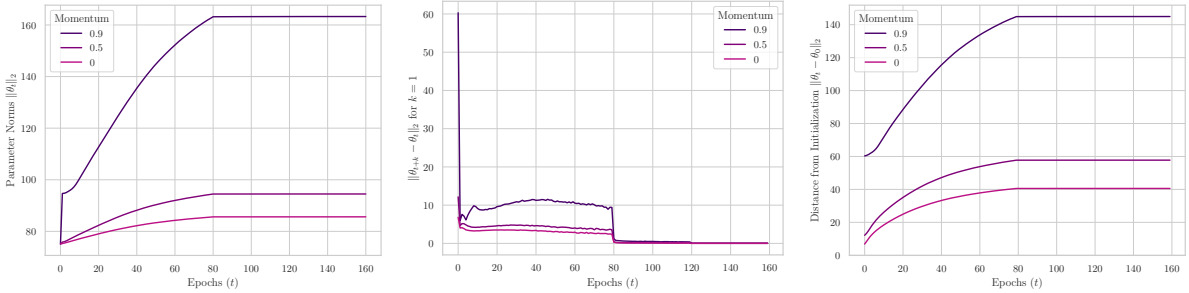


(f) Apex Angle at Initialization $\angle(\theta_t-\theta_0,\theta_1-\theta_0)$



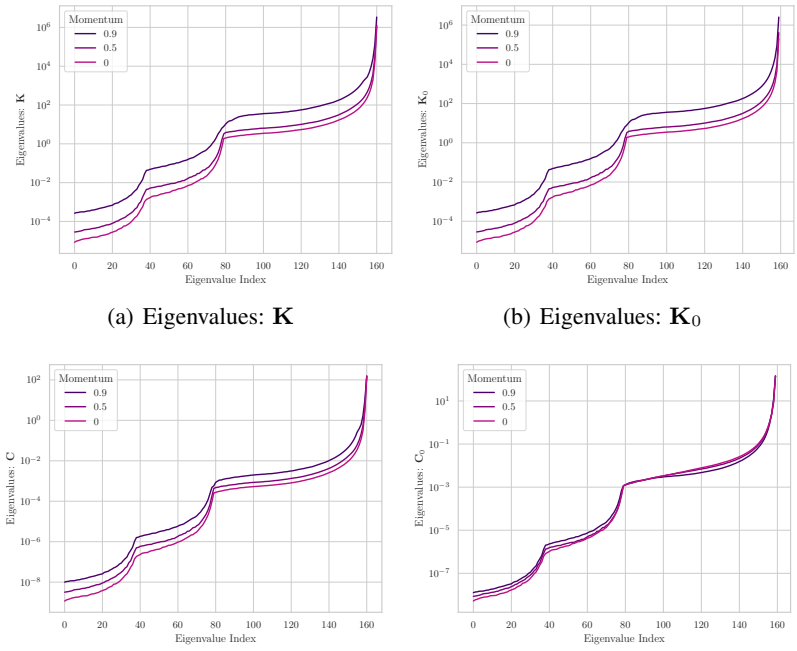
(g) Apex Angle at Origin $\angle(\theta_t,\theta_0)$

Figure 48. Angular measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 49. Norm-based measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 50. Spectral measures of the Trajectory for VGG16 models trained on CIFAR10.

G.7. VGG: Momentum Analysis, LR 0.01, WD 0.0001

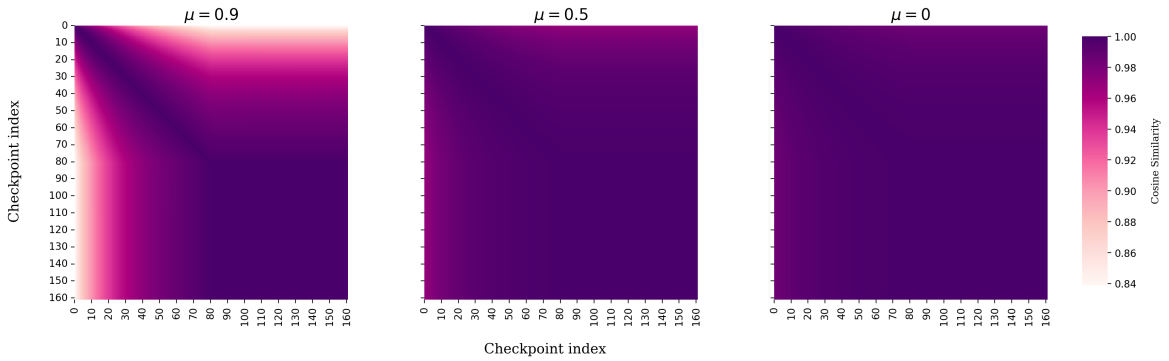


Figure 51. Trajectory Maps of VGG16 models across different amounts of momentum

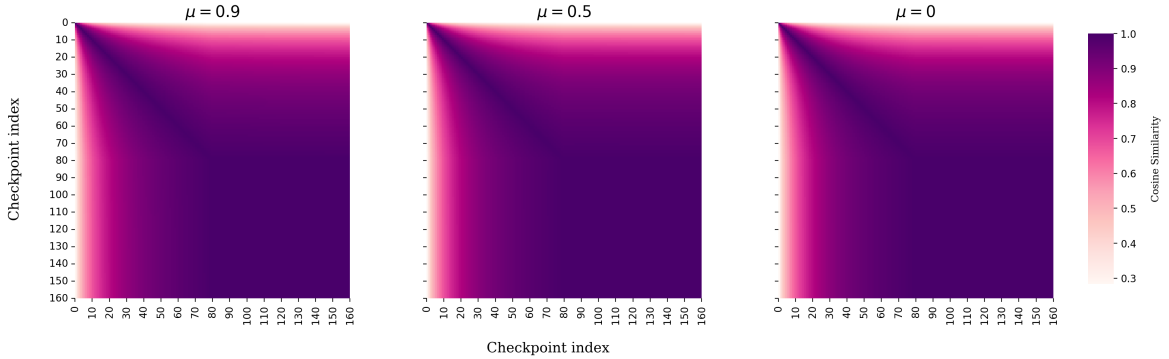
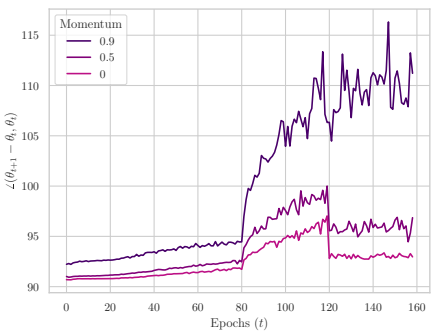
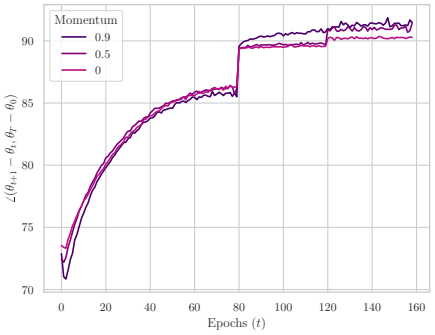


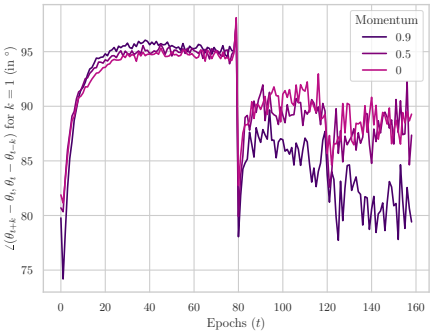
Figure 52. Relative Trajectory Maps, with respect to initialization, of VGG16 models across different amounts of momentum



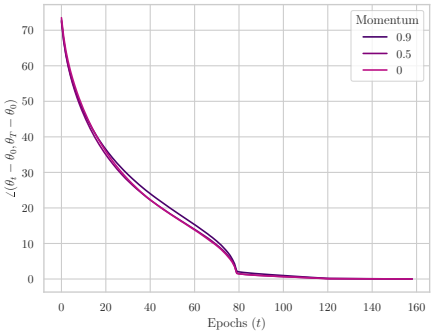
(a) $\angle(\theta_{t+1}-\theta_t,\theta_t)$



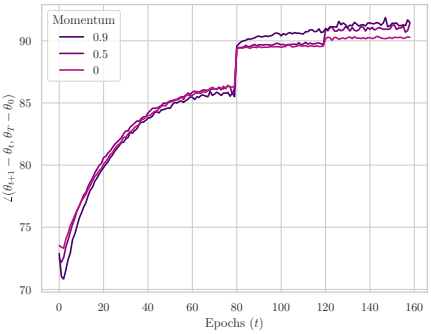
(b) $\angle(\theta_{t+1}-\theta_t,\theta_T-\theta_0)$



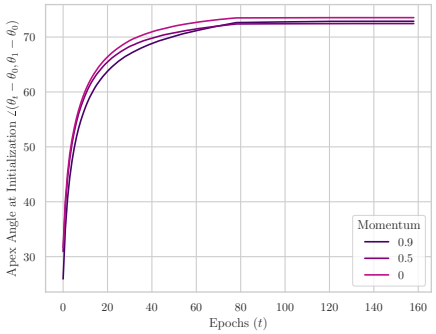
(c) $\angle(\theta_{t+k}-\theta_t,\theta_t-\theta_{t-k})$, for $k=1$



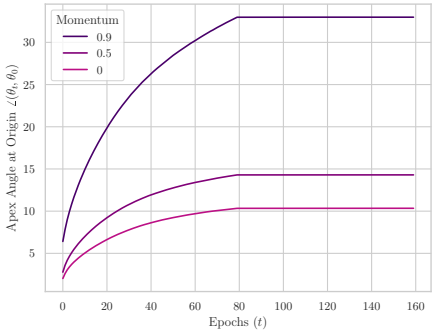
(d) $\angle(\theta_t-\theta_0,\theta_T-\theta_0)$



(e) $\angle(\theta_{t+1}-\theta_t,\theta_T-\theta_0)$

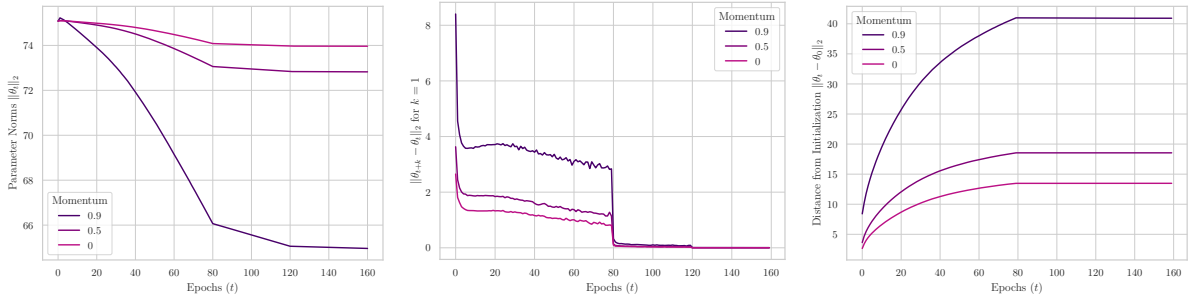


(f) Apex Angle at Initialization $\angle(\theta_t-\theta_0,\theta_1-\theta_0)$



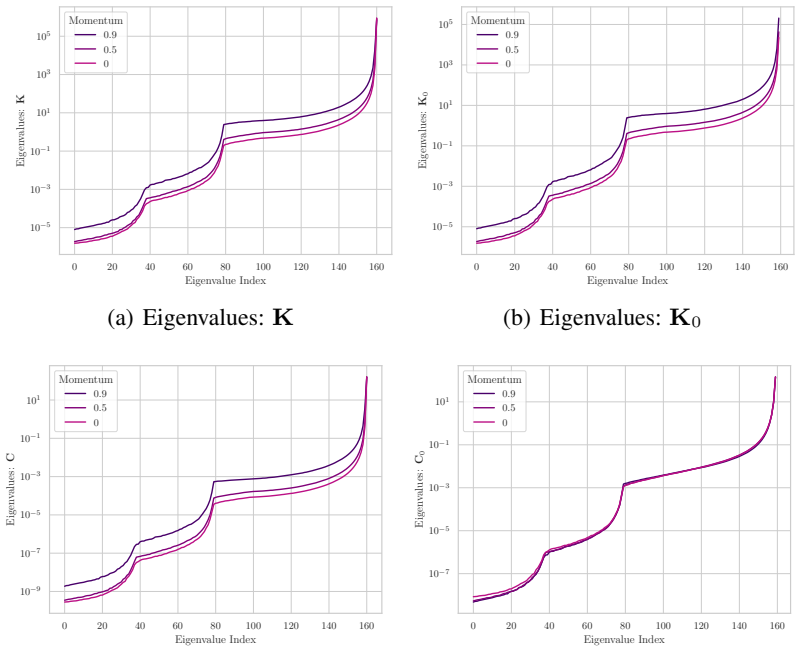
(g) Apex Angle at Origin $\angle(\theta_t,\theta_0)$

Figure 53. Angular measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 54. Norm-based measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 55. Spectral measures of the Trajectory for VGG16 models trained on CIFAR10.

G.8. VGG: Momentum Analysis, LR 0.01, WD 0

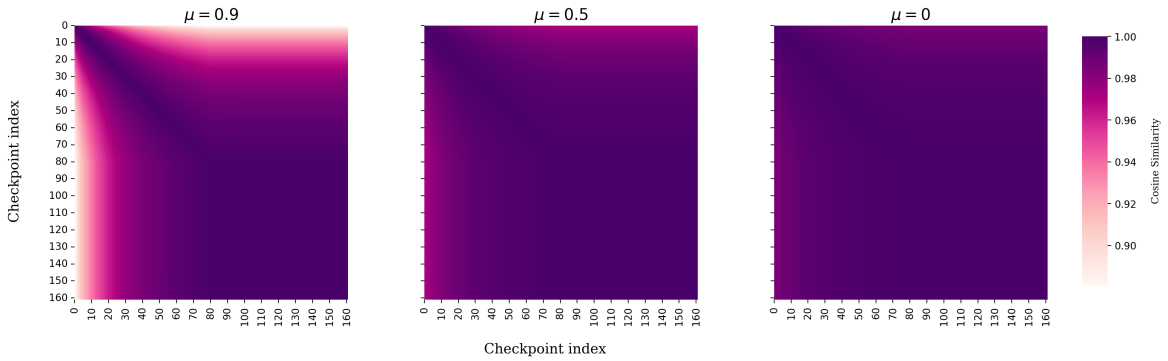


Figure 56. Trajectory Maps of VGG16 models across different amounts of momentum

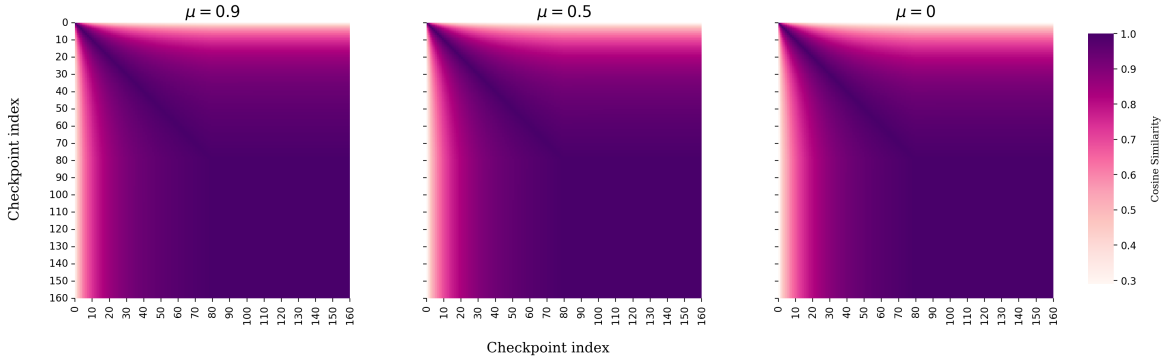
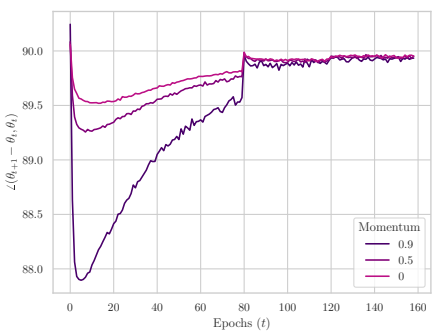
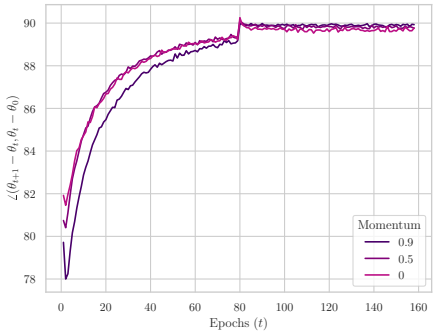


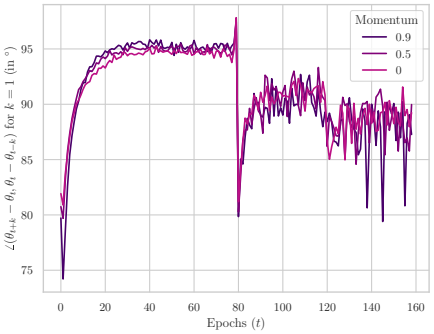
Figure 57. Relative Trajectory Maps, with respect to initialization, of VGG16 models across different amounts of momentum



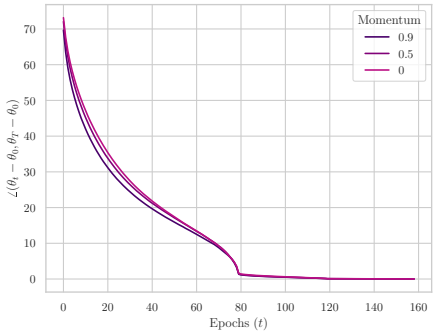
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



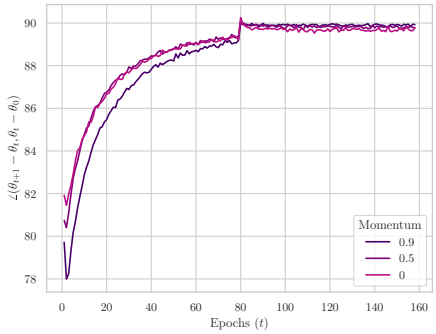
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



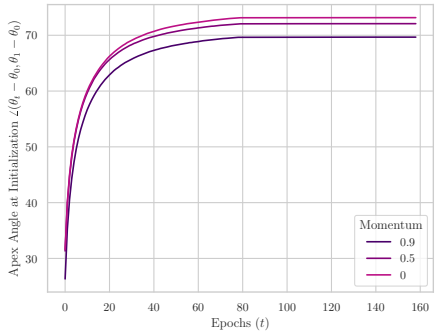
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



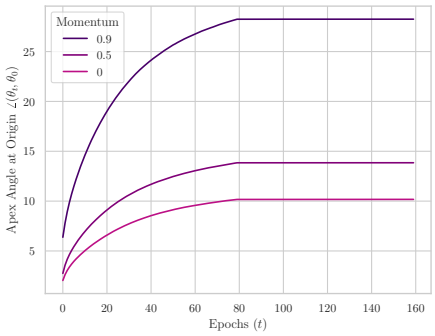
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

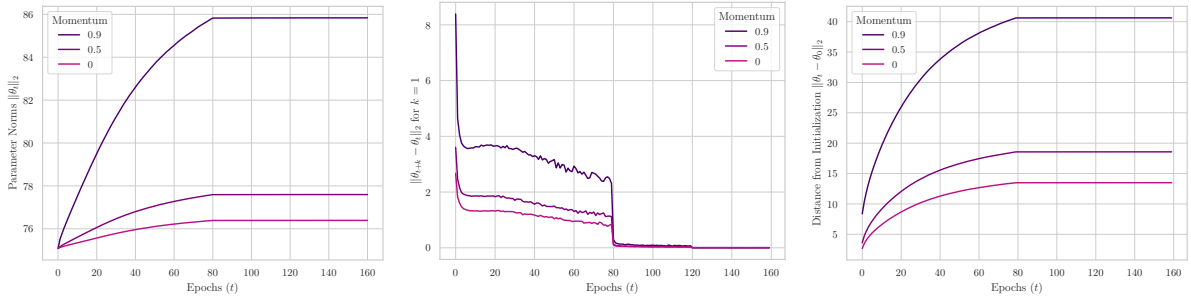


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



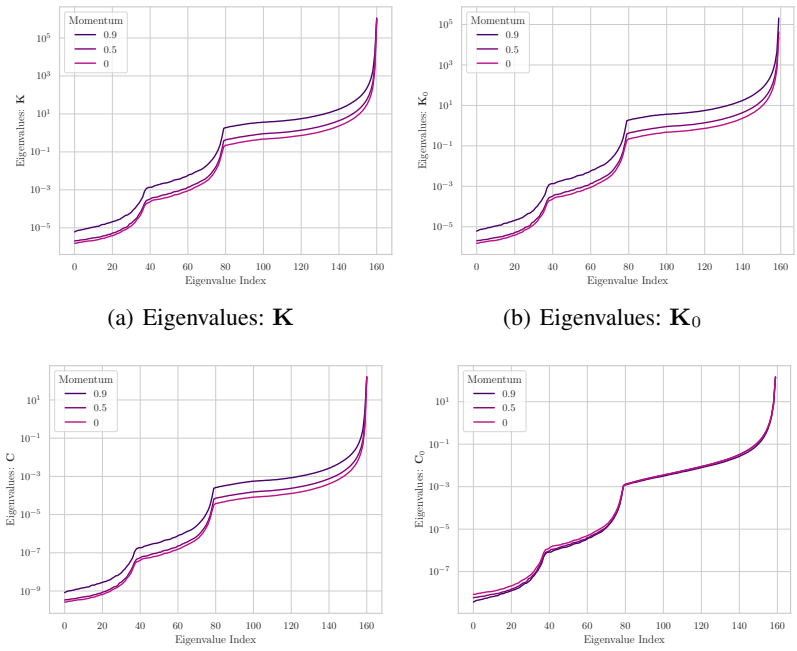
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 58. Angular measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

Figure 59. Norm-based measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$

(c) Eigenvalues: $\mathbf{C}$ (d) Eigenvalues: $\mathbf{C}_0$

Figure 60. Spectral measures of the Trajectory for VGG16 models trained on CIFAR10.

G.9. VGG16 Batch Size Analysis

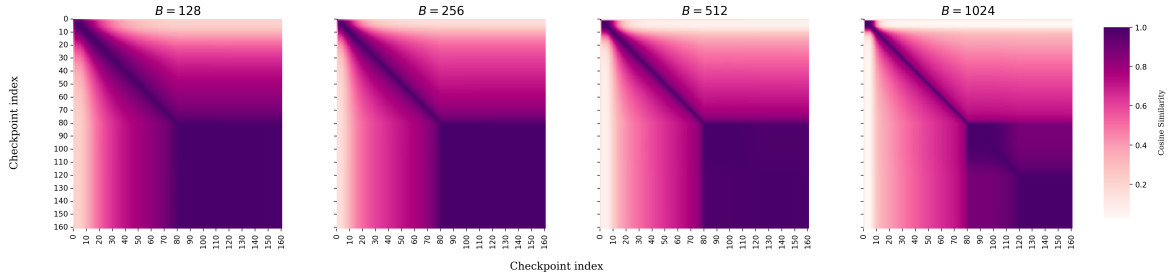


Figure 61. Trajectory Maps of VGG16 models across different batch sizes. The learning rates have been scaled in proportion to the batch size, and the training schedule was adjusted to ensure an equal number of steps (and not simply epochs) for all the runs. We also adjusted the learning rate schedule to drop learning rates at a corresponding number of steps across the experiments. The respective MDS values are $\omega = 0.753, 0.723, 0.660, 0.619$ and the test accuracies are $91.63\%, 91.82\%, 92.44\%, 92.39\%$.

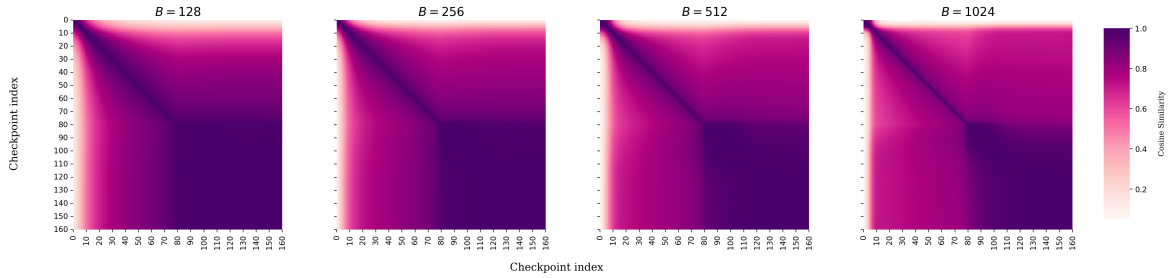
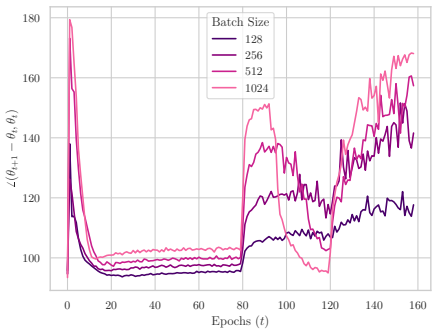


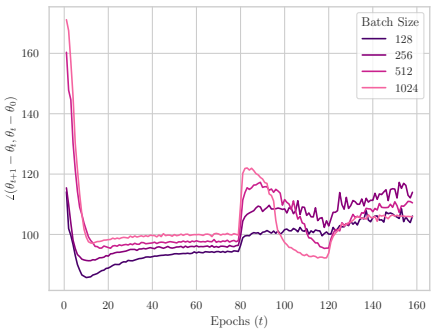
Figure 62. Relative Trajectory Maps, with respect to initialization, of VGG16 models across different batch sizes

G.10. Trajectory Maps in the presence of label noise

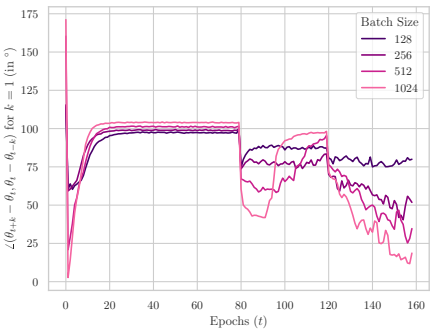
We observe that with increasing label noise, the network is required to undergo more directional exploration to find a solution that can interpolate the training set. The MDS scores decrease monotonically with increasing label noise.



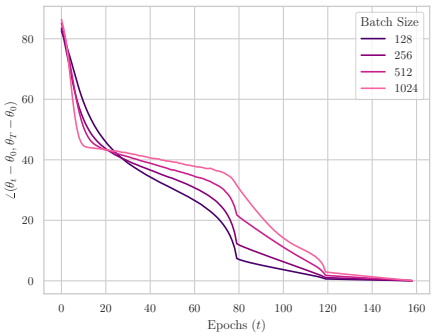
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



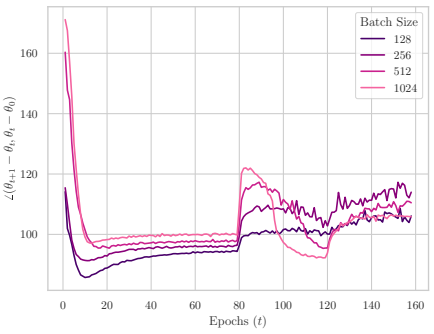
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



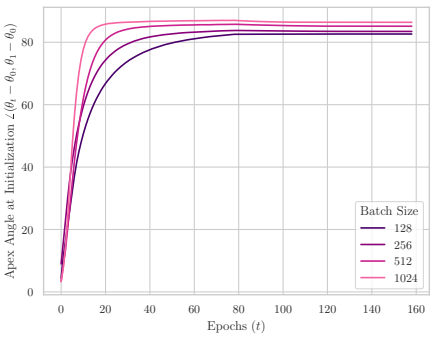
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



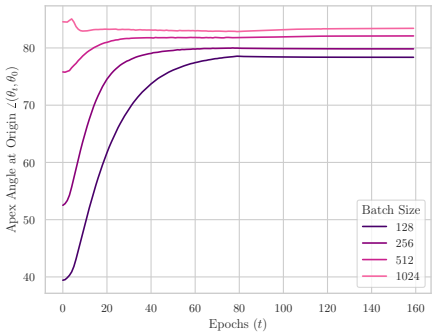
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

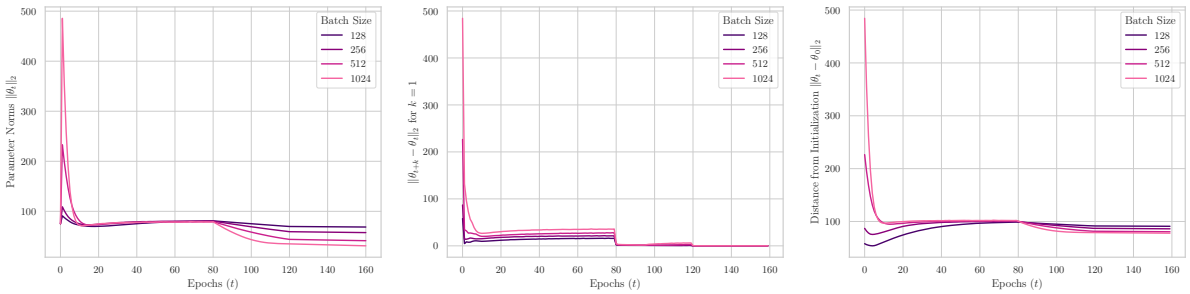


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



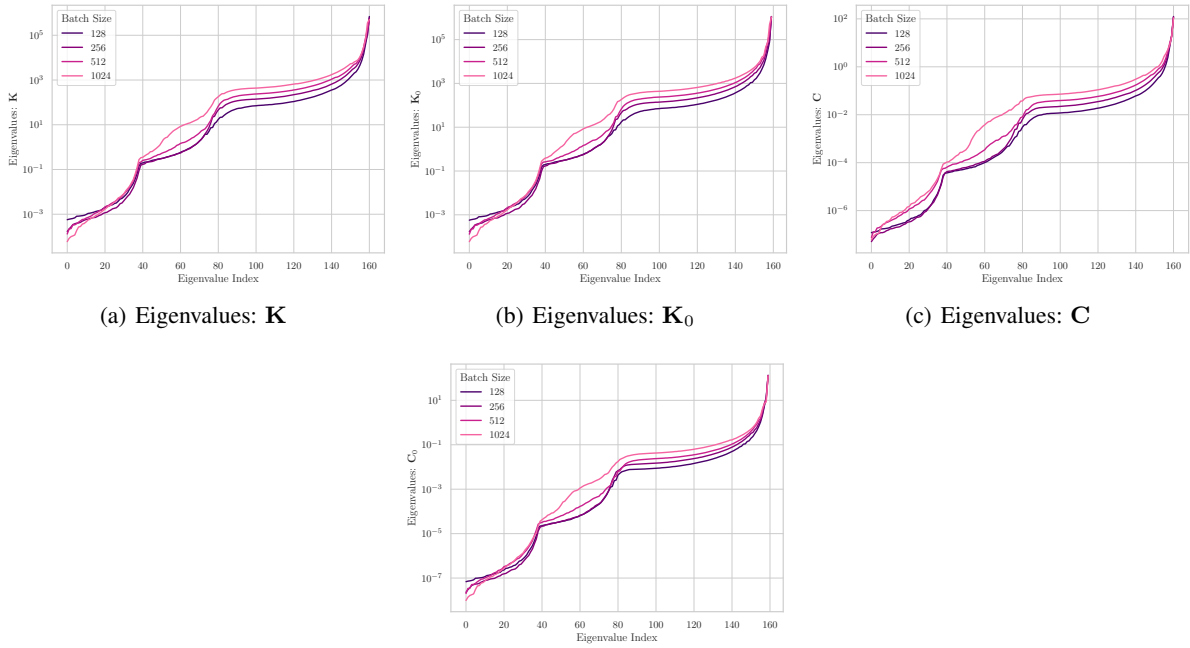
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 63. Angular measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

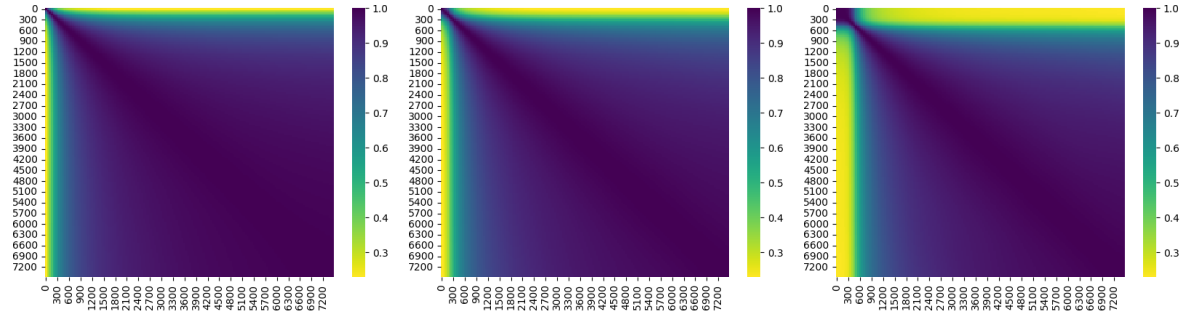
Figure 64. Norm-based measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) Eigenvalues: $\mathbf{K}$ (b) Eigenvalues: $\mathbf{K}_0$ (c) Eigenvalues: $\mathbf{C}$

(d) Eigenvalues: $\mathbf{C}_0$

Figure 65. Spectral measures of the Trajectory for VGG16 models trained on CIFAR10.



(a) Label Noise 0.4 (b) Label Noise 0.7 (c) Label Noise 1.0

Figure 66. Trajectory maps when a CNN is trained on CIFAR10 with different amounts of label noise, i.e., what fraction of samples have been assigned random labels.

H. GPT-NeoX Trajectory Analysis

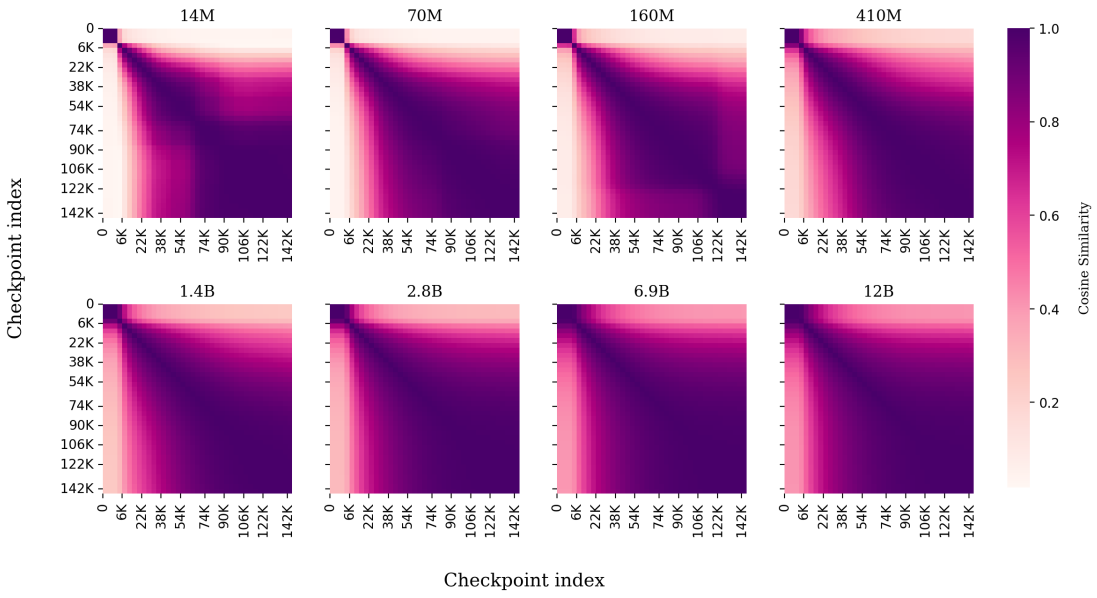


Figure 67. Trajectory Maps of Pythia GPT-NeoX models across two orders of model scales trained on Pile. The corresponding MDS values are $\omega = 0.650, 0.672, 0.678, 0.726, 0.759, 0.786, 0.818, 0.815$.

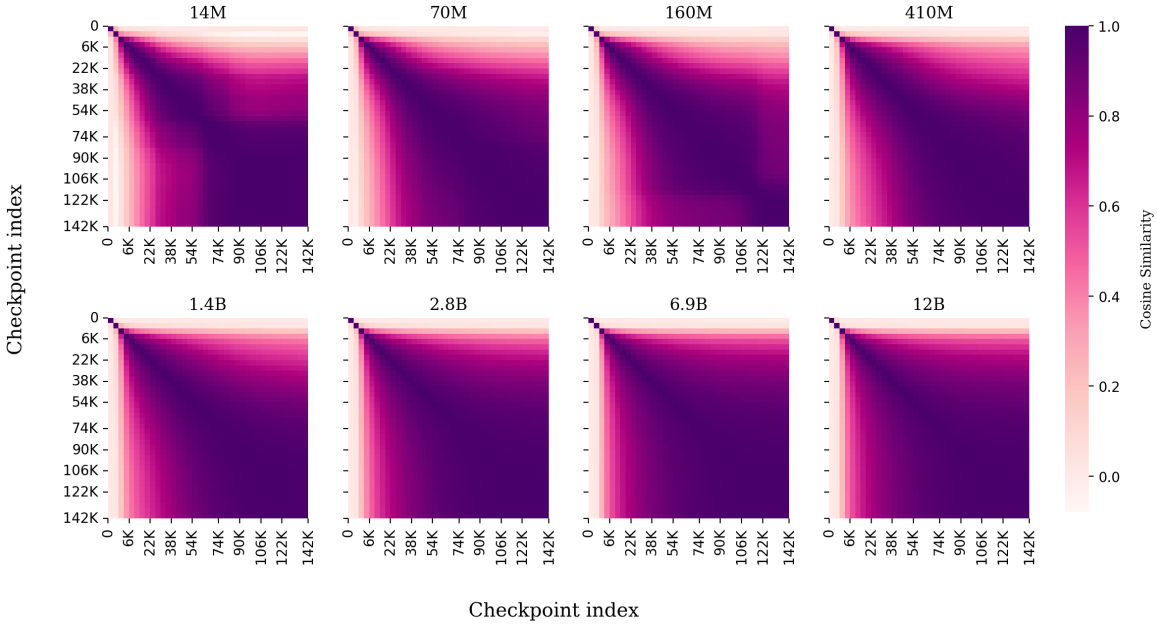
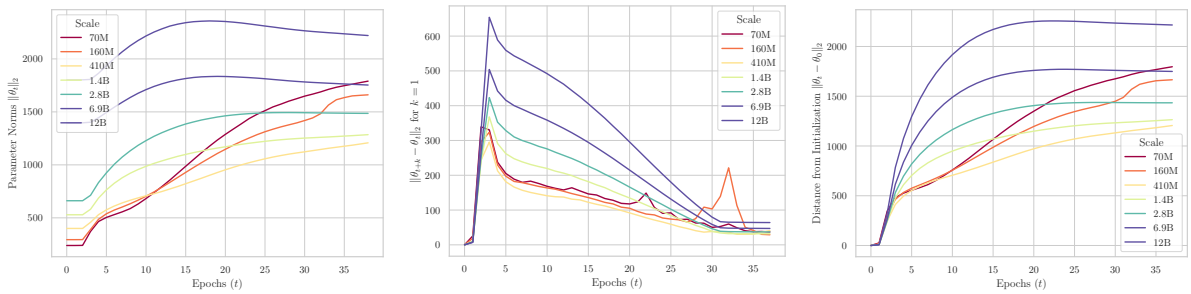
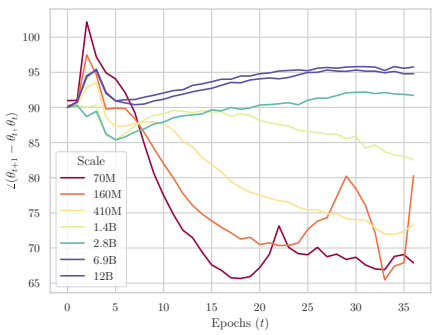


Figure 68. Relative Trajectory Maps, with respect to initialization, of Pythia GPT-NeoX models across two orders of model scales.

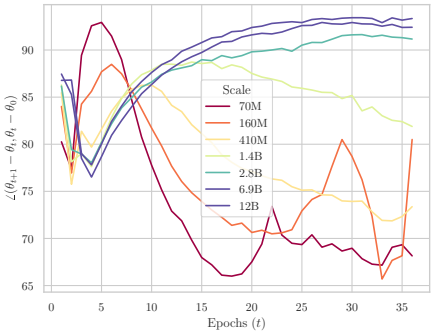


(a) $\|\theta_t\|_2$ (b) $\|\theta_{t+k} - \theta_t\|_2$ (c) $\|\theta_t - \theta_0\|_2$

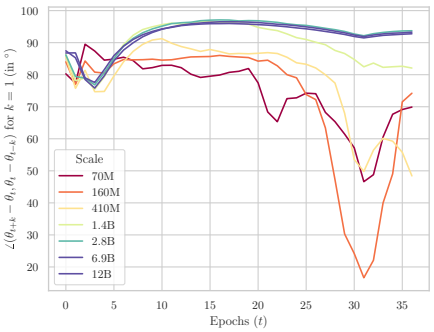
Figure 69. Norm-based measures of the Trajectory for GPT-NeoX trained on the Pile dataset



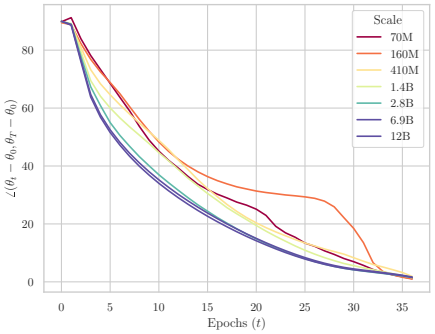
(a) $\angle(\theta_{t+1} - \theta_t, \theta_t)$



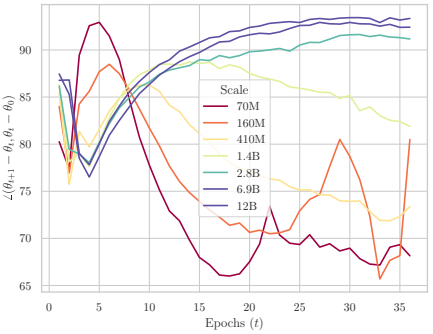
(b) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$



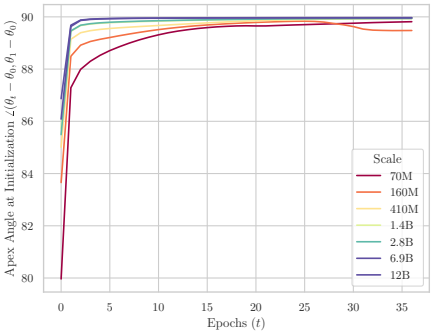
(c) $\angle(\theta_{t+k} - \theta_t, \theta_t - \theta_{t-k})$, for $k = 1$



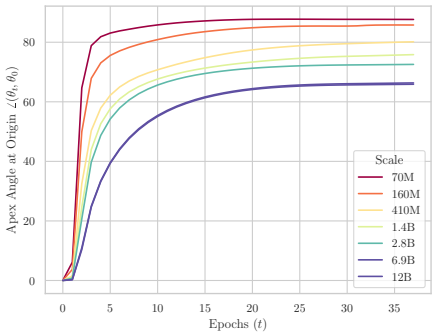
(d) $\angle(\theta_t - \theta_0, \theta_T - \theta_0)$



(e) $\angle(\theta_{t+1} - \theta_t, \theta_T - \theta_0)$

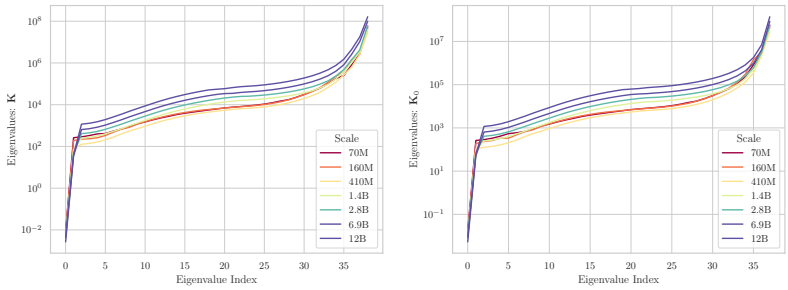


(f) Apex Angle at Initialization $\angle(\theta_t - \theta_0, \theta_1 - \theta_0)$



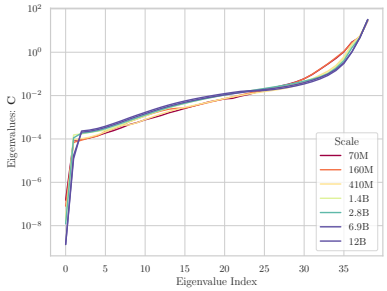
(g) Apex Angle at Origin $\angle(\theta_t, \theta_0)$

Figure 70. Angular measures of the Trajectory for GPT-NeoX trained on the Pile dataset

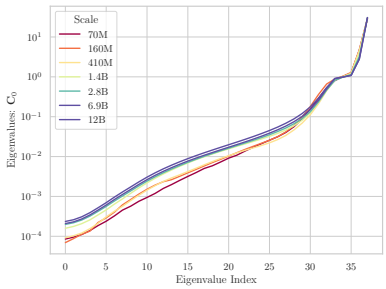


(a) Eigenvalues: $\mathbf{K}$

(b) Eigenvalues: $\mathbf{K}_0$



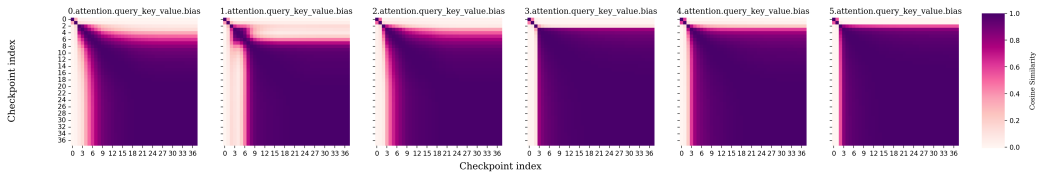
(c) Eigenvalues: $\mathbf{C}$



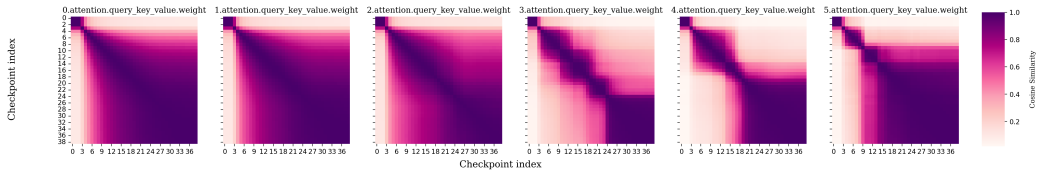
(d) Eigenvalues: $\mathbf{C}_0$

Figure 71. Spectral measures of the Trajectory for GPT-NeoX trained on the Pile dataset

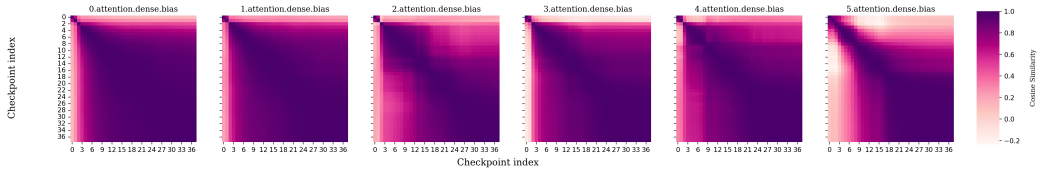
I. Layerwise-Trajectory Maps



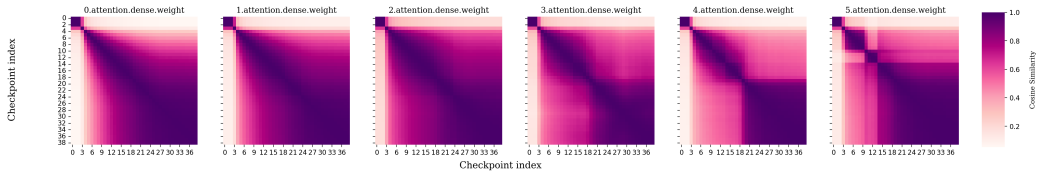
(a) query-key-value, bias



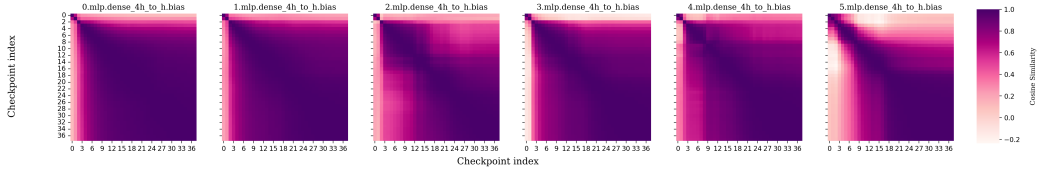
(b) query-key-value, weight



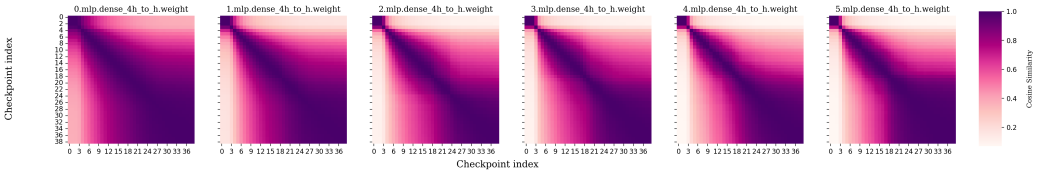
(c) dense, bias



(d) dense, weight

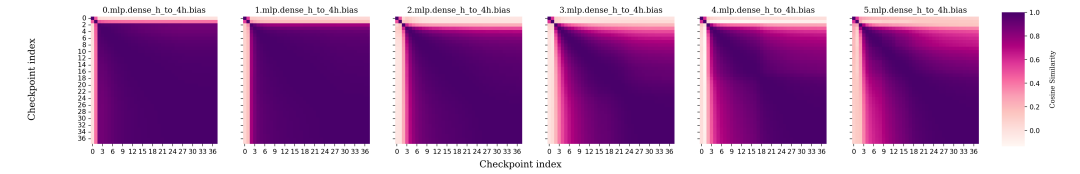


(e) dense-4h-to-h, bias

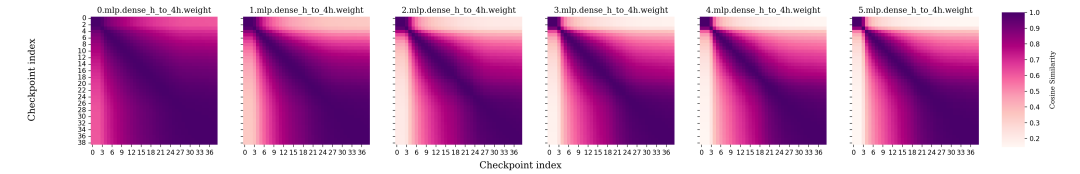


(f) dense-4h-to-h, weight

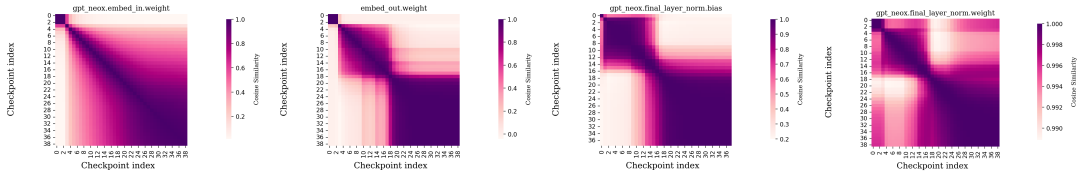
Figure 72. Layerwise Trajectory Maps, grouped by layer type, for the $14M$ GPT-NeoX model trained on the Pile dataset.



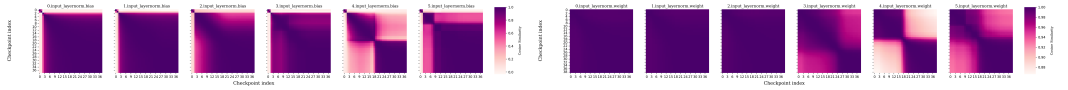
(a) dense-h-to-4h, bias



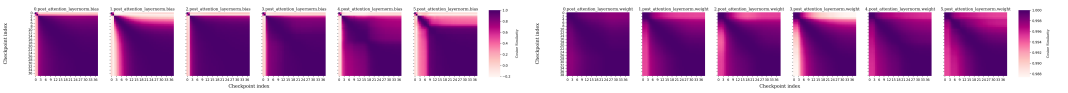
(b) dense-h-to-4h, weight



(c) embed-in, weight (d) embed-out, weight (e) final-layer-norm, bias (f) final-layer-norm, weight

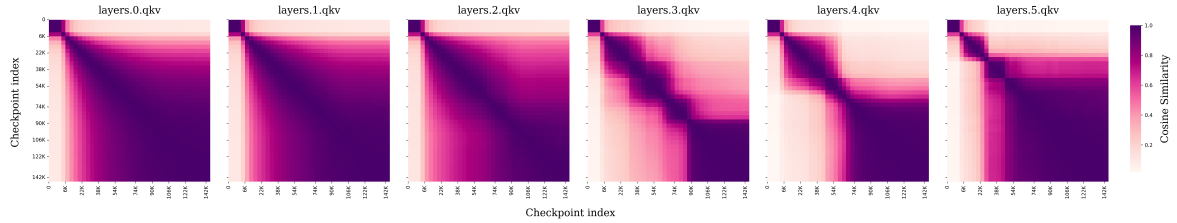


(g) input-layernorm, bias (h) input-layernorm, weight

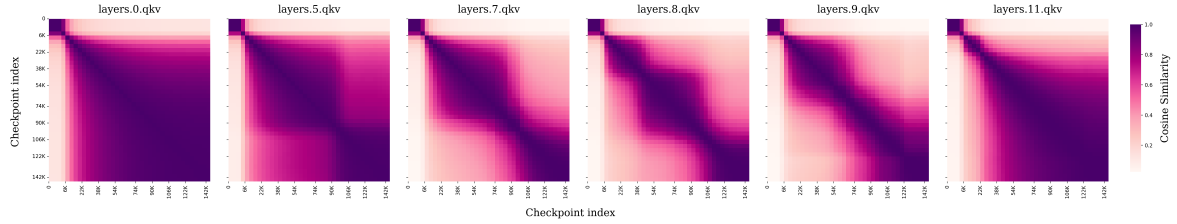


(i) post-attention-layernorm, bias (j) post-attention-layernorm, weight

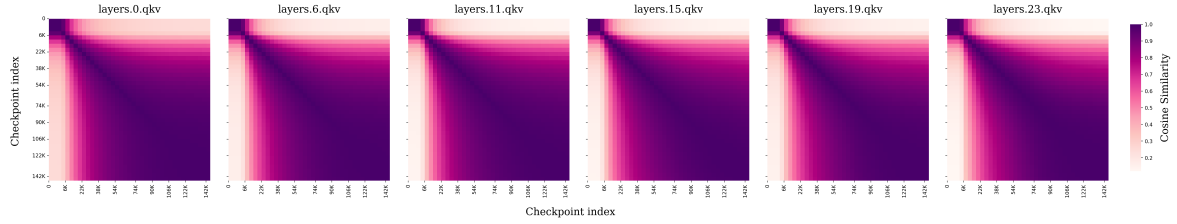
Figure 73. Layerwise Trajectory Maps, grouped by layer type, for the $14M$ GPT-NeoX model trained on the Pile dataset.



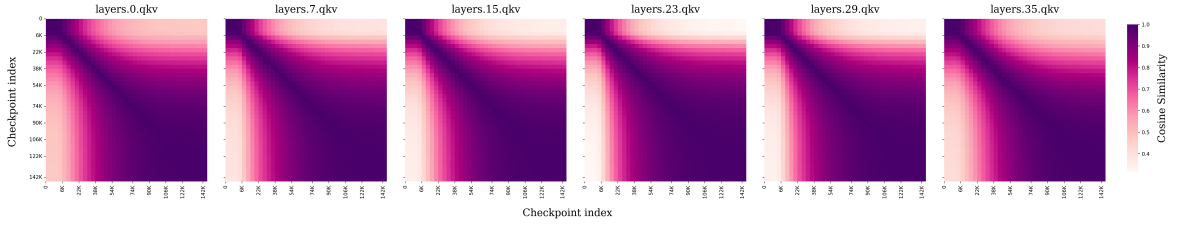
(a) 14 M



(b) 160 M



(c) 1.4 B

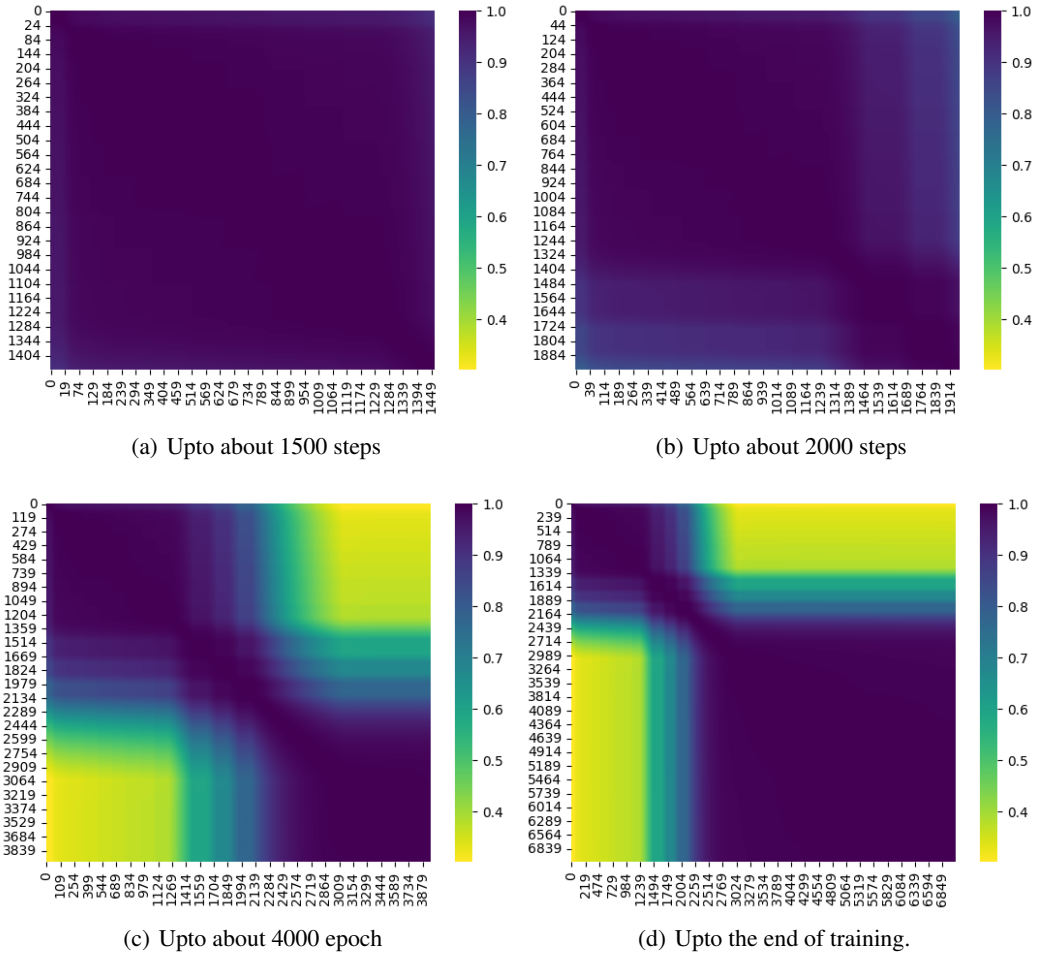


(d) 12 B

Figure 74. Trajectory maps of Q,K,V layers become homogenized over increasing scale.

J. Trajectory Maps for Grokking

In grokking (Power et al., 2022), we have that the performance on test samples significantly lags behind the training performance. Below, we look at the trajectory maps in this setting, considering the experimental setup of https://github.com/teddykoker/grokking. We can observe in Figure 75 that:



(a) Upto about 1500 steps

(b) Upto about 2000 steps

(c) Upto about 4000 epoch

(d) Upto the end of training.

Figure 75. Trajectory maps during the course of learning. Grokking (Power et al., 2022), or sudden increase in test accuracy while training accuracy is already at a ceiling, occurs where the trajectory map also shows a transition point.

- Upto about 1500 epochs: Everything is pitch blue. No directional exploration, test accuracy remains, more or less, random.
- Upto about 2000 epochs: Some directional movement starts to happen, and some initial signs of improvement in test performance.
- Upto about 4000 epoch: Transition point for directional exploration. Test performance visibly improves.

We think that without (appropriate) directional exploration, the training converges to a 'lazy'/'shortcut'/'dead-end' like solutions. Moreover, we believe that being 'lazy' in the directional sense is highly intertwined with being 'lazy' in the sense of feature learning (Chizat et al., 2020). Besides, the above experiments show that the resemblance with the lazy regime is more than an analogy. Kumar et al. (2024) have shown that grokking can be seen as the transition from the lazy to the non-lazy (rich) training regime. In particular, we find that the precise part of the training, where the test accuracy first shows a marked growth is also the part where the directional exploration starts to happen.

K. Putting Directional Redundancy to Test

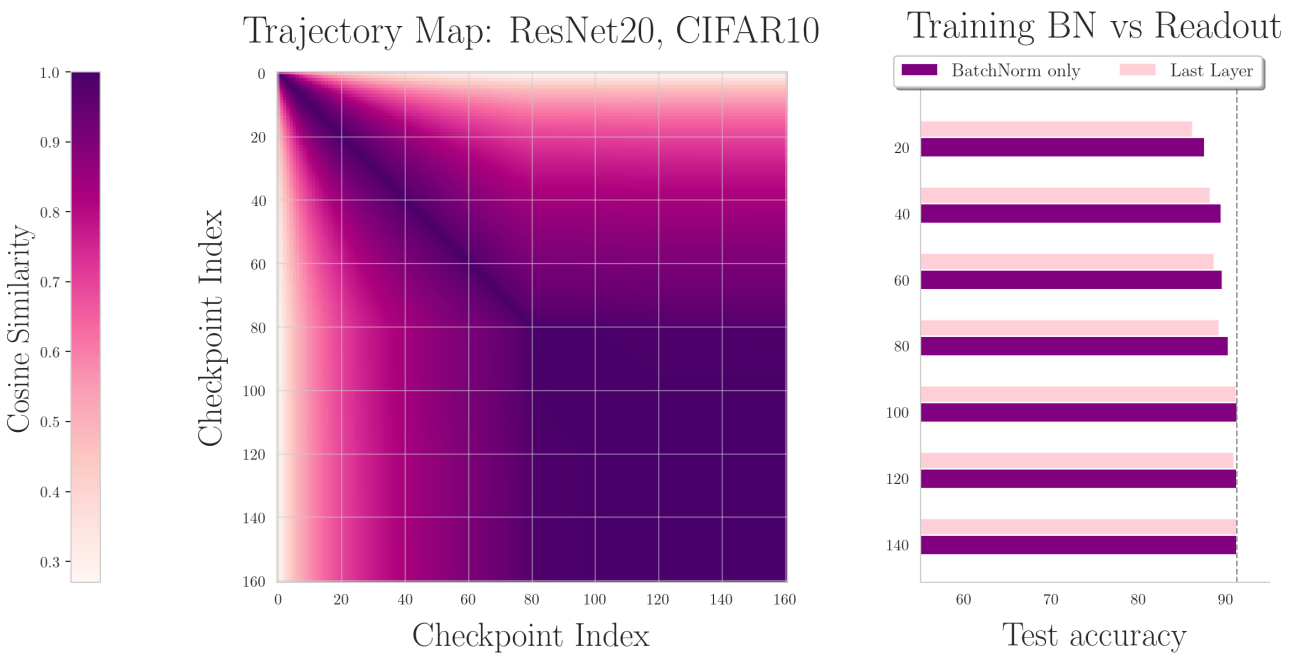


Figure 76. Comparison of training only batch norm parameters with training entire last layer (readout) parameters.

The presented results, and that in the main section, have been averaged over 3 seeds. The standard deviation is never more than 0.05 (for the batchnorm training, even less), and hence it is difficult to make it out in the plots and has been omitted.