The Weight Gram Matrix Captures Sequential Feature Linearization in Deep Networks

Taehun Cha
Department of Mathematics
Korea University
cth127@korea.ac.kr

Daniel Beaglehole
Computer Science and Engineering
UC San Diego
dbeaglehole@ucsd.edu

Adityanarayanan Radhakrishnan
MIT Mathematics
Broad Institute of MIT and Harvard
aradha@mit.edu

Donghun Lee*
Department of Mathematics
Korea University
holy@korea.ac.kr

Abstract

Understanding how deep neural networks learn representations remains a central challenge in machine learning theory. In this work, we propose a feature-centric framework for analyzing neural network training by relating weight updates to feature evolution. We introduce a simple identity, the *Feature Learning Equation*, which identifies the weight Gram matrix as the key object capturing feature dynamics. This enables us to interpret gradient descent as implicitly inducing a hypothetical evolution of features, whose covariance structure — termed the *Virtual Covariance* — characterizes how representations evolve during training. Building on this perspective, we introduce *Target Linearity*, a measure quantifying the linear alignment between features and targets. By analyzing the training and layer-wise dynamics, we show that deep networks learn to *sequentially* transform representations toward target-linear structure. This linearization perspective provides a unified interpretation of several empirical phenomena, including Neural Collapse and linear interpolation in generative models. Code is available at https://github.com/cth127/GramLin.

1 Introduction

Feature learning is central to the success of deep neural networks, yet a unified account of how representations evolve during training remains limited. Existing work falls into two directions. General theories of training dynamics are largely parameter-centric, as in kernel-based analyses where features remain effectively fixed [17]. In contrast, recent work studies feature learning through analytically tractable case studies — e.g., single and multi-index models, staircase, polynomials, and teacher-student setups — showing that networks can provably learn task-relevant representations [1, 3, 10, 11]. However, these results rely on specific problem structures, leaving open how feature learning behaves in general, realistic, high-dimensional settings.

A natural way to address this gap is to analyze training directly in feature space. As a thought experiment, suppose that instead of updating weights, we directly update hidden features by following the loss gradient. Concretely, fix the input x and define a virtual trajectory $\tilde{x}_t$ that accumulates the feature-space gradient at each training step:

$$\tilde{x}_{t+1} \leftarrow \tilde{x}_t - \gamma \nabla_x \mathcal{L}_t, \qquad \tilde{x}_0 = x,$$

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

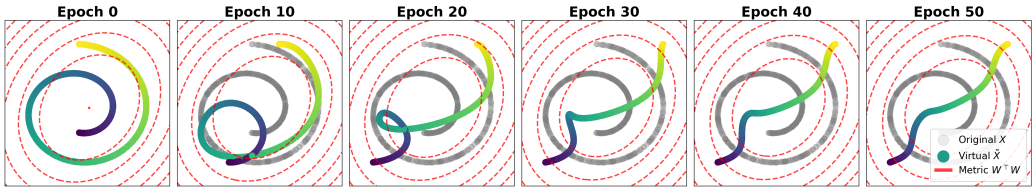

Figure 1: The virtual update $\tilde{X}_{t+1} \leftarrow \tilde{X}_t - \gamma \nabla_X \mathcal{L}_t$ progressively unrolls a Swiss roll into a near-linear curve along the target (encoded by color), shown across training epochs. Gray points denote the original input X; colored points denote the virtually updated $\tilde{X}$. The red dashed contours show the geometry induced by $W^\top W$ at the first layer — a connection developed in Section 3. The detailed experimental setup is provided in Section B.1.

where $\nabla_x \mathcal{L}_t$ is computed under the network with parameters θ_t with usual gradient descent update (GD) at step t. We call $\tilde{x}_t$ the *virtual update* of x. Note that $\tilde{x}_t$ does not affect the actual training trajectory; it tracks, in feature space, the direction GD would move features if features were the optimization variables. Figure 1 shows that the virtual update progressively unrolls Swiss roll data into a linear form. This raises a natural question: although GD operates in weight space, does it implicitly induce this feature-space evolution? Answering it requires a tool that connects weight-space updates to feature-space dynamics.

Such a tool follows directly from the chain rule. We begin by introducing a simple identity relating gradients in weight space to gradients in input (or hidden) space. For any function of the form $f(z)$ with $z = Wh$, the chain rule yields

$$\nabla_h f = W^\top \nabla_z f \quad \text{and} \quad \nabla_W f = \nabla_z f \cdot h^\top$$

Combining these gives

$$\nabla_h f \cdot h^\top = W^\top \nabla_W f. \tag{1}$$

The left-hand side describes how features evolve; the right-hand side describes how weights update. The equation allows us to translate between the two, and in particular, it identifies the weight Gram matrix $W^\top W$ as the natural object connecting them. We refer to this exact identity as the *Feature Learning Equation*, and use it throughout the paper to analyze feature evolution.

In Section 2, we examine the role of the weight Gram matrix more closely and present empirical evidence that it carries the dominant signal of feature learning. This enables us to analyze *what the weight Gram matrix learns* and, in turn, *how hidden features evolve* during training — i.e., *feature learning*. Building on this perspective, we make four theoretical contributions:

- Section 3: We prove that the weight Gram matrix learns the *Virtual Covariance*, which captures the covariance structure of virtually updated hidden states.
- Section 4.1: We introduce *Target Linearity*, a novel quantity measuring the linear relationship between hidden states and target labels, where the Gram structure appears again.
- Section 4.2: We characterize the training and layer-wise dynamics of Target Linearity. The training dynamics reveal that feature learning is a moving-target problem in which the network chases an evolving linear predictor, in contrast to the Neural Tangent Kernel regime (NTK, [17]) where the linear model is fixed. The layer-wise dynamics show that trained networks *sequentially* linearize input data with respect to the target, consistent with prior empirical findings on *linear probes* [2].
- Section 5: We show this linearization perspective can explain other feature learning phenomena. We show Neural Collapse [37] is an extreme case of target linearization, and that high Target Linearity is a sufficient condition for the linear interpolation property observed in the latent space of generative models [43], demonstrating the generality of our framework.

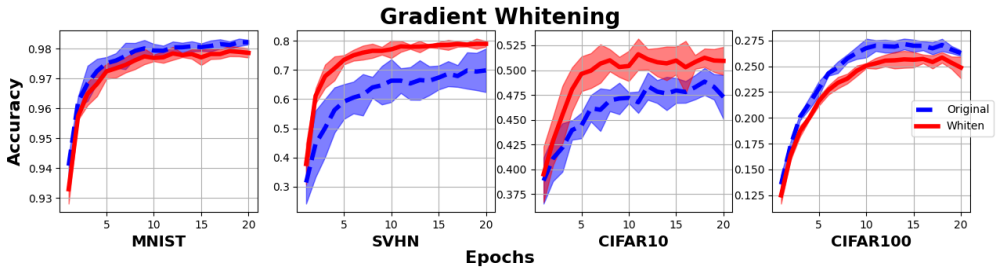

Figure 2: Test accuracy of standard gradient descent (Original, dashed) versus the whitened update (Whiten, solid) on MNIST, SVHN, CIFAR-10, and CIFAR-100, plotted across 20 training epochs. Shaded regions denote standard deviation across 5 runs.

2 Weight Gram Matrix as a Bridge

We now show how Equation (1) surfaces when analyzing the weight Gram matrix. Computing the Gram matrix of the gradient-descent-updated weight matrix yields

$$\begin{aligned}(W - \gamma \nabla_W \mathcal{L})^\top (W - \gamma \nabla_W \mathcal{L}) &= W^\top W - \gamma (W^\top \nabla_W \mathcal{L} + \nabla_W \mathcal{L}^\top W) + O(\gamma^2) \\ &\approx W^\top W - \gamma (\nabla_h \mathcal{L} \cdot h^\top + h \cdot \nabla_h \mathcal{L}^\top).\end{aligned} \tag{2}$$

where $\mathcal{L}$ is the loss function and γ is the learning rate. This identity makes a key point clear: *by taking the Gram of the weight matrix, we can directly access feature-space gradients and, through them, feature evolution*.

The weight Gram matrix is not an entirely novel object in the feature learning literature. Radhakrishnan et al. [40] recently proposed the *Neural Feature Ansatz* (NFA) as a step toward understanding feature learning in neural networks. They showed that the weight Gram matrix at each layer is closely related to the gradient of the network output with respect to that layer's input features. For a network f with l-th layer weight matrix W_l and corresponding input representation h_l, NFA conjectures

$$W_l^\top W_l \propto \mathbb{E}\left[\nabla_{h_l} f \cdot \nabla_{h_l} f^\top\right],$$

where the right-hand side is termed the *Average Gradient Outer Product* (AGOP). This relationship has been established theoretically in low-rank matrix completion, a widely studied setting for feature learning [41], and has also been validated empirically across a broad range of architectures and datasets.

The discussion so far establishes the weight Gram matrix as a natural object in feature learning analysis, both theoretically (via the FL equation) and through prior empirical observation (via NFA). A sharper question remains: Can the weight Gram matrix sufficiently capture the feature learning dynamics? We test this directly by isolating the Gram-modifying component of the gradient update.

Specifically, we consider an update direction Δ' that leaves the weight Gram matrix unchanged, found by solving

$$\arg\min_{\Delta'} \|\nabla_W \mathcal{L} - \Delta'\|_F \quad \text{subject to} \quad W^\top \Delta' + \Delta'^\top W = 0.$$

Setting $\Delta = \nabla_W \mathcal{L} - \Delta'$ and updating $W^+ = W - \gamma \Delta$ yields

$$(W^+)^\top W^+ = (W - \gamma \nabla_W \mathcal{L})^\top (W - \gamma \nabla_W \mathcal{L}) + O(\gamma^2),$$

so the Gram matrix evolves identically to that of standard GD, while the components of the gradient that leave the Gram unchanged are removed. We refer to Δ as the *whitened gradient*. If the weight Gram matrix carries the learning signal, training with Δ should recover the performance of standard GD; if it does not, performance should largely degrade.

We train three-layer fully connected networks on standard image classification benchmarks [23, 26, 35] using both standard GD and the whitened update. Figure 2 reports the results. On MNIST, SVHN, and CIFAR-10, the whitened update matches or exceeds the performance of standard GD; on SVHN and CIFAR-10 specifically, it reaches GD's best performance within four to five epochs and

shows lower variance across runs on SVHN. On CIFAR-100, it tracks standard GD with a small gap. The details for computing Δ are described in Section B.2, and comparable results for convolutional architectures [24] including ResNet [15] are deferred to Section B.4. Sensitivity to width, depth, and optimization hyperparameters is reported in Section B.3.

These results indicate that the Gram-modifying component of the gradient is largely sufficient to recover standard training performance: the residual component, while present in the gradient, contributes little to feature learning beyond what the Gram update already provides. This empirically substantiates the central role of the weight Gram matrix and motivates the analyses developed in the remainder of the paper, where we ask what the Gram matrix learns (Section 3) and what its evolution achieves at the level of features (Section 4).

3 Weight Gram Learns Virtual Covariance

Section 2 established the weight Gram matrix as a natural and empirically central object in feature learning analysis. We now ask: *what does the weight Gram matrix learn during training?* We show that each GD step shifts the weight Gram matrix by an amount that captures the covariance structure of *virtually updated* hidden states — the same virtual update introduced in Section 1. This formalizes the visualization in Figure 1: the Gram matrix tracks how features would evolve if features themselves were the optimization variables.

Settings We work with a standard L-layer fully connected network $f(x) = w^\top h_L$, where $h_l \in \mathbb{R}^{d_l}$ denotes the l-th hidden state, defined recursively as $h_l = \sigma(W_{l-1}h_{l-1})$ with $h_0 = x \in \mathbb{R}^{d_0}$ and σ a nonlinear activation function applied element-wise. The vector $w \in \mathbb{R}^{d_L}$ serves as the final predictor, and $W_l \in \mathbb{R}^{d_{l+1}\times d_l}$ are the weight matrices. For a loss function $\mathcal{L}$, we write $\nabla_{W_l}\mathcal{L} \in \mathbb{R}^{d_{l+1}\times d_l}$ for the gradient with respect to W_l.

Definition 1. *The **virtual update** of a hidden state h_l (or of the input $h_0 = x$) with learning rate γ is $h_l^+ = h_l - \gamma\nabla_{h_l}\mathcal{L}$. The **actual update** of a weight matrix W_l is $W_l^+ = W_l - \gamma\nabla_{W_l}\mathcal{L}$, the standard GD step. The **Virtual Covariance** (VC) is $\widetilde{\mathrm{cov}}(h_l^+) = h_l^+ \cdot (h_l^+)^\top$, and the **Virtual Covariance Shift** (VCS) is $\widetilde{\mathrm{cov}}(h_l^+) - \widetilde{\mathrm{cov}}(h_l)$.*

The virtual update represents how the input to the l-th layer would change under GD if hidden states themselves were the optimization variables. We use the term *virtual* because GD updates weights, not hidden states directly. The VCS therefore characterizes how the input covariance structure would evolve to reduce the loss.[2]

Weight Gram Dynamics The following theorem, our main result for this section, shows that the actual GD update on the weights produces precisely this VCS on the Gram matrix.

Theorem 1. *The shift in the Gram matrix of the gradient-descent-updated weight matrix equals the Virtual Covariance Shift up to $O(\gamma^2)$:*

$$(W_l^+)^\top W_l^+ - W_l^\top W_l = \widetilde{\mathrm{cov}}(h_l^+) - \widetilde{\mathrm{cov}}(h_l) + O(\gamma^2).^3$$

The proof is direct from Equation (2), as $\widetilde{\mathrm{cov}}(h_l^+) - \widetilde{\mathrm{cov}}(h_l) = -\gamma(\nabla_h\mathcal{L}\cdot h^\top + h\cdot\nabla_h\mathcal{L}^\top) + O(\gamma^2)$. We show this also holds for Convolutional neural network [24] in Section C.1.

The i-th column w_i of W determines how the i-th input entry x_i influences the next layer $h = \sigma(Wx)$. The Gram entry $(W^\top W)_{ij} = w_i^\top w_j$ therefore measures the correlation between the effects of x_i and x_j on h. In this sense, the weight Gram matrix is an *input covariance with respect to its influence on the next-layer representation*, while the VCS describes how this covariance *should* evolve to reduce the loss. Theorem 1 shows that GD on the weights aligns the two: the actual Gram update tracks the desired covariance shift, even though hidden states are never updated directly. The red dashed

[2] Strictly, $\widetilde{\mathrm{cov}}$ is not a covariance matrix since h_l^+ and h_l are not guaranteed to be centered; with standard normalization the two coincide.

[3] For a mini-batch update $W_l^+ = W_l - \gamma\sum_i \nabla_{W_l}\mathcal{L}_i$, the Gram shift accumulates the per-sample VCS, $(W_l^+)^\top W_l^+ - W_l^\top W_l \approx \sum_i[\widetilde{\mathrm{cov}}(h_{l,i}^+) - \widetilde{\mathrm{cov}}(h_{l,i})]$. Applying over τ consecutive steps, the Gram shift extends directly to the multi-step virtual update, $W_t^\top W_t - W_{t-\tau}^\top W_{t-\tau} \approx \tilde{h}_t\tilde{h}_t^\top - \tilde{h}_{t-\tau}\tilde{h}_{t-\tau}^\top$, where $\tilde{h}_t$ denotes the virtual trajectory defined in Section 1.

contours in Figure 1 make this evolution visible: each contour shows the density of a Gaussian with covariance $W^\top W$ at the first layer, adapting with the virtual unrolling of the spiral.

This result clarifies the relationship to the Neural Feature Ansatz of Radhakrishnan et al. [40], which conjectures $W^\top W \propto \frac{1}{N}\sum_{i=1}^{N} \nabla_{h_i} f \cdot \nabla_{h_i} f^\top$ (AGOP). An AGOP-like term — with the network output replaced by the loss — appears within the γ^2 residual of Theorem 1. The VCS therefore refines the AGOP correspondence by capturing the leading-order Gram update directly. Empirically, the VCS tracks the actual Gram update at $\rho = 0.98$, compared to $\rho = 0.71$ for AGOP (See Section B.5).

Relationship between Virtual vs. Actual Update Theorem 1 establishes that the weight Gram matrix learns the covariance structure of *virtually* updated features. A natural question is whether this is meaningful in practice: does the *actually* updated hidden state $\sigma(W_{l-1}^+ h_{l-1})$ behave similarly to the virtual update h_l^+? If actual and virtual updates diverge, W_l may be optimized to expect features that the previous layer never produces. The following proposition, stated for the single-sample case, shows the two are aligned in both sign and magnitude.

Proposition 1. *Assume σ is L-Lipschitz and $\|h_{l-1}\|_2 = 1$, and let W_{l-1}^+ denote the layer $l-1$ weight matrix after one gradient descent step on the single data point producing h_{l-1}, with all other layers held fixed. Then $|\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})| \le L^2 |h_l^+ - h_l|$ element-wise. If σ is additionally monotonically increasing, then $\operatorname{sgn}(\sigma(W_{l-1}^+ h_{l-1}) - \sigma(W_{l-1} h_{l-1})) = \operatorname{sgn}(h_l^+ - h_l)$; the sign equality also holds for non-decreasing σ such as ReLU under the convention $\operatorname{sgn}(0) \in \{-1, +1\}$.*

Proposition 1 shows that the actual update on the hidden state agrees with the virtual update in direction and is bounded by it in magnitude. For 1-Lipschitz activations such as ReLU, the magnitude bound reduces to $|h_l^+ - h_l|$ directly. The proof is given in Section C.2.

However, Proposition 1 assumes that only a single layer is updated. When all layers are trained simultaneously, the actual update can still be expressed as a preconditioned virtual update under gradient flow.

Proposition 2. *Consider a single input x and an L-layer network with $z_l = W_{l-1} h_{l-1}$ and $h_l = \sigma(z_l)$ for $l = 1, \dots, L$, where $h_0 = x$ and σ is differentiable. Suppose all weights evolve under gradient flow, $\dot{W}_l = -\nabla_{W_l} \mathcal{L}$ for $l = 0, \dots, L-1$. Then for every layer l,*

$$\frac{dh_l}{dt} = -M_l \nabla_{h_l} \mathcal{L}, \qquad M_l = \sum_{l'=1}^{l} \|h_{l'-1}\|_2^2 \, J_{l \leftarrow l'} J_{l \leftarrow l'}^\top \succeq 0,$$

where $J_{l \leftarrow l'} = \partial h_l / \partial z_{l'}$. Consequently, $\langle \frac{dh_l}{dt}, -\nabla_{h_l} \mathcal{L} \rangle \ge 0$.

Proposition 2 shows that, even when all layers are trained, the actual feature update is the virtual update $-\nabla_{h_l} \mathcal{L}$ preconditioned by a PSD matrix, and thus never moves against it. The proof is given in Section C.3.

We established that the weight Gram matrix tracks the Virtual Covariance Shift — the covariance structure that GD implicitly imposes on features. This tells us how features move, but not where they are going. Figure 1 already hinted at an answer: the virtual update doesn't just reshape covariance arbitrarily, it progressively aligns features with the target. To make this observation precise, we need a quantity that measures alignment between features and targets, expressed in terms of the same Gram structure that Theorem 1 identified as the carrier of feature dynamics. This motivates *Target Linearity*, introduced next: by construction, it is a function of the weight Gram matrix, so the covariance dynamics translate directly into dynamics of Target Linearity, letting us track not just how representations evolve, but how that evolution serves the task.

4 Weight Gram Captures Target Linearity

We now formalize the linearization phenomenon hinted in Section 3. We introduce *Target Linearity*, a quantitative measure of how well a linear model can predict the target from learned features, and analyze its dynamics during training and across layers. A central technical step is to identify a tractable surrogate that exposes the weight Gram matrix as the object that captures this linearization.

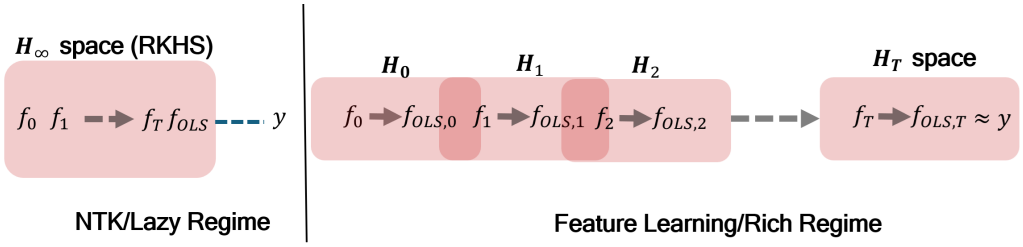

Figure 3: Training dynamics in the lazy (left) versus rich regimes (right). In the lazy regime, the network remains close to its linearization, and the feature map H_∞ is effectively fixed. In the rich regime, finite-width features H_t evolve during training, inducing a moving target $f_{\mathrm{OLS},t}$ that f_t chases, while feature updates simultaneously reshape $f_{\mathrm{OLS},t+1}$ toward y.

4.1 Target Linearity and its Surrogate

Definition 2. *For a data matrix $H = [h_1, \ldots, h_N] \in \mathbb{R}^{d_l \times N}$ and target vector $Y = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$, we define the **Target Linearity** (TL) as the coefficient of determination (R^2) of (ridge) linear regression: for $\lambda \geq 0$,*

$$\mathcal{T}_\lambda(H, Y) = 1 - \frac{\|Y - \hat{Y}_\lambda\|_2^2}{\|Y - \bar{Y}\|_2^2},$$

where $\hat{Y}_\lambda = H^\top (HH^\top + \lambda I)^{-1} HY$, $\bar{Y} = \bar{y} \cdot \mathbf{1}_N$, and $\bar{y} = \frac{1}{N}\sum_{i=1}^N y_i$.

At $\lambda = 0$ (and assuming $HH^\top$ is invertible), TL recovers the R^2 of ordinary least squares. Since TL is defined as the R^2 score of linear regression, it quantifies how well the target can be explained by a linear function of the features.

The features H appear in the squared error $\|Y - \hat{Y}_\lambda\|_2^2$ through multiple occurrences, making the dependence difficult to analyze directly. Applying the Woodbury identity yields $\|Y - \hat{Y}_\lambda\|_2^2 = \lambda^2 Y^\top (\lambda I + H^\top H)^{-2} Y$, which collapses the dependence on H into a single Gram-like term $H^\top H$. We define the Gram-induced error function $\mathcal{E}_\lambda(G_\sigma^l) = Y^\top (\lambda I + G_\sigma^l)^{-2} Y$, where $G_\sigma^l = \sigma(W_l H_l)^\top \sigma(W_l H_l)$. We drop the superscript l when the layer is clear from context.

However, $\mathcal{T}_\lambda$ still involves a squared matrix inverse and the nonlinearity σ inside G_σ, both of which obstruct direct analysis. To make the dependence on the Gram matrix tractable, we introduce a *surrogate* $\mathcal{S}(G) = Y^\top G Y$. Here G may denote either the nonlinear Gram $G_\sigma = \sigma(WH)^\top \sigma(WH)$ or the *linearized* Gram $G_{\mathrm{id}} = H^\top W^\top W H$, in which the activation has been removed. The surrogate admits the following lower-bound relation to TL.

Theorem 2. *Let $e_0 = \|Y - \bar{Y}\|_2^2$ and $e_1 = \|Y\|_2^2$, and assume $\|W\|_{\mathrm{op}} \leq c_0$, $\|H\|_F \leq c_1$, and $N > d$. Then*

$$\mathcal{T}_\lambda(H, Y) \geq 1 - C \cdot \mathcal{S}(G_{\mathrm{id}})^{-1},$$

for some fixed constant C.

Theorem 2 yields two benefits. First, the surrogate removes both the nonlinearity and the squared matrix inverse that obstruct direct analysis of $\mathcal{T}_\lambda$. Second, $\mathcal{S}(G_{\mathrm{id}}) = Y^\top H^\top W^\top W H Y$ exposes the weight Gram matrix $W^\top W$ explicitly, connecting the analysis of TL to the results of Section 3. The proof is given in Section C.4. We now use this surrogate to study how feature evolves during training and across layers.

4.2 Training & Layer-wise Dynamics

Training Dynamics We now analyze how $\mathcal{S}(G_{\mathrm{id}})$ evolves under GD. Throughout this subsection, we largely depend on a first-order Taylor expansion to characterize the leading-order behavior and see what this approach reveals about dominant portion of feature dynamics. We assume f is m-positively homogeneous in h: $f(\alpha h) = \alpha^m f(h)$ for any positive scalar α. We further assess the validity of this approach empirically in settings that go beyond these assumptions, including GELU activations in Section 4.3; further discussion and theoretical justification of our approach are in Section A.2.

Theorem 3. *For any loss function $\mathcal{L}$, let $G_{\mathrm{id}} = H^\top W^\top W H$ and $G_{\mathrm{id}}^+ = H^\top (W^+)^\top W^+ H$, where $W^+ = W - \gamma \nabla_W \mathcal{L}$. If f is 1-positively homogeneous in h, the following holds:*

$$\mathcal{S}(G_{\mathrm{id}}^+) - \mathcal{S}(G_{\mathrm{id}}) \approx 2\gamma\, (f^\top y) \cdot (y^\top K g),$$

[4] *where $K_{ij} = h_i^\top h_j$, $f_i = f(h_i)$ are the predictions on the training set, and $g = -\nabla_f \mathcal{L}$.*[5]

Proof Sketch. We rewrite $(W^+)^\top W^+ - W^\top W$ using the FL equation (Equation (1)), decompose $\nabla_h \mathcal{L} = -\nabla_h f \cdot g$, and resolve the cross terms $h_i^\top \nabla_{h_k} f$ via the m-homogeneity identity $h_i^\top \nabla_{h_k} f \approx f(h_i) + (m-1) f(h_k)$ (Lemma 3). The full proof is in Section C.5. □

The term $f^\top y$ captures the empirical correlation between predictions and labels. At initialization it is near zero, and as training proceeds it grows positive as f aligns with y.

To build intuition for $y^\top K g$, consider the binary classification setting. Under the binary cross-entropy loss with sigmoid predictions, $\hat{y} = \mathrm{sigmoid}(f(x)) \in (0,1)^n$, and labels $y \in \{0,1\}^n$, the negative gradient takes the form $g = y - \hat{y}$, so $g_i > 0$ when $y_i = 1$ and $g_i < 0$ when $y_i = 0$. Writing the quantity in feature space yields

$$y^\top K g = (Hy)^\top (Hg) = \langle \sum_{i \in S_1} h_i,\ \sum_{j \in S_1} (1 - \hat{y}_j)\, h_j - \sum_{k \in S_0} \hat{y}_k\, h_k \rangle,$$

where $S_a = \{i : y_i = a\}$. $y^\top K g$ measures the correlation, in feature space, between the positive-class centroid and a residual direction that pulls toward positive examples while pushing away from negative examples. Under feature learning regime where representations cluster by class, as $\langle h_i, h_j \rangle \geq \langle h_i, h_k \rangle$ holds for $i, j \in S_1, k \in S_0$. This makes the inner product positive whenever within-class similarity dominates cross-class similarity, before it tapers to zero as $\hat{y} \to y$.

A sharper interpretation of this gap comes from relating y to a classical statistical object. Let $f_{\mathrm{OLS}} = H^\top (H H^\top)^{-1} H y$ be the best linear predictor from current features. A direct calculation gives $\|y - f_{\mathrm{OLS}}\|_K = 0$, so y and f_{OLS} coincide under the K-norm. As training proceeds, f moves toward y and as a result $f \to f_{\mathrm{OLS}}$ in K-norm, while f_{OLS} itself moves toward y as TL increases.

This identifies feature learning as a *moving-target problem*. The network f_t chases $f_{\mathrm{OLS},t}$, but $f_{\mathrm{OLS},t+1}$ itself is reshaped as features H_{t+1} evolve: each gradient step closes the residual $\|f_{\mathrm{OLS},t} - f_t\|_{K_t}$ while simultaneously reopening it through feature updates that shift $f_{\mathrm{OLS},t+1}$ toward y. This contrasts sharply with the NTK regime [17], where, at infinite width, the kernel remains effectively fixed during training, and learning reduces to kernel regression in the corresponding RKHS [42]. In this regime, the solution is determined by the initial kernel, and training corresponds to fitting y within this static function space, without additional parametrization such as μP [51]. The lazy regime corresponds to this linearized setting in which the feature map remains effectively fixed, whereas in the feature learning regime the evolving representation H_t induces a sequence of changing targets $f_{\mathrm{OLS},t}$, so that optimizing f_t simultaneously reshapes the feature geometry itself (Figure 3). We empirically compare these two regimes in Section B.6.

Layer-wise Dynamics The role of depth in neural networks has been empirically validated in many practical architectures [15, 48], yet a unified theoretical account remains limited. Most existing work studies depth from an approximation perspective, showing that function composition is fundamentally more expressive than shallow expansion [45]. This line of work asks what functions are *learnable* with depth, rather than how a *learned* deep network processes input across its layers.

For the derivation we assume the Neural Feature Ansatz [40] holds. Boix-Adserà et al. [6] discuss why the NFA usually holds and the degenerate cases under which it does not. Under the NFA, we show that an interpolating (i.e., sufficiently trained) network *sequentially linearizes* input data toward the target at the final linear readout.

Theorem 4. *Let $f_l : h_l \mapsto f(x)$ denote the subnetwork mapping the l-th hidden state to the final output, and assume the activation σ is m-positively homogeneous. Assume the network interpolates*

[4] Up to this point, $\approx$ denotes equality up to $O(\gamma^2)$ terms. From here on, it may additionally absorb the error of first-order Taylor approximations (Lemma 3) and of the Neural Feature Ansatz (Theorem 4), so $\approx$ in the theorem statements can combine several approximations. We refer to the corresponding proofs for which approximation is used at each step.

[5] The same result extends to general m-positively homogeneous case under the centered label assumption, $\sum_i y_i = 0$.

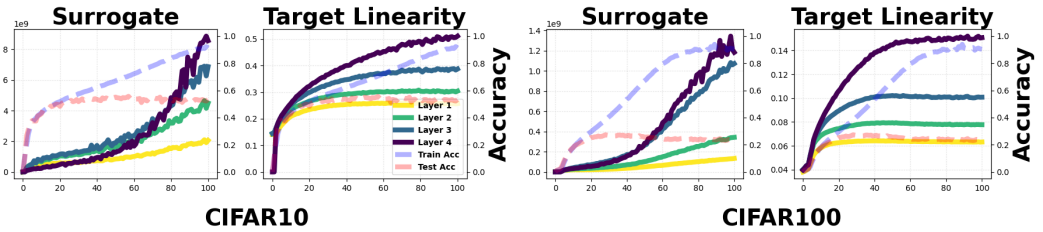

Figure 4: Training and layer-wise dynamics of the surrogate and Target Linearity for a 4-layer fully connected network trained on CIFAR-10 (left) and CIFAR-100 (right). Solid lines show layers 1–4 (yellow to dark purple); dashed lines show train and test accuracy.

the training data and $\exists C_0 > 0$ *s.t.* $\|W_l H_l\|_F^2 \geq C_0, \forall l$. *Under the Neural Feature Ansatz* $W_l^\top W_l \approx \frac{C_l}{N} \sum_i \nabla_{h_{l,i}} f_l \nabla_{h_{l,i}} f_l^\top$ *for layer-dependent constants* $C_l > 0$, *the surrogate satisfies*

$$\mathcal{S}(G_{\mathrm{id}}^l) \begin{cases} \gtrsim C_0 \|Y\|_2^2 \, \Phi_l^{-1}, & \text{if } m > 1 \text{ and } \sum_i y_i = 0, \\ \gtrsim C_0 \|Y\|_2^4 \, \Psi_l^{-1}, & \text{if } m = 1 \text{ and } f_l \text{ is piecewise linear,} \end{cases}$$

where Φ_l^{-1} *and* Ψ_l^{-1} *are monotonically increasing in* l.

Proof Sketch. Using the NFA, we decompose the surrogate into a sum of terms of the form $h_{l,i}^\top \nabla_{h_{l,j}} f$. A first-order Taylor expansion and homogeneity then gives $\mathcal{S}(G_{\mathrm{id}}^l) \approx C_l \|Y\|_2^4$ for both cases. Similarly, the NFA lets us decompose $\|W_l H_l\|_F^2 = \mathrm{tr}(H_l^\top W_l^\top W_l H_l) \geq C_0$. Relating the two expressions shows that the lower bound of C_l is monotonically increasing in l. The full proof is in Section C.6. □

The two cases of Theorem 4 differ in how the layer-wise complexity is measured. In the smooth m-homogeneous case ($m > 1$), Φ_l is governed by the homogeneity gap $\propto m^{L-l}$. In the piecewise-linear case ($m = 1$), the lower bound Ψ_l is governed by the number of linear regions of f_l [34]. Both quantities measure the same thing from different angles: how far the subnetwork f_l remains from being a linear function of its input. An interpolating network must produce y from x end-to-end, and Φ_l (or Ψ_l) measures the nonlinearity f_l still has to supply to close the gap between h_l and y. At shallow layers, f_l does most of the work and the complexity is large; at deep layers, f_l approaches a linear map and the complexity shrinks. Depth is the mechanism by which the total required nonlinearity is distributed across layers, and the surrogate $\mathcal{S}(G_{\mathrm{id}}^l)$ — inversely proportional to Φ_l or Ψ_l — correspondingly grows with depth.

This formalizes *sequential linearization*: as l increases, the residual nonlinearity f_l must supply shrinks, and the network leaves a progressively more linear problem for subsequent layers. The picture matches the standard empirical observation underlying linear probes [2]: deeper layers admit higher-accuracy linear classifiers, because by the time data reaches them, much of the nonlinear work has already been done.

4.3 Empirical Validation

We train a four-layer fully connected network with hidden dimension 256 and GELU activations [16] on CIFAR-10 and CIFAR-100 [23], using SGD with learning rate 0.005 for 100 epochs. To extend the Target Linearity to the multiple target cases, we extend the surrogate to C-class classification by replacing Y with the one-hot label matrix $Y \in \mathbb{R}^{N \times C}$ and redefining the surrogate as $\mathcal{S}_{\mathrm{cls}}(G) = \mathrm{tr}(Y^\top G Y)$. Since $\mathrm{tr}(Y^\top G Y) = \sum_c y_c^\top G y_c$, the dynamics from Section 4.2 carry over directly.

Figure 4 reports the training and layer-wise dynamics of the surrogate $\mathcal{S}_{\mathrm{cls}}(G_{\mathrm{id}}^l)$ and TL ($\lambda = 0$) at each layer. Both quantities grow throughout training, consistent with the regime identified in Theorem 3, and both increase with layer index, consistent with the sequential linearization predicted by Theorem 4. The surrogate grows roughly exponentially while TL grows roughly logarithmically; this is consistent with Theorem 2, where the lower bound on TL involves the inverse of the surrogate. Analogous results for the Adam optimizer [20] are in Section B.7.

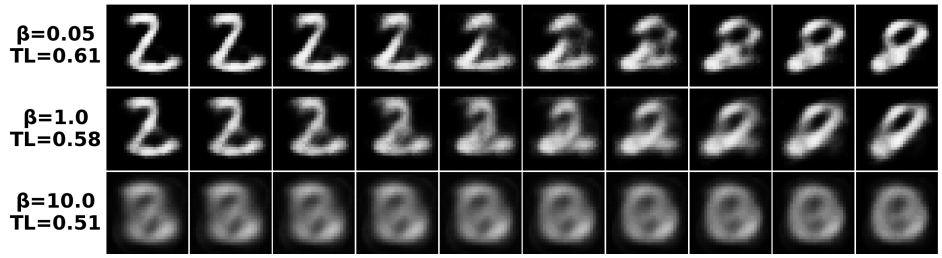

Figure 5: Latent-space interpolation between digits 2 and 8 in a VAE for different β values (the KL divergence term parameter). Lower β (higher TL) yields near-linear transitions in pixel space, while higher β leads to blurrier and less faithful interpolations.

A notable benefit of TL is that, by being defined as the R^2 of regression onto the target, it applies to any task with a continuous target — not just classification. We confirm this on self-supervised generation: TL grows during VAE training [21] when the target is the reconstructed input (Section B.8). This positions TL as a unifying measure of feature learning across discriminative and generative tasks, in contrast to phenomena like Neural Collapse [37] that are formulated specifically for classification.

5 Link to Other Feature Learning Phenomena

The framework developed in Section 4 is general — TL is defined for any task with a target, and its dynamics are governed by the weight Gram matrix. We now show that this generality recovers two well-known feature learning phenomena as special cases of our framework: Neural Collapse [37] as the maximum case of our surrogate in classification, and the linear interpolation property of generative models [43] as a consequence of high TL in latent space.

Neural Collapse as an Extreme Case In the classification setting, Papyan et al. [37] report the *Neural Collapse* (NC) phenomenon, in which the penultimate-layer features of trained networks (NC1) collapse to their class means with vanishing within-class variation, and (NC2) form a simplex equiangular tight frame. Existing theoretical accounts of NC rely largely on the Unconstrained Feature Model [33], which treats hidden states as freely trainable variables; recent work has begun to relax this [32] but only under separability assumptions.

Proposition 3. *Consider $f(x) = W^\top h(x) + b$ with hidden states centered ($\sum_i h(x_i) = 0$), W and b trained under squared loss on the centered one-hot label matrix, and $C \leq D$ for the last-layer hidden dimension D. Under NC1 and NC2, $\mathcal{S}_{\mathrm{cls}}(G_{\mathrm{id}})$ attains the maximum value allowed by a norm constraint $\|G_{\mathrm{id}}\|_F \leq c$ for some $c > 0$.*

This maximum is unique at the level of the Gram matrix: by the equality condition of the Cauchy–Schwarz inequality in Lemma 5, $\mathcal{S}_{\mathrm{cls}}(G_{\mathrm{id}})$ is maximized if and only if $G_{\mathrm{id}} = c\,YY^\top/\|YY^\top\|_F$, which is well defined since the centered one-hot label matrix satisfies $YY^\top \neq 0$ for $C \geq 2$.

Proposition 3 shows that NC corresponds to a maximally linearized representation in classification: the geometric structure observed empirically in trained networks [37] is precisely what saturates the surrogate $\mathcal{S}_{\mathrm{cls}}$. This recovers NC without invoking the Unconstrained Feature Model, instead deriving it from the same Gram-matrix framework that governs general feature learning. The full proof is in Section C.7. We note that Beaglehole et al. [4] independently show that target linearity (in the form of kernel linearization) controls the rate of NC formation in deep Recursive Feature Machines.

Linear Interpolation Property in Generative Models TL is defined as the R^2 of regression, so it applies to any task with a continuous target. We use this to address a phenomenon in generative models: Shao et al. [43] observed that decoders of trained Variational Autoencoders [21] approximately respect linear interpolation in latent space, $f(\alpha z_1 + (1-\alpha)z_2) \approx \alpha x_1 + (1-\alpha)x_2$, where f is the decoder, z_i are latent codes, and x_i are the corresponding inputs. This raises a natural question: when does a nonlinear decoder admit such linear interpolation?

Proposition 4. *Let $Z = [z_1, \ldots, z_N]^\top$ be latent vectors and $X = [x_1, \ldots, x_N]^\top$ the corresponding inputs. Let $g(z) = z^\top (Z^\top Z)^{-1} Z^\top X$ be the ordinary least-squares estimator, and assume $g \in \mathcal{F}$.*

Then for any $\alpha \in [0, 1]$*,*

$$\min_{f \in \mathcal{F}} \frac{1}{N^2} \sum_{i,j} \|f(\alpha z_i + (1-\alpha) z_j) - (\alpha x_i + (1-\alpha) x_j)\|_2 \;\leq\; \frac{1}{N} \sum_i \|g(z_i) - x_i\|_2.$$

The right-hand side is exactly the OLS reconstruction error, which equals square root of $1 - \mathcal{T}_0(Z, X)$ up to normalization. Proposition 4 therefore states that whenever the function class contains the OLS predictor and the latent code admits high TL with respect to the inputs, the decoder can interpolate linearly. Figure 5 empirically validates this finding in MNIST trained VAE (from class 2 to 8) and the full proof is in Section C.8.

6 Related Work & Discussion

Related Work Feature linearization has been observed empirically across several settings: [7] reported decreasing curvature of hidden representations with depth, and [2] showed linear probes attain higher accuracy at deeper layers. These observations are diagnostic; TL formalizes them as a quantity tied to the weight Gram matrix and explains, via the dynamics of Section 4.2, why such linearization emerges from training rather than treating it as a post-hoc finding.

More broadly, theoretical accounts of feature learning mainly fall into three streams: (1) *case studies* on structured targets [1, 3, 10, 11], whereas our results apply to general neural networks; (2) *metric studies* that propose descriptive measures of representation quality [8, 9, 22], which capture whether feature learning occurred but not how and why; and (3) phenomenon-driven studies that explain specific behaviors such as Neural Collapse [33, 37] or grokking [39, 28] in isolation. Relatedly, the lazy-versus-rich literature, spanning the NTK [17], mean-field analyses [30], and μP-style parameterizations [51, 13], mainly characterizes whether a network escapes the lazy regime by quantifying the degree of movement in weight space. Our framework is complementary: instead of measuring how far the weights move, it describes where the learned features move to, namely the progressive linearization toward the target captured by the weight Gram matrix. We aim to unify these threads by deriving feature dynamics directly from the Gram structure.

Closest to our analysis is the Neural Feature Ansatz [40], which conjectures $W_l^\top W_l \propto \mathbb{E}[\nabla_{h_l} f \cdot \nabla_{h_l} f^\top]$. Our VCS captures the Gram update dynamics directly (Theorem 1) without invoking the ansatz, tracking the actual update at $\rho = 0.98$ versus 0.71 for AGOP. A more comprehensive literature review is in Section A.1.

Target Linearity in Pre-trained Transformer We measure TL across every layer of BERT [12] on four downstream tasks comparing randomly initialized, pre-trained, and fine-tuned models in Section B.9. The goal is to check whether the dynamics theorized in Section 4.2 holds for complex transformers and track what TL illuminates on the pre-training and fine-tuning regime. We observe pre-training alone produces representations that are substantially more target linear with downstream labels than random initialization. Fine-tuning then converts this structure into a near-monotone TL profile that saturates within a single epoch and barely improves thereafter.

Link to Generalization From Figure 4, TL exhibits an interesting pattern when compared to generalization performance. In both CIFAR-10 and CIFAR-100, the training accuracy exceeds 0.9, while the test accuracy differs significantly (50% in CIFAR-10 and 20% in CIFAR-100). Meanwhile, the TL values differ substantially: the last-layer TL exceeds 0.5 for CIFAR-10 but only 0.15 for CIFAR-100. This is particularly interesting because TL is measured on the training set, where both models achieve high accuracy. To further investigate this phenomenon, we evaluate TL on a random-label dataset [52] and a modular arithmetic dataset which exhibit grokking [39] in Section B.10.

Acknowledgments and Disclosure of Funding

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-24873052 and RS-2025-25433902). We thank the anonymous reviewers for their constructive feedback, which substantially improved this paper. We also thank Seungsang Oh, Donghwan Kim, Dohyun Kwon, and Chulhee Yun for their thoughtful comments.

References

[1] E. Abbe, E. B. Adsera, and T. Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In Conference on Learning Theory, pages 4782–4887. PMLR, 2022.

[2] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. arXiv preprint arXiv:1610.01644, 2016.

[3] J. Ba, M. A. Erdogdu, T. Suzuki, Z. Wang, D. Wu, and G. Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. In Advances in Neural Information Processing Systems, 2022.

[4] D. Beaglehole, P. Súkeník, M. Mondelli, and M. Belkin. Average gradient outer product as a mechanism for deep neural collapse. Advances in Neural Information Processing Systems, 37: 130764–130796, 2024.

[5] D. Beaglehole, A. Radhakrishnan, E. Boix-Adsera, and M. Belkin. Toward universal steering and monitoring of ai models. Science, 391(6787):787–792, 2026.

[6] E. Boix-Adserà, N. R. Mallinar, J. B. Simon, and M. Belkin. FACT: a first-principles alternative to the neural feature ansatz for how networks learn representations. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=j4964wtJMz.

[7] P. P. Brahma, D. Wu, and Y. She. Why deep learning works: A manifold disentanglement perspective. IEEE transactions on neural networks and learning systems, 27(10):1997–2008, 2015.

[8] C.-N. Chou, H. Le, Y. Wang, and S. Chung. Feature learning beyond the lazy-rich dichotomy: Insights from representational geometry. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=gKdjHLrHDS.

[9] U. Cohen, S. Chung, D. D. Lee, and H. Sompolinsky. Separability and geometry of object manifolds in deep neural networks. Nature communications, 11(1):746, 2020.

[10] A. Damian, J. Lee, and M. Soltanolkotabi. Neural networks can learn representations with gradient descent. In Conference on Learning Theory, pages 5413–5452. PMLR, 2022.

[11] Y. Dandi, L. Pesce, L. Zdeborová, and F. Krzakala. The computational advantage of depth: Learning high-dimensional hierarchical functions with gradient descent. arXiv preprint arXiv:2502.13961, 2025.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pages 4171–4186, 2019.

[13] N. S. Dey, B. C. Zhang, L. Noci, M. Li, B. Bordelon, S. Bergsma, C. Pehlevan, B. Hanin, and J. Hestness. Don't be lazy: CompleteP enables compute-efficient deep transformers. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=lMU2kaMANl.

[14] S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro. Implicit bias of gradient descent on linear convolutional networks. Advances in neural information processing systems, 31, 2018.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[16] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.

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

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

[19] S. Karp, E. Winston, Y. Li, and A. Singh. Local signal adaptivity: Provable feature learning in neural networks beyond kernels. Advances in Neural Information Processing Systems, 34: 24883–24897, 2021.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[22] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In International conference on machine learning, pages 3519–3529. PMLR, 2019.

[23] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012.

[25] A. Kumar and J. Haupt. Early directional convergence in deep homogeneous neural networks for small initializations. arXiv preprint arXiv:2403.08121, 2024.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 2002.

[27] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, et al. Muon is scalable for llm training. arXiv preprint arXiv:2502.16982, 2025.

[28] Z. Liu, O. Kitouni, N. S. Nolte, E. Michaud, M. Tegmark, and M. Williams. Towards understanding grokking: An effective theory of representation learning. Advances in Neural Information Processing Systems, 35:34651–34663, 2022.

[29] K. Lyu and J. Li. Gradient descent maximizes the margin of homogeneous neural networks. In International Conference on Learning Representations, 2020. URL `https://openreview.net/forum?id=SJeLIgBKPS`.

[30] S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layer neural networks. Proceedings of the National Academy of Sciences, 115(33):E7665–E7671, 2018.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[32] H. Min, Z. Zhu, and R. Vidal. Neural collapse under gradient flow on shallow relu networks for orthogonally separable data. arXiv preprint arXiv:2510.21078, 2025.

[33] D. G. Mixon, H. Parshall, and J. Pi. Neural collapse with unconstrained features. Sampling Theory, Signal Processing, and Data Analysis, 20(2):11, 2022.

[34] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. Advances in neural information processing systems, 27, 2014.

[35] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, et al. Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, volume 2011, page 7. Granada, 2011.

[36] E. Nichani, A. Damian, and J. D. Lee. Provable guarantees for nonlinear feature learning in three-layer neural networks. Advances in Neural Information Processing Systems, 36:10828–10875, 2023.

[37] V. Papyan, X. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. Proceedings of the National Academy of Sciences, 117(40): 24652–24663, 2020.

[38] S. Parkinson, G. Ongie, and R. Willett. Relu neural networks with linear layers are biased towards single-and multi-index models. SIAM Journal on Mathematics of Data Science, 7(3): 1021–1052, 2025.

[39] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. arXiv preprint arXiv:2201.02177, 2022.

[40] A. Radhakrishnan, D. Beaglehole, P. Pandit, and M. Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. Science, 383(6690): 1461–1467, 2024.

[41] A. Radhakrishnan, M. Belkin, and D. Drusvyatskiy. Linear recursive feature machines provably recover low-rank matrices. Proceedings of the National Academy of Sciences, 122(13): e2411325122, 2025.

[42] A. Rahimi and B. Recht. Random features for large-scale kernel machines. Advances in neural information processing systems, 20, 2007.

[43] H. Shao, A. Kumar, and P. Thomas Fletcher. The riemannian geometry of deep generative models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 315–323, 2018.

[44] J. Sirignano and K. Spiliopoulos. Mean field analysis of neural networks: A law of large numbers. SIAM Journal on Applied Mathematics, 80(2):725–752, 2020.

[45] M. Telgarsky. Benefits of depth in neural networks. In Conference on learning theory, pages 1517–1539. PMLR, 2016.

[46] I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. In Proceedings of the 57th annual meeting of the association for computational linguistics, pages 4593–4601, 2019.

[47] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pages 142–147, 2003. URL https://www.aclweb.org/anthology/W03-0419.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[49] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

[50] A. R. Webb and D. Lowe. The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis. Neural Networks, 3(4):367–375, 1990.

[51] G. Yang, E. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tuning large neural networks via zero-shot hyperparameter transfer. Advances in Neural Information Processing Systems, 34:17084–17097, 2021.

[52] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In International Conference on Learning Representations, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

[53] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. arXiv preprint arXiv:2310.01405, 2023.

A Related Works & Limitations

A.1 Comprehensive Related Works

As feature learning is one of the distinguishing advantages of neural networks over classical machine learning models (such as kernel machines), an emerging body of work has focused on characterizing both theoretical and empirical aspects of feature learning. From the theoretical side, the study of feature learning in neural networks has largely been focused on fully connected networks used for multi-index modeling [1, 3, 10, 11]. Specifically, multi-index models are functions $f^* : \mathbb{R}^d \to \mathbb{R}$ of the form $f^*(x) = g(U^T x)$ where $U \in \mathbb{R}^{d \times r}$ satisfies $r << d$. It is known in such settings that neural networks outperform kernel machines by estimating the subspace U through feature learning (see e.g., [10] for an example of such an argument). Select work [19] has also theoretically analyzed feature learning in more general neural network architectures such as convolutional networks. Additional work has studied feature learning in deep fully connected networks, identifying problem settings for which feature learning in 2-hidden layer networks leads to improved sample complexity over feature learning in 1-hidden layer networks [36].

As a step toward more broadly characterizing the mechanism of feature learning in neural networks, the work of [40] posited the Neural Feature Ansatz (NFA) stating that Gram matrices of weights in layer ℓ of a neural network are proportional to the Average Gradient Outer Product (AGOP) of the network (where gradients are computed with respect into layer ℓ). This work presented empirical evidence for this claim across fully connected networks, convolutional networks, recurrent networks and transformers. Follow up work [41] proved that the claim of [40] was true in deep linear networks used for low rank matrix completion (a prominent setting for studying feature learning [14]). Additional follow up work [6] showed that the NFA did not hold with exact proportionality for all data generating processes and instead connected Gram matrices with an alternative object (based on gradient of the loss and activations). Related theoretical work [38] also connected the Expected Gradient Outer Product (EGOP) with the implicit bias of fully connected networks with a single nonlinearity. In particular, this work showed that the representation cost of such networks was bounded by Schatten-norms of the EGOP (implying that networks with minimum representation cost also had low rank EGOPs).

Lastly, linearization of features is a prominent observation in neural networks that has been studied in several settings [2, 7, 31]. Recently, there has been renewed interest in understanding the linearization of features in transformer-based language models, where linear representations of concepts can be used to monitor and steer such models [5, 53].

A.2 Limitations

On First-Order Approximations Our analysis of Target Linearity dynamics relies on first-order Taylor expansions, especially the homogeneity-based decompositions (Lemma 3). This is a deliberate trade-off. Existing dynamics analyses [1, 3, 10] achieve precise descriptions at the cost of restrictive assumptions on architecture or data. We aim for the opposite end of this trade-off: macroscopic dynamics under minimal technical assumptions, capturing leading-order behavior rather than exact trajectories.

While first-order approximations can be loose for individual terms given the highly nonlinear nature of deep networks, we conjecture that the aggregate quantities we study — particularly the surrogate $\mathcal{S}(G_{\mathrm{id}})$, which sums over all training samples — are more robust than any individual term within them, in line with the concentration patterns observed in mean-field analyses [30, 44]. We view this approximation-based approach as complementary to exact analyses: trading precision for generality may surface phenomena, such as the sequential linearization of Section 4.2, that restricted exact analyses would not naturally reveal. The following proposition partially justifies this approach. Although the proposition assumes exact homogeneity, we empirically find that the aggregate cancellation persists for non-homogeneous GELU networks, with an average error below 10^{-6} at every layer.

Proposition 5. *If a function f is m-positively homogeneous and input data and function values are centered, i.e., $\bar{x} = \frac{1}{N}\sum_i x_i = 0$ and $\sum_i f(x_i) = 0$, then sum of all pairs of first-order Taylor approximation errors among each data points is zero.*

Proof. By writing the sum of first-order Taylor approximation errors,

$$\begin{aligned}
\sum_{ij}\left[f(x_i)-f(x_j)-(x_i-x_j)^\top\nabla_{x_j}f\right] &= \sum_{ij}f(x_i)-\sum_{ij}f(x_j)-\sum_{ij}(x_i-x_j)^\top\nabla_{x_j}f \\
&= -\sum_{ij}(x_i-x_j)^\top\nabla_{x_j}f \\
&= -N\bar{x}^\top\left(\sum_j\nabla_{x_j}f\right)+N\sum_j x_j^\top\nabla_{x_j}f \\
&= N\sum_j x_j^\top\nabla_{x_j}f \quad \text{as } \bar{x}=\sum_i x_i=0 \\
&= mN\sum_j f(x_j) \quad \text{as } f \text{ is homogeneous} \\
&= 0 \quad \text{as } \sum_i f(x_i)=0
\end{aligned}$$

□

On Target Linearity and Generalization Since TL is measured on the training set, high TL does not by itself imply good generalization. In our random-label memorization experiment in Section B.10, TL keeps increasing during training even when all labels are randomly shuffled ($p = 1$), while the test accuracy remains near chance level. This shows that a network can linearize memorized, non-generalizable labels, so TL and generalization can diverge. On the other hand, we observe that the layer-wise TL profile appears more informative about generalization than the final-layer value alone: under random labels, the middle-layer TL exhibits a sudden increase for large p, and under grokking, the TL gap between intermediate and last layers closely tracks the generalization error. However, we have not established this relationship theoretically or validated it beyond these settings, and we leave a principled characterization of how layer-wise TL profiles relate to generalization as future work.

On Technical Assumptions Our analysis mainly adopts three assumptions that are standard in the deep learning theory literature: (i) m-positive homogeneity [18, 25, 29], satisfied exactly by bias-free ReLU networks; and (ii) lower-bounded pre-activation norm, which is automatically satisfied unless $WH = 0$ (a degenerate collapse). Additionally, (iii) the recent Neural Feature Ansatz [40] is adopted in dynamic analysis. NFA has been established theoretically in low-rank matrix completion, a widely studied setting for feature learning [41], and has also been validated empirically across a broad range of architectures and datasets. Additionally, Boix-Adserà et al. [6] discuss why the NFA usually holds and the degenerate cases under which it does not. Although its theoretical foundations are not yet fully understood, our analysis relies on extensive empirical validation in the original work.

B Experiment Details & Additional Experiments

We use a machine with AMD Ryzen 9 5900X 12-Core Processor CPU with one NVIDIA RTX 3090 GPU. Each experiment takes less than an hour, and the time can vary depending on the worker's settings. We attach our code for full reproduction.

B.1 Virtual Update on the Swiss Roll Dataset

Figure 1 is intended to give intuition for feature linearization before the formal framework is introduced, and we describe its experimental setup here. We generate 1,000 points from the Swiss roll dataset using `scikit-learn`, keep the two coordinates that span the spiral, and min-max normalize each coordinate to $[0, 1]$. The target y is the position of each point along the spiral, also min-max normalized to $[0, 1]$, which is shown by color in the figure. We train a fully connected network with two hidden layers of width 32 and GELU activations, where the first layer has no bias so that its weight matrix alone determines the Gram matrix $W^\top W$. The network is trained with the squared

loss using full-batch gradient descent with a learning rate of 0.05, so that each epoch corresponds to a single gradient step.

At each step t, we record the input gradient $\nabla_X \mathcal{L}_t$ under the current parameters θ_t, and compute the virtual update by accumulating these gradients, $\tilde{X}_t = X - \gamma \sum_{s \le t} \nabla_X \mathcal{L}_s$. Since the input gradients are small, we use $\gamma = 700$ for visualization and min-max normalize $\tilde{X}_t$ to $[0, 1]$ in each coordinate before plotting. Note that the virtual update is only tracked and never fed back into training, so it does not affect the actual optimization trajectory.

The red dashed contours show the density of a Gaussian centered at the mean of X with covariance $W_0^\top W_0 + 10^{-3} I$, where W_0 is the first-layer weight matrix at the corresponding epoch and the small ridge term ensures positive definiteness. We plot the original inputs, the virtually updated inputs, and the contours at epochs 0, 10, 20, 30, 40, and 50. As training proceeds, the virtual update unrolls the spiral into a near-linear curve ordered by the target, while the contours of $W_0^\top W_0$ reshape accordingly, consistent with Theorem 1.

B.2 Gradient Whitening Experiment

We solve the following optimization problem:

$$\arg_{W^+} \min \|W^+ - W\|_F \quad \text{subject to} \quad (W^+)^\top W^+ = (W - \gamma G)^\top (W - \gamma G),$$

where $G = \nabla_W \mathcal{L}$. By writing $W^+ = W - \gamma \Delta$, we can obtain the following problem:

$$\begin{aligned} \arg_{\Delta} \min \|\Delta\|_F \quad \text{subject to} \quad & (W - \gamma\Delta)^\top (W - \gamma\Delta) = (W - \gamma G)^\top (W - \gamma G) \\ & \Leftrightarrow \Delta^\top W + W^\top \Delta \approx G^\top W + W^\top G \end{aligned}$$

which is approximately equivalent to the original problem:

$$\begin{aligned} \arg_{\Delta'} \min \|G - \Delta'\|_F \quad \text{subject to} \quad & \Delta'^\top W + W^\top \Delta' \approx 0 \\ & \Leftrightarrow (W - \gamma\Delta')^\top (W - \gamma\Delta') \approx W^\top W, \end{aligned}$$

by letting $\Delta = G - \Delta'$.

Solution for the problem is well-known as an orthogonal procrustes problem:

$$W^+ = U V^\top \tilde{W^+},$$

for singular value decomposition $W(\tilde{W^+})^\top = U \Sigma V^\top$ and any $\tilde{W^+}$ satisfying $(W^+)^\top W^+ = (W - \gamma G)^\top (W - \gamma G)$. By simply setting $\tilde{W^+} = W - \gamma G$, we apply the Newton-Schulz method similar to Muon optimizer [27] to compute $UV^\top$.

B.3 Sensitivity of Gradient Whitening to Architecture and Hyperparameters

We further examine whether the gap between standard GD and the whitened update depends on the network size or on the optimization hyperparameters. We train fully connected networks on CIFAR-10 and CIFAR-100 while varying one factor at a time: the width $\in \{128, 256, 512\}$, the depth $\in \{2, 3, 4\}$, the learning rate $\in \{0.1, 0.05, 0.01, 0.005\}$, and the momentum $\in \{0.8, 0.6, 0.4, 0.2\}$. Table 1 reports the results for width and depth, and Table 2 reports the results for learning rate and momentum.

Width and depth. The gap between standard GD and the whitened update does not grow with network size. On CIFAR-10, the whitened update outperforms standard GD at every width and depth. On CIFAR-100, standard GD retains a small advantage, but the gap at epoch 20 shrinks as the width increases (3.56%p, 1.79%p, and 1.13%p for widths 128, 256, and 512) and stays within 2%p across depths.

Learning rate and momentum. The whitened update is notably more stable at high learning rates. On CIFAR-10, standard GD collapses to chance-level accuracy at learning rates of 0.1 and 0.05, whereas the whitened update maintains 44–47% accuracy. On CIFAR-100, the whitened update also outperforms standard GD at these high learning rates, while standard GD leads by about 1–2%p at

Table 1: Test accuracy (%) of standard GD (Orig.) and the whitened update (Whit.) for varying width and depth. We report mean ± standard deviation over 5 runs. The higher mean in each pair is in bold.

		Epoch 5		Epoch 10		Epoch 15		Epoch 20	
Dataset	Setting	Orig.	Whit.	Orig.	Whit.	Orig.	Whit.	Orig.	Whit.
CIFAR-10	Width 128	43.39 ± 1.69	**46.86** ± 0.36	45.45 ± 1.72	**49.72** ± 0.73	46.02 ± 1.61	**50.02** ± 0.72	46.89 ± 0.49	**50.32** ± 0.80
	Width 256	44.41 ± 0.64	**49.62** ± 0.87	47.20 ± 0.28	**50.42** ± 0.76	47.94 ± 0.63	**50.97** ± 0.70	47.29 ± 1.12	**50.93** ± 0.70
	Width 512	45.15 ± 1.12	**49.03** ± 0.40	48.21 ± 0.88	**51.20** ± 0.71	49.77 ± 0.66	**52.17** ± 0.36	49.91 ± 0.35	**52.42** ± 0.78
	Depth 2	44.41 ± 0.64	**49.62** ± 0.87	47.20 ± 0.28	**50.42** ± 0.76	47.94 ± 0.63	**50.97** ± 0.70	47.29 ± 1.12	**50.93** ± 0.70
	Depth 3	45.64 ± 0.73	**47.61** ± 1.32	49.26 ± 1.05	**51.08** ± 0.70	49.57 ± 0.73	**51.94** ± 0.44	50.19 ± 1.19	**51.53** ± 0.92
	Depth 4	45.71 ± 0.72	**46.67** ± 1.06	49.08 ± 0.64	**50.84** ± 0.49	50.77 ± 1.26	**51.32** ± 0.84	51.01 ± 0.44	**52.02** ± 1.02
CIFAR-100	Width 128	**20.58** ± 0.28	17.80 ± 0.20	**24.00** ± 0.26	20.76 ± 0.21	**25.40** ± 0.25	21.82 ± 0.48	**25.81** ± 0.23	22.25 ± 0.33
	Width 256	**19.77** ± 0.19	18.30 ± 0.24	**24.47** ± 0.34	22.27 ± 0.25	**26.56** ± 0.14	24.49 ± 0.14	**27.29** ± 0.33	25.50 ± 0.26
	Width 512	**22.03** ± 0.13	20.29 ± 0.35	**26.84** ± 0.11	24.80 ± 0.17	**28.46** ± 0.38	26.93 ± 0.30	**28.60** ± 0.22	27.47 ± 0.25
	Depth 2	**19.77** ± 0.19	18.30 ± 0.24	**24.47** ± 0.34	22.27 ± 0.25	**26.56** ± 0.14	24.49 ± 0.14	**27.29** ± 0.33	25.50 ± 0.26
	Depth 3	**19.12** ± 0.22	16.73 ± 0.20	**24.52** ± 0.11	22.11 ± 0.20	**26.53** ± 0.45	24.46 ± 0.14	**27.47** ± 0.16	25.48 ± 0.34
	Depth 4	**18.47** ± 0.28	16.48 ± 0.39	**24.52** ± 0.26	22.67 ± 0.09	**26.58** ± 0.35	25.02 ± 0.36	**26.08** ± 0.40	25.21 ± 0.45

Table 2: Test accuracy (%) of standard GD (Orig.) and the whitened update (Whit.) for varying learning rate and momentum. The higher value in each pair is in bold.

		Epoch 5		Epoch 10		Epoch 15		Epoch 20	
Dataset	Setting	Orig.	Whit.	Orig.	Whit.	Orig.	Whit.	Orig.	Whit.
CIFAR-10	LR 0.1	10.04	**44.06**	10.01	**45.22**	10.01	**45.39**	10.01	**44.56**
	LR 0.05	16.27	**46.41**	13.29	**46.95**	11.13	**47.29**	10.03	**46.63**
	LR 0.01	44.41	**49.62**	47.20	**50.42**	47.94	**50.97**	47.29	**50.93**
	LR 0.005	**48.29**	48.20	50.70	**51.13**	51.49	**51.95**	**52.89**	52.74
	Momentum 0.8	**45.41**	45.23	**51.14**	50.43	**51.88**	51.50	**53.55**	53.51
	Momentum 0.6	**42.13**	41.62	**48.41**	47.79	**48.88**	48.20	**51.70**	50.91
	Momentum 0.4	**39.12**	38.60	**46.13**	45.33	**45.76**	44.86	**49.40**	48.38
	Momentum 0.2	**36.72**	36.30	**43.65**	42.83	**43.18**	42.05	**47.00**	46.35
CIFAR-100	LR 0.1	14.81	**20.54**	15.09	**20.11**	15.37	**19.54**	15.31	**18.55**
	LR 0.05	22.18	**23.27**	21.78	**21.92**	20.94	**21.47**	20.24	**20.47**
	LR 0.01	**22.85**	21.62	**26.75**	25.05	**26.98**	25.63	**26.27**	24.87
	LR 0.005	**19.77**	18.30	**24.47**	22.27	**26.56**	24.49	**27.29**	25.50
	Momentum 0.8	**22.85**	21.62	**26.75**	25.05	**26.98**	25.63	**26.27**	24.87
	Momentum 0.6	**19.68**	18.32	**24.37**	22.31	**26.34**	24.47	**27.19**	25.43
	Momentum 0.4	**17.67**	16.37	**22.13**	20.00	**24.49**	22.59	**26.17**	24.19
	Momentum 0.2	**16.26**	14.70	**20.36**	18.51	**22.99**	20.83	**24.76**	22.71

lower learning rates. Across all momentum values, the whitened update stays within about 1%p of standard GD on CIFAR-10 and about 2%p on CIFAR-100.

Overall, these results are consistent with Section 2: the Gram-modifying component of the gradient recovers most of the performance of standard GD across architectures and hyperparameters, and it is more robust to large learning rates.

B.4 Gradient Whitening on Convolutional Neural Network

We present the gradient whitening results for a Convolutional Neural Network [24]. We train a model with 3 convolutional layers with 16 channels and 2-by-2 kernels, 2 max pooling layers, followed by a final linear classifier. We train the model using the SGD optimizer with a learning rate of 0.01 and momentum factor of 0.95 for 20 epochs. The result in Figure 6 shows negligible difference between original SGD and Gram-whitened SGD, similar to fully connected neural network case.

We further evaluate gradient whitening in a more complex training regime by training ResNet-18 [15] on TinyImageNet, which consists of 100,000/10,000/10,000 train/validation/test images across 200 classes. We train for 25 epochs using SGD with a learning rate of 0.1 and momentum of 0.9, and report the mean test accuracy over 3 runs in Table 3. Consistent with the CIFAR-100 results in Figure 2, standard GD and the whitened update follow similar trends, while the whitened update slightly underperforms.

We conjecture that this gap arises because the Gram matrix $W^\top W$ captures the structure of the input dimension of each layer but not its output dimension, so tasks with many classes, where the output dimension of the last layer is large, are comparatively affected. To test this, we additionally fine-tune

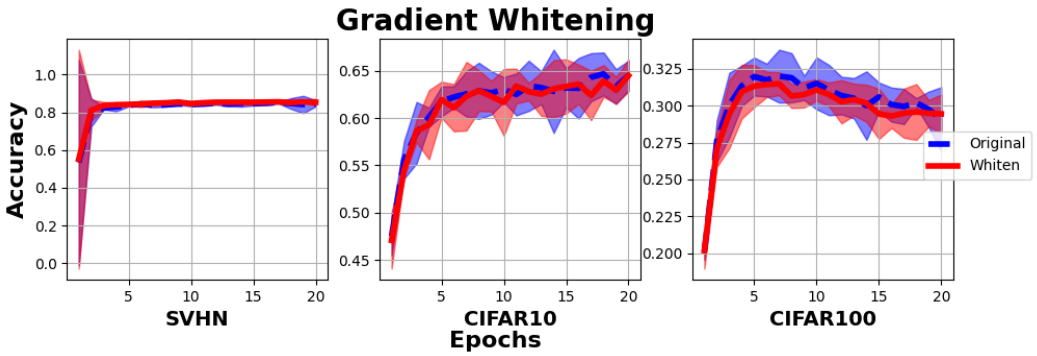

Figure 6: Performance comparison between gradient-whitened updates and standard gradient descent in convolutional neural network

only the last linear layer with standard SGD at the end of every epoch, while training all other layers with the whitened update. This reduces the gap to standard GD from 2.50%p to 0.71%p at epoch 25, and reaches 47.34% accuracy after only 5 epochs, substantially faster than standard GD. Although our goal is to show that the Gram matrix captures most of the training dynamics of GD, this result also suggests that Gram-based updates may lead to an efficient training algorithm, which we leave for future work.

Table 3: Test accuracy (%) of ResNet-18 on TinyImageNet for standard GD (Orig.), the whitened update (Whit.), and the whitened update with last-layer fine-tuning by standard SGD at the end of each epoch (Whit. + LL FT). We report the mean over 3 runs. The best value at each epoch is in bold.

Method	Epoch 5	Epoch 10	Epoch 15	Epoch 20	Epoch 25
Orig.	37.13	41.27	46.69	**49.21**	**49.23**
Whit.	36.46	38.45	46.24	46.72	46.73
Whit. + LL FT	**47.34**	**46.40**	**48.32**	48.53	48.52

B.5 Virtual Covariance vs. AGOP

In this section, we compare weight Gram ($W^\top W$) with AGOP ($\frac{1}{N}\sum_{i=1}^{N} \nabla_{h_i} f \cdot \nabla_{h_i} f^\top$) and the sum of our proposed VCS ($\sum_{t=1}^{\tau} \widetilde{\mathrm{cov}}(h_{l,t}^{+}) - \widetilde{\mathrm{cov}}(h_{l,t})$). We employ a two-layer neural network with a hidden dimension of 256 and train it on the CIFAR-10 dataset [23] using SGD with a learning rate of 0.05. We use the GELU function [16] as a σ. The model achieves 46% accuracy on the evaluation dataset. We then plot the reshaped diagonal components of $W^\top W$, AGOP, and the sum of the VCS in Figure 7. We also compute the Pearson correlation ρ between them.

We observe that both VC and AGOP exhibit a high correlation with the weight Gram. However, VC achieves a correlation of $\rho \approx 0.98$, which is substantially higher than that of AGOP ($\rho \approx 0.71$). This tendency is also evident in the visualizations: while both methods highlight the centered region, AGOP produces a relatively vague pattern, whereas VC closely resembles the weight Gram. Through this, we empirically verify that Theorem 1 holds, and the γ^2 factor does not significantly affect the resulting weight Gram.

B.6 Comparing Lazy vs. Rich Regimes

We train standard-parameterization MLPs (depth 3, ReLU, no bias) of widths $\{32, 128, 512, 1024\}$ on a synthetic staircase target $y = x_1x_2x_3x_4 + x_1x_2x_3 + x_1x_2 + x_1$ [1] with $n = 1000$ Gaussian inputs in $\mathbb{R}^{10}$. NTK Initialization [17] ($\mathrm{std} = 1/\sqrt{\text{fan-in}}$) and learning rate (0.01, full-batch SGD, 2000 epochs) are held fixed across widths, so width alone interpolates between rich and lazy regimes. Every 20 epochs we extract the penultimate features H and record $\|y - y_{\mathrm{OLS}}\|_2$, $\|\hat{y} - y_{\mathrm{OLS}}\|_2$, and $\|y - \hat{y}\|_2$, where $y_{\mathrm{OLS}} = H^\top(HH^\top + \epsilon I)^{-1}Hy$.

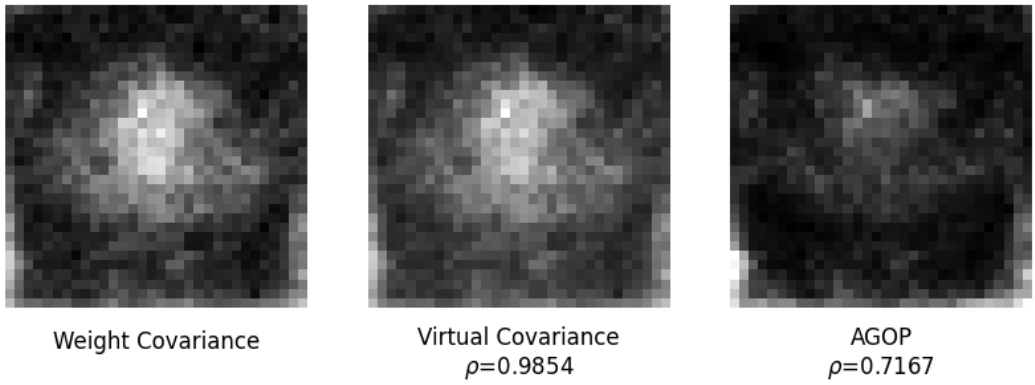

Figure 7: Visualization of the diagonal components of the weight Gram, AGOP, and our proposed Virtual Covariance Shift

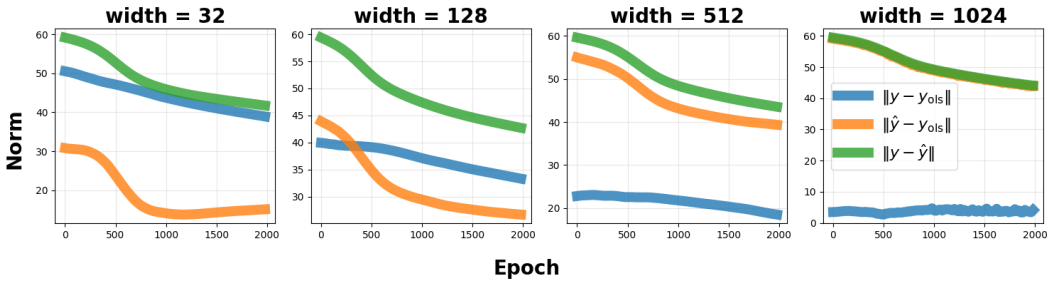

Figure 8: Training dynamics of MLPs of increasing width on a staircase target. In the rich regime (widths $32, 128$), y_{OLS} shifts substantially as features evolve, so the network chases a moving target. As width grows, features freeze: at width 1024, $\|y - y_{\text{OLS}}\|_2$ is flat near zero.

By the triangle inequality, the training loss decomposes as $\|\hat{y} - y\|_2 \leq \|\hat{y} - y_{\text{OLS}}\|_2 + \|y_{\text{OLS}} - y\|_2$, isolating two distinct sources of progress: how well the network fits the best linear predictor in its current feature space, and how well that feature space can linearly express y. Figure 8 shows that in the rich regime (widths $32, 128$), both terms decrease substantially: $\hat{y}$ chases y_{OLS} while y_{OLS} itself shifts toward y as features reorganize. As width grows, this gap flattens; at width 1024, $\|y - y_{\text{OLS}}\|_2$ stays near zero throughout training while $\|\hat{y} - y_{\text{OLS}}\|_2$ alone drives the decrease in $\|y - \hat{y}\|_2$, recovering the lazy / NTK behavior in which learning reduces to kernel regression in a fixed feature space. Consequently, the rich regime achieves a lower final $\|\hat{y} - y\|_2$ through successful feature learning, whereas the lazy regime is bottlenecked by the static expressivity of its initial features. This directly confirms the moving-target picture predicted in Section 4.2.

B.7 Target Linearity with ADAM Optimizer

We present the surrogate and TL values with ADAM [20] optimizer in Figure 9. Though our analysis mainly focused on the gradient descent optimizer, the results show a consistent pattern for various optimizers.

B.8 Target Linearity in Generative Model

We empirically measure how TL evolves in a self-supervised VAE [21] trained on an image generation task. Assuming Gaussianity of the latent vector, the (β-)VAE is trained with the following loss function:

$$\mathcal{L} = \underbrace{-E_{q(z|x)}[\log p(x|z)]}_{\text{reconstruction error}} + \underbrace{\beta D_{KL}(q(z|x)||p(z))}_{\text{regularization error}},$$

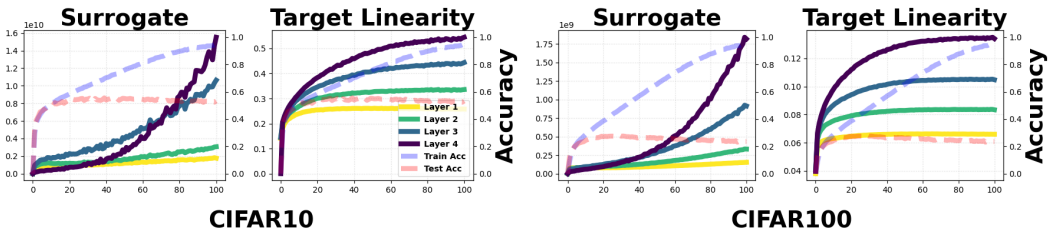

Figure 9: Surrogate and Target Linearity when trained with ADAM optimizer

where the regularization term controlled by β constrains the latent space to follow a Gaussian distribution.

We train the VAE on two image datasets: MNIST and CIFAR-10. We use a 2-layer fully-connected neural network with hidden dimension 512 for both the encoder and decoder, and set the latent dimension to 32. The VAE is trained with the ADAM optimizer [20] with a learning rate of 0.0001 for 10 epochs for MNIST and 30 epochs for CIFAR-10. During training, we measure the TL between latent vectors Z and the prediction target X. We visualize the TL along with the validation set reconstruction error for varying β from 0.05 to 10.0. The results are shown in Figure 10.

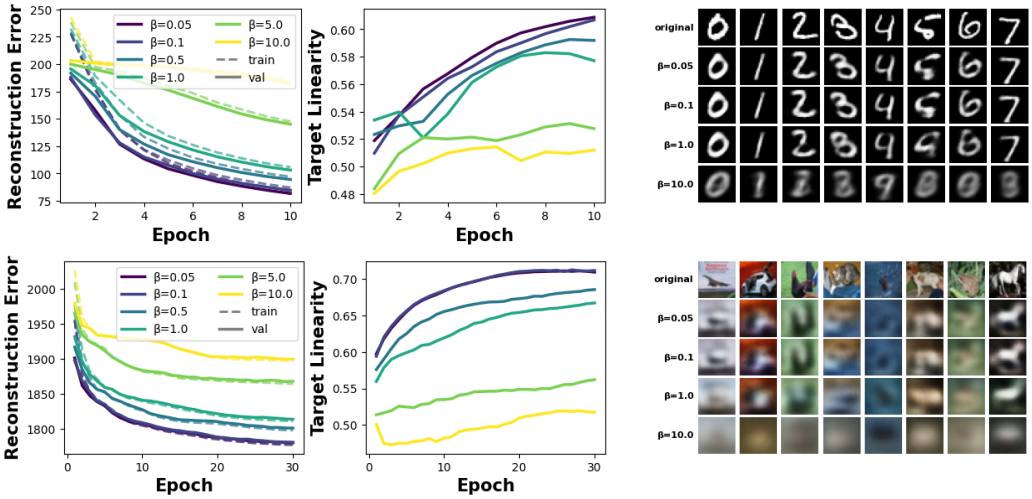

Figure 10: Reconstruction error, Target Linearity, and decoded output for VAE trained on MNIST (upper) and CIFAR-10 (lower)

The results demonstrate several key observations about the relationship between the regularization parameter β, reconstruction quality, and TL in VAE training.

First, we observe a clear trade-off between reconstruction error and TL across different β values. For both MNIST and CIFAR-10 datasets, lower β values (0.05, 0.1) achieve significantly lower reconstruction errors but also exhibit higher TL. Conversely, higher β values (5.0, 10.0) result in increased reconstruction error while maintaining lower TL. This trade-off is visually evident in the generated images: lower β values produce sharper, more detailed reconstructions that closely resemble the original images, whereas higher β values yield increasingly blurred and abstract representations.

Second, the evolution of TL during training shows distinct patterns depending on β. For low β values (0.05, 0.1), TL steadily increases throughout training, suggesting that the latent representations become progressively more linearly related to the input as the model prioritizes reconstruction accuracy. In contrast, moderate to high β values (1.0, 5.0, 10.0) maintain relatively stable or slightly increasing TL, indicating that stronger regularization constrains the latent space structure.

Finally, the generated samples reveal the practical implications of the β parameter. For MNIST, even with $\beta = 10.0$, the digits remain somewhat recognizable, albeit heavily smoothed. However, for CIFAR-10, high β values ($\beta = 10.0$) produce nearly uniform gray images with minimal structure, indicating that excessive regularization can completely collapse the model's ability to capture meaningful image features in complex domains.

B.9 Target Linearity in Pre-trained Transformers

We fine-tune BERT [12] on four tasks spanning two task families: sequence classification (SST-2 and MNLI datasets [49]) and token classification (CoNLL-2003 POS tagging and NER datasets [47]). All models are trained for 3 epochs with AdamW [20] at learning rate 2e-5. To compute target linearity at each stage, we sample 2,000 training examples and reduce hidden states to 256 dimensions via PCA. We report TL for five model states per task: randomly initialized (architecture only), pre-trained (no fine-tuning), and the three fine-tuned epoch checkpoints. The results are in Figure 11.

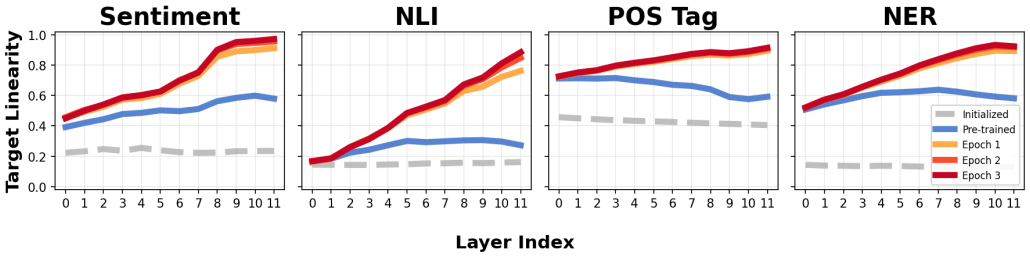

Figure 11: Layerwise Target Linearity across four tasks. TL at each layer of BERT for randomly initialized (gray dashed), pre-trained (blue), and fine-tuned (orange and red, epochs 1–3) models. Measured on SST-2 and MNLI, CONLL 2003 POS and NER datasets.

Across all four tasks, the pre-trained model's TL curve sits well above the randomly initialized baseline at every layer, despite the pre-training objective (MLM + NSP) never seeing any of these labels. This is the cleanest evidence in our experiments that pre-training does not merely produce a useful initialization for optimization but actually learns representations that are relatively linearly separable for downstream tasks. The size of the gap, however, depends sharply on how aligned the task is with the pre-training objective: it is largest for POS and NER (which depend on local lexical and syntactic regularities that MLM directly rewards), moderate for sentiment, and smallest for NLI — where the pre-trained TL barely separates from the random baseline. NLI requires cross-sentence reasoning that token-level reconstruction has little incentive to encode, so the linearly-extractable signal is correspondingly weaker.

Pre-training and fine-tuning produce qualitatively different layerwise profiles. The pre-trained curves recapitulate a pattern reported in earlier probing work [46]: TL for semantic tasks (sentiment) climbs monotonically toward the top of the network, while TL for syntactic tasks (POS, NER) peaks in the early-to-middle layers and slightly decays afterward. This is consistent with the view that BERT progressively assembles a classical NLP pipeline, with surface and syntactic information concentrated low and semantic information high. Fine-tuning, however, erases this division: after a single epoch, all four tasks exhibit the same near-monotone increasing TL profile, with the pre-trained curve serving as a lower envelope. Even POS — for which the pre-trained model's optimal layer is layer 3 — is reorganized so that the final layer carries the most target linear information. This implies the layerwise specialization seen during pre-training appears to be a property of the pre-training objective rather than an intrinsic property of the architecture.

B.10 Target Linearity and Generalization

Random Label Memorization From Figure 4, TL exhibits an interesting pattern when compared to generalization performance. In both CIFAR-10 and CIFAR-100, the training accuracy exceeds 0.9, while the test accuracy differs significantly. The model achieves over 50% test accuracy on CIFAR-10 but only over 20% on CIFAR-100, indicating a much larger generalization gap for CIFAR-100. The TL values also differ substantially: the last-layer TL exceeds 0.5 for CIFAR-10 but only 0.15 for CIFAR-100. This is particularly interesting because TL is measured on the training set, where both

models achieve high accuracy. These results suggest a potential relationship between generalization and TL, although we do not provide a theoretical analysis.

To further investigate this phenomenon, we measure TL on randomly labeled datasets. It is well known that neural networks can memorize randomly labeled data, resulting in high generalization error [52]. We report TL values and train/test accuracy for various values of $p \in [0, 1]$, where p denotes the fraction of data with randomly assigned labels. When $p = 0$, the labels are identical to the original dataset, whereas when $p = 1$, all labels are randomly shuffled. We train a 3-layer neural network with a hidden dimension of 256 on CIFAR-10 for 200 epochs to allow sufficient memorization. The model is optimized using stochastic gradient descent with a learning rate of 0.005. The results are presented in Figure 12.

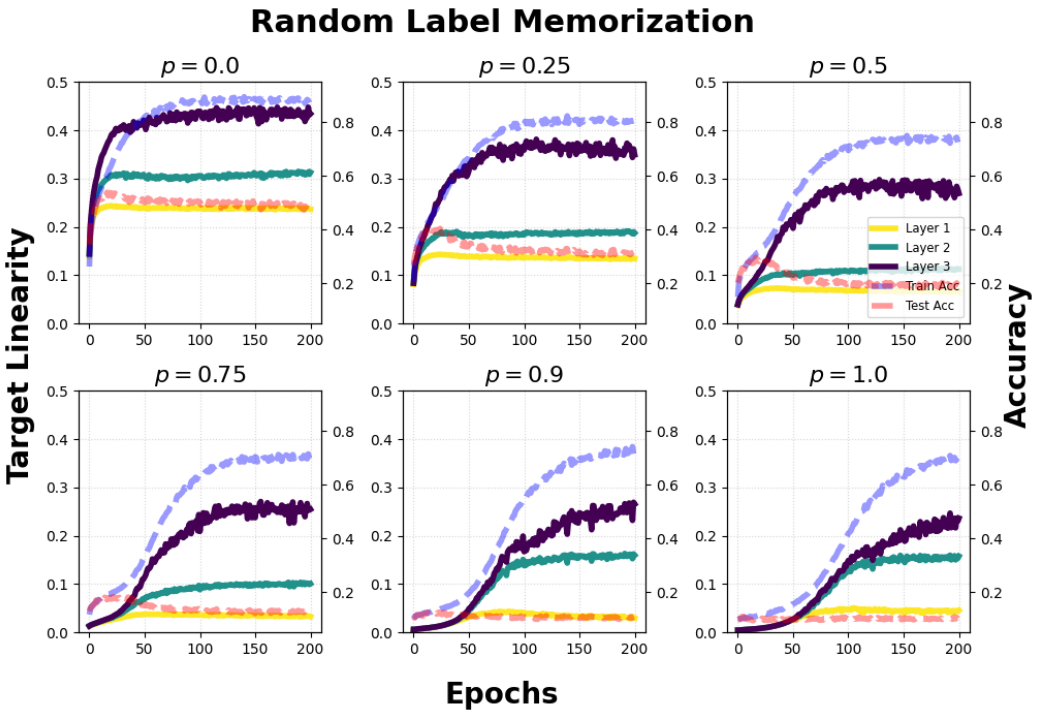

Figure 12: Target Linearity measured on randomly labeled CIFAR 10

Figure 12 shows that increasing the random label probability p leads to both a larger generalization error and a decrease in last-layer TL. As p increases, the overall training accuracy decreases up to $p = 0.5$, while it converges to approximately 0.7 for $p \geq 0.75$. In contrast, the test accuracy decreases more rapidly, from 0.5 ($p = 0.0$) to 0.3 ($p = 0.25$) and 0.2 ($p = 0.5$), resulting in an increasing generalization gap. Although the final training accuracy remains around 0.7 and the test accuracy around 0.1 for $p \geq 0.75$, higher values of p require more training time to reach high training accuracy. As a result, during the middle stages of training, the generalization gap is larger for higher p.

Interestingly, TL also decreases substantially: the last-layer TL drops from 0.45 ($p = 0.0$) to 0.3 ($p = 0.5$), while the training accuracy decreases from 0.85 to 0.75. TL in other layers also decreases over this range. For $p \geq 0.75$, the last-layer TL at the end of training remains around 0.25. However, similar to the slower increase in training accuracy, the growth of TL also slows for higher p. Overall, the alignment between decreasing TL and an increasing generalization gap provides further evidence that TL may serve as a measure of generalization.

More interestingly, the TL of the middle layers exhibits a sudden increase for $p \geq 0.75$, unlike the cases with $p \leq 0.75$. This may indicate the need for additional layers to fully memorize highly randomized labels. For $p \leq 0.75$, the model can achieve sufficiently low loss using modestly processed intermediate representations exploiting the last layer, resulting in rapid early increases. However, for $p \geq 0.75$, the model explores the parameter space more extensively, showing slow

early increases followed by sudden changes across multiple layers. This observation suggests that layer-wise TL patterns may reveal characteristics related to data complexity.

Grokking Another interesting generalization-related phenomenon is grokking [39]. Grokking refers to a phenomenon in which a neural network first memorizes the training data with poor generalization, and only after prolonged training suddenly transitions to a solution that generalizes well. This behavior was observed in algorithmic tasks such as modular arithmetic, where models trained with standard optimization initially achieve near-perfect training accuracy while test accuracy remains low, followed by an abrupt improvement in generalization after many additional training steps. The transition is often associated with implicit regularization effects of optimization, such as weight decay, which gradually favor simpler or more structured solutions over memorizing ones.

Here, we examine how TL evolves under various grokking settings. We primarily vary the weight decay parameter λ, which is known to control the timing of grokking. We use a modular addition dataset with a prime modulus of 61. A 3-layer neural network with a hidden dimension of 256 is trained using the Adam optimizer [20] with a learning rate of 0.001 for 500 epochs. By varying $\lambda \in (0, 1)$, we analyze how TL changes as grokking emerges. The results are presented in Figure 13.

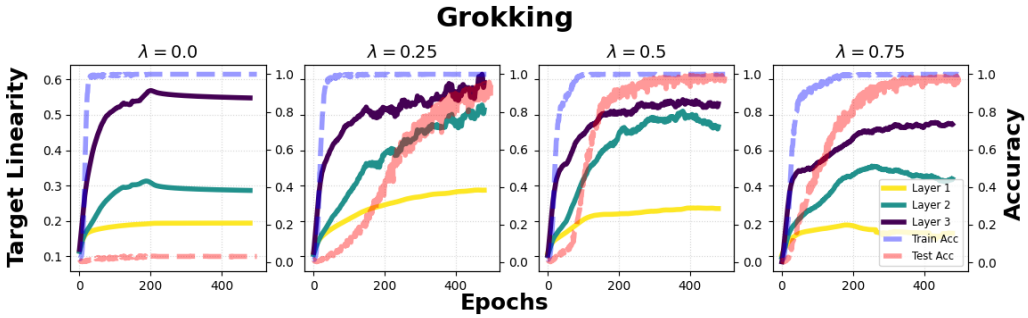

Figure 13: Target Linearity dynamics under Grokking setting

Figure 13 shows diverse patterns of overfitting and grokking across different values of the weight decay parameter λ. One notable observation is that, for $\lambda \geq 0.25$, TL continues to increase even after memorization, i.e., when the training accuracy is approximately 1.0. TL increases until grokking emerges and then saturates in the final stage of training. These results suggest a potential relationship between TL dynamics and generalization.

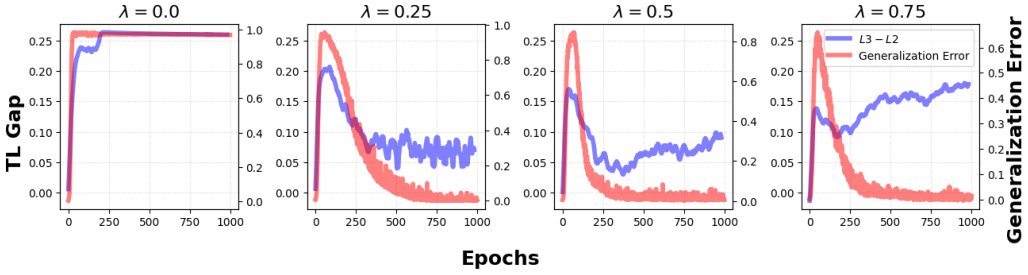

Figure 14: Comparing Target Linearity gap and generalization error

An interesting observation is that, as grokking emerges, the TL of the second layer (Layer 2) increases sharply. As a result, the gap between the second and last layers (Layer 3) narrows, although it widens again after generalization in the $\lambda = 0.75$ case. For a more direct comparison, we plot the TL gap between the second and last layers alongside the generalization error in Figure 14. Although the TL gap increases after generalization, particularly for $\lambda = 0.75$, it exhibits a similar decreasing pattern at the onset of grokking. These results suggest another potential use of layer-wise TL patterns as a measure of generalization.

C Proofs

C.1 Proof of Theorem 1 in Convolutional Neural Network

Without loss of generality, we study a single convolutional layer with nonlinear σ and a linear readout,

$$f(x) \;=\; a^\top \sigma(W * x),$$

where $x \in \mathbb{R}^{B\times C_{\text{in}}\times H\times W}$, $W \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times k_h\times k_w}$, and $a \in \mathbb{R}^{C_{\text{out}}}$ is a per-channel linear readout (applied at each spatial site). Let

$$D := C_{\text{in}}\, k_h\, k_w, \qquad N := B\, H'\, W'.$$

We use the standard im2col operator $\mathcal{U}$ that stacks receptive-field patches into columns:

$$h \;:=\; \mathcal{U}(x) \in \mathbb{R}^{D\times N}, \quad \widetilde{W} \in \mathbb{R}^{C_{\text{out}}\times D} \text{ (flattened kernels).}$$

Define pre-activations in patch space:

$$z \;=\; \widetilde{W}\, h \in \mathbb{R}^{C_{\text{out}}\times N}.$$

By the chain rule,

$$\nabla_{\widetilde{W}}\mathcal{L} \;=\; \nabla_z\mathcal{L}\, h^\top, \qquad \nabla_h\mathcal{L} \;=\; \widetilde{W}^\top\nabla_z\mathcal{L}.$$

So we obtain FL equation (Equation (1))

$$\widetilde{W}^\top\, \nabla_{\widetilde{W}}\mathcal{L} \;=\; \nabla_h\mathcal{L}\cdot h^\top,$$

which leads to the CNN version of Theorem 1.

C.2 Proof for Proposition 1

Statement 1. *Assume σ is L-Lipschitz and $\|h_{l-1}\|_2 = 1$, and let W^+_{l-1} denote the layer $l-1$ weight matrix after one gradient descent step on the single data point producing h_{l-1}, with all other layers held fixed. Then $|\sigma(W^+_{l-1}h_{l-1}) - \sigma(W_{l-1}h_{l-1})| \le L^2|h^+_l - h_l|$ element-wise. If σ is additionally monotonically increasing, then $\operatorname{sgn}(\sigma(W^+_{l-1}h_{l-1}) - \sigma(W_{l-1}h_{l-1})) = \operatorname{sgn}(h^+_l - h_l)$; the sign equality also holds for non-decreasing σ such as ReLU under the convention $\operatorname{sgn}(0) \in \{-1, +1\}$.*

Proof. Define the pre-activation hidden state $z_l = W_{l-1}h_{l-1}$ and its actual update $z^+_l = W^+_{l-1}h_{l-1}$. Then we can observe

$$\begin{aligned}
z^+_l &= W^+_{l-1}h_{l-1} \\
&= W_{l-1}h_{l-1} - \gamma(\nabla_{W_{l-1}}\mathcal{L})h_{l-1} \\
&= z_l - \gamma\nabla_{z_l}\mathcal{L}\langle h_{l-1}, h_{l-1}\rangle \quad \text{as } \nabla_{W_{l-1}}\mathcal{L} = \nabla_{z_l}\mathcal{L}\cdot h^\top_{l-1} \\
&= z_l - \gamma\nabla_{z_l}\mathcal{L} \quad \text{as } \|h_{l-1}\| = 1.
\end{aligned}$$

By the L-Lipschitz property,

$$\begin{aligned}
|\sigma(W^+_{l-1}h_{l-1}) - \sigma(W_{l-1}h_{l-1})| &\le L|W^+_{l-1}h_{l-1} - W_{l-1}h_{l-1}| \\
&= L|z^+_l - z_l| \\
&= L|\gamma\nabla_{z_l}\mathcal{L}| \\
&= L|\gamma\sigma'(z_l)\odot\nabla_{h_l}\mathcal{L}| \quad \text{as } \nabla_{z_l}\mathcal{L} = \sigma'(z_l)\odot\nabla_{h_l}\mathcal{L} \\
&\le L^2|\gamma\nabla_{h_l}\mathcal{L}| \quad \text{as } \sigma'(z_l)_i \le L, \forall i \\
&= L^2|h^+_l - h_l|
\end{aligned}$$

For the sign case, as σ is a non-decreasing function,

$$\begin{aligned}
\operatorname{sgn}(\sigma(W^+_{l-1}h_{l-1}) - \sigma(W_{l-1}h_{l-1})) &= \operatorname{sgn}(\sigma(z^+_l) - \sigma(z_l)) \\
&= \operatorname{sgn}(z^+_l - z_l) \quad \text{by non-decreasing property} \\
&= \operatorname{sgn}(-\nabla_{z_l}\mathcal{L}) \\
&= \operatorname{sgn}(-\sigma'(z_l)\odot\nabla_{h_l}\mathcal{L}) \\
&= \operatorname{sgn}(-\nabla_{h_l}\mathcal{L}) \quad \text{by non-decreasing property} \\
&= \operatorname{sgn}(h^+_l - h_l).
\end{aligned}$$

□

C.3 Proof for Proposition 2

Statement 2. *Consider a single input x and an L-layer network with $z_l = W_{l-1}h_{l-1}$ and $h_l = \sigma(z_l)$ for $l = 1, \dots, L$, where $h_0 = x$ and σ is differentiable. Suppose all weights evolve under gradient flow, $\dot{W}_l = -\nabla_{W_l}\mathcal{L}$ for $l = 0, \dots, L-1$. Then for every layer l,*

$$\frac{dh_l}{dt} = -M_l \nabla_{h_l}\mathcal{L}, \qquad M_l = \sum_{l'=1}^{l} \|h_{l'-1}\|_2^2 \, J_{l\leftarrow l'} J_{l\leftarrow l'}^\top \succeq 0,$$

where $J_{l\leftarrow l'} = \partial h_l / \partial z_{l'}$. Consequently, $\langle \frac{dh_l}{dt}, -\nabla_{h_l}\mathcal{L}\rangle \geq 0$.

Proof. For $1 \leq l' \leq l \leq L$, define the diagonal activation-derivative matrices and the Jacobians

$$D_l := \mathrm{diag}(\sigma'(z_l)), \qquad J_{l\leftarrow l'} := \frac{\partial h_l}{\partial z_{l'}} = D_l W_{l-1} D_{l-1} W_{l-2} \cdots W_{l'} D_{l'},$$

with the convention $J_{l\leftarrow l} = D_l$. Since h_l depends only on $W_0, \dots, W_{l-1}$, the chain rule gives

$$\frac{dh_l}{dt} = \sum_{l'=1}^{l} \partial_{W_{l'-1}} h_l[\dot{W}_{l'-1}],$$

where $\partial_W h_l[A]$ denotes the directional derivative of h_l with respect to W in direction A. Since $W_{l'-1}$ enters h_l only through $z_{l'} = W_{l'-1}h_{l'-1}$, and $h_{l'-1}$ does not depend on $W_{l'-1}$,

$$\partial_{W_{l'-1}} h_l[A] = \frac{\partial h_l}{\partial z_{l'}}(A h_{l'-1}) = J_{l\leftarrow l'} A h_{l'-1}.$$

Inserting $A = \dot{W}_{l'-1} = -\nabla_{W_{l'-1}}\mathcal{L} = -\nabla_{z_{l'}}\mathcal{L}\, h_{l'-1}^\top$ yields

$$\frac{dh_l}{dt} = -\sum_{l'=1}^{l} J_{l\leftarrow l'} \nabla_{z_{l'}}\mathcal{L}\, h_{l'-1}^\top h_{l'-1} = -\sum_{l'=1}^{l} \|h_{l'-1}\|_2^2 \, J_{l\leftarrow l'} \nabla_{z_{l'}}\mathcal{L}.$$

Since every path from $z_{l'}$ to the loss passes through h_l for $l' \leq l$, we have $\nabla_{z_{l'}}\mathcal{L} = J_{l\leftarrow l'}^\top \nabla_{h_l}\mathcal{L}$, and therefore

$$\frac{dh_l}{dt} = -\Big(\sum_{l'=1}^{l} \|h_{l'-1}\|_2^2 \, J_{l\leftarrow l'} J_{l\leftarrow l'}^\top\Big)\nabla_{h_l}\mathcal{L} = -M_l \nabla_{h_l}\mathcal{L}.$$

Each summand of M_l is PSD, so $M_l \succeq 0$ and $\langle \frac{dh_l}{dt}, -\nabla_{h_l}\mathcal{L}\rangle = \nabla_{h_l}\mathcal{L}^\top M_l \nabla_{h_l}\mathcal{L} \geq 0$. □

C.4 Proof for Theorem 2

Statement 3. *Let $e_0 = \|Y - \bar{Y}\|_2^2$ and $e_1 = \|Y\|_2^2$, and assume $\|W\|_{\mathrm{op}} \leq c_0$, $\|H\|_F \leq c_1$, and $N > d$. Then*

$$\mathcal{T}_\lambda(H, Y) \geq 1 - C \cdot \mathcal{S}(G_{\mathrm{id}})^{-1},$$

for some fixed constant C.

If W, H, G are l-layer weight, hidden state, and Gram, define W', H', G' as $l-1$-layer quantities. We first show the relationship between $\mathcal{S}(G_{\mathrm{id}})$ and $\mathcal{S}(G'_\sigma)$.

Lemma 1. *With a regularity condition on weight matrix $\|W\|_{op} < c$, $\mathcal{S}(G'_\sigma) > c^{-2} \cdot \mathcal{S}(G_{id})$.*

Proof for Lemma 1. We can show

$$\mathcal{S}(G_{\mathrm{id}}) = Y^\top H^\top W^\top W H Y < c^2 \cdot Y^\top H^\top H Y = c^2 \cdot Y^\top \sigma(W'H')^\top \sigma(W'H')Y = c^2 \cdot \mathcal{S}(G'_\sigma),$$

where the inequality holds by the definition of the matrix operator norm. □

Lemma 1 shows that increasing the l-th layer linear $\mathcal{S}(G_{\mathrm{id}})$ increases the lower bound of $l-1$-th layer non-linear $\mathcal{S}(G'_\sigma)$. This arguments holds for all $l = 2, ..., L$ except the first 1-st layer where the hidden state H is equal to the fixed input data X.

Since we showed the relationship between two surrogates, we now show how $\mathcal{S}(G_\sigma)$ is related to our target quantity $\mathcal{E}_\lambda(G_\sigma)$.

Lemma 2. *For a symmetric positive (semi-) definite matrix $G \in \mathbb{R}^{N\times N}$, vector $Y \in \mathbb{R}^N$, and a positive constant $\lambda > 0$, the following holds:*

$$Y^\top(\lambda \cdot I + G)^{-2}Y \le \kappa \cdot \|Y\|_2^4 \cdot \left[Y^\top(\lambda \cdot I + G)Y\right]^{-1}$$

where $\kappa = \frac{(2\lambda+\lambda_{\max}+\lambda_{\min})^2}{4(\lambda+\lambda_{\min})^3}$, and $\lambda_{\max}, \lambda_{\min}$ are maximum and minimum eigenvalue of G.

Proof for Lemma 2. By spectral decomposition, $G = Q\Lambda Q^\top$, $Y^\top[\lambda I + G]Y = \sum_i(\lambda+\lambda_i)\cdot\langle q_i, Y\rangle^2$. Similarly, $Y^\top[\lambda I + G]^{-a}Y = \sum_i(\lambda+\lambda_i)^{-a}\cdot\langle q_i, Y\rangle^2$ for integer a.

By the Kantorovich inequality,

$$\left(Y^\top[\lambda\cdot I + G]Y\right)\left(Y^\top[\lambda\cdot I + G]^{-1}Y\right) \le \|Y\|_2^4 \frac{(2\lambda+\lambda_{\max}+\lambda_{\min})^2}{4(\lambda+\lambda_{\max})\cdot(\lambda+\lambda_{\min})}$$

and the bounds

$$\begin{aligned}
&(\lambda+\lambda_{\max})^{-1}\cdot\sum_i\langle q_i, Y\rangle^2 \le Y^\top[\lambda\cdot I + G]^{-1}Y \quad \text{and}\\
&Y^\top[\lambda\cdot I + G]^{-2}Y \le (\lambda+\lambda_{\min})^{-2}\cdot\sum_i\langle q_i, Y\rangle^2\\
&\Rightarrow Y^\top[\lambda\cdot I + G]^{-2}Y \le (\lambda+\lambda_{\max})^{-1}\cdot\sum_i\langle q_i, Y\rangle^2\cdot\frac{\lambda+\lambda_{\max}}{(\lambda+\lambda_{\min})^2}\\
&\qquad\qquad \le Y^\top[\lambda\cdot I + G]^{-1}Y\cdot\frac{\lambda+\lambda_{\max}}{(\lambda+\lambda_{\min})^2}.
\end{aligned}$$

So we can conclude

$$\begin{aligned}
Y^\top[\lambda\cdot I + G]^{-2}Y &\le \kappa\cdot\|Y\|_2^4\cdot\left(Y^\top[\lambda\cdot I + G]Y\right)^{-1},\\
\text{where } \kappa &= \frac{(2\lambda+\lambda_{\max}+\lambda_{\min})^2}{4(\lambda+\lambda_{\max})\cdot(\lambda+\lambda_{\min})}\cdot\frac{\lambda+\lambda_{\max}}{(\lambda+\lambda_{\min})^2}\\
&= \frac{(2\lambda+\lambda_{\max}+\lambda_{\min})^2}{4(\lambda+\lambda_{\min})^3}.
\end{aligned}$$

□

Since $\mathcal{S}(G_\sigma) = Y^\top G_\sigma Y \propto Y^\top(\lambda\cdot I + G_\sigma)Y$ and $\mathcal{E}_\lambda(G_\sigma) = Y^\top(\lambda I + G_\sigma)^{-2}Y$, we can conclude that increasing $\mathcal{S}(G_\sigma)$ decreases the upper bound of $\mathcal{E}_\lambda(G_\sigma)$.

Combining these lemmas, we obtain the resulting proof.

Proof. Since $\mathcal{T}_\lambda(H, Y) = 1 - \frac{\|Y-\hat{Y}_\lambda\|_2^2}{\|Y-\bar{Y}\|_2^2} = 1 - \frac{\lambda^2\mathcal{E}_\lambda(G'_\sigma)}{e_0}$, where

$$\begin{aligned}
\mathcal{E}_\lambda(G'_\sigma) &\le \kappa_0\cdot\|Y\|_2^4\cdot\left[Y^\top(\lambda\cdot I + G'_\sigma)Y\right]^{-1} \quad \text{by Lemma 2}\\
&\le \kappa_0 e_1^2\cdot\left[Y^\top G'_\sigma Y\right]^{-1} \quad \text{since } \lambda Y^\top Y \ge 0\\
&= \kappa_0 e_1^2\cdot\mathcal{S}(G'_\sigma)^{-1}\\
&\le \kappa_0 e_1^2 c_0^2\cdot\mathcal{S}(G_{\text{id}})^{-1} \quad \text{by Lemma 1,}
\end{aligned}$$

where

$$\begin{aligned}
\kappa_0 &= \frac{(2\lambda+\lambda_{\max}+\lambda_{\min})^2}{4(\lambda+\lambda_{\min})^3} \quad \text{where } \lambda_{min}, \lambda_{max} \text{ are min/max eigenvalues of } G'_\sigma\\
&= \frac{(2\lambda+\lambda_{\max})^2}{4\lambda^3} \quad \text{since } \lambda_{min} = 0 \text{ as } N > d\\
&\le \frac{\left(2\lambda+c_1^2\right)^2}{4\lambda^3}
\end{aligned}$$

where the last line holds as $\lambda_{max} = \|H\|_{op}^2 \le \|H\|_F^2 \le c_1^2$. We obtain the result by setting $C = e_0^{-1}e_1^2c_0^2\cdot\frac{(2\lambda+c_1^2)^2}{4\lambda}$. □

C.5 Proof for Theorem 3

Statement 4. *For any loss function $\mathcal{L}$, let $G_{\text{id}} = H^\top W^\top W H$ and $G_{\text{id}}^+ = H^\top (W^+)^\top W^+ H$, where $W^+ = W - \gamma \nabla_W \mathcal{L}$. If f is 1-positively homogeneous in h, the following holds:*

$$\mathcal{S}(G_{\text{id}}^+) - \mathcal{S}(G_{\text{id}}) \approx 2\gamma\, (f^\top y) \cdot (y^\top K g),$$

where $K_{ij} = h_i^\top h_j$, $f_i = f(h_i)$ are the predictions on the training set, and $g = -\nabla_f \mathcal{L}$.

We first state and prove a lemma:

Lemma 3. *If a function f is m-positively homogeneous in h, then $h_i^\top \nabla_{h_k} f \approx f(h_i) + (m-1) f(h_k)$.*

Proof of Lemma 3. By Taylor expansion, we can write $f(h_i) \approx f(h_k) + (h_i - h_k)^\top \nabla_{h_k} f$. Since f is the m-positively homogeneous, $h^\top \nabla_h f = m f(h)$ holds. So we obtain

$$\begin{aligned} f(h_i) - f(h_k) &\approx h_i^\top \nabla_{h_k} f - h_k^\top \nabla_{h_k} f \\ &= h_i^\top \nabla_{h_k} f - m f(h_k). \\ \Rightarrow h_i^\top \nabla_{h_k} f &\approx f(h_i) + (m-1) f(h_k) \end{aligned}$$

□

Now we prove the main statement.

Proof. Recall the Feature Learning Equation (Equation (1)) and its Gram form (Equation (2)),

$$(W^+)^\top W^+ - W^\top W \approx -\gamma \left[\sum_i h_i \cdot \nabla_{h_i} \mathcal{L}^\top + \nabla_{h_i} \mathcal{L} \cdot h_i^\top \right].$$

By writing $\mathcal{S}(G_{\text{id}}) = \sum_{ij} [\mathcal{S}(G_{\text{id}})]_{ij}$ where $[\mathcal{S}(G_{\text{id}})]_{ij} = y_i y_j h_i^\top W^\top W h_j$, we have

$$\begin{aligned} [\mathcal{S}(G_{\text{id}}^+)]_{ij} - [\mathcal{S}(G_{\text{id}})]_{ij} &= y_i y_j h_i^\top \left((W^+)^\top W^+ - W^\top W \right) h_j \\ &\approx -\gamma y_i y_j h_i^\top \left[\sum_k h_k \cdot \nabla_{h_k} \mathcal{L}^\top + \nabla_{h_k} \mathcal{L} \cdot h_k^\top \right] h_j \\ &= -\gamma \sum_k y_i y_j h_i^\top h_k \cdot \nabla_{h_k} \mathcal{L}^\top h_j - y_i y_j h_i^\top \nabla_{h_k} \mathcal{L} \cdot h_k^\top h_j. \end{aligned}$$

Note $\nabla_h \mathcal{L} = \nabla_h f \cdot \nabla_f \mathcal{L} = -\nabla_h f \cdot g$, and $h_i^\top \nabla_{h_k} f \approx f(h_i) + (m-1) f(h_k) = f(h_i)$ by Lemma 3 and $m = 1$. So we can write

$$\begin{aligned} \mathcal{S}(G_{\text{id}}^+) - \mathcal{S}(G_{\text{id}}) &= \sum_{ij} [\mathcal{S}(G_{\text{id}}^+)]_{ij} - [\mathcal{S}(G_{\text{id}})]_{ij} \\ &\approx -\gamma \left[\sum_{ijk} y_i y_j h_i^\top h_k \cdot \nabla_{h_k} \mathcal{L}^\top h_j + y_i y_j h_i^\top \nabla_{h_k} \mathcal{L} \cdot h_k^\top h_j \right] \\ &= -\gamma \left[\sum_{ijk} 2 y_i y_j h_i^\top \nabla_{h_k} \mathcal{L} \cdot h_k^\top h_j \right] \quad \text{with symmetry} \\ &= \gamma \sum_{ijk} 2 y_i y_j h_i^\top \nabla_{h_k} f \cdot g_k \cdot h_k^\top h_j \\ &\approx \gamma \sum_{ijk} 2 y_i y_j g_k f(h_i) \cdot h_k^\top h_j \\ &= 2\gamma (f^\top y) \cdot (y^\top K g) \quad \text{where } K_{ij} = h_i^\top h_j \text{ and } f_i = f(h_i). \end{aligned}$$

□

C.6 Proof for Theorem 4

Statement 5. *Let $f_l : h_l \mapsto f(x)$ denote the subnetwork mapping the l-th hidden state to the final output, and assume the activation σ is m-positively homogeneous. Assume the network interpolates the training data ($f(x_i) \approx y_i$, equivalently $f_l(h_{l,i}) \approx y_i$ for all i and l) and $\exists C_0 > 0$ s.t. $\|W_l H_l\|_F^2 \geq C_0, \forall l$. Under the Neural Feature Ansatz $W_l^\top W_l \approx \frac{C_l}{N} \sum_i \nabla_{h_{l,i}} f_l \nabla_{h_{l,i}} f_l^\top$ for layer-dependent constants $C_l > 0$, the surrogate satisfies*

$$\mathcal{S}(G_{\mathrm{id}}^l) \begin{cases} \gtrsim C_0 \|Y\|_2^2 \, \Phi_l^{-1}, & \text{if } m > 1 \text{ and } \sum_i y_i = 0, \\ \gtrsim C_0 \|Y\|_2^4 \, \Psi_l^{-1}, & \text{if } m = 1 \text{ and } f_l \text{ is piecewise linear,} \end{cases}$$

where Φ_l^{-1} and Ψ_l^{-1} are monotonically increasing in l.

For the first case ($m > 1$), we use Lemma 3, while for piecewise linear case, we use the following lemma.

Lemma 4. *For a ReLU network f with bounded inputs $\|a\|_2, \|b\|_2 < B$, suppose that the gradient difference across any pair of adjacent linear regions is bounded by δ, i.e.,*

$$\|\nabla f(x) - \nabla f(y)\|_2 < \delta,$$

for any x and y lying in two adjacent linear regions. Let M denote the number of linear-region boundaries crossed by the segment $[a, b]$. Then

$$b^\top \nabla f(a) = f(b) + \epsilon, \quad \text{where} \quad |\epsilon| < 2MB\delta.$$

Proof of Lemma 4. Recall the ReLU network f is a 1-positively homogeneous and piece-wise linear function. By the fundamental theorem of calculus, for a piecewise linear function f and points a, b,

$$\begin{aligned} f(b) - f(a) &= \int_0^1 \nabla f(\gamma(t))^\top (b - a) \, dt \\ &= \sum_{j=0}^{M} (t_{j+1} - t_j) \cdot v_j^\top (b - a), \end{aligned}$$

where $\gamma(t) = a + t(b - a)$, the segment $[a, b]$ crosses M boundaries between linear regions at parameter values $0 = t_0 < t_1 < \cdots < t_M < t_{M+1} = 1$, and $v_j = \nabla f(\gamma(t))$ for $t \in (t_j, t_{j+1})$ denotes the (constant) gradient on the j-th sub-interval. In particular, $v_0 = \nabla f(a)$ and $v_M = \nabla f(b)$.

Subtracting the leading term $v_0^\top (b - a) = \nabla f(a)^\top (b - a)$ from both sides yields the first-order Taylor expansion with explicit remainder:

$$\begin{aligned} f(b) &= f(a) + \nabla f(a)^\top (b - a) - \epsilon, \\ \epsilon &= \sum_{j=1}^{M} (t_{j+1} - t_j)(v_j - v_0)^\top (b - a). \end{aligned}$$

By the δ bound of ∇f for two adjacent linear regions, we have

$$\|v_j - v_0\|_2 = \|\sum_{k=1}^{j} v_k - v_{k-1}\|_2 \leq j\delta.$$

As all input points are normalized, $\|b-a\|_2 < 2B$,

$$\begin{aligned}
|\epsilon| &\le \|b-a\|_2 \cdot \sum_{j=1}^{M}(t_{j+1}-t_j)\|v_j - v_0\|_2 \\
&\le 2B\delta \sum_{j=1}^{M}(t_{j+1}-t_j)j \\
&= 2B\delta \sum_{j=1}^{M}(t_{j+1}-t_j)\sum_{k=1}^{j}1 \\
&= 2B\delta \sum_{j=1}^{M}\sum_{k=1}^{M}(t_{j+1}-t_j)\mathbf{1}_{\{k\le j\}} \\
&= 2B\delta \sum_{k=1}^{M}\sum_{j=k}^{M}(t_{j+1}-t_j) \\
&= 2B\delta \sum_{k=1}^{M}(1-t_k) \le 2MB\delta
\end{aligned}$$

Since $f(a) = a^\top \nabla f(a)$ holds by the 1-homogeneity, the result holds. □

From Lemma 3, we showed that via Taylor expansion, $b^\top \nabla f(a)$ can be approximated by $f(b)$ for 1-homogeneous f. In Lemma 4, we characterize the approximation error, which is analogous to the norm of the Hessian. However, the Hessian does not exist for piecewise-linear functions, as the second-order derivative is zero within linear regions and undefined at the boundaries. We therefore measure a generalized notion of the Hessian in terms of the number of boundaries between linear regions, and use it to bound ϵ. Lemma 4 intuitively shows the error of the first order Taylor expansion can be bounded by the function complexity measured by the number of jumps (M) and the degree of the jump (δ).

Now we prove the main theorem.

Proof. As we focus on the fixed l-th layer, we omit the l superscripts. We can write $\mathcal{S}(G^l_{\text{id}}) = \sum_{ij}\mathcal{S}(G^l_{\text{id}})_{ij}$ where $\mathcal{S}(G^l_{\text{id}})_{ij} = y_i y_j h_i^\top W^\top W h_j$. By applying the Neural Feature Ansatz,

$$\begin{aligned}
\mathcal{S}(G^l_{\text{id}})_{ij} &= C_l \cdot y_i y_j h_i^\top \left[\frac{1}{N}\sum_k \nabla_{h_k} f \cdot \nabla_{h_k} f^\top\right] h_j \\
&= \frac{C_l}{N}\sum_k y_i y_j h_i^\top \nabla_{h_k} f \cdot \nabla_{h_k} f^\top h_j.
\end{aligned}$$

Note that f is l'-homogeneous, where $l' = m^{L-l}$ is the degree of homogeneity for the remaining layers. With Lemma 3, we can rewrite

$$\begin{aligned}
h_i^\top \nabla_{h_k} f \cdot \nabla_{h_k} f^\top h_j &\approx [f(h_i) + (l'-1)f(h_k)] \cdot [f(h_j) + (l'-1)f(h_k)] \\
&\approx [y_i + (l'-1)y_k] \cdot [y_j + (l'-1)y_k] \quad \text{(interpolation assumption)} \\
&= y_i y_j + (l'-1)y_k y_i + (l'-1)y_k y_j + (l'-1)^2 y_k^2.
\end{aligned}$$

So

$$\sum_k y_i y_j h_i^\top \nabla_{h_k} f \cdot \nabla_{h_k} f^\top h_j \approx \sum_k y_i^2 y_j^2 + (l'-1)y_k y_i^2 y_j + (l'-1)y_k y_i y_j^2 + (l'-1)^2 y_k^2 y_i y_j \quad \text{and}$$

$$\begin{aligned}
\mathcal{S}(G^l_{\text{id}}) &\approx \frac{C_l}{N}\sum_{ijk} y_i^2 y_j^2 + (l'-1)y_k y_i^2 y_j + (l'-1)y_k y_i y_j^2 + (l'-1)^2 y_k^2 y_i y_j \\
&= C_l \|Y\|_2^4 + 2(l'-1)(\sum_i y_i)^2 \|Y\|_2^2 + (l'-1)^2 (\sum_i y_i)^2 \|Y\|_2^2
\end{aligned}$$

Case 1 ($m > 1$ and $\sum_i y_i = 0$) As $\sum_i y_i = 0$, the latter terms in $\mathcal{S}(G^l_{\text{id}})$ is 0, i.e. $\mathcal{S}(G^l_{\text{id}}) = C_l \|Y\|_2^4$. Note for the pre-activation norm lower bound,

$$\begin{aligned}
\|WH\|_F^2 = \text{tr}\left[H^\top W^\top W H\right] &= \frac{C_l}{N} \sum_{ik} h_i^\top \nabla_{h_k} f \cdot \nabla_{h_k} f^\top h_i \\
&\approx \frac{C_l}{N} \sum_{ik} \left[f(h_i) + (l'-1) f(h_k)\right]^2 \quad \text{with Lemma 3} \\
&\approx \frac{C_l}{N} \sum_{ik} y_i^2 + \left[(l'-1) y_k\right]^2 + 2(l'-1) y_i y_k \\
&= C_l \left((l'-1)^2 + 1\right) \cdot \|Y\|_2^2 \gtrsim C_0 \quad \text{by assumption.} \\
\Rightarrow C_l \|Y\|_2^2 \gtrsim C_0 \left((l'-1)^2 + 1\right)^{-1} &= C_0 \left(\left(m^{L-l} - 1\right)^2 + 1\right)^{-1}
\end{aligned}$$

So we obtain

$$\mathcal{S}(G^l_{\text{id}}) \approx C_l \|Y\|_2^4 \gtrsim C_0 \|Y\|_2^2 \left(\left(m^{L-l} - 1\right)^2 + 1\right)^{-1} := C_0 \|Y\|_2^2 \, \Phi_l^{-1},$$

where $\Phi_l = \left(\left(m^{L-l} - 1\right)^2 + 1\right)$ is monotonically decreasing in l, as C_0, L, and $\|Y\|_2$ are fixed for all layers, when $m > 1$.

Case 2 ($m = 1$, piecewise linear) As $m = 1$, $l' - 1 = m^{L-l} - 1 = 0$, so the latter term in $\mathcal{S}(G^l_{\text{id}})$ is 0, i.e. $\mathcal{S}(G^l_{\text{id}}) = C_l \|Y\|_2^4$. Define M_l be the number of linear regions in the input space of l-th layer. Define global constants B and δ which bounds the norm of inputs and gradient difference between adjacent linear regions accordance with Lemma 4. Then the normalizing constant,

$$\begin{aligned}
\|WH\|_F^2 = \text{tr}\left[H^\top W^\top W H\right] &= \frac{C_l}{N} \sum_{ik} h_i^\top \nabla_{h_k} f \cdot \nabla_{h_k} f^\top h_i \\
&\approx \frac{C_l}{N} \sum_{ik} \left[f(h_i) + \epsilon_{ik}\right]^2 \quad \text{with Lemma 4} \\
&\approx \frac{C_l}{N} \sum_{ik} \left[y_i + \epsilon_{ik}\right]^2 \gtrsim C_0 \quad \text{by interpolation assumption.}
\end{aligned}$$

This implies

$$\begin{aligned}
C_l &\gtrsim N C_0 \left[\sum_{ik} \left[y_i + \epsilon_{ik}\right]^2\right]^{-1} \\
&\geq N C_0 \left[\left[\sum_{ik} \left|y_i + \epsilon_{ik}\right|\right]^2\right]^{-1} \\
&\geq N C_0 \left[\left[\sum_{ik} |y_i| + |\epsilon_{ik}|\right]^2\right]^{-1} = N C_0 \left(N\|Y\|_1 + \sum_{ik} |\epsilon_{ik}|\right)^{-2}
\end{aligned}$$

So we obtain

$$\begin{aligned}
\mathcal{S}(G_{\text{id}}) \approx C_l \|Y\|_2^4 &\gtrsim N C_0 \|Y\|_2^4 \left(N\|Y\|_1 + \sum_{ik} |\epsilon_{ik}|\right)^{-2} \\
&\geq N C_0 \|Y\|_2^4 (N\|Y\|_1 + 2N^2 M_l B\delta)^{-2} \quad \text{by Lemma 4.} \\
&= N^{-1} C_0 \|Y\|_2^4 (\|Y\|_1 + 2N M_l B\delta)^{-2} := C_0 \|Y\|_2^4 \Psi_l^{-1}
\end{aligned}$$

where $\Psi_l = N(\|Y\|_1 + 2N M_l B\delta)^2$ is monotonically decreasing in l as the number of linear region in input space decreases with l [34]. □

C.7 Proof for Proposition 3

Statement 6. *Consider $f(x) = W^\top h(x) + b$ with hidden states centered ($\sum_i h(x_i) = 0$), W and b trained under squared loss on the centered one-hot label matrix, and $C \leq D$. The surrogate $\mathcal{S}_{\text{cls}}(G_{\text{id}})$ attains the maximum value allowed by a norm constraint $\|G_{\text{id}}\|_F \leq c$ for some $c > 0$ under NC1 and NC2.*

To prove the proposition, we need the following lemma.

Lemma 5. *For a matrix $A \in \mathbb{R}^{m\times n}$ with $AA^\top \neq 0$ and a symmetric matrix $G \in \mathbb{R}^{m\times m}$ with $\|G\|_F \leq c$, $\operatorname{tr}(A^\top GA)$ attains its maximum if and only if $G = \frac{c}{\|AA^\top\|_F} AA^\top$.*

Proof of Lemma 5. Note that $\operatorname{tr}(A^\top GA) = \operatorname{tr}(GAA^\top) = \langle G, AA^\top\rangle_F \leq \|G\|_F \cdot \|AA^\top\|_F \leq c\|AA^\top\|_F$, where equality holds if and only if G is a nonnegative multiple of $AA^\top$ with $\|G\|_F = c$, i.e., $G = \frac{c}{\|AA^\top\|_F} AA^\top$. □

Now we prove the proposition.

Proof. Since the hidden states are normalized, we can write the total, between-class, and within-class variation matrices as $\Sigma_T = \text{Average}_i h_i \cdot h_i^\top$, $\Sigma_W = \text{Average}_{c,i}(h_{i,c} - \mu_c) \cdot (h_{i,c} - \mu_c)^\top$, and $\Sigma_B = \text{Average}_c \mu_c \cdot \mu_c^\top$, where $h_{i,c}$ is the i-th hidden state belonging to the c-th class, and $\mu_c = \text{Average}_i h_{i,c}$ is the class mean vector. Let $M = [\mu_c]_{c=1}^C$ be the class mean matrix. Note that $\Sigma_T = \Sigma_B + \Sigma_W$, and (NC1) implies $\Sigma_W = 0$, so $\Sigma_T = \Sigma_B$.

Following Proposition 1 in Papyan et al. [37], the weight matrix satisfies:

$$W = \frac{1}{C} M^\top \Sigma_T^+ = \frac{1}{C} M^\top \Sigma_B^+ = M^\top (MM^\top)^+ = M^+,$$

where $+$ denotes the Moore-Penrose pseudoinverse. The first equality holds by Webb and Lowe [50], and the second equality holds by the (NC1) condition. The third equality holds by the definition of Σ_B, and the last equality holds by properties of the pseudoinverse.

Since the number of classes C is smaller than the hidden dimension D and the hidden states form the simplex ETF (NC2), $M^+M = I - \frac{1}{C}\mathbf{1}\cdot\mathbf{1}^\top$. This implies $M^+ h_{i,c} = M^+\mu_c$ is a centered one-hot vector with $1 - \frac{1}{C}$ in the c-th element and $-\frac{1}{C}$ the others, so that $M^+H = Y^\top$. Thus, the Gram matrix is $G_{\text{id}} = H^\top W^\top W H = YY^\top$ and the surrogate is $\mathcal{S}_{\text{cls}}(G_{\text{id}}) = \operatorname{tr}\left[Y^\top YY^\top Y\right]$. Lemma 5 shows that $\mathcal{S}_{\text{cls}}(G_{\text{id}})$ is maximized when $G_{\text{id}} = \frac{\|YY^\top\|_F}{\|YY^\top\|_F} YY^\top = YY^\top$, which is achieved by setting $c = \|YY^\top\|_F$. □

C.8 Proof for Proposition 4

Statement 7. *Let $Z = [z_1, \ldots, z_N]^\top$ be latent vectors and $X = [x_1, \ldots, x_N]^\top$ the corresponding inputs. Let $g(z) = z^\top (Z^\top Z)^{-1} Z^\top X$ be the ordinary least-squares estimator, and assume $g \in \mathcal{F}$. Then for any $\alpha \in [0, 1]$,*

$$\min_{f\in\mathcal{F}} \tfrac{1}{N^2} \sum_{i,j} \|f(\alpha z_i + (1-\alpha)z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2 \;\leq\; \tfrac{1}{N} \sum_i \|g(z_i) - x_i\|_2.$$

Proof. Using the Triangle inequality,

$$\begin{aligned}
&\|f(\alpha z_i + (1-\alpha)z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2 \\
&= \|f(\alpha z_i + (1-\alpha)z_j) - g(\alpha z_i + (1-\alpha)z_j) + g(\alpha z_i + (1-\alpha)z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2 \\
&\leq \|f(\alpha z_i + (1-\alpha)z_j) - g(\alpha z_i + (1-\alpha)z_j)\|_2 + \|g(\alpha z_i + (1-\alpha)z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2.
\end{aligned}$$

So we obtain,

$$\begin{aligned}
&\min_{f\in\mathcal{F}} \sum_{ij} \|f(\alpha z_i + (1-\alpha)z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2 \\
&\qquad \leq \min_{f\in\mathcal{F}} \sum_{ij} \|f(\alpha z_i + (1-\alpha)z_j) - g(\alpha z_i + (1-\alpha)z_j)\|_2 \\
&\qquad\qquad + \sum_{ij} \|g(\alpha z_i + (1-\alpha)z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2
\end{aligned}$$

We can show the first term is 0 as $g \in \mathcal{F}$. For the second term, as g is a linear function, we can show

$$g(\alpha z_i + (1-\alpha)z_j) = \alpha g(z_i) + (1-\alpha)g(z_j).$$

Then the error term becomes

$$\begin{aligned}
&\sum_{ij} \|\alpha g(z_i) + (1-\alpha)g(z_j) - (\alpha x_i + (1-\alpha)x_j)\|_2 \\
&\quad \leq \sum_{ij} \alpha\|g(z_i) - x_i\|_2 + (1-\alpha)\|g(z_j) - x_j\|_2 \\
&\quad = \alpha \sum_{ij} \|g(z_i) - x_i\|_2 + (1-\alpha) \sum_{ij} \|g(z_j) - x_j\|_2] \\
&\quad = \alpha N \sum_{i} \|g(z_i) - x_i\|_2 + (1-\alpha) N \sum_{i} \|g(z_i) - x_i\|_2 \\
&\quad = N \cdot \sum_{i} \|g(z_i) - x_i\|_2,
\end{aligned}$$

and the result follows by dividing by N^2. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper's contributions are consistent with its claims made in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation of our work in Section A.2.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The full set of assumptions are correctly stated in each theorem.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We included the experiment details in the supplementary code submission.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We are planning to provide open access code after paper acceptance.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We included the details of experiments in the supplementary code submission.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide error bars for each experiment, if necessary.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on the computer resources in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Authors conducted the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We do not think there is a potential societal impact of our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not think there is a potential risk to mitigate.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited all data and models utilized through the manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We included the documentation in the supplementary code submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

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

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

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

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We mainly used the LLM for writing and editing.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.