NEURAL TANGENT KERNEL PERSPECTIVE ON PARAMETER SPACE SYMMETRIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-space symmetries are transformations that modify a model's parameters without changing its outputs. These transformations can be leveraged to accelerate optimization and enhance generalization. Remarkably, applying a single transformation, either before or during training, often suffices to realize these benefits. While the effectiveness of this approach is promising, its underlying mechanisms remain poorly understood. In this paper, we provide an explanation within the Neural Tangent Kernel (NTK) framework, by analyzing how such transformations affect the properties of the kernel. We show that maximizing the loss gradient norm via a symmetry is equivalent to maximizing the alignment between the NTK and the network. Since kernel alignment is known to correlate with the optimization rate in the NTK limit, this result elucidates how loss gradient optimization facilitates faster training. To establish the validity of this approach, we prove that parameter-space symmetries preserve the NTK limit.

1 INTRODUCTION

Parameter-space symmetries are transformations that modify a neural network's parameters while preserving its outputs. Formally, let F be a hypothesis function, parameterized by $\theta \in \mathbb{R}^N$, mapping an input space $X \subseteq \mathbb{R}^{d_x}$ to an output space $Y \subseteq \mathbb{R}^{d_y}$, i.e.,

$$F(\theta) : X \to Y \ . \tag{1}$$

A parameter-space symmetry is a transformation $\theta \mapsto g\left(\theta\right) = \theta'$, such that:

$$F(\theta) = F(\theta') \ . \tag{2}$$

Parameter-space symmetries are common in neural networks, and are useful both in applications and in the analyses of their performance. Continuous symmetries, for example, have been identified in many network architectures, including networks with homogeneous activations Badrinarayanan et al. (2015), ReLU networks Neyshabur et al. (2015), radial rescaling activations Ganev et al. (2021), and components such as softmax and batch normalization Kunin et al. (2020).

Parameter permutation symmetries have been linked to the structure of minima in optimization landscapes Simsek et al. (2021); Entezari et al. (2021); Hecht-Nielsen (1990); Chen et al. (1993). Quiver representation theory provides a unified framework for analyzing symmetries in neural networks with pointwise Armenta & Jodoin (2021) and rescaling activations Ganev & Walters (2022). Additionally, Zhao et al. (2022b) identify a class of nonlinear, data-dependent symmetries. Parameter-space symmetries have also proven useful in the analysis of mode connectivity Ainsworth et al. (2022).

An important application of parameter-space symmetries is their potential to accelerate the optimization of neural networks Kunin et al. (2020); Van Laarhoven (2017); Grigsby et al. (2022). One approach, known as *neural teleportation*, achieves this effect by inserting teleportation steps during gradient descent, where the norm of the empirical loss gradient is maximized over the training data via symmetry transformations Armenta et al. (2023); Zhao et al. (2022a). Although this approach reduces the number of gradient descent steps needed to minimize the loss, it is not as efficient if many teleportation steps are needed, as maximizing the gradient norm is computationally expensive. However, Zhao et al. (2023) have demonstrated that even a single loss gradient norm maximization transformation prior to training can meaningfully improve the optimization rate.

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

While the theoretical foundations of how neural teleportation can improve optimization have been studied Zhao et al. (2022a; 2023), the mechanisms by which a single symmetry transformation can accelerate long-term optimization remain poorly understood.

One principal approach to studying such complex questions in neural networks is to examine them in the large-width limit. In this regime, known as the *neural tangent kernel* (NTK) limit, networks behave linearly with respect to their parameters, and their training dynamics are greatly simplified Jacot et al. (2018); Lee et al. (2019); Li et al. (2019); Yang (2019b; 2020); Yang & Littwin (2021); Shem-Ur & Oz (2024); Chizat et al. (2019); Liu et al. (2020). Despite its simplicity, this first-order approximation captures many of the core properties of realistic finite networks, making it a powerful tool for analyzing their behavior Li et al. (2019); Yang & Hu (2020); Littwin et al. (2021).

The primary objective of this paper is to elucidate the effects of a single parameter-space symmetry transformation through the lens of the neural tangent kernel limit.

Main Contributions

- We extend the results of Liu et al. (2020) on the role of the Hessian spectral norm in the linearization of wide neural networks, by explicitly analyzing its effect on the linearization in the function space (Section 3.3).
- We prove that the NTK linearization rate is preserved under bounded linear parameter-space symmetry transformations (Section 4).
- We introduce a new definition of NTK alignment, and show that it serves as an effective indicator of the optimization rate in the NTK regime (Section 5, Appendix H).
- We show that maximizing the norm of the loss gradient once, an established technique for improving optimization, is equivalent to maximizing our modified kernel alignment (Section 5).
- We present a method to extend any hypothesis function such that any linear invertible transformation, not necessarily a symmetry of the original function, can be realized as a symmetry (Appendix E).

2 Background

2.1 An Example of Parameter-Space Symmetries in Fully Connected Neural Networks

To illustrate the concept of parameter-space symmetries, we consider the fundamental example of fully connected neural networks with a homogeneous activation function.

Fully connected neural networks are characterized by $L+1 \in \mathbb{N}$ parameter vectors, the biases, denoted by $\theta^{(1)}, \ldots, \theta^{(L+1)}$, $L+1$ parameter matrices, the weights, denoted by $\theta^{(1,0)}, \ldots, \theta^{(L+1,L)}$, and an activation function $\phi : \mathbb{R} \to \mathbb{R}$. Given an input $x \in X$, the network's layers are recursively defined for every layer $l = 1, ..., L$ as follows:

$$F^{(1)} = \theta^{(1,0)} x + \theta^{(1)}, \quad F^{(l+1)} = \theta^{(l+1,l)} \phi\left(F^{(l)}\right) + \theta^{(l+1)}, \tag{3}$$

where the network's output is given by the final layer:

$$F(\theta)(x) = F^{(L+1)}(\theta)(x) . \tag{4}$$

The depth of the network is defined as the number of hidden layers, $L \in \mathbb{N}$, while the width of each layer $l = 0, \ldots, L+1$ is denoted by n_l, $n_0 = d_x$ and $n_{L+1} = d_y$.

Assuming a homogeneous activation function ϕ of degree d, i.e., for all $r, c \in \mathbb{R}$, $\phi(cr) = c^d \phi(r)$, then for any inner layer $l = 1, \ldots, L$ and neuron index $i_l = 1, \ldots, n_l$, the following transformation constitutes a symmetry, as defined in Equation 2, for any $c_{i_l} \neq 0$:

$$\theta^{(l+1,l)}_{\cdot,i_l} \mapsto \tfrac{1}{c^d_{i_l}} \theta^{(l+1,l)}_{\cdot,i_l}, \quad \theta^{(l,l-1)}_{i_l,\cdot} \mapsto c_{i_l} \theta^{(l,l-1)}_{i_l,\cdot}, \quad \theta^{(l)}_{i_l} \mapsto c_{i_l} \theta^{(l)}_{i_l} . \tag{5}$$

Moreover, any combination of such transformations constitutes a symmetry as well. This result extends to activation functions that are homogeneous over a subset of coefficients, with c_{i_l} restricted accordingly. For instance, in the case of ReLU activation, any $c_{i_l} > 0$ is permissible.

2.2 THE NEURAL TANGENT KERNEL LIMIT

In this section, we review the main analytical tool used in this work, the Neural Tangent Kernel limit. To simplify notation and reduce clutter, we omit explicit summation in the equations. In Appendix B, we expand all summations explicitly, and present all indices for clarity.

2.2.1 DEFINITION

Analyzing neural networks is notoriously challenging due to their highly nonlinear architectures, and vast parameter spaces. Questions such as how a single parameter-space symmetry transformation can improve optimization are exceedingly difficult to address analytically.

A principal approach to tackling such questions is to study neural networks in the large-width limit. In this regime, wide networks trained via gradient descent-based algorithms, evolve as though they were linear in their parameters Jacot et al. (2018); Lee et al. (2019); Li et al. (2019). Specifically, under appropriate quantitative assumptions, the network function along training can be approximated by its first-order Taylor expansion around the initialization Lee et al. (2019); Yang (2019a); Hanin & Nica (2019); Seleznova & Kutyniok (2022a;b); Lee et al. (2022); Huang & Yau (2020):

$$F(\theta) = F(\theta_0) + \nabla F(\theta_0)^T (\theta - \theta_0) + O\left(\frac{1}{\sqrt{n}}\right) . \tag{6}$$

Here, ∇F denotes the Jacobian with respect to θ, θ_0 is the initialization, and $n = \min_{1 \leq l \leq L} n_l$. The $O(1/\sqrt{n})$ term is in the stochastic big-O notation.

Formally, let F^{lin} denote the linear approximation of F around θ_0, as given in Equation 6. Suppose both F and F^{lin} are trained using the same gradient descent-based algorithm, under the setup outlined in Yang & Littwin (2021) and the quantitative bound above. Then, for any fixed training step $s \in \mathbb{N}$, and input $x \in X$, the difference between the outputs of F and F^{lin} is asymptotically bounded by $O(1/\sqrt{n})$:

$$\left\| F(\theta(s))(x) - F^{\text{lin}}(s)(x) \right\| = O\left(\frac{1}{\sqrt{n}}\right) , \tag{7}$$

where $\theta(s)$ is θ at step s, and $F^{\text{lin}}(s)$ is F^{lin} for the s-th learning step.

This limit holds across a broad class of wide neural network architectures, and compatible gradient-based optimization methods Yang (2019b; 2020); Yang & Littwin (2021); Chizat et al. (2019); Shem-Ur & Oz (2024); Liu et al. (2020).

2.2.2 PROPERTIES

A fundamental result of the neural tangent kernel limit, is that the network's evolution during training becomes decoupled from its parameters θ. Instead, it is governed by a fixed kernel, defined at initialization. This kernel is a two-point matrix function of size $n_{L+1} \times n_{L+1}$:

$$\forall x, a \in X : \quad \Theta(x, a) = \eta \nabla F(\theta_0)(x)^T \nabla F(\theta_0)(a) , \tag{8}$$

where $\eta > 0$ is the learning rate, and the inner product is taken over all network parameters.

This behavior is well illustrated in the setting of deterministic gradient descent, where $X' \subseteq X$ is a finite set of inputs, with corresponding outputs $y(X')$, and the network is trained by minimizing the empirical loss $\mathcal{L}$, derived from a loss function $\mathcal{C}$, such as:

$$\Delta\theta = -\eta \nabla \mathcal{L}(\theta) , \tag{9}$$

with the loss defined as:

$$\mathcal{L}(\theta) = \frac{1}{|X'|} \sum_{x \in X'} \mathcal{C}(F(\theta)(x), y(x)) . \tag{10}$$

In the NTK regime, the output evolution follows a semi-linear equation of motion that is decoupled from θ, once the kernel and initial state are chosen:

$$\begin{aligned} \forall x \in X : \quad & \Delta F(\theta)(x) = F(\theta + \Delta\theta)(x) - F(\theta)(x) = \\ & -\tfrac{1}{|X'|} \Theta(x, X') \mathcal{C}'\big(F(\theta)(X'), y(X')\big) + O\left(\tfrac{1}{\sqrt{n}}\right) , \end{aligned} \tag{11}$$

where $\mathcal{C}'$ denotes the derivative of the loss with respect to the network output. This derivative is a vector indexed by $i = 1, \ldots, n_{L+1}$ and $x \in X'$, which we also denote by f:

$$f_i(\theta)(x) = \mathcal{C}'_i(F(\theta)(x), y(x)) \ . \tag{12}$$

Likewise, $\Theta(x, X')$ is a matrix indexed by i and (j, x') for $i, j = 1, \ldots, n_{L+1}$ and $x' \in X'$. The product in Equation 11 is standard matrix-vector multiplication, as we show in Appendix B.

Despite being a simplification, this first-order approximation captures many of the key properties of finite-width neural networks, which makes it a valuable approach to understanding finite neural networks Li et al. (2019); Yang & Hu (2020); Littwin et al. (2021).

3 Preliminaries

3.1 Assumptions and Notation

We begin by summarizing the assumptions and notation used in our analysis.

Our results apply to the class of hypothesis functions defined in Equation 1, trained with deterministic gradient descent as described in Equation 9. We assume that both the hypothesis function and the loss function are twice continuously differentiable, with locally Lipschitz output-space loss gradient. We also assume that the hypothesis function admits a parameter-space symmetry, satisfying the properties outlined in Section 3.2.

When discussing linearization rate, we do so in the context of a sequence of hypothesis functions $\{F_n\}_{n=0}^{\infty}$. We assume that all functions in this sequence are trained with the same algorithm, using the same loss and training data. While a function may admit more than one symmetry, we associate each function with a single symmetry, and assume that all of these symmetries across the sequence share the same C bound specified in Equation 16.

Throughout this work, we primarily interpret n as the width of the network. However, n may represent any measure of system scale. Thus, our analysis is not limited to wide neural networks, but also applies to more general sequences of function models.

To simplify notations, we often omit the explicit index n and simply refer to the network sequence as a function, as we already did in Section 1. We also omit explicit dependencies on $x \in X'$ and θ in the functions throughout the paper.

All limits use fixed training horizons and input collections, and bounded initial outputs. We provide the details and generalization of these assumptions in Appendix A.

3.2 Our Symmetry Transformations

We assume symmetry transformations of the form given in Equation 2, such that they are well-defined, and twice continuously differentiable in a neighborhood of the parameter θ where the symmetry is applied.

We mainly discuss linear symmetry transformations, such as scaling and permutation symmetries, since these constitute the vast majority of the transformations discussed in the literature Zhao et al. (2025). However, in Appendix F.2 we also address nonlinear transformations.

We define the *transformation Jacobian*, such that for any two parameter indices $\alpha, \beta = 1, \ldots, N$:

$$\Gamma^{-1}_{\alpha\beta} = \frac{\partial g_\beta(\theta)}{\partial \theta_\alpha} \ . \tag{13}$$

We assume the transformation are invertible, as we can always remove redundant parameters, such as the gradient descent dynamics are conserved, as we show in Appendix I.1.1.

Since the hypothesis function remains invariant by definition, the transformation of the hypothesis function's derivatives under symmetry follows from the chain rule, as shown in Appendix I.1.2:

$$\nabla \mapsto \Gamma \nabla \ . \tag{14}$$

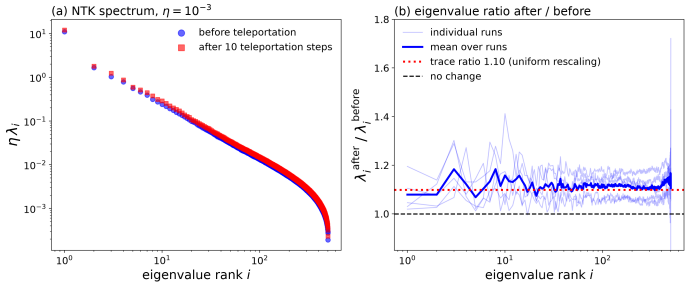


Figure 1: **Empirical NTK eigenvalues before and after symmetry teleportation**. (a) Sorted eigenvalues λ_i of the empirical NTK, summed over output classes and computed on a fixed batch of 500 samples, averaged over 5 runs and scaled by learning rate $\eta = 10^{-3}$ (SGD with momentum 0.9). (b) Ratio of the eigenvalues after vs. before teleportation for individual runs and their average. Teleportation raises the whole spectrum by about 10%, nearly uniformly, while preserving its shape, meaning the kernel is rescaled only modestly. (The full setup is explained in Appendix D).

To ensure finite-size learning steps, we impose a bound on the rescaling of the transformed gradient norm by some constant $0 < C$:

$$\|\Gamma \nabla F\| \leq C \, \|\nabla F\| \ . \tag{15}$$

For our theoretical results, we use the stronger sufficient condition:

$$\|\Gamma\| \leq C \ , \tag{16}$$

in operator norm, uniformly in n, and throughout the fixed-radius neighborhoods considered in our analysis.

We analyze the effect of this bound in Appendix C, where we show that the scalar C enters the transformed dynamics in the same way as changing the effective learning rate to $\eta_{\text{eff}} = C^2\eta$. Thus, increasing C has the same effect on both the training dynamics and the deviation from the NTK limit as increasing the learning rate. In particular, choosing a large C does not introduce a new source of deviation from linearization, it is equivalent to choosing an excessively large learning rate.

As is well known, gradient descent remains stable only when the largest eigenvalue of the normalized NTK remains below the corresponding stability threshold, and the optimal learning rate is also generally of the same order as this threshold Cohen et al. (2021); Arora et al. (2022); Damian et al. (2022); Chemnitz & Engel (2025). Therefore, when the learning rate is already well chosen, C must remain approximately bounded by one. Equivalently, when the learning rate is not well chosen, choosing the largest useful value of C is effectively equivalent to retuning the learning rate to a well-optimized value, a regime in which the NTK approximation is known to remain informative in practice Li et al. (2019); Yang & Hu (2020); Littwin et al. (2021).

While we impose the bound explicitly, practical implementations also control this scale by limiting the size and number of symmetry-optimization steps. For example, Zhao et al. (2023) use only a small number of such steps. In our validation, ten teleportation steps increase the kernel eigenvalues by only about 10% (Figure 1), consistent with only a modest change in the effective learning-rate scale. We discuss this further in Appendix D.

In Appendix E, we show that any invertible matrix Γ can be interpreted as a linear symmetry of a hypothesis function, by extending the function with an additional set of hyperparameters. This suggests that analyzing neural network symmetries through their associated transformation matrices is both natural and powerful.

3.3 Underlying Cause of the Neural Tangent Kernel Limit

Many works proving the NTK limit heavily rely on independent parameter initialization. While this assumption simplifies the analysis, it offers limited intuition for why the limit holds. Moreover, it is unsuitable for our purposes, since symmetry transformations may completely violate it. However, several works go beyond proving the NTK limit, aiming instead to uncover its underlying cause Liu et al. (2020); Chizat et al. (2019); Dyer & Gur-Ari (2019); Liu et al. (2022); Shem-Ur & Oz (2024).

A notable example is Liu et al. (2020), which argues that the decay of the Hessian's spectral norm as $n \to \infty$ is the driving mechanism behind linearization. Formally, for any $x \in X'$ and $n \in \mathbb{N}$ (suppressing x and n from the notation to avoid clutter):

$$\left\|\eta\nabla^{\times 2}F_i\right\| = \|H\| = \sup_{\|v\|=1} \left|v^T H v\right| = \widetilde{O}\left(\frac{1}{\sqrt{n}}\right) , \tag{17}$$

within any arbitrarily large but fixed radius $0 < R$ around initialization, in the NTK parametrization, where i is an output coordinate, and $\widetilde{O}$ allows logarithmic factors.

To make this connection explicit for the training trajectories in Equation 7, while tracking the gradient scale, we introduce the following lemma:

Lemma 3.1 (Hessian Norm Governs Linearization Rate) Given the setup in Section 3.1, if there exists a parametrization such that:

$$\|H\| = O(m_n) , \tag{18}$$

uniformly on every fixed-radius ball around initialization, where $m_n \to 0$ characterizes the Hessian decay rate, and the gradient is appropriately normalized at initialization, i.e.:

$$\|\sqrt{\eta_n}\nabla F_n\| = O(1) , \quad \|\eta_n \nabla F_n\| = O(1) , \tag{19}$$

then the system linearizes as in Equation 7, with error $O(m_n)$ at every fixed training step.

We explain the meaning of the NTK parametrization, and how it gives the normalization in Equation 19, in Appendix A.6. The proof of Lemma 3.1 is given in Appendix F.1.

Since these conditions hold for the networks above, it follows that, provided parameter-space symmetries do not disrupt them, such symmetries preserve the NTK limit.

4 Neural Tangent Kernel Limit Preservation under Parameter-Space Symmetry

To study neural teleportation within the NTK limit, we must first verify that parameter-space symmetry transformations preserve the linearization property of the NTK limit. This step is essential to ensure the applicability of the NTK framework to our analysis.

Theorem 4.1 (NTK Limit Preservation) Given the setup of Lemma 3.1, after applying a linear parameter-space symmetry at a fixed training step $s' \in \mathbb{N}$ as described in Section 3.1, $\theta(s') \mapsto g(\theta(s'))$, the neural tangent kernel limit, as defined in Equation 7 with m_n replacing $\frac{1}{\sqrt{n}}$, remains valid for every fixed number of subsequent steps.

Proof. Having established the right tool for the job, the proof unfolds naturally. The key idea is to track how the Hessian and gradient norm transform under parameter-space symmetries. From Equation 14, we know that:

$$H \mapsto \Gamma H \Gamma^T, \quad \nabla F \mapsto \Gamma \nabla F , \tag{20}$$

since the effect of symmetry transformations on the network's derivatives follows directly from the chain rule. This observation allows us to control the transformed gradient and Hessian norms.

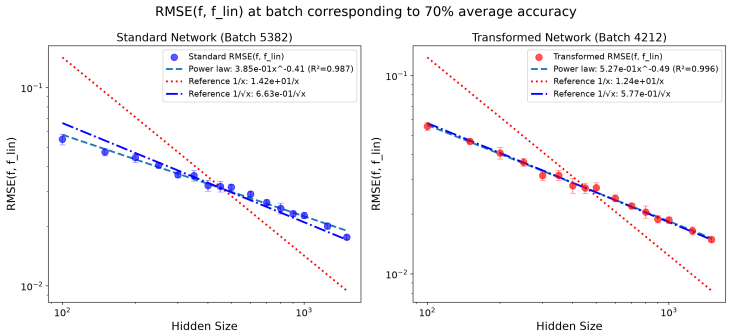


Figure 2: **Preservation of linearization under parameter-space transformations.** Root mean squared error (RMSE) between the outputs of neural networks and their linearized counterparts as a function of network width. **Left:** Standard two-hidden-layer MLP. **Right:** Transformed two-hidden-layer MLP. Each panel is evaluated at the first batch at which the test accuracy, averaged over all widths and trials, reaches 70%. Results are averaged over five independent trials, with error bars denoting standard deviation.

Before the transformation, the gradient and Hessian norms are asymptotically bounded by Equations 18 and 19. Because the gradient norm rescaling is bounded by Equation 15, Equation 19 remains valid after the transformation. Furthermore, the Hessian spectral norm transforms as:

$$\|H\| \mapsto \left\|\Gamma H \Gamma^T\right\| = \sup_{\|v\|=1} \left|v^T \Gamma H \Gamma^T v\right| \leq C^2 \|H\| = O\left(m_n\right) \ . \tag{21}$$

Since $Dg^{-1} = \Gamma^T$, a transformed radius-R ball pulls back into a radius-CR ball around $\theta(s')$, which is $O(1)$ from initialization. The uniform bounds therefore still apply, and Lemma 3.1 yields the desired linearization bound. □

This result is unexpected, as parameter symmetry transformations can completely unravel the independent-initialization assumptions used in the standard NTK proofs. Yet the symmetry identity, together with the transformation bound, preserves the derivative estimates that ensure linearization.

In Appendix F.2, we extend the theorem's analysis beyond the linear case and show that the contraction of the transformation's second-derivative tensor with the network gradient can also contribute to deviations from linearization. Since departure from the NTK regime can affect learning, as shown by Jacot et al. (2022), we suggest that this term may serve as a design principle for constructing new nonlinear symmetries that could either mitigate or promote linearization.

We provide empirical evidence for the preservation of linearization in Figure 2, where we train both standard and transformed networks on FashionMNIST Xiao et al. (2017) for image classification using plain SGD (without momentum), a batch size of 512, and a width-independent learning rate of 1 (NTK parametrization). We find that, in both cases, the error between the network and its linearized counterpart decreases with width at a comparable rate. Further details on the experimental setup and results are provided in Appendix G.

Finally, the kernel itself transforms under symmetry as follows:

$$\Theta = \eta \nabla F(\theta_0)^T \nabla F(\theta_0) \mapsto \eta \nabla F(\theta_0)^T \Gamma^T \Gamma \nabla F(\theta_0) \ . \tag{22}$$

5 Optimization Acceleration

5.1 Optimization Relation to Kernel Alignment

As we discussed in the introduction, one of the primary applications of parameter-space symmetries is to accelerate optimization. This is achieved by maximizing the norm of the empirical loss gradient as defined in Equation 10, via symmetry transformations:

$$\max \|\nabla \mathcal{L}\left(\theta_0\right)\|^2 , \tag{23}$$

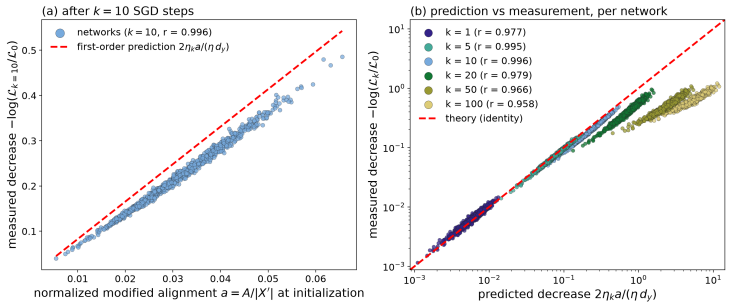


Figure 3: Modified alignment and early optimization speed for 1,000 independently initialized MLPs, trained with MSE loss on image classification. Dashed lines show the first-order (NTK) prediction of the relative decrease of the training loss after k SGD steps, $-\log(\mathcal{L}_k/\mathcal{L}_0) \approx 2\eta_k a/\eta d_y$, where $a = A/|X'|$ and η_k is the cumulative step size of SGD with momentum (App. H.2). **(a)** After $k = 10$ steps. **(b)** After $k = 1$ to 100 steps. The prediction is accurate early and tends to slightly overestimate the decrease later.

Zhao et al. (2023) demonstrated that such a transformation can be effectively applied once prior to learning, offering a computationally inexpensive way to accelerate optimization. While effective, the mechanisms by which this method enhances optimization throughout training remain poorly understood. We argue that since symmetry transformations preserve the NTK limit, this phenomenon can be analyzed through the lens of the NTK limit, by examining how such transformations affect the kernel.

A key indicator of a kernel's capacity to learn, is its alignment with the data, defined here as the normalized inner product between the kernel matrix and the label outer product Cortes et al. (2012); Baratin et al. (2021); Arora et al. (2019); Wang et al. (2022); Kopitkov & Indelman (2020); Khalafi et al. (2023):

$$A_o = \frac{y\left(X'\right)^T \Theta\left(X', X'\right) y\left(X'\right)}{\left\|y\left(X'\right)\right\|_2^2 \left\|\Theta\left(X', X'\right)\right\|_{\mathrm{F}}} . \tag{24}$$

We propose that gradient norm maximization can accelerate learning in the NTK regime by maximizing a modified version of this quantity.

5.2 Our Modified Alignment

To better capture the effect described above, we slightly modify the standard definition of kernel alignment. First, instead of aligning with the data directly, we follow Kopitkov & Indelman (2020); Khalafi et al. (2023) in considering alignment with the gradient of the loss function with respect to the model output as defined in Equation 12.

Second, we remove the kernel norm from the denominator, following Khalafi et al. (2023), for two reasons: (i) up to a certain scale, larger kernels improve optimization Lewkowycz et al. (2020); Hayou et al. (2022); and (ii) the overall kernel scale can be controlled through the learning rate, as discussed in Appendix C.

With these modifications, for $f(X') \neq 0$, we redefine alignment as:

$$A = \frac{f\left(X'\right)^T \Theta\left(X', X'\right) f\left(X'\right)}{\left\|f\left(X'\right)\right\|_2^2} . \tag{25}$$

We establish the effectiveness of this definition in Proposition H.1, by showing that small initial alignment relative to $|X'|$ limits the loss-reduction rate.

We demonstrate empirically that our modified alignment reflects the optimization rate by training 1,000 independently initialized ReLU MLPs with two hidden layers. For each network, we compute the modified alignment at initialization and record the relative decrease of the training loss during

the first steps of training. As illustrated in Figure 3, the alignment predicts this decrease quantitatively. After 10 steps, the measured decrease follows the first-order prediction closely ($r = 0.996$), indicating that larger initial alignment values are associated with faster optimization (Appendix H.2).

5.3 Alignment and Loss Gradient Norm Equivalence

With our new definition for alignment as an indicator for optimization rate, we can now show its proportionality to the squared loss gradient norm.

Proposition 5.1 (Alignment and Loss Gradient Norm Equivalence) Under the setup detailed in Section 3.1, the squared loss gradient norm (Equation 23), and the kernel alignment (Equation 25), are proportional at the same parameter value, for $f(X') \neq 0$:

$$\|\nabla \mathcal{L}\|_2^2 = \frac{\|f(X')\|^2}{\eta |X'|^2} A \propto A \,. \tag{26}$$

Proof. The proposition follows directly from Equations 8, 10, and 25, applying the chain rule within our analytical framework:

$$\begin{aligned} \|\nabla \mathcal{L}\|_2^2 = \left\| \nabla \tfrac{1}{|X'|} \textstyle\sum_{x \in X'} \mathcal{C}\left(F(x), y(x)\right) \right\|_2^2 = \\ \left\| \tfrac{1}{|X'|} \textstyle\sum_{x \in X'} \nabla F(x) f(x) \right\|_2^2 = \tfrac{1}{\eta |X'|^2} f(X')^T \Theta(X', X') f(X') = \tfrac{\|f(X')\|^2}{\eta |X'|^2} A \,. \end{aligned} \tag{27}$$

□

This concludes our explanation of why maximizing the normed loss gradient improves optimization from the NTK perspective, as it is equivalent to maximizing kernel alignment, which is correlated with the optimization rate.

6 Conclusion

In this work, we explored the effects of applying parameter-space symmetries in neural network optimization through the lens of the NTK limit. We established that bounded linear symmetries preserve the NTK limit, thereby validating the use of this framework for our analysis. Leveraging this result, we explained how even a single application of parameter-space symmetry can improve the optimization rate in the NTK regime.

Our findings suggest potential directions for the development of new optimization algorithms. In particular, they highlight more direct approaches to maximizing kernel alignment through parameter-space symmetries, or other related methods. By clarifying why parameter-space symmetries work, our work may help inspire new algorithms that achieve similar benefits without requiring the costly optimization often associated with parameter-space symmetries. Furthermore, our results for nonlinear symmetries may guide the design of new, useful classes of symmetries tailored for optimization.

Looking forward, we aim to extend this line of work by combining insights on both optimization and generalization. We hope to develop approaches that simultaneously accelerate training and improve the generalization capacity of neural networks.

A primary limitation of our work is that our analysis is restricted to the NTK limit, which remains an approximation of finite neural networks. Addressing this limitation in future work, we plan to investigate the effects of parameter-space symmetries on higher-order approximations and conduct more extensive empirical studies. Nevertheless, as the NTK framework captures many of the most relevant properties of neural networks, we believe that our findings hold meaningful value, and could facilitate practical advancements in the field.

Our work is also significant to the general understanding of the NTK limit, beyond the context of parameter-space symmetries. We provide a clear example of wide neural networks that do not satisfy a common assumption in NTK proofs, namely, independent parameter initialization, yet still exhibit linearization.

AI USE STATEMENT

All of the main mathematical ideas in this work were conceptualized, formulated, derived, and structured by humans. Generative AI was used later to help polish some of these arguments, check them for technical problems, and correct technical mistakes. Similarly, the proofs were first drafted by humans, and were then checked, polished, and in some cases completed with the help of AI. AI was not responsible for the central ideas, theoretical framework, or critical ingredients behind the main mathematical results.

The main text was written by humans and later edited through a combination of human and AI-assisted editing. AI was used more heavily in parts of the appendices, mainly for polishing technical explanations based on arguments and results developed by humans.

AI assisted substantially with the review of related work, including finding potentially relevant papers, summarizing parts of the literature, and helping organize the discussion and citations.

The experiments were conceived and designed at the conceptual level by humans, including the main questions being tested and the quantities of interest. AI helped with some aspects of the technical realization, including deriving selected technical quantities and variables required for the experiments. It was used primarily for verification and, to a lesser extent, to assist with writing parts of the codebase. AI also assisted with the technical analysis and explanation of the results. However, the high-level conclusions drawn from the experiments were determined by humans.

AI was not involved in any substantial way in formulating the research questions or hypotheses, developing the overall theoretical framework, or determining the high-level research methodology or experimental design. AI was also not used for generating, cleaning, or reformatting datasets, translation, or qualitative/thematic data analysis. While AI assisted with technical experimental analysis and explanations, it did not determine the central scientific interpretation of the results. All AI-assisted mathematical arguments, writing, literature review, experimental analysis, and code were reviewed by humans before being included in the paper. We take responsibility for the final content of this work, including all text, claims, analyses, and code produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All of our assumptions and proofs are provided in the main text and appendix. Experimental settings are described in Appendices D, G, and H.2. Code will be included in the camera ready version.

REFERENCES

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Marco Armenta and Pierre-Marc Jodoin. The representation theory of neural networks. *Mathematics*, 9(24):3216, 2021.

Marco Armenta, Thierry Judge, Nathan Painchaud, Youssef Skandarani, Carl Lemaire, Gabriel Gibeau Sanchez, Philippe Spino, and Pierre-Marc Jodoin. Neural teleportation. *Mathematics*, 11(2):480, 2023.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pp. 322–332. PMLR, 2019.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pp. 948–1024. PMLR, 2022.

Vijay Badrinarayanan, Bamdev Mishra, and Roberto Cipolla. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754*, 2015.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pp. 2269–2277. PMLR, 2021.

Dennis Chemnitz and Maximilian Engel. Characterizing dynamical stability of stochastic gradient descent in overparameterized learning. *Journal of Machine Learning Research*, 26(134):1–46, 2025.

An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

Iordan Ganev and Robin Walters. Quiver neural networks. *arXiv preprint arXiv:2207.12773*, 2022.

Iordan Ganev, Twan van Laarhoven, and Robin Walters. Universal approximation and model compression for radial neural networks. *arXiv preprint arXiv:2107.02550*, 2021.

J Elisenda Grigsby, Kathryn Lindsey, Robert Meyerhoff, and Chenxi Wu. Functional dimension of feedforward relu neural networks. *arXiv preprint arXiv:2209.04036*, 2022.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *arXiv preprint arXiv:1909.05989*, 2019.

Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. The curse of depth in kernel regime. In *I (Still) Can't Believe It's Not Better! Workshop at NeurIPS 2021*, pp. 41–47. PMLR, 2022.

Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pp. 129–135. Elsevier, 1990.

Jiaoyang Huang and Horng-Tzer Yau. Dynamics of deep neural networks and neural tangent hierarchy. In *International conference on machine learning*, pp. 4542–4551. PMLR, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Arthur Jacot, Franck Gabriel, Francois Ged, and Clement Hongler. Freeze and chaos: Ntk views on dnn normalization, checkerboard and boundary artifacts. In *Mathematical and Scientific Machine Learning*, pp. 257–270. PMLR, 2022.

Shervin Khalafi, Saurabh Sihag, and Alejandro Ribeiro. Neural tangent kernels motivate graph neural networks with cross-covariance graphs. *arXiv preprint arXiv:2310.10791*, 2023.

Miltiadis Kofinas, Boris Knyazev, Yan Zhang, Yunlu Chen, Gertjan J Burghouts, Efstratios Gavves, Cees GM Snoek, and David W Zhang. Graph neural networks for learning equivariant representations of neural networks. *arXiv preprint arXiv:2403.12143*, 2024.

Dmitry Kopitkov and Vadim Indelman. Neural spectrum alignment: Empirical study. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part II 29*, pp. 168–179. Springer, 2020.

Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel LK Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *arXiv preprint arXiv:2012.04728*, 2020.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Jongmin Lee, Joo Young Choi, Ernest K Ryu, and Albert No. Neural tangent kernel analysis of deep narrow neural networks. In *International Conference on Machine Learning*, pp. 12282–12351. PMLR, 2022.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Sanjeev Li, Arora, Simon S Du, Zhiyuan Hu, Wei, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

Derek Lim, Haggai Maron, Marc T Law, Jonathan Lorraine, and James Lucas. Graph metanetworks for processing diverse neural architectures. *arXiv preprint arXiv:2312.04501*, 2023.

Etai Littwin, Tomer Galanti, and Lior Wolf. On random kernels of residual architectures. In *Uncertainty in Artificial Intelligence*, pp. 897–907. PMLR, 2021.

Chaoyue Liu, Libin Zhu, and Misha Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33: 15954–15964, 2020.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Transition to linearity of wide neural networks is an emerging property of assembling weak models. *arXiv preprint arXiv:2203.05104*, 2022.

Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. *Advances in neural information processing systems*, 28, 2015.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019.

Rose STL Lab. Teleportation-optimization: Github repository, 2024. URL `https://github.com/Rose-STL-Lab/Teleportation-Optimization`.

Mariia Seleznova and Gitta Kutyniok. Analyzing finite neural networks: Can we trust neural tangent kernel theory? In *Mathematical and Scientific Machine Learning*, pp. 868–895. PMLR, 2022a.

Mariia Seleznova and Gitta Kutyniok. Neural tangent kernel beyond the infinite-width limit: Effects of depth and initialization. In *International Conference on Machine Learning*, pp. 19522–19560. PMLR, 2022b.

Ori Shem-Ur and Yaron Oz. Weak correlations as the underlying principle for linearization of gradient-based learning systems. *arXiv preprint arXiv:2401.04013*, 2024.

Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clément Hongler, Wulfram Gerstner, and Johanni Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pp. 9722–9732. PMLR, 2021.

Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

Haonan Wang, Wei Huang, Ziwei Wu, Hanghang Tong, Andrew J Margenot, and Jingrui He. Deep active learning by leveraging training dynamics. *Advances in Neural Information Processing Systems*, 35:25171–25184, 2022.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019a.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in Neural Information Processing Systems*, 32, 2019b.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

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

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, pp. 11762–11772. PMLR, 2021.

Bo Zhao, Nima Dehmamy, Robin Walters, and Rose Yu. Symmetry teleportation for accelerated optimization. *Advances in neural information processing systems*, 35:16679–16690, 2022a.

Bo Zhao, Iordan Ganev, Robin Walters, Rose Yu, and Nima Dehmamy. Symmetries, flat minima, and the conserved quantities of gradient flow. *arXiv preprint arXiv:2210.17216*, 2022b.

Bo Zhao, Robert M Gower, Robin Walters, and Rose Yu. Improving convergence and generalization using parameter symmetries. *arXiv preprint arXiv:2305.13404*, 2023.

Bo Zhao, Robin Walters, and Rose Yu. Symmetry in neural network parameter spaces. *arXiv preprint arXiv:2506.13018*, 2025.

A Assumptions and Generalizations

In this part of the appendix, we elaborate on Section 3.1, detailing and justifying our assumptions, and discuss their generalizations.

A.1 Hypothesis Functions

Our results apply to the function class described in Equation 1, with twice continuously differentiable parameter dependence. Hessian bounds apply to each output coordinate on every fixed-radius ball around the initialization, for the training inputs and any fixed test inputs under consideration. The data and output dimension are fixed, and the initial outputs are $O(1)$. Vector norms are Euclidean and matrix norms are operator norms unless otherwise specified. uniform bounds under random initialization are in the stochastic big-O sense.

The same argument also extends to piecewise-smooth networks, such as ReLU networks. In this case, the activation changes introduce additional contributions to the Jacobian variation and Taylor remainder, but under the standard NTK activation-stability argument these contributions vanish with width. We explain this extension, together with the role of the Barycentric Mean Value Corollary (Corollary I.1), in Appendix F.1. When analyzing linearization rate, we consider a sequence of hypothesis functions $\{F_n\}_{n=1}^{\infty}$ from this class. Each F_n depends on a corresponding parameter vector $\theta^n \in \mathbb{R}^{N_n}$, and each network inherits its own symmetry while being trained under the same algorithm.

The primary interpretation of n will be the network width. However n can represent any parameter that governs the scale of the system. For notational simplicity, we will often omit the explicit index n and refer to the network as a single function rather than a sequence.

This framework is extremely general and encompasses a very broad class of neural network architectures, as well as more general model sequences satisfying these assumptions.

A.2 Symmetry Transformation

Our symmetries, as defined in Equation 2, are assumed to be twice continuously differentiable and well-defined around the parameter values at which they are applied. The symmetry identity holds throughout the neighborhoods used in the analysis. Our main result, Theorem 4.1, is primarily applicable to linear symmetries. While this may seem restrictive, linear symmetries are the main class of transformations currently studied in the literature Zhao et al. (2025). Moreover, in Appendix F.2, we extend our discussion to the generalization of this assumption.

When analyzing how these symmetries affect the linearization rate, we further assume they satisfy the bound described in Equation 16 uniformly for every $n \in \mathbb{N}$. This gives uniform control of the transformed derivatives, as discussed in Appendix C. Limiting the size and number of symmetry-optimization steps, as in Zhao et al. (2023), is one way to keep rescaling modest in practice. The selected linear transformation is held fixed when differentiating, even when its choice depends on the current parameters.

A.3 Hessian Spectral Norm Bound

For Theorem 4.1, we assume that the spectral norm of the Hessian before transformation is bounded as specified in Equation 18. This follows from the arguments presented in Liu et al. (2020), as discussed in the introduction, where it was shown that the scale of the Hessian norm governs linearization, and that for the architectures covered there the Hessian norm vanishes as $\widetilde{O}\left(1/\sqrt{n}\right)$.

While Liu et al. (2020) demonstrated this behavior for a wide variety of networks, there remains a significant class of networks not explicitly covered. Nevertheless, because their proof relies on the semi-linear structure of wide networks, it is highly plausible that the linearization property extends to any network describable by the tensor program formalism Yang (2019b; 2020); Yang & Littwin (2021). Since this formalism captures practically any wide neural network, we believe this assumption should stay valid, although a full formal proof remains outstanding.

A.4 Gradient Descent

We work withing the framework of deterministic gradient descent, as defined in Equation 9, mainly to keep the analysis simple. The fixed-horizon linearization argument extends naturally to stochastic gradient descent and momentum under the usual boundedness and stability assumptions.

In particular, for momentum with a fixed coefficient $\beta < 1$, the momentum variable can be treated as an additional bounded state. Over any fixed number of steps, bounded gradients imply bounded momentum and parameter updates, while the same Jacobian-variation and Taylor-remainder bounds used in Appendix F.1 continue to control the deviation from the linearized dynamics.

The quantitative alignment result in Appendix H.1 is stated for plain gradient descent for simplicity. With momentum, each kernel eigendirection instead follows the corresponding second-order recurrence, and analogous bounds can be obtained under a stable choice of learning rate and momentum.

A.5 Loss Function

We assume that the loss function takes the form given in Equation 10, is twice continuously differentiable, and has a locally Lipschitz output gradient. The alignment extension additionally assumes a nonnegative convex loss with a globally β-Lipschitz output gradient. These assumptions include most standard loss functions.

A.6 Gradient Normalization and NTK Parametrization

For Lemma 3.1, and Theorem 4.1, we assume that the gradient norm is properly normalized according to Equation 19.

The left-hand side of this equation represents the necessary condition for the kernel not to diverge. It holds when the scaled NTK on the fixed input collection remains bounded. The right-hand side, by contrast, depends on the choice of parametrization.

Different parametrizations of the learning dynamics can be obtained by rescaling the parameters and then compensating by multiplying them externally by the same constant. Such transformations do not affect the training dynamics (provided the learning rate is scaled appropriately), but they allow us to view the system from different perspectives. To illustrate this idea, we consider the example of fully connected neural networks as introduced in Jacot et al. (2018).

Under the *standard parametrization*, defined in Equation 3, the typical parameter scales for layers $l = 0, \ldots, L$ are:

$$\theta^{(l+1,l)} \sim \frac{1}{\sqrt{n_l}}, \quad \theta^{(l+1)} \sim 1, \tag{28}$$

and the learning rate scales as:

$$\eta \sim \frac{1}{n}. \tag{29}$$

In contrast, the *NTK parametrization* is defined as:

$$\begin{aligned} F^{(1)} &= \tfrac{1}{\sqrt{n_0}}\theta^{(1,0)}x + \theta^{(1)}, \\ \forall l = 1, \ldots, L : F^{(l+1)} &= \tfrac{1}{\sqrt{n_l}}\theta^{(l+1,l)}\phi\left(F^{(l)}\right) + \theta^{(l+1)}, \end{aligned} \tag{30}$$

with initialization for parameters at all layers $l = 0, \ldots, L$:

$$\theta^{(l+1,l)} \sim 1, \quad \theta^{(l+1)} \sim 1, \tag{31}$$

and a learning rate scaling typically given by:

$$\eta \sim 1. \tag{32}$$

Both parametrizations give equivalent dynamics when learning rates are matched blockwise. This reparametrization trick can always be applied to learning algorithms (though in some cases multiple learning rates may be required, which is only a technical complication) Chizat et al. (2019). It is in the NTK parametrization, where $\eta \sim 1$ and the gradient norm is $O(1)$, that the result of Liu et al. (2020), on which we rely, was established.

B Neural Tangent Kernel Limit Notations

In this appendix we clarify and expand upon the primary notations introduced in Section 2.2.

In the context of deterministic gradient descent, defined in Equation 9, replacing a function input with the set X', signifies that we substitute the function with a vector, by combining the input from X' with the output index of the original function.

Considering for example the derivative of the relative loss function from Equation 11. For any $i, j = 1, \ldots, d_y$ and $x, a \in X'$, we can express the derivative of the loss as:

$$\mathcal{C}'\left(F\left(\theta\right)\left(X'\right), y\left(X'\right)\right)_{(i,x)} = \mathcal{C}'\left(F\left(\theta\right)\left(x\right), y\left(x\right)\right)_i \ . \tag{33}$$

Similarly, for the kernel with one fixed input as in Equation 11:

$$\Theta\left(x, X'\right)_{i,(j,a)} = \Theta\left(x, a\right)_{ij} \ . \tag{34}$$

And for the kernel with no fixed inputs, as in Equations 24 and 25:

$$\Theta\left(X', X'\right)_{(i,x),(j,a)} = \Theta\left(x, a\right)_{ij} \ . \tag{35}$$

Neglecting the linearization remainder, we can now explicitly expand Equations 6, 8, 9, 11, detailing all indices, variables, and summations. For any $i, j = 1, \ldots, n_{L+1}$ and $x, y \in X$, we have:

$$F_i\left(\theta\right)\left(x\right) = F_i\left(\theta_0\right)\left(x\right) + \sum_{\alpha=1}^{N} \nabla_\alpha F_i\left(\theta_0\right)\left(x\right)\left(\theta - \theta_0\right)_\alpha \ , \tag{36}$$

$$\Theta_{ij}\left(x, y\right) = \eta \sum_{\alpha=1}^{N} \nabla_\alpha F_i\left(\theta_0\right)\left(x\right) \nabla_\alpha F_j\left(\theta_0\right)\left(y\right) \ , \tag{37}$$

$$\forall \alpha = 1, \ldots, N: \quad \Delta\theta_\alpha = -\eta \nabla_\alpha \mathcal{L}\left(\theta\right) \ , \tag{38}$$

$$\Delta F_i\left(\theta\right)\left(x\right) = -\frac{1}{|X'|} \sum_{x' \in X'} \sum_{k=1}^{n_{L+1}} \Theta_{ik}\left(x, x'\right) \mathcal{C}'_k\left(F\left(\theta\right)\left(x'\right), y\left(x'\right)\right) \ . \tag{39}$$

Here, θ_α denotes the α-th component of the parameter vector, with $\alpha = 1, \ldots, N$, where N is the total number of parameters. The operator $\nabla_\alpha = \nabla_{\theta_\alpha}$ denotes differentiation with respect to the α-th parameter. The Jacobian of the network at initialization is denoted as:

$$\nabla_\alpha F_i\left(\theta_0\right)\left(x\right) = \left.\nabla_\alpha F_i\left(\theta\right)\left(x\right)\right|_{\theta=\theta_0} \ . \tag{40}$$

C Symmetry Bound Effect

In Equation 16, we bound the transformation matrix by some constant C. Here we explain the role of this constant, and show that its overall scaling effect is equivalent to changing the learning rate.

To separate the scale of the transformation from its remaining effect, we split this scale from the geometry of the transformation:

$$\Gamma = C\bar{\Gamma}, \qquad \|\bar{\Gamma}\| \leq 1 \ . \tag{41}$$

The matrix $\bar{\Gamma}$ then describes how the symmetry changes the relative importance of different parameter-space directions, while C controls their overall scale.

This distinction can be seen directly from gradient descent. With our convention, a linear symmetry is given by $\theta' = \Gamma^{-T}\theta$. Pulling the transformed trajectory back to the original coordinates, and denoting it by $u = g^{-1}(\theta')$, gives:

$$\Delta u = -\eta \Gamma^T \Gamma \nabla \mathcal{L}(u) = -(\eta C^2) \bar{\Gamma}^T \bar{\Gamma} \nabla \mathcal{L}(u) \ . \tag{42}$$

Thus, once $\bar{\Gamma}$ is fixed, the scalar C enters the dynamics only through the effective learning rate:

$$\eta_{\text{eff}} = C^2 \eta \ . \tag{43}$$

This is an exact relation, and does not rely on the NTK approximation. Therefore, the scaling effect of increasing C is the same as increasing the learning rate, and can be compensated exactly by $\eta \mapsto \eta/C^2$.

The same relation appears in the kernel. Writing $J = \nabla F$ at the point where the symmetry is gives us:

$$\Theta' = \eta J^T \Gamma^T \Gamma J = \eta C^2 J^T \bar{\Gamma}^T \bar{\Gamma} J \; . \tag{44}$$

The normalized matrix $\bar{\Gamma}$ does not itself need to define a symmetry, it is only a convenient way to separate the overall scale of the actual transformation from its geometric effect.

Therefore, choosing a large C does not introduce a new mechanism by which the NTK approximation can fail. Its scalar effect is exactly the same as choosing a larger effective learning rate. If C becomes large enough to substantially increase the deviation from linearization, this corresponds to the same behavior obtained from an excessively large learning rate.

D Empirical Check of Bounded Rescaling

To complement the theoretical analysis in Section 3.2, we empirically examine how a finite sequence of symmetry teleportation steps affects both the parameterization and the empirical NTK in a controlled setting. Following Zhao et al. (2023) and their implementation (Rose STL Lab, 2024), we consider a 3-layer MLP (784→16→10→10) without biases and with LeakyReLU activations (negative slope 0.1) on FashionMNIST, with weights drawn uniformly from $[-1/\sqrt{\text{fan_in}},\, 1/\sqrt{\text{fan_in}}]$ (PyTorch default initialization of linear layers).

Prior to any training, we apply 10 teleportation steps, each performing gradient ascent (step size 0.05) on the squared norm of the cross-entropy loss gradient wrt. the transformation parameters of the data-dependent group action of Zhao et al. (2022a). The loss is computed on a batch of 500 training samples, drawn at random for each run. We compute the empirical NTK on a separate fixed batch of 500 samples, held constant across all runs to ensure a fair comparison. As is common in practice, we sum the kernel over the output classes, $\sum_c \nabla F_c(x)^T \nabla F_c(x')$, instead of forming the full output-by-output kernel of Equation 8, but note that these two have the same trace. We repeat the experiment for 5 independent random initializations; $\pm$ values denote the standard deviation over runs.

Figure 4 shows the distribution of elementwise weight ratios r before and after teleportation. The distributions are sharply peaked at one, with median $\tilde{r} = 0.998 \pm 0.001$ for $w_{\text{after}}/w_{\text{before}}$ and 1.002 ± 0.001 for its inverse, indicating that most parameters undergo only mild rescaling. The mean absolute weight change, $\langle |\Delta W| \rangle = 5.5 \times 10^{-4} \pm 2.1 \times 10^{-4}$, is small compared with the mean absolute weight magnitude of approximately 0.020. Rare outliers arise from near-zero weights and occasional sign flips ($0.43\% \pm 0.20\%$ of parameters per run). The outliers affect the mean ratio, $\langle r \rangle = 1.143 \pm 0.284$ for $w_{\text{after}}/w_{\text{before}}$ and 1.014 ± 0.016 for its inverse, which is why we report medians as the typical rescaling.

At the level of function-space geometry, Figure 1 reports the sorted NTK eigenvalues scaled by the step size $\eta = 10^{-3}$ used to train this network with SGD with momentum 0.9. Teleportation induces a modest, nearly uniform increase across the spectrum ($\lambda_{\max}$ ratio 1.08 ± 0.07, trace ratio 1.10 ± 0.05), while preserving the overall spectral decay.

D.1 Estimating the Rescaling

While the weight ratios in Figure 4 are informative, they do not determine C directly. Since the group action mixes the weights within each layer, the ratio of a single weight is not a scale factor of Γ. Instead, we measure how the transformation rescales the network gradients, which is the quantity directly bounded in Equation 15.

Strictly speaking, this does not verify the stronger operator-norm condition in Equation 16, which we use in Theorem 4.1 to guarantee that the gradient and Hessian bounds remain controlled after the transformation. However, this condition is a worst-case bound over all parameter-space directions, including directions that may have little effect on the actual learning dynamics. What matters dynamically is how strongly Γ acts on the gradient and curvature directions explored by the network. Measuring the gradient rescaling, together with the change in the NTK spectrum, therefore

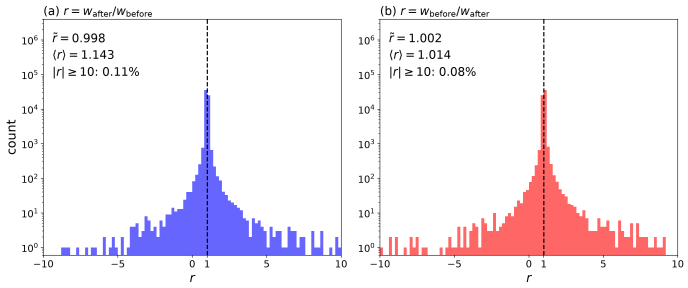


Figure 4: Distribution of element-wise weight ratios r before and after symmetry teleportation, (a) $r = w_{\text{after}}/w_{\text{before}}$ and (b) its inverse, pooled over 5 runs. The dashed line marks $r = 1$, and $\tilde{r}$ and $\langle r \rangle$ denote the median and the mean of r. The distributions are sharply concentrated at one: about 91% of the weights change by less than 10%, and the median ratio is 0.998 in (a) and 1.002 in (b). The tails come from weights close to zero, for which even a small absolute change produces a large ratio. Sign flips (0.43% of the weights) give negative ratios. The vertical axis is logarithmic to make rare outliers more visible. Ratios with $|r| \geq 10$ (0.11% in (a), 0.08% in (b)) lie outside the plotted range but are included in the reported statistics.

gives a natural estimate of the effective rescaling in these relevant directions, even though it does not directly bound the full operator norm $\|\Gamma\|$.

By Equation 8, the diagonal of the empirical NTK gives the squared norm of the network Jacobian for each input. Thus, taking the square root of the ratio between its diagonal before and after the transformation gives the corresponding gradient-rescaling factor.

Across the five runs, this factor averages 1.05 ± 0.02 over the 500 samples of the NTK batch and reaches at most 1.30 ± 0.10, while for some samples it is below one. The largest rescaling factor observed in any run is 1.43. Thus, in the gradient directions explored by the network, the effective rescaling remains close to one. This is consistent with the weight ratios, about 91% of which lie within 10% of one. The rare extreme weight ratios do not correspond to large gradient rescaling, since they mostly arise from weights close to zero, where a small absolute change can produce a large ratio.

The same picture appears at the level of the kernel. The nearly uniform increase of the NTK spectrum by about 10% in Figure 1 indicates that the transformation acts primarily as a modest overall rescaling in the directions seen by the network, corresponding to an effective kernel-scale factor of approximately $C_{\text{eff}}^2 \approx 1.1$. The full norm $\|\Gamma\|$ may still be larger, since the NTK batch probes only a subset of parameter-space directions.

Finally, the group action used in this experiment preserves the network function only approximately. After transforming a hidden layer, the implementation compensates in the following layer using the pseudo-inverse of the hidden activations on the teleportation batch. Such compensation can be exact only when the layer has at least as many units as the batch has samples. Here, the hidden layers contain only 16 and 10 units, compared with a teleportation batch of 500 samples, so the compensation is necessarily approximate.

E Generalized Symmetries

E.1 Hypothesis Function Extension

Here we demonstrate that any reversible matrix transformation Γ, as defined in Equation 13, can be regarded as a symmetry for any hypothesis function, as noted in Section 3.2.

To formalize this, consider a hypothesis function from the class described in Equation 1. We define its *extension* as:

$$\tilde{F}\left(\tilde{\Gamma}, \theta\right) = F\left(\tilde{\Gamma}\theta\right) , \tag{45}$$

where $\tilde{\Gamma}$ is an invertible matrix.

Setting $\tilde{\Gamma} = I$ recovers the original hypothesis function. Importantly, the new hyperparameter $\tilde{\Gamma}$ is *not* updated during learning; rather, it defines a transformation that induces a symmetry.

Starting from an initial choice $\tilde{\Gamma} = I$, any subsequent reversible matrix Γ defines a symmetry, provided we simultaneously transform the parameters as:

$$\tilde{\Gamma} = \Gamma^T, \quad \theta \mapsto \Gamma^{-T}\theta \,. \tag{46}$$

Verifying that this defines a symmetry is straightforward:

$$\tilde{F}\left(I, \theta\right) \mapsto \tilde{F}\left(\Gamma^T, \Gamma^{-T}\theta\right) = F\left(\Gamma^T\Gamma^{-T}\theta\right) = F\left(\theta\right) = \tilde{F}\left(I, \theta\right) \,. \tag{47}$$

The transposes follow from Equation 13. Generalizing to arbitrary initializations of $\tilde{\Gamma}$ is similarly simple but unnecessary for our purposes. Since we only apply the transformation once during the learning process, it suffices to assume $\tilde{\Gamma} = I$ initially.

If we wish to restrict ourselves to vector transformations, we can replace the matrix Γ with a vector c and define the extension and corresponding symmetry using the Hadamard (element-wise) product, with $c_\alpha \neq 0$:

$$\tilde{F}\left(c, \theta\right) = F\left(c \odot \theta\right) \quad , \quad \theta \mapsto c^{-1} \odot \theta \,. \tag{48}$$

E.2 Example - Fully Connected Neural Networks

In the case of fully connected neural networks, as with most architectures, the parameter vector θ is structured into a sequence of matrices and vectors (as shown in Equations 3 and 4). Consequently, to preserve the structure of the Hadamard product under the transformation, the hyperparameter vector c must likewise be organized into a corresponding sequence:

- Hyperparameter matrices: $c^{(1,0)} \in \mathbb{R}^{n_1 \times n_0}, \ldots, c^{(L+1,L)} \in \mathbb{R}^{n_{L+1} \times n_L}$,
- Hyperparameter vectors: $c^{(1)} \in \mathbb{R}^{n_1}, \ldots, c^{(L+1)} \in \mathbb{R}^{n_{L+1}}$.

To extend the network, we simply replace each parameter θ_α with $c_\alpha \odot \theta_\alpha$. Thus, Equation 3 becomes:

$$\begin{aligned} F^{(1)} &= \left(c^{(1,0)} \odot \theta^{(1,0)}\right) x + c^{(1)} \odot \theta^{(1)}, \\ F^{(l+1)} &= \left(c^{(l+1,l)} \odot \theta^{(l+1,l)}\right) \phi\left(F^{(l)}\right) + c^{(l+1)} \odot \theta^{(l+1)} \,, \end{aligned} \tag{49}$$

where $\odot$ denotes the element-wise product between matrices or vectors.

For the linear symmetries described for fully connected networks in Section 2.1, it can be verified that all derivatives of the network after applying the real symmetry, and those of the extended network after applying the generalized symmetry, coincide. In fact, they agree wherever the original linear symmetry identity holds, without requiring analyticity.

This supports the idea that generalized symmetries capture and extend the notion of symmetry for any transformation Jacobian Γ, including cases that would not naturally arise from the structure of the function itself. This perspective connects to recent architectural approaches such as graph metanetworks Lim et al. (2023) and equivariant representations of neural networks Kofinas et al. (2024).

F Wide Neural Networks Linearization

In this section we prove Lemma 3.1 and discuss the generalization of Theorem 4.1 for nonlinear symmetries.

F.1 Lemma 3.1 Proof

We begin by proving the lemma. Writing $J(\theta) = \nabla F(\theta)(X')$ and $M = |X'|$, suppressing n as before. The fixed number of inputs and output coordinates allows us to use the coordinatewise Hessian bounds for this stacked Jacobian.

We proceed by induction. Assume that at step $s-1$ the following conditions hold:

1. Lazy training:
$$\|\theta(s-1)-\theta_0\| = O(1), \qquad \|(\theta(s-1)-\theta_0)/\sqrt{\eta}\| = O(1). \tag{50}$$
2. Gradient stability:
$$\|\sqrt{\eta}(J(\theta(s-1))-J(\theta_0))\| = O\left(m_n\right), \quad \|\eta(J(\theta(s-1))-J(\theta_0))\| = O\left(m_n\right). \tag{51}$$
3. Bounded outputs:
$$\|F(\theta(s-1))(X')\| = O(1), \qquad \|f(s-1)\| = O(1). \tag{52}$$
4. Linearization:
$$\|F(\theta(s-1))(X') - F^{\mathrm{lin}}(s-1)(X')\| = O\left(m_n\right). \tag{53}$$

The base case follows from the initialization assumptions. We show that all four conditions hold at the next step. Notice that the gradient we keep nearly fixed is the network Jacobian, not the loss gradient, which changes as the error signal changes.

Lazy training:

The gradient descent update is $\Delta\theta = -\eta J(\theta(s-1))f(s-1)/M$. By the induction assumptions and Equation 19,
$$\|\Delta\theta\| = O(1), \qquad \|\Delta\theta/\sqrt{\eta}\| = O(1). \tag{54}$$
Adding this to the previous displacement proves Equation 50. Since the number of steps is fixed, the trajectory remains inside a fixed-radius ball around initialization.

Gradient stability:

Set $d = \theta(s) - \theta_0$. For each output coordinate, integrating the Hessian along the segment gives
$$\eta\left(\nabla F_i(\theta(s)) - \nabla F_i(\theta_0)\right) = \int_0^1 H_i(\theta_0 + td)\, d\, dt\ . \tag{55}$$
The η-scaled Jacobian difference is therefore bounded by $O\left(m_n\right)\|d\|$, and the $\sqrt{\eta}$-scaled difference by $O\left(m_n\right)\|d/\sqrt{\eta}\|$. Both are $O\left(m_n\right)$, establishing Equation 51. Writing $\Theta_s = \eta J(\theta(s))^T J(\theta(s))$, it follows that
$$\|\Theta_s - \Theta_0\| = O\left(m_n\right), \qquad \|\Theta_s\| = O(1)\ . \tag{56}$$

Linearization:

The Taylor remainder of a single update is bounded by a constant times $\sup_i \|H_i\|\,\|\Delta\theta/\sqrt{\eta}\|^2 = O\left(m_n\right)$. Thus,
$$F(\theta(s))(X') = F(\theta(s-1))(X') - \frac{1}{M}\Theta_{s-1} f(s-1) + O\left(m_n\right)\ . \tag{57}$$
The linearized model satisfies the exact update
$$F^{\mathrm{lin}}(s)(X') = F^{\mathrm{lin}}(s-1)(X') - \frac{1}{M}\Theta_0 f_{\mathrm{lin}}(s-1)\ . \tag{58}$$
The local Lipschitz continuity of the loss gradient gives $\|f(s-1) - f_{\mathrm{lin}}(s-1)\| = O\left(m_n\right)$. Subtracting these updates and using kernel stability proves Equation 53 at step s. The same calculation with a cross-kernel gives the result at any fixed test input.

Finally, the output update is $O(1)$, and the loss gradient remains bounded on the resulting bounded output set. This proves Equation 52 and completes the induction. Under random initialization, the argument applies on events where the uniform bounds hold, giving the same stochastic big-O conclusion. Constants may depend on the fixed number of steps.

For a positive semidefinite preconditioner P, held fixed during training, with $\|P\| \le 1$, the argument is unchanged after replacing the update by $-\eta P J f/M$ and the kernel by $\eta J^T P J$. This is the normalized dynamics used in Appendix C.

For piecewise-smooth networks, such as ReLU networks, the argument requires one additional observation. As the Hessian is not defined at the activation boundaries, we cannot directly integrate it along the entire parameter-space segment as in the smooth case. However, the same linearization argument still goes through once the contribution of activation changes is controlled. Under standard random initialization and NTK scaling, the training trajectory remains inside the fixed-radius neighborhoods considered above, while only neurons whose preactivations are sufficiently close to zero can change the activation state under the corresponding small per-neuron perturbations.

By the usual anti-concentration argument, only a vanishing fraction of neurons satisfy this condition, and under NTK normalization their combined contribution to the Jacobian variation and Taylor remainder also vanishes with width Jacot et al. (2018); Lee et al. (2019). On each region between activation changes, the usual smooth argument applies. The Barycentric Mean Value Corollary (Corollary I.1) then combines these smooth pieces, while the activation-stability bounds control the Jacobian changes introduced at the boundaries.

These activation-stability estimates are taken uniformly over the relevant fixed-radius neighborhoods. As in the smooth case, a radius-R neighborhood after a bounded linear symmetry pulls back into a radius-CR neighborhood before it, so the same estimates continue to apply after the transformation. Thus, if the resulting Jacobian-variation and Taylor-remainder terms decay at rate $O(m_n)$, the induction above is unchanged and gives the same $O(m_n)$ linearization error after the symmetry.

F.2 Theorem 4.1 Nonlinear Case

In Section 4, we proved that parameter-space symmetries do not alter the conditions of Lemma 3.1 in the case of linear symmetries. Here we extend this discussion to nonlinear symmetries.

We define the second-derivative tensor of the transformation as:

$$T_{\alpha\beta\gamma} = \partial_\alpha \partial_\beta g_\gamma \,, \tag{59}$$

which captures the deviation of the symmetry from linearity.

For nonlinear symmetries, the scaled Hessian of each network output transforms as:

$$H'_{\alpha\beta} = \eta \Gamma_{\alpha\mu} \partial_\mu \left(\Gamma_{\beta\nu} \partial_\nu F\right) = \Gamma_{\alpha\mu}\Gamma_{\beta\nu} H_{\mu\nu} + \eta \Gamma_{\alpha\mu} (\partial_\mu \Gamma_{\beta\nu}) \partial_\nu F \,, \tag{60}$$

where we suppress the output coordinate and use Einstein notation. The first term is $\Gamma H \Gamma^T$, which has already been analyzed. The second term requires further examination.

We connect this second term to T as follows. Starting from:

$$\Gamma^{-1}_{\alpha\beta}\, \Gamma_{\beta\nu} = \delta_{\alpha\nu} \,, \tag{61}$$

differentiating with respect to θ_μ gives:

$$\partial_\mu \Gamma^{-1}_{\alpha\beta}\, \Gamma_{\beta\nu} + \Gamma^{-1}_{\alpha\beta}\, \partial_\mu \Gamma_{\beta\nu} = 0 \,. \tag{62}$$

Multiplying on the left by $\Gamma_{\gamma\alpha}$ yields:

$$\partial_\mu \Gamma_{\beta\nu} = -\,\Gamma_{\beta\rho} \left(\partial_\mu \Gamma^{-1}_{\rho\sigma}\right) \Gamma_{\sigma\nu} \,. \tag{63}$$

Combining with Equation 13 we obtain:

$$\partial_\mu \Gamma^{-1}_{\rho\sigma} = \partial_\mu \partial_\rho g_\sigma = T_{\mu\rho\sigma} \,. \tag{64}$$

Thus:

$$\partial_\mu \Gamma_{\beta\nu} = -\,\Gamma_{\beta\rho}\, T_{\mu\rho\sigma}\, \Gamma_{\sigma\nu} \,. \tag{65}$$

Hence, denoting the second term by E, we obtain

$$E_{\alpha\beta} = -\eta \Gamma_{\alpha\mu}\Gamma_{\beta\rho} T_{\mu\rho\sigma} (\Gamma \nabla F)_\sigma, \qquad \|E\| \le C^2 \eta \|T \cdot (\Gamma \nabla F)\| \,, \tag{66}$$

where $(T \cdot v)_{\mu\rho} = T_{\mu\rho\sigma} v_\sigma$. The same linearization rate is therefore preserved if $\eta \|T \cdot (\Gamma \nabla F)\| = O(m_n)$ for every output, uniformly on the relevant neighborhoods. As in the linear case, the local inverse must map these neighborhoods into fixed-radius neighborhoods where the original estimates hold.

This provides a guideline for designing nonlinear symmetries. Keeping the additional term small preserves the Hessian-based linearization guarantee, while allowing it to grow provides a possible way to move beyond that guarantee. Whether this improves learning is then a question about the resulting dynamics, rather than the tensor norm alone.

G EXPERIMENTAL VERIFICATION OF THE NTK LIMIT PRESERVATION

In this appendix, we test the preservation of the predicted linearization behavior under the studied transformations. We implemented our experiments in Python using JAX, leveraging the Neural Tangents library Novak et al. (2019) with a customized Stax layer for the generalized symmetry. All experiments use fully-connected feedforward networks (MLPs) with the *NTK parametrization*, trained on the FashionMNIST dataset Xiao et al. (2017). All computations were performed on a single NVIDIA RTX 4090 GPU.

To verify the NTK limit for standard and transformed networks, we tracked the root mean squared error (RMSE) between the outputs of a neural network and its linearized counterpart, computed over all ten outputs on the 10,000 test images. We compare this metric with the reference scalings $1/\sqrt{n}$ and $1/n$, where n is the hidden-layer width, and report the exponent of a fitted power law.

We used networks with two hidden layers of equal width $n \in \{100, 150, \ldots, 500, 600, 700, \ldots, 1000, 1250, 1500\}$, ReLU activations, and the mean squared error loss, averaged over samples and the ten outputs, with one-hot labels. The standard network was initialized using the NTK parametrization, and for simplicity, biases were omitted.

The transformed network was obtained by using the generalized symmetry, as described in Section E, with scaling hyperparameters $c^{(l+1,l)}$. The generalized symmetry transformation consists of scaling each parameter at initialization, and compensating for that through the architectural change (see below). Specifically, each initialized weight was divided by the corresponding hyperparameter $c^{(l+1,l)}$, which remained fixed throughout training. These hyperparameters were obtained through an optimization process with the objective of maximizing the empirical loss gradient norm $\|\nabla\mathcal{L}(\theta_0)\|$, evaluated on a random batch of 512 training images at the same initialization θ_0 that is subsequently trained. Starting with uniform values (all ones), the method employs 500 steps of the Adam optimizer with learning rate $\eta = 0.01$ to iteratively update these hyperparameters.

To maintain stability, the optimization includes regularization that penalizes large deviations from initial values via a term $\lambda\|c-1\|^2$, summed over all entries, where $\lambda = 10^{-5}$ was used. Additional constraints include clipping the gradient of this objective to norm 1 to prevent large updates, and projecting the hyperparameters onto $[c_{\min}, c_{\max}] = [0.1, 10]$ after every step; the lower bound excludes $c_i = 0$, for which the transformation is undefined. In practice, no hyperparameter decreases below 0.9999, since the gradient of the transformed network is $c \odot \nabla_W \mathcal{L}$ and maximizing its norm increases c. The resulting hyperparameters are sparse: the mean deviation $|c-1|$ is between 0.04% ($n = 1500$) and 1.1% ($n = 100$), while a few entries grow to $c \approx 7$, increasing the gradient norm by a factor of 1.9 to 3.9.

The modification in the transformed network's architecture consisted of applying a Hadamard product between the hyperparameters and corresponding weights in the forward pass, which compensates for the rescaling of the weights and thereby realizes the generalized symmetry. Both networks (standard and transformed) were trained together with their linearizations, each taken around the network's own initial parameters, using SGD without momentum for 200 epochs (23,400 batches of size 512 from the 60,000 training images), with a learning rate of 1. All four models saw the identical sequence of minibatches, reshuffled every epoch. To show that the result holds beyond plain SGD, we repeated the experiment with momentum 0.9 and otherwise identical settings.

The results are presented in Figure 2 (SGD) and Figure 5 (SGD with momentum). Each data point represents the average over five independent trials, with error bars indicating the standard deviation across these trials. For each network, all widths are evaluated at the batch where the test accuracy, averaged across all widths and trials, first reaches 70% (Figure 6). The threshold lies where training starts to saturate, with the accuracy gain dropping from about 5 to 1 percentage points per 1000 batches (SGD). The fitted exponents are -0.41 for the standard and -0.49 for the transformed network in both settings. At later accuracy thresholds, the exponents become smaller in magnitude for both networks, and the linearization error of the transformed network still decreases with width at a rate comparable to the standard one, as predicted. Pairs of networks (standard and transformed) in each trial share the same initialization up to a symmetry transformation, as well as the same minibatches, which induces a correlation between their outputs and partially aligns the fluctuations around the fitted scaling law.

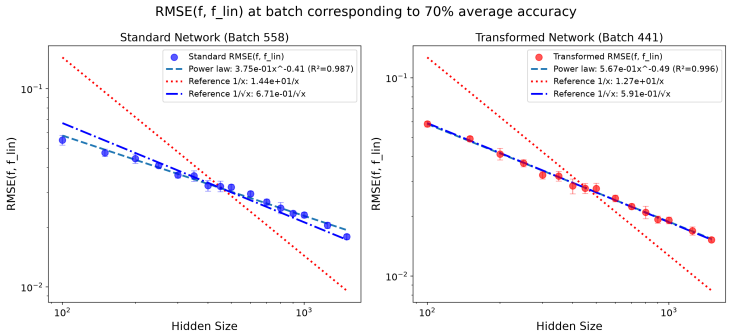


Figure 5: Same experiment as in Figure 2, but trained with SGD with momentum 0.9 all other settings are identical (App. G). **Left:** standard MLP at batch 558. **Right:** transformed MLP at batch 441. Each panel is evaluated at the batch at which the test accuracy, averaged over all widths, first reaches the threshold of 70% (see Figure 6(b)).

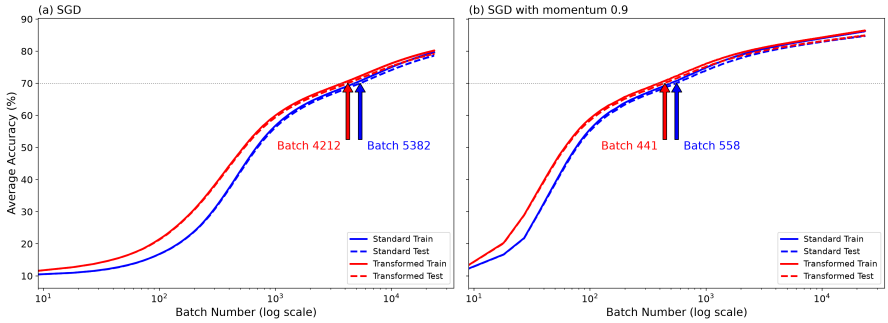


Figure 6: Training (solid) and test (dashed) accuracy, averaged across all hidden sizes and 5 independent trials, over training iterations (batches, logarithmic axis). **(a)** Plain SGD, the setting of Figure 2. **(b)** SGD with momentum 0.9, the setting of Figure 5. The batch at which each network's average test accuracy reaches 70% is highlighted; these are the batches at which the RMSE is evaluated. The transformed network reaches the threshold earlier after optimization of the hyperparameters $c^{(l+1,l)}$: batch 4212 vs. 5382 with SGD ($1.28\times$ speedup) and batch 441 vs. 558 with momentum ($1.27\times$). The threshold of 70% lies where training starts to saturate.

H Modified Alignment and Training Dynamics

In this section we show, both numerically and analytically, that our new definition of neural tangent alignment in Equation 25 is an indicator of the optimization rate.

H.1 Theoretical Justification

We begin with the theoretical perspective, showing that if the initial alignment is small, then the training rate must also be small.

We work with the setup described in Section 2.2, with MSE loss in the strictly linear limit. Write $\bar{\Theta} = \Theta(X', X')/|X'|$. The training dynamics are:

$$f(s+1) = \left(I - \bar{\Theta}\right) f(s) \,, \tag{67}$$

and the loss is:

$$\mathcal{L}(s) = \frac{1}{2|X'|} \, \|f(s)\|_2^2 \,. \tag{68}$$

We make the following argument:

Proposition H.1 (Small Kernel Alignment Limits Optimization Rate) Given the setup above, with $f(0) \neq 0$, let A be the initial alignment in Equation 25 and $a = A/|X'|$. Then, for every integer $s \geq 1$,

$$\frac{\mathcal{L}(s)}{\mathcal{L}(0)} \geq (1-a)^{2s} \geq 1 - 2sa \,, \tag{69}$$

and, for $0 \leq a < 1$, the average exponential loss-decay rate is bounded above by:

$$2\ln\left(\frac{1}{1-a}\right) = 2a + O\left(a^2\right) \,. \tag{70}$$

Proof. Since $\bar{\Theta}$ is symmetric positive semi-definite (PSD), diagonalize it as:

$$\bar{\Theta} = U\,\Lambda\,U^T \,, \qquad \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m) \,, \;\; \lambda_i \geq 0 \,. \tag{71}$$

Here $m = |X'|d_y$. Writing $f(0) = Uv$ with $v \in \mathbb{R}^m$, the linear dynamics in equation 67 imply

$$f(s) = (I - \bar{\Theta})^s f(0) = U\,(I - \Lambda)^s v \,. \tag{72}$$

Therefore,

$$\frac{\|f(s)\|_2^2}{\|f(0)\|_2^2} = \sum_{i=1}^{m} w_i\,(1-\lambda_i)^{2s} \qquad \text{with} \qquad w_i := \frac{v_i^2}{\sum_{j=1}^m v_j^2} \,, \;\; w_i \geq 0 \,, \;\; \sum_{i=1}^{m} w_i = 1 \,. \tag{73}$$

For fixed $s \geq 1$, the function $g(\lambda) := (1-\lambda)^{2s}$ satisfies

$$g''(\lambda) = (2s)(2s-1)\,(1-\lambda)^{2s-2} \geq 0 \,, \tag{74}$$

hence g is convex on $[0, \infty)$. By Jensen's inequality applied to Equation 73,

$$\sum_{i=1}^{m} w_i\, g(\lambda_i) \geq g\left(\sum_{i=1}^{m} w_i \lambda_i\right) = \left(1 - \sum_{i=1}^{m} w_i \lambda_i\right)^{2s} \,. \tag{75}$$

Observe that

$$\sum_{i=1}^{m} w_i \lambda_i = \frac{f(0)^T \bar{\Theta} f(0)}{\|f(0)\|_2^2} = \frac{A}{|X'|} = a \,. \tag{76}$$

Combining Equations 73, 75, and 76 gives

$$\frac{\|f(s)\|_2^2}{\|f(0)\|_2^2} \;\geq\; \left(1-a\right)^{2s} \,. \tag{77}$$

The common loss normalization cancels, giving:

$$\frac{\mathcal{L}(s)}{\mathcal{L}(0)} \geq (1-a)^{2s} \geq 1 - 2sa \,, \tag{78}$$

The second inequality follows from convexity and the tangent bound at $a = 0$.

Finally, when $0 \leq a < 1$:

$$\frac{\mathcal{L}(s)}{\mathcal{L}(0)} \geq e^{\ln\left((1-a)^{2s}\right)} = e^{-\left(2\ln\left(\frac{1}{1-a}\right)\right)s} \,, \tag{79}$$

showing that the average exponential loss-decay rate is bounded above by $2\ln\left(\frac{1}{1-a}\right)$. □

The conclusion that small initial alignment limits optimization also extends beyond MSE. Assume that each sample loss is nonnegative and convex on $\mathbb{R}^{d_y}$, with a β-Lipschitz output gradient for some $\beta > 0$, and that $\beta\|\Theta\|/|X'| \leq 2$. We continue to work in the strictly linear NTK regime and write $M = |X'|$.

The function-space update is $\Delta F = -\Theta f(s)/M$. Convexity gives

$$\mathcal{L}(s+1) - \mathcal{L}(s) \geq \frac{1}{M}\langle f(s), \Delta F\rangle = -\frac{f(s)^T \Theta f(s)}{M^2} \,. \tag{80}$$

To relate later steps to initialization, let B_s be the averaged, block-diagonal Hessian of the sample losses along this update. Then $f(s+1) - f(s) = B_s \Delta F$ and $0 \preceq B_s \preceq \beta I$. For $u_s = \Theta^{1/2} f(s)$,

$$u_{s+1} = \left(I - \frac{1}{M} \Theta^{1/2} B_s \Theta^{1/2} \right) u_s \, . \tag{81}$$

The step-size assumption places every eigenvalue of this update matrix in $[-1, 1]$, so $\|u_{s+1}\| \leq \|u_s\|$. Summing Equation 80 therefore gives

$$\mathcal{L}(s) \geq \mathcal{L}(0) - \frac{s}{M^2} f(0)^T \Theta f(0) \, . \tag{82}$$

Finally, nonnegativity and β-smoothness give $\|f(0)\|^2 \leq 2\beta M \mathcal{L}(0)$. Apply the smoothness upper bound to each loss at the output shifted by minus its gradient divided by β, and use that its value is nonnegative. Hence, when $\mathcal{L}(0) > 0$ and $f(0) \neq 0$,

$$\frac{\mathcal{L}(s)}{\mathcal{L}(0)} \geq 1 - \frac{2\beta s A}{M} \, . \tag{83}$$

Thus, the small-alignment limitation survives for smooth convex losses under a stable step size. The sharper exponential expression in Proposition H.1 uses the fixed error recurrence of MSE, but its main conclusion does not depend on that special quadratic form.

H.2 Empirical Demonstration

In this appendix we show that our modified alignment (Equation 25), when measured at initialization, quantitatively predicts early optimization speed of the network. The experiment was implemented in Python using JAX and the Neural Tangents library, and executed on a single NVIDIA RTX 4090 GPU. It uses MLPs with two hidden layers of width $n = 1024$, ReLU, and the NTK parametrization, without biases for simplicity. The networks are trained on a subset of FASHION-MNIST consisting of $|X'| = 5000$ training images, with the per-output MSE loss $\mathcal{L} = \frac{1}{2|X'|d_y} \sum_{x \in X'} \|F(x) - y(x)\|^2$ on one-hot labels ($d_y = 10$), for which the output-space loss gradient f is proportional to the residual $F - y$.

For each of 1,000 independent random initializations, we evaluate the modified alignment (Equation 25) on all training images and normalize it as $a = A/|X'|$, as in Proposition H.1. Each network is then trained with stochastic gradient descent (learning rate $\eta = 1$, momentum $\beta = 0.9$, batch size 128), and the loss on the full training set is recorded after every step.

To first order, the NTK dynamics (Equation 11) predict the relative loss decrease after k steps as $-\log(\mathcal{L}_k/\mathcal{L}_0) \approx 2\eta_k a/\eta d_y$, where $\eta_k = \eta \sum_{t \leq k}(1-\beta^t)/(1-\beta)$ is the cumulative step size with momentum and the factor $1/d_y$ comes from averaging the loss over the outputs.

Figure 3 compares this prediction with the measured loss decrease. After $k = 10$ steps, the two agree closely (Pearson $r = 0.996$ between a and the measured decrease). The prediction is accurate early in training and later overestimates the decrease, consistent with the bound in Proposition H.1, while networks with larger alignment still train faster ($r \geq 0.96$ up to $k = 100$). In contrast, the standard kernel-target alignment (Equation 24) of the same networks does not describe any of the studied quantities well.

I Additional Derivations and Proofs

Here we present some of the mathematical calculations and details used throughout the paper.

I.1 Section 3.2

I.1.1 Reduction to the Invertibility of the Symmetry

First, we justify the reduction to invertible transformations in the global linear case. Suppose $g(\theta) = B\theta$ and $F(B\theta) = F(\theta)$ for every θ. Choose k such that $\ker B^k = \ker B^{k+1}$, and set $V = \ker B^k$. Every $v \in V$ is a redundant direction, since

$$F(\theta + v) = F(B^k(\theta + v)) = F(B^k \theta) = F(\theta) \, . \tag{84}$$

Let the columns of U be an orthonormal basis of $V^{\perp}$, and write $q = U^T\theta$. Then $F(\theta) = \widehat{F}(q)$ with $\widehat{F}(q) = F(Uq)$. Consequently,

$$\nabla_\theta \mathcal{L} = U\nabla_q \widehat{\mathcal{L}}, \qquad \Delta q = -\eta \nabla_q \widehat{\mathcal{L}}\,. \tag{85}$$

Thus the reduction preserves both the outputs and gradient descent exactly. Since $BV \subseteq V$, the induced symmetry is $q \mapsto U^T BUq$. This map is invertible: $U^T BUq = 0$ implies $BUq \in V$, hence $Uq \in \ker B^{k+1} = V$, so $q = 0$. Orthogonal restriction also preserves upper bounds on gradient and Hessian norms. We can therefore carry out the linear analysis on these effective parameters without changing the learning problem. For general nonlinear or only locally defined transformations, local invertibility is an explicit assumption; curved reductions require retaining the induced optimization metric.

I.1.2 Equation 14 Proof

Second, we prove that the transformation Jacobian defined in Equation 13 indeed governs how the derivatives of the network transform under the symmetry, as illustrated in Equation 14.

This is shown by observing that for any two parameter indices $\alpha, \beta = 1, \ldots, N$, according to the chain rule, the gradient with respect to the transformed parameters ∇' where $\theta' = g(\theta)$ is related to the original parameters θ, after the symmetry transformation as follows:

$$\left(\nabla' F\left(\theta'\right)\right)_\alpha = \sum_{\beta=1}^{N} \frac{\partial \theta_\beta}{\partial \theta'_\alpha} \frac{\partial F\left(g\left(\theta\right)\right)}{\partial \theta_\beta} = \sum_{\beta=1}^{N} \Gamma_{\alpha\beta} \frac{\partial F\left(\theta\right)}{\partial \theta_\beta}\,. \tag{86}$$

This calculation is valid because the symmetry is assumed to be well-defined and differentiable in a neighborhood of θ. We can then use the fact that in this neighborhood, by definition:

$$F\left(g\left(\theta\right)\right) = F\left(\theta\right)\,, \tag{87}$$

which yields the desired result.

For linear symmetries, the same rule applies to each derivative index. For nonlinear symmetries, derivatives of Γ contribute the additional terms analyzed above.

I.2 Section F.1

Corollary I.1 (Barycentric Mean Value) Let $a < b$ and let $f : [a, b] \to \mathbb{R}$ be continuous on $[a, b]$ and differentiable on (a, b) except at finitely many points. Then there exist

$$c_1, \ldots, c_N \in (a, b) \qquad \text{and} \qquad \lambda_1, \ldots, \lambda_N > 0,\ \sum_{j=1}^{N} \lambda_j = 1\,, \tag{88}$$

such that

$$\frac{f(b) - f(a)}{b - a} \;=\; \sum_{j=1}^{N} \lambda_j\, f'(c_j)\,. \tag{89}$$

Proof.

Denote by $a < d_1 < \cdots < d_k < b$ the (finite) set of points where f fails to be differentiable and set

$$d_0 := a, \qquad d_{k+1} := b\,. \tag{90}$$

Because f is continuous on $[a, b]$ and differentiable on each open sub-interval

$$I_j = (d_{j-1}, d_j) \quad (j = 1, \ldots, k + 1)\,, \tag{91}$$

the classical Mean Value Theorem applies to f on every $[d_{j-1}, d_j]$:

$$\exists\, c_j \in (d_{j-1}, d_j) \quad \text{s.t.} \quad f'(c_j) = \frac{f(d_j) - f(d_{j-1})}{d_j - d_{j-1}} \quad (j = 1, \ldots, k + 1)\,. \tag{92}$$

Define positive weights:

$$\lambda_j := \frac{d_j - d_{j-1}}{b-a}, \qquad j = 1, \dots, k+1 \,. \tag{93}$$

They satisfy $\sum_{j=1}^{k+1} \lambda_j = 1$ and $\lambda_j > 0$ because $d_j > d_{j-1}$.

Now compute:

$$\sum_{j=1}^{k+1} \lambda_j \, f'(c_j) = \frac{1}{b-a} \sum_{j=1}^{k+1} (d_j - d_{j-1}) \frac{f(d_j) - f(d_{j-1})}{d_j - d_{j-1}} = \frac{1}{b-a} \sum_{j=1}^{k+1} \big[f(d_j) - f(d_{j-1}) \big] \,. \tag{94}$$

The sum telescopes:

$$\sum_{j=1}^{k+1} \big[f(d_j) - f(d_{j-1}) \big] = f(d_{k+1}) - f(d_0) = f(b) - f(a) \,. \tag{95}$$

Hence:

$$\sum_{j=1}^{k+1} \lambda_j \, f'(c_j) = \frac{f(b) - f(a)}{b-a} \,. \tag{96}$$

Setting $N := k + 1$ completes the proof.

□