BREAKING BALANCE FOR SPEED: RESCALING NORMALIZATION AT INITIALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Gradient flow in homogeneous neural networks is known to preserve a conservation law between adjacent weight matrices, which follows from a known rescaling symmetry that has strong implications for the theoretical analysis of deep learning dynamics. Normalization breaks this symmetry, yet, as we show, its affine parameters introduce another conservation law with the downstream weights that determines how quickly the two parameter groups learn relative to one another. Contemporary initialization schemes bring them out of balance, which typically induces significant benefits for generalization. Leveraging this insight, we deliberately rescale the initialization to move normalization and weight parameters further out of balance and control their relative learning speeds. Across convolutional networks and Transformers, we find that this can accelerate optimization and yield consistent performance gains over standard and balanced initializations.

1 INTRODUCTION

Normalization layers, such as BatchNorm (BN) (Ioffe & Szegedy, 2015) and LayerNorm (LN) (Ba et al., 2016), are standard components of modern deep neural networks. Beyond improving optimization and training stability (Santurkar et al., 2018b; Xiong et al., 2020), normalization introduces learnable affine parameters that are jointly optimized with the surrounding weights. Their interaction can substantially alter optimization dynamics and feature learning. Given the widespread use of normalization in convolutional networks and Transformers, understanding these parameter-level interactions may provide new levers for steering their training dynamics.

A natural perspective on these interactions comes from the theory of homogeneous neural networks and their neuron-wise rescaling symmetries. In such networks, parameters can be rescaled without changing the function represented by the network. By Noether's theorem (Noether, 1918; Olver, 1993), continuous symmetries induce conservation laws under gradient flow. In particular, for each hidden unit j, the difference δ_j between the squared norms of its incoming and outgoing weights remains constant throughout training (Arora et al., 2018; Du et al., 2018). Because δ_j is fixed at initialization, the relative parameter scales chosen before training constrain the subsequent optimization trajectory. This notion of *balance* has become an important tool for understanding optimization dynamics and implicit regularization in deep linear and homogeneous networks (Saxe et al., 2013; Domine et al., 2025; Jacobs & Montúfar, 2026).

However, normalization fundamentally changes this picture. When a normalization layer separates consecutive weights, the classical weight-to-weight conservation law is broken as shown in Figure 1a. We show that balance does not disappear—it *relocates*. A new rescaling symmetry arises between the normalization layer's affine parameters and the downstream weights, inducing a *feature-wise* conserved quantity C_j that couples their scales. This raises a natural question: *How does balance at the normalization–weight interface shape optimization and convergence speed?*

We answer this question by exploiting the fact that C_j is determined at initialization and remains conserved under gradient flow. Concretely, we multiply downstream weights by a constant $\lambda > 0$ and inversely scale the corresponding normalization gain and shift parameters by λ. This rescaling preserves the network function and initial loss, while changing C_j and the relative learning speeds of the two parameter groups. It therefore provides a controlled intervention for varying balance without changing the initial predictor.

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

Our analysis reveals that this function-preserving reparameterization can substantially alter optimization. We derive the effective dynamics at normalization–weight interfaces and show that C_j directly controls the induced preconditioning. In particular, moving the interface further out of balance can accelerate convergence (Figure 1c). Experiments across convolutional networks and Transformers corroborate this prediction: deliberately introducing imbalance at initialization yields faster optimization and improved performance compared with both standard initialization and a λ-balanced initialization.

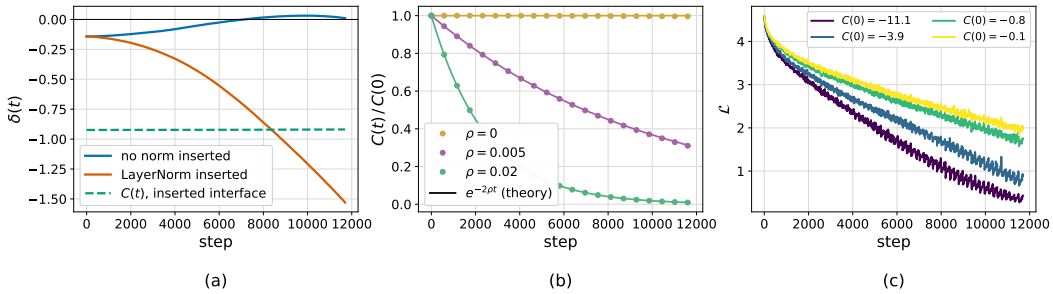


Figure 1: **Normalization breaks classic balance and relocates it to the Norm–Weight interface.** With a DeiT-Tiny on CIFAR-100, 60 epochs of SGD with learning rate $\eta = 0.01$. (a) Inserting a normalization layer between adjacent weights breaks the classical balance $\delta(t)$, while a new conserved quantity C_j emerges between the normalization parameters and downstream weights (Theorem 4.1). (b) Under weight decay, $C_j(t)$ decays exponentially toward 0 (Proposition 4.3). (c) Greater initial imbalance leads to faster convergence (Theorem 4.6).

Our main contributions are:

- **Conservation law.** We derive a feature-wise conservation law for normalization–weight interfaces that couples each downstream weight column to the corresponding normalization parameters (§4).
- **Optimization dynamics.** Based on this insight, we describe the effective dynamics of a normalization–weight interface and show that the conserved quantity C_j directly shapes the induced preconditioning. This explains how greater initial imbalance can yield faster convergence (§4.2).
- **Function-preserving initialization.** Leveraging the normalization–weight rescaling symmetry associated with the conserved quantity, we propose an initialization that moves the respective interfaces out of balance while preserving the initial network function.
- **Experimental validation.** Across vision and language tasks, we validate the predicted relationship between initial balance and optimization dynamics and show that appropriate rescaling can accelerate optimization and improve performance over standard and balanced initializations (§5).

2 Related Work

Normalization. BatchNorm and LayerNorm were introduced to improve the stability and speed of neural-network training (Ioffe & Szegedy, 2015; Ba et al., 2016). Subsequent empirical work links BatchNorm's benefits to its ability to support larger learning rates (Bjorck et al., 2018a), while scale-invariance analyses explain how weight norms and weight decay affect the effective learning rate (Hoffer et al., 2018). Normalization also influences implicit regularization: BatchNorm induces uniform-margin biases in linear models and certain two-layer linear convolutional networks (Cao et al., 2023), while weight normalization can change the bias also towards sparsity (Chou et al., 2023; Adnan et al., 2026). In Transformers, recent approaches normalize representations and weight vectors to unit norm (Loshchilov et al., 2025) or replace normalization layers with learned element-wise transformations (Zhu et al., 2025). Karagodin et al. (2025) theoretically show that normalization in attention mechanisms act as a form of speed regulation. Training only affine normalization parameters can yield strong performance with frozen random weights (Frankle et al., 2021), with network-reconstruction and universal-approximation guarantees under suitable assumptions (Giannou et al., 2023; Burkholz,

2024). Our work shows that normalization breaks the classical $\delta(t)$ rescaling symmetry, while inducing a new feature-wise symmetry that couples BatchNorm and LayerNorm affine parameters to downstream weight columns and yields a new conserved quantity throughout training (Figure 1).

Initialization. Neural network weight initializations tailored to specific architectures have been proposed (Balduzzi et al., 2017; Burkholz & Dubatovka, 2019; De & Smith, 2020; Zhang et al., 2018; Fischer et al., 2022; Gadhikar & Burkholz, 2022; Mustafa et al., 2023; Price et al., 2024; Nowak et al., 2024; Dent & Tanner, 2026; Saada et al., 2025) to increase the trainability of neural networks without relying on normalization layers. Still, normalization has remained a critical component of most modern architectures for its many benefits, including increased training stability with large learning rates over long training horizons (Bjorck et al., 2018b), conditioning (Santurkar et al., 2018a; Daneshmand et al., 2021; Lange et al., 2022; Burkholz, 2025), mitigating neural collapse (Yaras et al., 2022), facilitating finetuning (Kopiczko et al., 2024; Basu et al., 2024; Zhou et al., 2025) and even sparsification (Gadhikar & Burkholz, 2024; Qu & Horváth, 2025). As normalization has been designed to mitigate initialization issues, surprisingly, the initialization of trainable normalization parameters has received less attention. As we show, their relation to neural network weights has great impact on their relative learning speed and presents an effective lever to tune learning dynamics and boost generalization.

Balance equations and conservation laws. Gradient flow in positively homogeneous networks preserves differences between the squared norms of adjacent layers, so an approximately balanced initialization remains approximately balanced throughout training (Du et al., 2018). In deep linear networks, these balance identities enable exact analyses of learning dynamics (Saxe et al., 2013; Domine et al., 2025). The same balancedness principle extends beyond feed-forward architectures: Mustafa et al. (2023) derive a conservation law for the gradient flow dynamics of Graph Attention Networks, showing that it explains why a large fraction of parameters under standard initialization fail to move during training and even limit their expressiveness (Mustafa & Burkholz, 2024b), an effect that worsens with depth. More generally, conservation laws can be characterized through the model Jacobian (Marcotte et al., 2023), with extensions to non-Euclidean geometries and momentum dynamics (Marcotte et al., 2024; Jacobs & Montúfar, 2026; Jacobs et al., 2026; Wang & Klabjan, 2024), as well as modern architectures (Marcotte et al., 2025; Tran et al., 2026). Based on such insights, Mustafa & Burkholz (2024a) improve feature learning in graph neural networks by breaking the hindering conservation law. Their work does not translate to modern neural network architectures with normalization layers, however. Our key contribution is to take normalization into consideration, identifying feature-wise balance laws arising from the joint rescaling symmetry of downstream weight columns and affine normalization parameters, and relating their initial imbalance to learning speed.

Implicit bias and learning speed. Balance and imbalance are not merely conserved quantities; they actively shape optimization. Balance identities underpin studies of implicit acceleration and regularization from overparameterization Arora et al. (2018; 2019); Pesme et al. (2021); Gunasekar et al. (2017); Woodworth et al. (2020); Chou et al. (2021). More broadly, product parameterizations are known to control learning speed and representation dynamics Jacobs & Burkholz (2025); Domine et al. (2025); Kunin et al. (2024), with exact solutions further revealing how initialization imbalance can promote rapid feature learning and shape the transition between lazy and rich regimes Kunin et al. (2024); Domine et al. (2025). Our defined Norm–Weight interfaces precisely have this product structure: the normalization gain and downstream weight jointly define an effective parameter whose dynamics are preconditioned by the two factors. Explicit regularization also modifies these structures: uniform weight decay turns conservation into a controlled exponentially decaying quantities Marcotte et al. (2025); Jacobs et al. (2025). Our imbalance lower bound on convergence speed (Theorem 4.6) contributes directly to this line of work, showing that the initial imbalance at a Norm–Weight interface controls the rate at which the interface's effective parameter converges.

3 Preliminaries

Throughout, we write $[n] := \{1,2,\ldots,n\}$ for $n \in \mathbb{N}$, $\|\cdot\|_2$ for the Euclidean norm, and $\|\cdot\|_F$ for the Frobenius norm. We study feedforward networks trained by gradient flow, focusing on how positive homogeneity of the activation function interacts with normalization layers. To do this, we first recall the relevant structural facts about homogeneous networks and gradient flow, and then introduce a unified way of describing normalization layers through interfaces.

Definition 3.1. A function $f:\mathbb{R}^m \to \mathbb{R}^n$ is called positive homogeneous if and only if for all $\theta \in \mathbb{R}^m$ and $\lambda > 0$ we have that $f(\lambda\theta) = \lambda f(\theta)$.

Example 3.2. *Examples of positive homogeneous activation functions $\sigma:\mathbb{R}\to\mathbb{R}$ are $\sigma(z)=z$ (Linear), $\sigma(z)=max\{z,0\}$ (ReLU), and $\sigma(z)=max\{z,cz\}$ (Leaky-ReLU) where $c>0$.*

Gradient flow and balance equation. Let θ denote all trainable parameters and let $\mathcal{L}(\theta)$ be a (sub)-differentiable loss. Under gradient flow,

$$d\theta_t = -\nabla_\theta \mathcal{L}(\theta_t)dt, \qquad \theta_0 = \theta_{\text{init}}, \tag{1}$$

where θ_{init} is the model parameters at initialization. As a consequence of positive homogeneity and training with gradient flow, the two layers of a network stay "balanced" relative to one another for all time. The gap between their squared norms is a conserved quantity, fixed by the initialization. We recall this in Lemma 3.3, for the two layer network $f(W_1,W_2\,|\,x) := W_2\sigma(W_1x)$ with homogeneous activation σ.

Lemma 3.3. *As established in (Du et al., 2018; Arora et al., 2018), consider a two layer neural network $f(W_1,W_2|x) := W_2\sigma(W_1x)$, where σ is positive homogeneous, and let $W_{1,t}$ and $W_{2,t}$ denote the two weight matrices at time t along the gradient flow Eq (1). Define the neuron imbalance for each hidden dimension $j \in [d]$,*

$$\delta_j(t) := \|W_{2,\cdot,j,t}\|_2^2 - \|W_{1,j\cdot,t}\|_2^2 = \|W_{2,\cdot,j,0}\|_2^2 - \|W_{1,j\cdot,0}\|_2^2 \tag{2}$$

the difference of two squared Euclidean norms and let $\delta(t) := \sum_{j=1}^d \delta_j(t)$. Then for each $j \in [d]$, δ_j is conserved, $\delta_j(t) = \delta_j(0)$ for all $t \geq 0$ and similarly for $\delta(t)$.

We refer to (2) as the δ-balance property. We will later analytically and empirically study how normalization layers affect this conversation law in Section 4.

Normalization layers. In practice, homogeneous layers are rarely applied directly to raw pre-activations. We consider LayerNorm and BatchNorm layers with learnable feature-wise scale and shift parameters $\gamma,\beta \in \mathbb{R}^d$.

Definition 3.4. Let $\mathcal{N} \in \{\text{LN},\text{BN}\}$. For a batch $H \in \mathbb{R}^{b\times d}$ with batch size b and feature dimension d, we write $\hat{H} := \mathcal{N}_{1,0}(H)$ for the corresponding normalized activations. The affine normalization layer is $\mathcal{N}_{\gamma,\beta}(H)_{i,j} = \gamma_j \hat{H}_{i,j} + \beta_j$, where γ_j and β_j are the learnable scale and shift associated with feature j.

LayerNorm and BatchNorm differ only in how the normalized activations $\hat{H}$ are computed. Their full definitions are given in Appendix A. The placement of normalization relative to linear layers and nonlinearities depends on the architecture (Ioffe & Szegedy, 2015; Ba et al., 2016). Our analysis focuses on the local structure in which the output of a normalization layer is consumed by a downstream weight, allowing a positively homogeneous activation in between. We formalize this in the following definition.

Definition 3.5. A *Norm–Weight interface* is a normalization layer $\mathcal{N}_{\gamma,\beta}$ and a downstream weight $W \in \mathbb{R}^{p\times d}$, with $p \geq 1$, that consumes its output, with a positively homogeneous activation ϕ in between. The case $p=1$ corresponds to a downstream weight vector $w \in \mathbb{R}^d$. For $H \in \mathbb{R}^{b\times d}$, the interface output is $Z_{ik} := \sum_{j=1}^d W_{kj}\phi(\mathcal{N}_{\gamma,\beta}(H)_{ij})$, where $i \in [b]$, $k \in [p]$, and $\phi = \text{id}$ is allowed. We assume that the normalized activations $\mathcal{N}_{\mathbf{1},\mathbf{0}}(H)$ do not depend on (W,γ,β), and that the remainder of the network depends on these parameters only through Z. Figure 2 illustrates the resulting Norm–Weight interface.

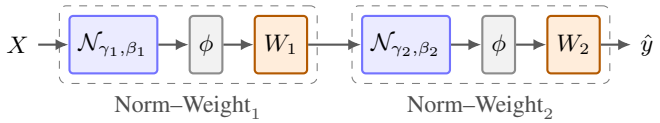


Figure 2: **Norm–Weight interface.** Each highlighted block has a function-preserving rescaling symmetry between the normalization affine parameters and downstream weight.

Using interfaces as in Definition 3.5 we can ask how the δ-balance property of Lemma 3.3 is affected once W_1 is replaced by a Norm–Weight interface.

4 THEORETICAL ANALYSIS

The δ-balance property (Lemma 3.3) follows from the rescale symmetry between consecutive homogeneous weights. Normalization breaks this symmetry and instead a new rescaling symmetry arises between the affine parameters (γ_j, β_j) and the corresponding downstream weight column $W_{\cdot j}$. We first prove this symmetry yields a new conserved quantity at every Norm–Weight interface (Theorem 4.1); this holds true for both LN and BN, for any numerical stabilizer $\epsilon \geq 0$, and allows a positively homogeneous activation between the normalization layer and downstream weight. We then study how weight decay exponentially drives the conserved quantity to zero (Proposition 4.3) and minimizing the weight decay penalty over all function preserving rescaling symmetries is attained at $C_j = 0$ and induces a feature-wise group Lasso cost. Lastly, Section 4.2 shows that this quantity controls how fast the layers on the two sides of an interface learn, and therefore how fast training converges.

4.1 CONSERVATION AT THE NORM–WEIGHT INTERFACE

For a norm-weight interface, multiplying $W_{\cdot j}$ by $\lambda > 0$ and dividing (γ_j, β_j) by the same factor leaves the resulting output unchanged, including when a positively homogeneous activation intervenes. We present the corresponding balance equation in Theorem 4.1.

Theorem 4.1. *Let (W, γ, β) from a Norm–Weight interface and let θ evolve under the gradient flow. Then for every $j \in [d]$ and all $t \geq 0$:*

$$\|W_{\cdot j,t}\|_2^2 - \gamma_{j,t}^2 - \beta_{j,t}^2 = \|W_{\cdot j,0}\|_2^2 - \gamma_{j,0}^2 - \beta_{j,0}^2 =: C_j(t). \tag{3}$$

Proof sketch. Differentiating the feature-wise rescaling symmetry gives $\langle \nabla_{W_{\cdot j}} \mathcal{L}, W_{\cdot j} \rangle = \gamma_j \partial_{\gamma_j} \mathcal{L} + \beta_j \partial_{\beta_j} \mathcal{L}$, so $\dot{C}_j = 0$ under gradient flow (see Theorem C.1).

It follows from Theorem 4.1 that the symmetry couples each downstream weight column to the corresponding normalization parameters: their squared norms change at equal rates under gradient flow (Eq (1)). Normalization therefore relocates the classical balance as in (Eq (2)) between adjacent weights to a feature-wise balance at the Norm–Weight interface. Since Theorem 4.1 preserves the imbalance set at initialization, we next characterize this quantity under a standard initialization.

Corollary 4.2. *For He initialization $W_{2,ij} \sim N(0,\, 2/d)$ and $W_{1,jk} \sim N(0,\, 2/n)$ with $i \in [m],\ j \in [d], k \in [n]$, $\gamma = \mathbf{1}$ and $\beta = 0$ we have that $\mathbb{E}[C_j] = 2\frac{m}{d} - 1$ and $\mathrm{Var}(C_j) = \frac{8m}{d^2}$. Moreover, without normalization we have that $\mathbb{E}[\delta_j] = 2(\frac{m}{d} - 1)$ and $\mathrm{Var}(\delta_j) = 8(\frac{m}{d^2} + \frac{1}{n})$.*

Proof: It follows directly from summing second moments of the centered Gaussian distribution. See Corollary B.2 for a general calculation.

■

Furthermore, Corollary 4.2 shows that normalization can change both the mean and variance of the initialization imbalance. In particular, when $m = d = n$, He initialization is balanced without normalization, $\mathbb{E}[\delta_j] = 0$, but becomes imbalanced with normalization, $\mathbb{E}[C_j] = 1$. More generally, normalization also reduces the variance of the imbalance, as shown in Lemma B.3; additional initialization schemes are compared in Tables 1 and 2. These imbalances persist under gradient flow and shape the effective learning dynamics studied in Section 4.2.

Norm conservation. Normalization also induces an independent scale symmetry in the upstream weights when $\epsilon = 0$, which preserves their norms under gradient flow. Denoting such a weight matrix by W_1, for both LayerNorm and BatchNorm, $\|W_{1,t}\|_F^2$ remains constant; for BatchNorm, this conservation additionally holds column-wise. Thus, the upstream weights can learn by changing direction while their norms remain fixed. This conservation is separate from the Norm–Weight balance, which couples the downstream weights to the normalization parameters (see Lemma C.2).

Weight decay. The conserved quantity of Theorem 4.1 is a statement about the loss: it requires the objective to depend on (W, γ, β) only through the interface output. Weight decay violates this requirement, since the penalty is a function of the parameters themselves and not of what the network computes with them. Consequently, the invariant is no longer conserved and Proposition 4.3 shows it decays exponentially, at a rate set by the regularization strength alone (Figure 1b).

Proposition 4.3. *Let (W,γ,β) form a Norm–Weight interface and let θ evolve under gradient flow on the regularized objective $\widetilde{\mathcal{L}} = \mathcal{L} + \frac{\rho}{2}\|\theta\|_2^2$ with $\rho > 0$. Then for every $j \in [d]$,*

$$\dot{C}_j(t) = -2\rho C_j(t), \text{ hence } C_j(t) = C_j(0)e^{-2\rho t} \quad \text{for all } t \geq 0.$$

Proof sketch. Weight decay adds a shrinkage term $-\rho\theta$ to the velocity of every parameter. So each squared norm acquires a factor of $-2p$, and so does their signed difference. Thus, weight decay drives the imbalance exponentially towards zero (see Proposition C.3).

Representational cost of weight decay. The Norm–Weight symmetry together with weight decay also gives a variational interpretation. Although rescaling the affine parameters and downstream weights leaves the network function unchanged, it changes their weight decay cost. Among all function-equivalent parameterizations obtained by rescaling a Norm–Weight interface, the weight decay cost is minimized at balance, $C_j = 0$. Minimizing over this redundant scale, the quadratic penalty becomes a feature-wise group Lasso penalty (Kolb et al., 2025; 2026; Parhi & Nowak, 2023).

Proposition 4.4. *Let (W,γ,β) form a Norm–Weight interface. For feature j, let $a_j := (\gamma_j, \beta_j)$ and define its affine magnitude $r_j := \|a_j\|_2$, with effective weight $v_j := r_j W_{\cdot j} \in \mathbb{R}^p$, which is invariant under the Norm–Weight rescaling. The minimum weight decay cost of the interface over all per-feature rescalings is*

$$\inf_{\lambda_1,\ldots,\lambda_d} \frac{\rho}{2}\sum_{j=1}^{d}\left(\lambda_j^2\|W_{\cdot j}\|_2^2 + \lambda_j^{-2} r_j^2\right) = \rho\sum_{j=1}^{d}\|v_j\|_2 =: \rho\|V\|_{2,1}. \tag{4}$$

The minimum for feature j is attained at $\lambda_j^\star = \left(\frac{r_j^2}{\|W_{\cdot j}\|_2^2}\right)^{1/4}$, hence $C_j^{(\lambda_j^\star)} = 0$.

Proof sketch. The effective weight is unchanged by the Norm–Weight rescaling, while its weight decay cost varies with how this fixed scale is split between r_j and $W_{\cdot j}$. By AM–GM, this cost is lower bounded by $\rho\|v_j\|_2$, with equality when the two factors are balanced, $C_j = 0$. Summing over all features gives an $\ell_{2,1}$ group Lasso penalty (see Proposition C.4).

Having characterized the implicit bias of weight decay toward balance, we next study how C_j itself shapes the effective learning dynamics and convergence speed.

4.2 Imbalance and convergence speed

Our conserved quantity of Theorem 4.1 not only stays preserved throughout training, but also controls the effective learning dynamics of a Norm–Weight interface. The weight column and normalization gain act on the network through their product, whose gradient flow is preconditioned (Lemma 4.5). In the scalar setting, the conserved quantity enters this preconditioner directly and provides a uniform lower bound, yielding a convergence rate that increases with the initial imbalance (Theorem 4.6).

Lemma 4.5. *Let (W,γ,β) form a Norm–Weight interface with $\beta = 0$, and for each $j \in [d]$ let $\Pi_{\cdot j} := \gamma_j W_{\cdot j} \in \mathbb{R}^p$ be the effective parameter of feature j. Under gradient flow,*

$$\dot{\Pi}_{\cdot j} = -\underbrace{\left(\gamma_j^2 I_p + W_{\cdot j} W_{\cdot j}^\top\right)}_{=:M_j} \nabla_{\Pi_{\cdot j}} \mathcal{L}. \tag{5}$$

Thus $\Pi_{\cdot j}$ follows a preconditioned gradient flow with preconditioner M_j. The preconditioner scales every direction orthogonal to $W_{\cdot j}$ by γ_j^2, and the direction parallel to $W_{\cdot j}$ by $\gamma_j^2 + \|W_{\cdot j}\|_2^2$. When M_j is positive definite, this can equivalently be viewed as a Riemannian gradient flow with metric tensor M_j^{-1} The statement holds for both LayerNorm and BatchNorm and for any $\epsilon \geq 0$.

Proof sketch. The interface depends on $W_{\cdot j}$ and γ_j only through their product $\Pi_{\cdot j} := \gamma_j W_{\cdot j}$. Under gradient flow, both factors move, so differentiating their product gives two contributions that combine to form the preconditioner M_j. The $\gamma_j^2 I_p$ term acts equally in every direction, while $W_{\cdot j} W_{\cdot j}^\top$ adds motion only along the direction of $W_{\cdot j}$ (see Lemma C.5 for full proof).

Lemma 4.5 shows how the scale is split between $W_{\cdot j}$ and γ_j directly determines the effective preconditioning seen by the feature. We consider the case when $p = 1$, to show this preconditioning can

be written explicitly in terms of C_j. Let $w,\gamma \in \mathbb{R}^d$ and $\pi := \gamma \odot w$. Then the effective dynamics reduce to

$$\dot{\pi} = -M(\pi,C)\nabla\mathcal{L}(\pi), \qquad M_{jj}(\pi,C) = \sqrt{C_j^2 + 4\pi_j^2}.$$

Importantly, $M_{jj}(\pi,C) \geq |C_j|$. This lower bound holds throughout training because C_j is conserved by Theorem 4.1. Hence, imbalance prevents the effective preconditioner from becoming arbitrarily small; this geometric effect is visualized in Figure 3. The full metric characterization and bound are provided in Lemma C.5 and Corollary C.7, respectively. The restriction $p = 1$ is only needed for an arbitrary-sign guarantee, for general p, the same convergence bound holds whenever $C_j < 0$ for all features (see Remark C.9).

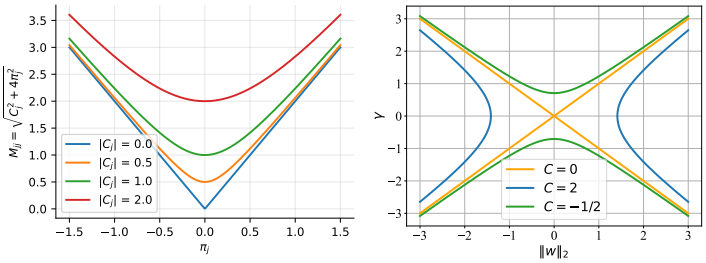


Figure 3: **Left**: Effective preconditioner M_{jj} as a function of π_j for different imbalances. All curves approach $2|\pi_j|$ away from the origin, so imbalance matters most near $\pi_j = 0$. **Right**: The Norm–Weight conserved quantity C changes depending on the dimension ratio of w.

Theorem 4.6. *Suppose $\mathcal{L}$ satisfies the Polyak-Łojasiewicz inequality in π with constant τ (Definition A.3) and let $C_{\min} := \min_j |C_j|$. Then under gradient flow on (w,γ):*

$$\mathcal{L}(\pi_t) - \mathcal{L}^* \leq (\mathcal{L}(\pi_0) - \mathcal{L}^*)e^{-2\tau C_{\min} t}.$$

Proof sketch. From Corollary C.7, the effective parameter moves at least $|C_j|$ per unit of gradient. Since C_j is fixed by initialization, this bound holds at every t and can be integrated along the trajectory. The PL inequality then replaces the gradient magnitude by the remaining loss, giving exponential decay at a rate proportional to $C_{\min}$ (see Theorem C.8 for full proof).

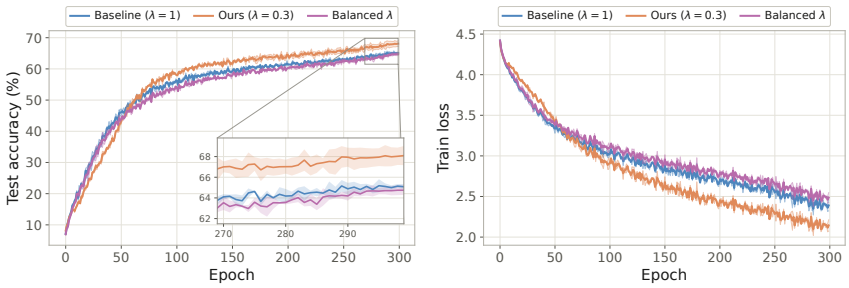
Thus, a larger initialization imbalance yields a faster guaranteed convergence rate. This motivates our initialization scaling mechanism we introduce in §5: applying the same rescale symmetry once, at initialization, deliberately imbalances every interface without changing the network function, and thereby sets the speed at which its two sides learn.

5 Experiments

Controlling balance at initialization. We can directly set the imbalance at initialization to control the relative learning of the downstream weight parameters and normalization affine parameters. Our intervention applies the Norm–Weight rescaling symmetry once, at initialization. Starting from a standard initialization $\bar{\theta}_0$, we multiply each selected downstream weight by a constant $\lambda > 0$ and divide its corresponding gain and shift by the same factor at initialization for each Norm–Weight interface:

$$W^{(\lambda)}_{\cdot j,0} = \lambda \bar{W}_{\cdot j,0}, \qquad \gamma^{(\lambda)}_{j,0} = \lambda^{-1}\bar{\gamma}_{j,0}, \qquad \beta^{(\lambda)}_{j,0} = \lambda^{-1}\bar{\beta}_{j,0}, \qquad j \in [d]. \tag{6}$$

By rescale invariance, the transformation in Eq (6) preserves the initial network function and loss, while changing the parameter scales, C_j, and subsequent learning dynamics. The choice of $\lambda = 1$ recovers the standard initialization. Moreover, we construct a λ-balanced initialization by setting $\lambda_{\text{bal}} = \left(\frac{\sum_{i\in\mathcal{I}} \sum_{j=1}^{d_i} (\gamma_j^{(i)2} + \beta_j^{(i)2})}{\sum_{i\in\mathcal{I}} \sum_{j=1}^{d_i} \|W_{\cdot j}^{(i)}\|_2^2} \right)^{1/4}$, where $\mathcal{I}$ denotes the selected Norm–Weight interfaces. The sum runs over every feature, such that the mean conserved quantity across $\mathcal{I}$ is balanced, satisfying $\mathbb{E}[C_j] = 0$. The λ_{bal} is a single global scalar applied identically to every selected interface, computed once at initialization. This provides a direct baseline for isolating the effect of introducing imbalance at initialization. Together, $\lambda = 1$ and λ_{bal} provide standard and balanced baselines against which we compare our proposed imbalanced initialization. Details for determining the optimal λ in Appendices D and E.



(a) DeiT-Tiny: Test Accuracy

(b) DeiT-Tiny: Train Loss

Figure 4: **Increasing initialization imbalance accelerates optimization and improves generalization.** For DeiT-Tiny on CIFAR-100, the imbalanced initialization at $\lambda = 0.3$ yields improved final test accuracy (a) and faster convergence (b) compared to both the standard initialization ($\lambda = 1$) and the λ_{bal}.

Larger imbalance accelerates optimization. The change in initialization imbalance induced by λ can be computed directly. By applying our initialization transformation of Eq. (6) to Corollary B.2, for $W_{\cdot,j,0} \sim \mathcal{N}(0, \sigma^2 I_m)$, $\gamma_{j,0} = 1$, and $\beta_{j,0} = 0$, gives

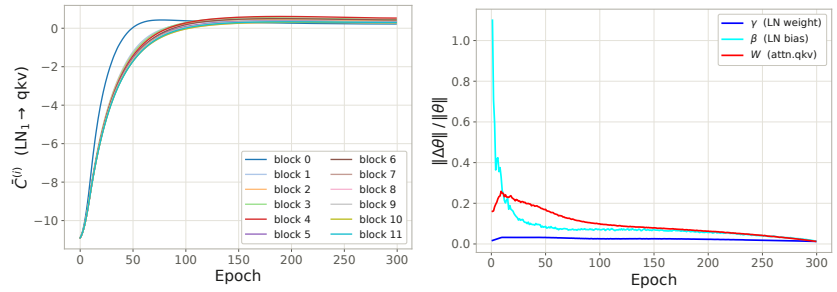
$$\mathbb{E}[C_j^{(\lambda)}(0)] = \lambda^2 m \sigma^2 - \lambda^{-2}. \tag{7}$$

Thus, λ changes the magnitude of C_j while leaving the initial network function unchanged. For DeiT-Tiny, the imbalance predicted by Eq (7) closely matches the conserved quantities observed in Figure 5a, where the optimal choice of $\lambda = 0.3$ begins substantially farther from balance than the standard initialization (Figure 10c). This larger initial imbalance is accompanied by significantly faster convergence in Figure 4b. This behaviour is consistent with Theorem 4.6, which shows in a tractable setting that larger $|C_j|$ results in a faster convergence rate. The same rescaling also improves final test accuracy in Figure 4a, showing that the optimization gain yields improved generalization performance. We also observe improved validation perplexity for GPT-2 Small (Radford et al., 2019) with our $\lambda = 0.5$ initialization, as shown in Figure 6b. See Appendices D and E for details on the experimental setup and results on ResNet50/ImageNet.

The direction of imbalance determines what learns faster. While the absolute magnitude of C_j controls the strength of the imbalance and, as shown above, is associated with faster convergence, its signed direction (i.e., $\text{sign}(C_j)$) determines how learning is distributed across the two parameter groups of the Norm–Weight interface. More specifically, for a nonzero parameter group θ, we measure its relative gradient speed as $s_\theta := \|\nabla_\theta \mathcal{L}\|_2 / \|\theta\|_2$.

At initialization, the function-preserving rescaling in Eq. (6) implies that applying $\lambda > 1$ shifts scale toward the downstream weights and moves each C_j in the *positive direction*. Thus, increasing the relative speed of the normalization parameters with respect to the downstream weights. Conversely, $\lambda < 1$ shifts scale toward the normalization parameters and moves C_j in the *negative direction*, increasing the relative speed of the downstream weights with respect to the normalization parameters. Figure 5b verifies this effect for the DeiT-Tiny attention interface with $\lambda = 0.3$, the downstream `qkv` weights move substantially faster relative to their scale than the LayerNorm affine parameters. This separation is much weaker under the standard (Figure 11c) and λ-balanced initializations (Figure 11e). Hence, for the Transformer experiments, moving C_j in the negative direction favours faster relative learning of the downstream weights starting from initialization, and coincides with improved optimization as shown in Figure 4. In contrast, ResNet-20 benefits from moving C_j in the positive direction (Figure 7a) and favours faster learning of γ and slower learning of the weights in the initial phase of training (Figure 7b), resulting in improved generalization performance (Figure 6a).

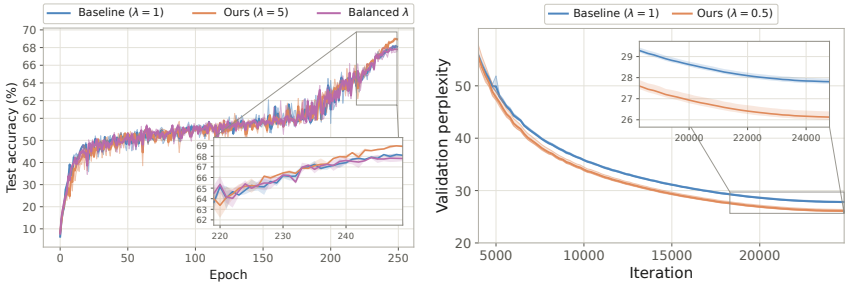
Weight decay gradually restores balance. The optimization benefit of starting imbalanced directly connects to the implicit bias of weight decay. Proposition 4.3 shows that weight decay progressively brings C_j toward zero, consistent with the trajectories in Figure 1. Thus, the network can be initialized far from balance, where the initial imbalance accelerates the effective learning dynamics, while weight decay gradually drives the network toward balance over the course of training. By the representation cost argument in Proposition 4.4, this balanced regime is also the minimum-cost parameterization



(a) DeiT-Tiny: conserved quantity $\bar{C}^{(i)}(t)$. (b) DeiT-Tiny: relative parameter speed.

Figure 5: **Initialization imbalance shapes relative learning speed.** DeiT-Tiny on CIFAR-100 with $\lambda = 0.3$ at the $\mathrm{LN}_1 \to \mathtt{qkv}$ Norm–Weight interface. In (a), for each Transformer block i, we plot the feature-wise mean $\bar{C}^{(i)}(t) = \frac{1}{d}\sum_{j=1}^{d} C_j^{(i)}(t)$. The initialization begins strongly imbalanced, and weight decay progressively brings each $C_j^{(i)}$ toward zero (Proposition 4.3). In (b), relative parameter updates of the downstream qkv weights and LayerNorm affine parameters averaged over all blocks.

preferred by weight decay and induces a feature-wise sparsity bias on the effective representation. As we demonstrated, the initial imbalance can thus improve optimization without changing the final induced regularization by weight decay.



(a) ResNet20: test accuracy. (b) GPT2-Small: validation perplexity.

Figure 6: **Improved generalization across vision and language.** The imbalanced initialization improves held-out performance for both ResNet-20 on CIFAR-100 with $\lambda = 5$ (a) and GPT-2 Small on SlimPajama-6B with $\lambda = 0.5$ (b).

6 Conclusion and Future Work

Initialization and normalization are often treated as separate mechanisms for making deep networks trainable. We show instead that normalization couples initialization to optimization through the relative scale of its affine parameters and downstream weights. Normalization relocates the classical balance structure to this Norm–Weight interface, where the magnitude of imbalance controls learning speed and its direction determines which side of the interface learns faster. Standard initializations are already substantially imbalanced, suggesting that this asymmetry is a consequential part of how normalized networks train rather than an artifact. Moreover, weight decay progressively restores balance for each interface, with the balanced limit corresponding to the minimum-cost representation and inducing a feature-wise sparsity bias. This insight yields a remarkably simple intervention: rescale the two sides of a Norm–Weight interface once at initialization. Across convolutional networks and Transformers, deliberately moving these interfaces further out of balance accelerates optimization and improves generalization performance over standard and balanced initializations. On top of that, the results show that normalization does not make initialization irrelevant — it highlights a new initialization lever for shaping learning dynamics and generalization. Understanding its interaction with adaptive optimizers and dynamically controlling imbalance during training are promising directions for future work.

AI USE STATEMENT

We used generative AI tools to improve the overall writing and polishing of the manuscript. We also used them to assist with generating code to run the experiments, and to check the correctness and presentation of proofs after the primary structure of the proofs had already been derived by hand. Generative AI tools were not used to develop the theoretical models or conceptual framework, formulate mathematical claims, provide critical ingredients for proofs, propose or refine hypotheses, or design or provide feedback on the research methodology or experiments.

REPRODUCIBILITY STATEMENT

Complete proofs are provided in Appendix C and full experimental details for reproducibility are in Appendices D and E.

REFERENCES

Mohammed Adnan, Rohan Jain, Tom Jacobs, Ekansh Sharma, Rahul G Krishnan, Rebekka Burkholz, and Yani Ioannou. Sparseopt: Addressing normalization-induced gradient skew in sparse training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=o9JP7N8YO9.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization, 2018. URL https://arxiv.org/abs/1802.06509.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *ArXiv*, abs/1905.13655, 2019. URL https://api.semanticscholar.org/CorpusID:173188705.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016. URL https://api.semanticscholar.org/CorpusID:8236317.

David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 342–350, 2017.

Samyadeep Basu, Shell Xu Hu, Daniela Massiceti, and Soheil Feizi. Strong baselines for parameter-efficient few-shot fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11024–11031, 2024. doi: 10.1609/aaai.v38i10.28978.

Johan Bjorck, Carla Gomes, Bart Selman, and Kilian Q. Weinberger. Understanding batch normalization. In *The Thirty-second Annual Conference on Neural Information Processing Systems*, 2018a. URL https://api.semanticscholar.org/CorpusID:46956723.

Nils Bjorck, Carla Gomes, Bart Selman, and Kilian Weinberger. Understanding batch normalization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf.

Rebekka Burkholz. Batch normalization is sufficient for universal function approximation in CNNs. In *International Conference on Learning Representations*, 2024.

Rebekka Burkholz. Random features for normalization layers, 2025. URL https://openreview.net/forum?id=E361DSJEyT.

Rebekka Burkholz and Alina Dubatovka. Initialization of ReLUs for dynamical isometry. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yuan Cao, Difan Zou, Yuanzhi Li, and Quanquan Gu. The implicit bias of batch normalization in linear models and two-layer linear convolutional neural networks. In *The Thirty Sixth Annual Conference on Learning Theory*, pp. 5699–5753. PMLR, 2023.

Hung-Hsu Chou, Johannes Maly, and Holger Rauhut. More is less: Inducing sparsity via overparameterization. *ArXiv*, abs/2112.11027, 2021. URL https://api.semanticscholar.org/CorpusID:245353663.

Hung-Hsu Chou, Holger Rauhut, and Rachel A. Ward. Robust implicit regularization via weight normalization. *ArXiv*, abs/2305.05448, 2023. URL https://api.semanticscholar.org/CorpusID:258564503.

Hadi Daneshmand, Amir Joudaki, and Francis Bach. Batch normalization orthogonalizes representations in deep random networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=_RSgXL8gNnx.

Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems*, 33:19964–19975, 2020.

Emily Dent and Jared Tanner. How controlling the variance can improve training stability of sparsely activated dnns and cnns, 2026. URL https://arxiv.org/abs/2602.05779.

Clementine Domine, Nicolas Anguita, Alexandra M Proca, Lukas Braun, Daniel Kunin, Pedro Mediano, and Andrew Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. In *International Conference on Learning Representations*, volume 2025, pp. 102485–102536, 2025.

Simon Shaolei Du, Wei Hu, and J. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *ArXiv*, abs/1806.00900, 2018. URL https://api.semanticscholar.org/CorpusID:46927169.

Jonas Fischer, Advait Gadhikar, and Rebekka Burkholz. Lottery tickets with nonzero biases, 2022. URL https://arxiv.org/abs/2110.11150.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Training BatchNorm and only BatchNorm: On the expressive power of random features in CNNs. In *International Conference on Learning Representations*, 2021.

Advait Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. In *Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qODvxQ8TXW.

Advait Harshal Gadhikar and Rebekka Burkholz. Dynamical isometry for residual networks, 2022.

Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. The expressive power of tuning only the normalization layers. In *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195, 2023.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/58191d2a914c6dae66371c9dcdc91b41-Paper.pdf.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a0160709701140704575d499c997b6ca-Paper.pdf.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Tom Jacobs and Rebekka Burkholz. Mask in the mirror: Implicit sparsification. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=U47ymTS3ut.

Tom Jacobs and Guido Montúfar. Implicit bias of mirror flow in homogeneous neural networks: Sparse and dense feature learning. 2026. URL https://api.semanticscholar.org/CorpusID:288654339.

Tom Jacobs, Chao Zhou, and Rebekka Burkholz. Mirror, mirror of the flow: How does regularization shape implicit bias? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=MLiR9LS5PW.

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

Nikita Karagodin, Shu Ge, Yury Polyanskiy, and Philippe Rigollet. Normalization in attention dynamics. In *The Thirteenth Annual Conference on Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2510.22026.

Chris Kolb, Tobias Weber, Bernd Bischl, and David Rügamer. Deep weight factorization: Sparse learning through the lens of artificial symmetries. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vNdOHr7mn5.

Chris Kolb, Christian L. Müller, Bernd Bischl, and David Rügamer. Smoothing the edges: Smooth optimization for sparse regularization using hadamard overparametrization. *Machine Learning*, 115(4), April 2026. ISSN 1573-0565. doi: 10.1007/s10994-026-06997-0. URL http://dx.doi.org/10.1007/s10994-026-06997-0.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.

Daniel Kunin, Allan Raventós, Clémentine Dominé, Feng Chen, David Klindt, Andrew Saxe, and Surya Ganguli. Get rich quick: exact solutions reveal how unbalanced initializations promote rapid feature learning, 2024. URL https://arxiv.org/abs/2406.06158.

Susanna Lange, Kyle Helfrich, and Qiang Ye. Batch normalization preconditioning for neural network training. *Journal of Machine Learning Research*, 23(72):1–41, 2022. URL http://jmlr.org/papers/v23/20-1135.html.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. ngpt: Normalized transformer with representation learning on the hypersphere, 2025. URL https://arxiv.org/abs/2410.01131.

Sibylle Marcotte, Rémi Gribonval, and Gabriel Peyr'e. Abide by the law and follow the flow: Conservation laws for gradient flows. *ArXiv*, abs/2307.00144, 2023. URL https://api.semanticscholar.org/CorpusID:259316537.

Sibylle Marcotte, Rémi Gribonval, and Gabriel Peyré. Keep the momentum: Conservation laws beyond euclidean gradient flows, 2024. URL https://arxiv.org/abs/2405.12888.

Sibylle Marcotte, Rémi Gribonval, and Gabriel Peyré. Transformative or conservative? conservation laws for resnets and transformers, 2025. URL https://arxiv.org/abs/2506.06194.

Nimrah Mustafa and Rebekka Burkholz. Dynamic rescaling for training gnns. *Advances in Neural Information Processing Systems 37*, 2024a. URL https://api.semanticscholar.org/CorpusID:276318335.

Nimrah Mustafa and Rebekka Burkholz. GATE: How to keep out intrusive neighbors. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=Sjv5RcqfuH.

Nimrah Mustafa, Aleksandar Bojchevski, and Rebekka Burkholz. Are GATs out of balance? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=qY7UqLoora.

Emmy Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1918:235–257, 1918.

Aleksandra Nowak, Łukasz Gniecki, Filip Szatkowski, and Jacek Tabor. Sparser, better, deeper, stronger: Improving sparse training with exact orthogonal initialization, 2024. URL https://openreview.net/forum?id=3mY9aGiMn0.

Peter J. Olver. *Applications of Lie Groups to Differential Equations*, volume 107 of *Graduate Texts in Mathematics*. Springer, New York, 2 edition, 1993.

Rahul Parhi and Robert D. Nowak. Deep learning meets sparse regularization: A signal processing perspective. *IEEE Signal Process. Mag.*, 40(6):63–74, 2023. doi: 10.1109/MSP.2023.3286988. URL https://doi.org/10.1109/MSP.2023.3286988.

Scott Pesme, Loucas Pillaud-Vivien, and Nicolas Flammarion. Implicit bias of SGD for diagonal linear networks: a provable benefit of stochasticity. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=vvi7KqHQiA.

Ilan Price, Nicholas Daultry Ball, Adam Jones, Samuel Chun Hei Lam, and Jared Tanner. Deep neural network initialization with sparsity inducing activations. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35082–35108, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/96d328a1f6d8396d8c8a62f2beee252a-Paper-Conference.pdf.

Xingyu Qu and Samuel Horváth. Vanishing feature: Diagnosing model merging and beyond. In Beidi Chen, Shijia Liu, Mert Pilanci, Weijie Su, Jeremias Sulam, Yuxiang Wang, and Zhihui Zhu (eds.), *Conference on Parsimony and Learning*, volume 280 of *Proceedings of Machine Learning Research*, pp. 1051–1086. PMLR, 24–27 Mar 2025. URL https://proceedings.mlr.press/v280/qu25a.html.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Thiziri Nait Saada, Alireza Naderi, and Jared Tanner. Mind the gap: a spectral analysis of rank collapse and signal propagation in attention layers. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sRKtbGsebH.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018a. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *The Thirty-second Annual Conference on Neural Information Processing Systems*, 2018b.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120, 2013. URL https://api.semanticscholar.org/CorpusID:17272965.

Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining, 2025. URL https://arxiv.org/abs/2509.01440.

Viet-Hoang Tran, Vinh Khanh Bui, Tan Lai Ngoc, Nam Nguyen, Tuan Quang Dam, and Tan Minh Nguyen. Conservation laws for modern neural architectures. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ay4Q69fAJL.

Shuyang Wang and Diego Klabjan. A mirror descent perspective of smoothed sign descent. In *Conference on Uncertainty in Artificial Intelligence*, 2024. URL https://api.semanticscholar.org/CorpusID:273482612.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In Jacob Abernethy and Shivani Agarwal (eds.), *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 3635–3673. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *The Thirty-seventh International Conference on Machine Learning*, 2020.

Can Yaras, Peng Wang, Zhihui Zhu, Laura Balzano, and Qing Qu. Neural collapse with normalized features: A geometric analysis over the riemannian manifold. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 11547–11560. Curran Associates, Inc., 2022. doi: 10.52202/068431-0839. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/4b3cc0d1c897ebcf71aca92a4a26ac83-Paper-Conference.pdf.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2018.

Chao Zhou, Tom Jacobs, Advait Gadhikar, and Rebekka Burkholz. Pay attention to small weights. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=XKnOA7MhCz.

Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization, 2025. URL https://arxiv.org/abs/2503.10622.

A Preliminary Definitions

In this section we recall some standard optimization and deep learning definitions that we will use throughout the paper.

Definition A.1 (LayerNorm)**.** Let $\gamma,\beta\in\mathbb{R}^d$ and $\epsilon\geq 0$. For a batch $H\in\mathbb{R}^{b\times d}$, the LayerNorm operation $\mathrm{LN}_{\gamma,\beta}:\mathbb{R}^{b\times d}\to\mathbb{R}^{b\times d}$ is defined componentwise as

$$\mathrm{LN}_{\gamma,\beta}(H)_{i,j}:=\gamma_j\frac{H_{i,j}-\bar{H}_i}{\sqrt{\hat{\sigma}_i^2+\epsilon}}+\beta_j,$$

where $\bar{H}_i:=\frac{1}{d}\sum_{j=1}^d H_{i,j}$, and $\hat{\sigma}_i^2:=\frac{1}{d}\sum_{j=1}^d\left(H_{i,j}-\bar{H}_i\right)^2$.

Definition A.2 (BatchNorm)**.** Let $\gamma,\beta\in\mathbb{R}^d$ and $\epsilon\geq 0$. For a batch $H\in\mathbb{R}^{b\times d}$, the BatchNorm operation $\mathrm{BN}_{\gamma,\beta}:\mathbb{R}^{b\times d}\to\mathbb{R}^{b\times d}$ is defined componentwise by

$$\mathrm{BN}_{\gamma,\beta}(H)_{i,j}:=\gamma_j\frac{H_{i,j}-\mu_j^{\mathrm{B}}}{\sqrt{(\sigma_j^{\mathrm{B}})^2+\epsilon}}+\beta_j,$$

where $\mu_j^{\mathrm{B}}:=\frac{1}{b}\sum_{i=1}^b H_{i,j}$, and $(\sigma_j^{\mathrm{B}})^2:=\frac{1}{b}\sum_{i=1}^b\left(H_{i,j}-\mu_j^{\mathrm{B}}\right)^2$.

Definition A.3 (Polyak-Łojasiewicz inequality)**.** Let $\tau>0$ be a constant and $\mathcal{L}\in C^1(\mathbb{R}^m,\mathbb{R})$ then we say $\mathcal{L}$ satisfies the PL-inequality if for all $\theta\in\mathbb{R}^m$:

$$||\nabla\mathcal{L}(\theta)||_2^2\geq 2\tau(\mathcal{L}(\theta)-\mathcal{L}^*),$$

where $\mathcal{L}^*=\inf_{\theta\in\mathbb{R}^m}\mathcal{L}(\theta)$.

B IMBALANCE UNDER STANDARD INITALIZATIONS

Here we provide a list of the resulting expected imbalance from standard initalization and compare to resulting balance without normalization in between. Recall the notation of the main text $W_2 \in \mathbb{R}^{m\times d}$ and $W_1 \in \mathbb{R}^{d\times n}$. For a two layer neural network $W_2\sigma(W_1x)$ with input $x \in \mathbb{R}^n$ and homogeneous activation. We have the balance equation per neuron as in Theorem 2.1 of Du et al. (2018):

$$\|W_{2,\cdot j,t}\|_2^2 - \|W_{1,j\cdot,t}\|_2^2 = \|W_{2,\cdot j,0}\|_2^2 - \|W_{1,j\cdot,0}\|_2^2 =: \delta_j.$$

This allows us to compare δ_j and C_j under standard initialization schemes. We first provide a general calculation split into moment calculations in Lemma B.1 and the application to the respective balance equations in Corollary B.2. For the standard initialization schemes we report all of them in Tables 1 and 2. We observe as shown in Corollary B.2 there is a variance reduction when normalization is applied. We provide a more general statement in Lemma B.3. Moreover, while under standard initialization the expected balance becomes zero for square matrices i.e. $m=d=n$, with normalization the expected balance in expectation is also zero under LeCun initialization but is 1 under both Glorot and He initialization.

Lemma B.1. *Denote the element wise variance of a Gaussian with $\sigma_2^2 = Var(W_2)$ and $\sigma_1^2 = Var(W_1)$ and let the element wise expectation be zero i.e. $\mathbb{E}[W_1]=0$ and $\mathbb{E}[W_2]=0$. Then we have:*

$$\mathbb{E}[\|W_{2,\cdot j,0}\|_2^2] = m\sigma_2^2 \quad \textit{and} \quad \mathbb{E}[\|W_{1,j\cdot,0}\|_2^2] = n\sigma_1^2.$$

Moreover, for variance:

$$Var(\|W_{2,\cdot j,0}\|_2^2) = 2m\sigma_2^4 \quad \textit{and} \quad Var(\|W_{1,j\cdot,0}\|_2^2) = 2n\sigma_1^4.$$

Proof: As the squared L_2 norm is independently summable, the first part follows directly from summing independent i.i.d random variables and the definition. For the fourth moments we have:

$$\mathbb{E}[\|W_{2,\cdot j,0}\|_2^4] = \mathbb{E}[\sum_i w_{2,i,j}^4 + \sum_{i\neq k} w_{2,i,j}^2 w_{2,k,j}^2] = 3m\sigma_2^4 + m(m-1)\sigma_2^4 = m(m+2)\sigma_2^4.$$

Now the variance follows:

$$\begin{aligned}\mathrm{Var}(\|W_{2,\cdot j,0}\|_2^2) &= \mathbb{E}[\|W_{2,\cdot j,0}\|_2^4] - \mathbb{E}[\|W_{2,\cdot j,0}\|_2^2]^2 \\ &= m(m+2)\sigma_2^4 - m^2\sigma_2^4 = 2m\sigma_2^4\end{aligned}$$

and similarly for W_1,

$$\mathbb{E}[\|W_{1,j\cdot,0}\|_2^4] = \mathbb{E}[\sum_i w_{1,j,i}^4 + \sum_{i\neq k} w_{1,j,i}^2 w_{1,j,k}^2] = 3n\sigma_1^4 + n(n-1)\sigma_1^4 = n(n+2)\sigma_1^4,$$

and the variance is now given by

$$\begin{aligned}\mathrm{Var}(\|W_{1,j\cdot,0}\|_2^2) &= \mathbb{E}[\|W_{1,j\cdot,0}\|_2^4] - \mathbb{E}[\|W_{1,j\cdot,0}\|_2^2]^2 \\ &= n(n+2)\sigma_1^4 - n^2\sigma_1^4 = 2n\sigma_1^4\end{aligned}$$

Moreover, we used that $\mathbb{E}[w_{\ell,i,j}^4] = 3\sigma_\ell^4$ for $\ell \in \{1,2\}$ for Gaussian random variables. This completes the proof.

■

Corollary B.2. *In the same setting as Lemma B.1, we have that*

$$\mathbb{E}[\delta_j] = m\sigma_2^2 - n\sigma_1^2 \qquad \textit{and } \mathbb{E}[C_j] = m\sigma_2^2 - 1$$

and

$$Var(\delta_j) = 2m\sigma_2^4 + 2n\sigma_1^4 \textit{ and } Var(C_j) = 2m\sigma_2^4$$

Proof: The expectation calculations follow directly. For the variance we get by variance calculation rules

$$\begin{aligned}\mathrm{Var}(\delta_j) &= \mathrm{Var}(\|W_{2,\cdot j,0}\|_2^2 - \|W_{1,j\cdot,0}\|_2^2) \\ &= \mathrm{Var}(\|W_{2,\cdot j,0}\|_2^2) + \mathrm{Var}(\|W_{1,j\cdot,0}\|_2^2) \\ &= 2m\sigma_2^4 + 2n\sigma_1^4\end{aligned}$$

and similarly:

$$\begin{aligned}\text{Var}(C_j) &= \text{Var}(\|W_{2,\cdot j,0}\|_2^2 - 1) \\ &= \text{Var}(\|W_{2,\cdot j,0}\|_2^2) \\ &= 2m\sigma_2^4\end{aligned}$$

■

Lemma B.3. *(Variance reduction) Let W_1 and W_2 be random matrices with i.i.d. entries and finite second and fourth moments then the following holds for all $j \in [d]$:*

$$\textit{Var}(\delta_j) \geq \textit{Var}(C_j)$$

Proof: It follows from the definition of variance of independent variables:

$$\begin{aligned}\text{Var}(\delta_j) &= \text{Var}(\|W_{2,\cdot j,0}\|_2^2 - \|W_{1,j\cdot,0}\|_2^2) \\ &= \text{Var}(\|W_{2,\cdot j,0}\|_2^2) + \text{Var}(\|W_{2,j\cdot,0}\|_2^2) \\ &\geq \text{Var}(\|W_{2,\cdot j,0}\|_2^2) \\ &= \text{Var}(C_j)\end{aligned}$$

■

Init.	Var(W_2)	Var(W_1)	$\mathbb{E}[\delta_j]$	$Var(\delta_j)$
LeCun	$\frac{1}{d}$	$\frac{1}{n}$	$\frac{m}{d} - 1$	$2\left(\frac{m}{d^2} + \frac{1}{n}\right)$
Glorot	$\frac{2}{d+m}$	$\frac{2}{n+d}$	$2\left(\frac{m}{d+m} - \frac{n}{n+d}\right)$	$2\left[\frac{4m}{(d+m)^2} + \frac{4n}{(n+d)^2}\right]$
He	$\frac{2}{d}$	$\frac{2}{n}$	$2\left(\frac{m}{d} - 1\right)$	$8\left(\frac{m}{d^2} + \frac{1}{n}\right)$

Table 1: Expectation and variance of δ_j.

Init.	Var(W_2)	$\mathbb{E}[C_j]$	Var(C_j)
LeCun	$\frac{1}{d}$	$\frac{m}{d} - 1$	$\frac{2m}{d^2}$
Glorot	$\frac{2}{d+m}$	$\frac{2m}{d+m} - 1$	$\frac{8m}{(d+m)^2}$
He	$\frac{2}{d}$	$\frac{2m}{d} - 1$	$\frac{8m}{d^2}$

Table 2: Expectation and variance of C_j.

C Proofs of Theoretical Statements

Here we provide detailed proofs of the main statements in the paper and additional statements we referred to in the paper. The correspondence is:

- The proof of Theorem 4.1 is in C.1.
- The proof of Proposition 4.3 is in C.3.
- The proof of Proposition 4.4 is in C.4.
- The proof of Lemma 4.5 is in C.5.
- The proof of Theorem 4.6 is in C.8.

Theorem C.1 (*Theorem 4.1*). *Let (W,γ,β) from a Norm–Weight interface and let θ evolve under the gradient flow. Then for every $j\in[d]$ and all $t\geq 0$:*

$$\|W_{\cdot j,t}\|_2^2-\gamma_{j,t}^2-\beta_{j,t}^2=\|W_{\cdot j,0}\|_2^2-\gamma_{j,0}^2-\beta_{j,0}^2=:C_j.$$

Proof: Fix a feature $j\in[d]$ and let $\lambda_j>0$. Consider the per-feature scaling transformations,

$$W_{\cdot j}\mapsto\lambda_j W_{\cdot j},\qquad(\gamma_j,\beta_j)\mapsto(\gamma_j,\beta_j)/\lambda_j,\qquad\lambda_j>0. \tag{8}$$

Since

$$\mathcal{N}_{\lambda^{-1}\gamma_j,\lambda^{-1}\beta_j}(H)_{ij}=\lambda^{-1}\mathcal{N}_{\gamma,\beta}(H)_{ij},$$

positive homogeneity of ϕ gives

$$\lambda W_{kj}\phi\big(\lambda^{-1}\mathcal{N}_{\gamma,\beta}(H)_{ij}\big)=W_{kj}\phi(\mathcal{N}_{\gamma,\beta}(H)_{ij}).$$

Since the rescaling in (8) leaves the network function, and therefore the loss, unchanged, we have

$$\mathcal{L}\big(\lambda W_{\cdot j},\lambda^{-1}\gamma_j,\lambda^{-1}\beta_j\big)=\mathcal{L}(W_{\cdot j},\gamma_j,\beta_j)\qquad\text{for all }\lambda>0.$$

Setting $\lambda=e^s$ and differentiating this invariance at $s=0$ gives

$$0=\big\langle\nabla_{W_{\cdot j}}\mathcal{L},W_{\cdot j}\big\rangle-\gamma_j\partial_{\gamma_j}\mathcal{L}-\beta_j\partial_{\beta_j}\mathcal{L}. \tag{9}$$

This is the Noether identity associated with the Norm–Weight rescaling symmetry.

Now define

$$C_j(t)=\|W_{\cdot j,t}\|_2^2-\gamma_{j,t}^2-\beta_{j,t}^2.$$

Under gradient flow,

$$\dot W_{\cdot j}=-\nabla_{W_{\cdot j}}\mathcal{L},\qquad\dot\gamma_j=-\partial_{\gamma_j}\mathcal{L},\qquad\dot\beta_j=-\partial_{\beta_j}\mathcal{L}.$$

Therefore,

$$\begin{aligned}\frac{d}{dt}C_j(t)&=2\Big\langle W_{\cdot j,t},\dot W_{\cdot j,t}\Big\rangle-2\gamma_{j,t}\dot\gamma_{j,t}-2\beta_{j,t}\dot\beta_{j,t}\\&=-2\big\langle W_{\cdot j,t},\nabla_{W_{\cdot j}}\mathcal{L}\big\rangle+2\gamma_{j,t}\partial_{\gamma_j}\mathcal{L}+2\beta_{j,t}\partial_{\beta_j}\mathcal{L}\\&=-2\big[\big\langle\nabla_{W_{\cdot j}}\mathcal{L},W_{\cdot j,t}\big\rangle-\gamma_{j,t}\partial_{\gamma_j}\mathcal{L}-\beta_{j,t}\partial_{\beta_j}\mathcal{L}\big]\\&=0,\end{aligned}$$

where the last equality follows from (9). Hence $C_j(t)=C_j(0)$ for all $t\geq 0$.

■

Lemma C.2. *If $\epsilon = 0$ and $\sigma^2 > 0$ then LayerNorm and BatchNorm are scale invariant. Therefore for all $t \geq 0$ we have that $\|W_{1,t}\|_F^2 = \|W_{1,0}\|_F^2$. Moreover, for BatchNorm we have $\|w_{1\cdot j,t}\|_2^2 = \|w_{1\cdot j,0}\|_2^2$ for all $j \in [d]$.*

Proof: We use the LayerNorm and BatchNorm definitions given in Definitions A.1 and A.2. Let $H = XW_1$ denote the pre-normalization activations.

For LayerNorm, when $\epsilon = 0$, scaling H by any $\lambda > 0$ gives

$$\overline{\lambda H}_i = \lambda \bar{H}_i, \qquad \hat{\sigma}_i^2(\lambda H) = \lambda^2 \hat{\sigma}_i^2(H).$$

Therefore,

$$\frac{\lambda H_{i,j} - \lambda \bar{H}_i}{\sqrt{\lambda^2 \hat{\sigma}_i^2}} = \frac{H_{i,j} - \bar{H}_i}{\sqrt{\hat{\sigma}_i^2}},$$

and hence

$$\mathrm{LN}_{\gamma,\beta}(\lambda H) = \mathrm{LN}_{\gamma,\beta}(H).$$

Since $X(\lambda W_1) = \lambda H$, the transformation $W_1 \mapsto \lambda W_1$ leaves the network function, and therefore the loss, unchanged. Differentiating this loss invariance at $\lambda = 1$ gives the corresponding identity

$$\langle \nabla_{W_1} \mathcal{L}, W_1 \rangle_F = 0. \tag{10}$$

By (10), under gradient flow,

$$\frac{d}{dt} \|W_{1,t}\|_F^2 = 2\langle W_{1,t}, \dot{W}_{1,t} \rangle_F = -2\langle W_{1,t}, \nabla_{W_1} \mathcal{L} \rangle_F = 0,$$

so

$$\|W_{1,t}\|_F^2 = \|W_{1,0}\|_F^2 \qquad \text{for all } t \geq 0.$$

For BatchNorm, let $H = Xw_1$. Since the statistics in Definition A.2 are computed independently for each feature. Thus, for $\lambda = (\lambda_1, \ldots, \lambda_d) \in \mathbb{R}_{>0}^d$,

$$\mu_j^{\mathrm{B}}(H\mathrm{diag}(\lambda)) = \lambda_j \mu_j^{\mathrm{B}}(H), \qquad (\sigma_j^{\mathrm{B}}(H\mathrm{diag}(\lambda)))^2 = \lambda_j^2 (\sigma_j^{\mathrm{B}}(H))^2,$$

which implies

$$\mathrm{BN}_{\gamma,\beta}(H\mathrm{diag}(\lambda)) = \mathrm{BN}_{\gamma,\beta}(H).$$

Since

$$X(w_1 \mathrm{diag}(\lambda)) = H\mathrm{diag}(\lambda),$$

each column admits an independent rescaling $w_{1,\cdot j} \mapsto \lambda_j w_{1,\cdot j}$. Differentiating this loss invariance gives

$$\left\langle \nabla_{w_{1,\cdot j}} \mathcal{L}, w_{1,\cdot j} \right\rangle = 0 \qquad \text{for every } j \in [d].$$

Therefore, under gradient flow,

$$\frac{d}{dt} \|w_{1,\cdot j,t}\|_2^2 = -2\left\langle \nabla_{w_{1,\cdot j}} \mathcal{L}, w_{1,\cdot j,t} \right\rangle = 0,$$

and consequently

$$\|w_{1,\cdot j,t}\|_2^2 = \|w_{1,\cdot j,0}\|_2^2 \qquad \text{for all } j \in [d], t \geq 0.$$

■

Proposition C.3 (*Proposition 4.3*). *Let (W, γ, β) form a Norm–Weight interface and let θ evolve under gradient flow on the regularized objective $\widetilde{\mathcal{L}} = \mathcal{L} + \frac{\rho}{2}\|\theta\|_2^2$ with $\rho > 0$. Then for every $j \in [d]$,*

$$\dot{C}_j(t) = -2\rho C_j(t), \text{ hence } C_j(t) = C_j(0)e^{-2\rho t} \quad \text{for all } t \geq 0.$$

Proof: Throughout, $\nabla_{W_{\cdot j}} \mathcal{L}$, $\partial_{\gamma_j} \mathcal{L}$ and $\partial_{\beta_j} \mathcal{L}$ denote the partial derivatives of the loss with respect to the indicated block, evaluated at the current iterate θ_t; they are not derivatives with respect to t. Applying the gradient flow (1) to $\widetilde{\mathcal{L}}$ gives $\dot{\theta}_t = -\nabla_\theta \mathcal{L}(\theta_t) - \rho\theta_t$. so that $\dot{W}_{\cdot j,t} = -\nabla_{W_{\cdot j}} \mathcal{L} - \rho W_{\cdot j,t}$ and likewise for $\gamma_{j,t}$ and $\beta_{j,t}$. Differentiating the three blocks of $C_j(t) = \|W_{\cdot j,t}\|_2^2 - \gamma_{j,t}^2 - \beta_{j,t}^2$:

$$\frac{d}{dt}\|W_{\cdot j,t}\|_2^2 = 2\left\langle W_{\cdot j,t}, \dot{W}_{\cdot j,t} \right\rangle = -2\left\langle \nabla_{W_{\cdot j}} \mathcal{L}, W_{\cdot j,t} \right\rangle - 2\rho\|W_{\cdot j,t}\|_2^2,$$

$$\frac{d}{dt}\gamma_{j,t}^2 = -2\gamma_{j,t}\partial_{\gamma_j}\mathcal{L} - 2\rho\gamma_{j,t}^2, \qquad \frac{d}{dt}\beta_{j,t}^2 = -2\beta_{j,t}\partial_{\beta_j}\mathcal{L} - 2\rho\beta_{j,t}^2.$$

By the Noether identity (9) the loss contributions cancel out and we are only left with the contraction terms,

$$\dot{C}_j = \underbrace{-2\Big[\langle \nabla_{W_{\cdot j}} \mathcal{L}, W_{\cdot j}\rangle - \gamma_j \partial_{\gamma_j}\mathcal{L} - \beta_j \partial_{\beta_j}\mathcal{L}\Big]}_{=0} - 2\rho\Big(\|W_{\cdot j,t}\|_2^2 - \gamma_{j,t}^2 - \beta_{j,t}^2\Big) = -2\rho C_j(t). \tag{11}$$

Equation (11) is a linear ODE. Multiplying by the integrating factor $e^{2\rho t}$ gives

$$\frac{d}{dt}\Big(C_j(t)e^{2\rho t}\Big) = e^{2\rho t}\Big(\dot{C}_j(t) + 2\rho C_j(t)\Big) = 0,$$

so $t \mapsto C_j(t)e^{2\rho t}$ is constant and equals its value $C_j(0)$ at $t=0$. Hence $C_j(t) = C_j(0)e^{-2\rho t}$ for every initial condition, including $C_j(0) = 0$.

■

Proposition C.4 (*Proposition 4.4*)**.** *Let (W,γ,β) form a Norm–Weight interface. For feature j, let $a_j := (\gamma_j, \beta_j)$ and define its affine magnitude $r_j := \|a_j\|_2$, with effective weight $v_j := r_j W_{\cdot j} \in \mathbb{R}^p$, which is invariant under the Norm–Weight rescaling. The minimum weight decay cost of the interface over all per-feature rescalings is*

$$\inf_{\lambda_1,\dots,\lambda_d} \frac{\rho}{2}\sum_{j=1}^{d}\big(\lambda_j^2\|W_{\cdot j}\|_2^2 + \lambda_j^{-2} r_j^2\big) = \rho\sum_{j=1}^{d}\|v_j\|_2 =: \rho\|V\|_{2,1}. \tag{12}$$

The minimum for feature j is attained at $\lambda_j^\star = \Big(\frac{r_j^2}{\|W_{\cdot j}\|_2^2}\Big)^{1/4}$, hence $C_j^{(\lambda_j^\star)} = 0$.

Proof: The argument in an application of variational characterizations for the ℓ_2 regularization under multiplicative parameterizations, where minimizing a smooth quadratic penalty over a redundant factorization induces an $\ell_{2,1}$-type penalty on the effective parameter (Kolb et al., 2026; Parhi & Nowak, 2023; Kolb et al., 2025). In our case, the factorization arises naturally from the rescaling symmetry of a Norm–Weight interface.

■

Lemma C.5 (*Lemma 4.8*)**.** *Let (W,γ,β) form a Norm–Weight interface with $\beta = 0$, and for each $j \in [d]$ let $\Pi_{\cdot j} := \gamma_j W_{\cdot j} \in \mathbb{R}^p$ be the effective parameter of feature j. Under gradient flow,*

$$\dot{\Pi}_{\cdot j} = -\underbrace{\big(\gamma_j^2 I_p + W_{\cdot j} W_{\cdot j}^\top\big)}_{=:M_j} \nabla_{\Pi_{\cdot j}} \mathcal{L}.$$

Thus $\Pi_{\cdot j}$ follows a preconditioned gradient flow with preconditioner M_j. The preconditioner scales every direction orthogonal to $W_{\cdot j}$ by γ_j^2, and the direction parallel to $W_{\cdot j}$ by $\gamma_j^2 + \|W_{\cdot j}\|_2^2$. When M_j is positive definite, this can equivalently be viewed as a Riemannian gradient flow with metric tensor M_j^{-1} The statement holds for LayerNorm and BatchNorm alike and for any $\epsilon \geq 0$.

Proof: Since $\mathcal{L}$ depends on $(W_{\cdot j}, \gamma_j)$ only through $\Pi_{\cdot j}$, let $G := \nabla_{\Pi_{\cdot j}}\mathcal{L}$ then chain rule gives $\nabla_{W_{\cdot j}}\mathcal{L} = \gamma_j G$ and $\partial_{\gamma_j} = \langle W_{\cdot j}, G\rangle$. Hence,

$$\dot{\Pi}_{\cdot j} = \dot{\gamma}_j W_{\cdot j} + \gamma_j \dot{W}_{\cdot j} = -\langle W_{\cdot j}, G\rangle W_{\cdot j} - \gamma_j^2 G = -\big(\gamma_j^2 I_p + W_{\cdot j} W_{\cdot j}^\top\big) G.$$

Thus the effective parameter follows a preconditioned gradient flow with

$$M_j = \gamma_j^2 I_p + W_{\cdot j} W_{\cdot j}^\top.$$

The matrix $W_{\cdot j} W_{\cdot j}^\top$ is rank one. It has eigenvalue $\|W_{\cdot j}\|_2^2$ along $W_{\cdot j}$ and zero in orthogonal directions. Hence, M_j has eigenvalue $\gamma_j^2 + \|W_{\cdot j}\|_2^2$ once and γ_j^2 with multiplicity $p-1$.

■

Lemma C.6. *Under gradient flow on (w,γ), the effective parameter obeys*

$$\dot{\pi} = -M(\pi,C)\nabla L(\pi), \qquad M(\pi,C) = \operatorname{diag}\left(\sqrt{C_j^2+4\pi_j^2}\right)_{j\in[d]}.$$

Equivalently, π follows Riemannian gradient flow for the metric tensor M^{-1}.

Proof: Write $g_j = \partial\mathcal{L}/\partial\pi_j$. Since $\mathcal{L}$ depends on (w,γ) only through π, the chain rule gives $\partial\mathcal{L}/\partial w_j = \gamma_j g_j$ and $\partial\mathcal{L}/\partial\gamma_j = w_j g_j$. Hence, $\dot{w}_j = -\gamma_j g_j, \dot{\gamma}_j = -w_j g_j$ and

$$\dot{\pi}_j = \dot{w}_j\gamma_j + w_j\dot{\gamma}_j = -(\gamma_j^2+w_j^2)g_j.$$

Since $w_j^2-\gamma_j^2 = C_j$ and $w_j^2\gamma_j^2 = \pi_j^2$,

$$(w_j^2+\gamma_j^2)^2 = (w_j^2-\gamma_j^2)^2+4w_j^2\gamma_j^2 = C_j^2+4\pi_j^2,$$

and $w_j^2+\gamma_j^2>0$, so $w_j^2+\gamma_j^2 = \sqrt{C_j^2+4\pi_j^2}$.

■

Corollary C.7. *For every $j\in[d]$ and $t\geq 0$,*

$$M_{jj} = \sqrt{C_j^2+4\pi_j^2} \geq \max\{|C_j|, 2|\pi_j|\}, \qquad M \succeq C_{\min} I := \min_j |C_j|.$$

Proof: Since $C_j^2, 4\pi_j^2 \geq 0$,

$$\sqrt{C_j^2+4\pi_j^2} \geq |C_j|, \qquad \sqrt{C_j^2+4\pi_j^2} \geq 2|\pi_j|.$$

Because M is diagonal, $M_{jj} \geq C_{\min}$ for every j, which gives $M \succeq C_{\min} I$. As we established that C_j is conserved, this lower bound holds uniformly throughout training. At balance, $C_j = 0$ and hence $M_{jj} = 2|\pi_j|$, which vanishes as $\pi_j \to 0$. Thus, near the origin, a balanced parameterization can have an arbitrarily small preconditioner.

■

Theorem C.8 (*Theorem 4.6*)**.** *Suppose $\mathcal{L}$ satisfies the Polyak-Łojasiewicz inequality in π with constant τ (Definition A.3) and let $C_{\min} := \min_j |C_j|$. Then under gradient flow on (w,γ):*

$$\mathcal{L}(\pi_t) - \mathcal{L}^* \leq (\mathcal{L}(\pi_0)-\mathcal{L}^*)e^{-2\tau C_{\min} t}, \qquad \pi_0 = \pi_{\text{init}} \text{ and } t \geq 0.$$

Proof: Set $E(t) = \mathcal{L}(\pi_t) - \mathcal{L}^* \geq 0$. Since $\mathcal{L}^*$ is constant, by the chain rule and Lemma C.6,

$$\dot{E} = \frac{d}{dt}(\mathcal{L}(\pi_t)-\mathcal{L}^*) = \langle \nabla\mathcal{L}, \dot{\pi}\rangle = -\nabla\mathcal{L}^\top M \nabla\mathcal{L}.$$

Since M is diagonal with positive entries,

$$\begin{aligned}
\nabla\mathcal{L}^\top M\nabla\mathcal{L} &= \sum_{j=1}^{d} M_{jj}(\partial_j\mathcal{L})^2 \\
&\geq (\min_j M_{jj})\|\nabla\mathcal{L}\|^2 \\
&\geq C_{\min}\|\nabla\mathcal{L}\|^2 \quad \text{by Corollary C.7.}
\end{aligned}$$

Hence,

$$\begin{aligned}
\dot{E} &\leq -C_{\min}\|\nabla\mathcal{L}\|^2 \\
&\leq -2\tau C_{\min} E \quad \text{by Definition A.3}
\end{aligned}$$

Then,

$$\begin{aligned}
\frac{d}{dt}\left(Ee^{2\tau C_{\min} t}\right) &= e^{2\tau C_{\min} t}(\dot{E}+2\tau C_{\min} E) \quad \text{by the product rule} \\
&\leq 0.
\end{aligned}$$

So, $Ee^{2\tau C_{\min} t}$ is non-increasing $\Rightarrow Ee^{2\tau C_{\min} t} \leq E(0)e^{2\tau C_{\min} 0} = E(0)$.

Therefore, $E(t) \leq E(0)e^{-2\tau C_{\min} t}$.

■

Remark C.9. The restriction $p = 1$ in Theorem C.8 is only needed for an arbitrary-sign bound. For general p, if $C_j < 0$ for all j, then

$$M_j = \gamma_j^2 I_p + W_{\cdot j} W_{\cdot j}^\top \succeq \gamma_j^2 I_p \succeq |C_j| I_p,$$

since

$$C_j = \|W_{\cdot j}\|_2^2 - \gamma_j^2 < 0.$$

Hence, the same PL argument yields the convergence rate of Theorem C.8 with

$$C_{\min} := \min_j |C_j|.$$

D Details on Convolutional Network experiments

Selecting the optimal λ. We first fix the best-performing baseline at $\lambda = 1$. We then identify a lower bound that constrains the search space, and perform a uniform sweep over values beyond this bound. For convolutional networks, improved performance occurs for $\lambda > 1$, so the sweep is carried out over progressively larger values of λ. The final value is selected according to test performance.

Hyperparameters. For both convolutional settings we follow the standard training recipes, summarized in Table 3, and verified that our unscaled baselines reach the accuracy expected of these recipes: $\approx$67-68% top-1 for ResNet20 on CIFAR-100 and $\approx$75-76% top-1 for ResNet50 on ImageNet.

For every architecture we use exactly the same hyperparameters for the standard initialization ($\lambda = 1$), λ-balanced initialization, and our proposed initialization with the optimal λ. The three runs therefore differ only in the rescaling of Eq. (6) applied once at initialization, which leaves the initial network function unchanged, so that any difference in training is attributed to the initial imbalance alone. For ResNet20 on CIFAR-100 our runs are averaged over 2 unique seeds.

Table 3: Training hyperparameters for the convolutional networks. The same values are used for the standard, λ-balanced, and the proposed initialization with optimal λ.

	ResNet-20 CIFAR-100	ResNet-50 ImageNet
Optimizer	SGD + momentum	SGD + momentum
Momentum	0.9	0.9
Batch size	256	1024 (4×256)
Epochs	250	100
Peak learning rate	0.1	0.25
Learning rate schedule	Triangular	Triangular
Warmup	10 epochs	5 epochs
Weight decay	5×10^{-4}	5×10^{-5}
Loss	cross-entropy	Label smoothing loss

Additional results. We provide conserved-quantity trajectories and the corresponding relative parameter updates for our proposed initialization, the standard initialization ($\lambda = 1$), and the λ-balanced initialization (λ_{bal}). Results are shown for ResNet-20 on CIFAR-100 in Figure 7 and for ResNet-50 on ImageNet in Figure 9. We additionally report ResNet-50/ImageNet test performance under the optimal λ rescaling in Figure 8.

E Details on Transformer experiments

Selecting the optimal λ. After fixing the best-performing baseline at $\lambda = 1$, we first perform a coarse sweep over a wide range of values to identify the direction in which performance improves. For the Transformer models, this consistently occurs for $\lambda < 1$. We then determine a lower bound on λ, typically where further decreasing the scale no longer improves performance, and refine the search between this lower bound and $\lambda = 1$. The final value of λ is selected according to the test performance.

Transformers are usually trained with adaptive optimizers, so there is no standard SGD recipe to adopt, and we tuned one for each model before running any initialization experiments. In general, our goal was to bring the SGD baseline as close to the standard AdamW baselines. The resulting hyperparameters are summarized in Table 4.

DeiT-Tiny on CIFAR-100. We swept the peak learning rate over $\{0.01, 0.05, 0.09, 0.1, 0.3, 0.5, 0.7, 0.9\}$ with the standard initialization and selected the best-performing value. Our runs are averaged over 2 unique seeds.

GPT-2 Small on SlimPajama-6B. We train a GPT-2 Small model (Radford et al., 2019) ($n_{\text{layer}} = 12$, `n_head`$= 12$, `n_embd`$= 768$, $\sim$ 124M parameters with tied input/output embeddings) on the SlimPajama-6B subset using the GPT-2 BPE tokenizer (vocabulary size 50304). The code used

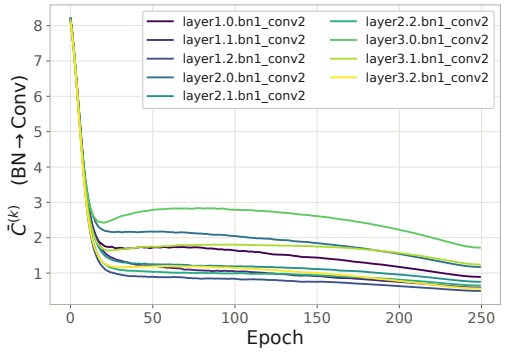
is based mainly on the repository by Semenov et al. (2025).We swept the peak learning rate over $\{0.005, 0.009, 0.01, 0.03, 0.07, 0.1, 0.3, 0.5, 0.7, 0.9\}$ and the weight decay over $\{0, 5\times10^{-5}, 10^{-4}, 4\times10^{-4}\}$, and selected a learning rate of 0.3 with no weight decay. We use a micro-batch of 32 sequences of length 512 with 4 steps of gradient accumulation, giving 128 sequences, or $65{,}536$ tokens, per optimizer step. Training runs for $25{,}000$ iterations, approximately 1.64 billion tokens in total.

For every architecture we use exactly the same hyperparameters for the standard initialization ($\lambda = 1$), λ-balanced initialization, and the proposed initialization with the optimal λ. Our runs are averaged over 3 unique seeds.

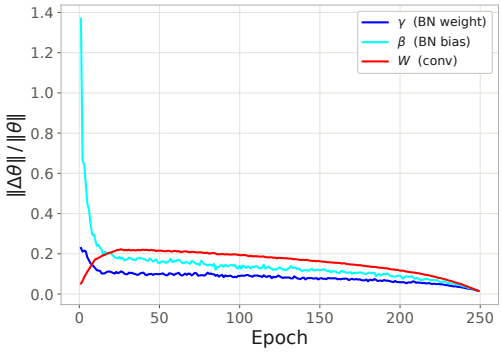
Table 4: Training hyperparameters for the Transformers. The same values are used for the standard, rescaled, and λ-balanced initializations.

	DeiT-Tiny CIFAR-100	GPT-2 Small SlimPajama-6B
Optimizer	SGD + momentum	SGD + momentum
Momentum	0.9	0.95
Batch size	256 images	128 sequences (32×4)
Sequence length	—	512
Training length	300 epochs	25,000 iterations ($\approx$ 1.64B tokens)
Peak learning rate	0.1	0.3
Learning rate schedule	triangular	onecyle + cosine
Warmup	10 epochs	3000 steps
Weight decay	10^{-4}	0

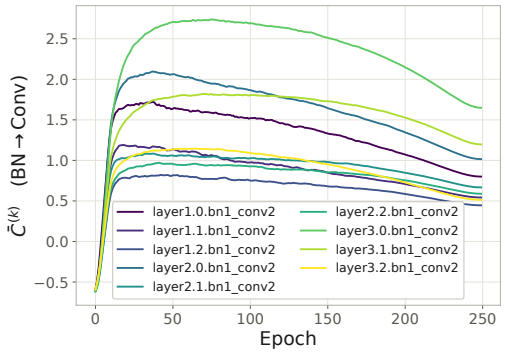
Additional results. For DeiT-Tiny on CIFAR-100, we report conserved-quantity trajectories for both the Attention and MLP Norm–Weight interfaces under the respective initialization scalings in Figure 10. The corresponding relative parameter updates for the attention and MLP parameter pairs are shown in Figure 11. For GPT-2 Small on SlimPajama-6B, we additionally find that the proposed initialization yields faster convergence than the standard initialization, consistently across three seeds, as shown in Figure 12.



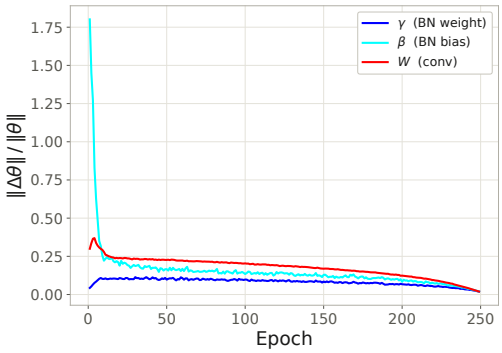
(a) ResNet20: conserved quantity ($\lambda = 5$).



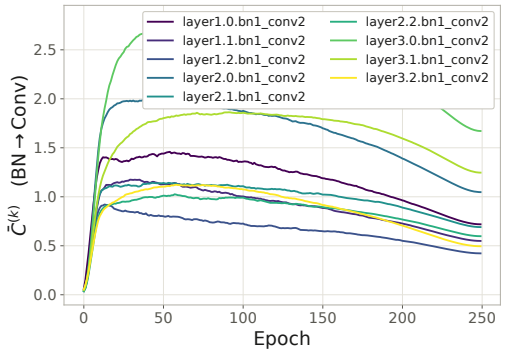
(b) Resnet20: relative parameter speed ($\lambda = 5$).



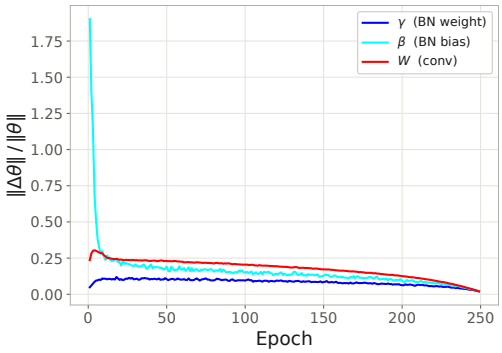
(c) ResNet20: conserved quantity ($\lambda = 1$).



(d) ResNet20: relative parameter speed ($\lambda = 1$).



(e) ResNet20: conserved quantity ($\lambda_{\text{bal}} \approx 1.32$).



(f) ResNet20: relative parameter speed ($\lambda_{\text{bal}} \approx 1.32$).

Figure 7: **ResNet20 on CIFAR-100: conserved quantities with corresponding relative parameter speed with the respective initializations**. Left column: at the $\mathrm{BN}_1 \to \texttt{conv2}$ Norm–Weight interface, the only such interface in a BasicBlock: BN_2 feeds the residual addition and has no single downstream convolution. For each residual block $k \in [9]$ (three stages of three blocks) we plot the feature-wise mean $\bar{C}^{(k)}(t) = \frac{1}{d_k}\sum_{j=1}^{d_k} C_j^{(k)}(t)$, where j indexes the input channels of $\texttt{conv2}$, $C_j = \|W_{:,j,:,:}\|_2^2 - \gamma_j^2 - \beta_j^2$, and $d_k \in \{16, 32, 64\}$ is the stage width. (Right column): we plot the relative parameter speed $s_\theta(t) = \|\theta_t - \theta_{t-1}\|_2 / \|\theta_{t-1}\|_2$, the per-epoch relative displacement of W, γ and β, averaged over all 9 Norm–Weight interfaces.

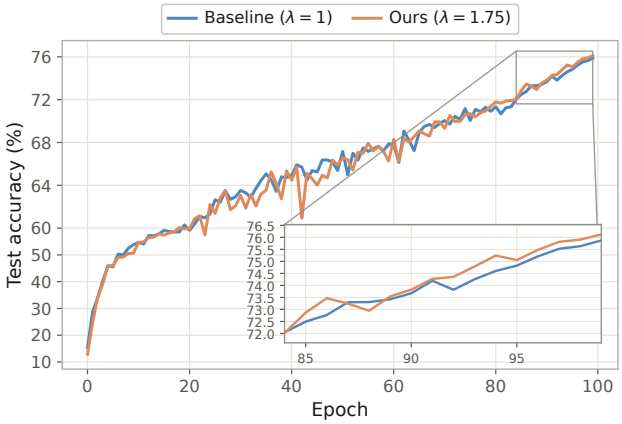
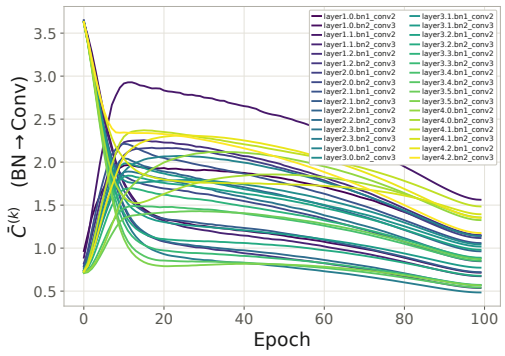
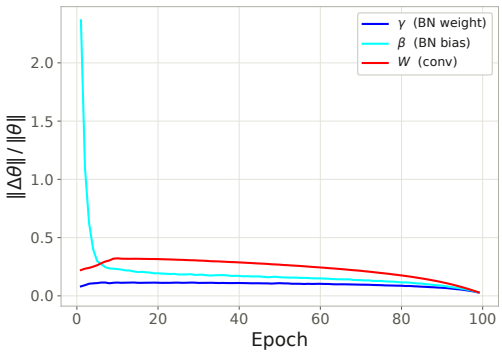


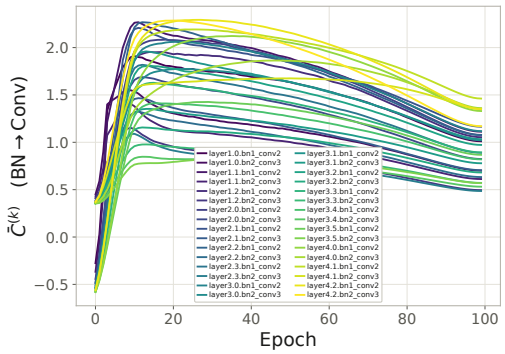
Figure 8: ResNet50 on ImageNet: top 1 accuracy. Comparing the proposed initialization with $\lambda = 1.75$ vs. the standard initialization ($\lambda = 1$).



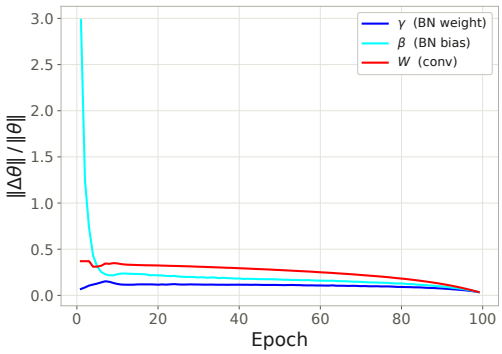
(a) ResNet50: conserved quantity ($\lambda = 1.75$).



(b) ResNet50: relative parameter speed ($\lambda = 1.75$).

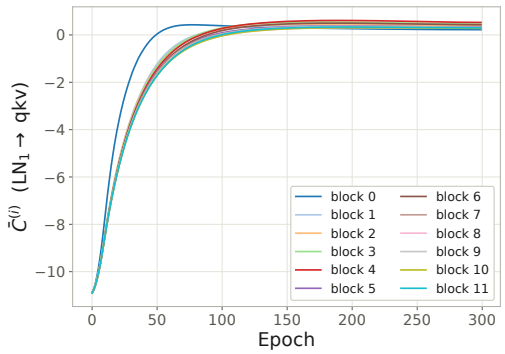


(c) ResNet50: conserved quantity ($\lambda = 1$).

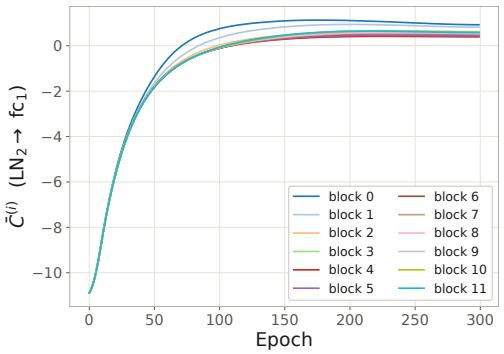


(d) ResNet50: relative parameter speed ($\lambda = 1$).

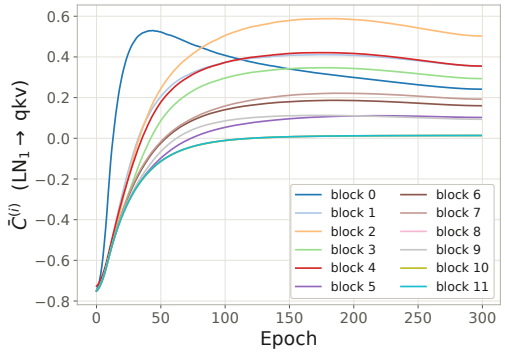
Figure 9: **ResNet50 on ImageNet: conserved quantities with corresponding relative parameter speed with the respective initializations.** Left column: at the two Norm–Weight interfaces of each bottleneck: $\mathrm{BN}_1 \to \texttt{conv2}$ (3×3) in (a) and $\mathrm{BN}_2 \to \texttt{conv3}$ (1×1) in (b); BN_3 feeds the residual addition and is excluded. For each bottleneck $k \in [16]$ we plot the feature-wise mean $\bar{C}^{(k)}(t) = \frac{1}{d_k} \sum_{j=1}^{d_k} C_j^{(k)}(t)$, where j indexes the input channels of the downstream convolution and $d_k \in \{64,128,256,512\}$ is the stage width, giving 32 interfaces in total. (Right column): we plot the relative parameter speed $s_\theta(t) = \|\theta_t - \theta_{t-1}\|_2 / \|\theta_{t-1}\|_2$, the per-epoch relative displacement of W, γ and β, averaged over all 32 Norm–Weight interfaces.



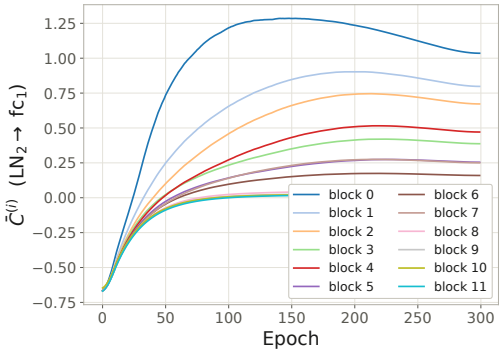
(a) DeiT-Tiny: conserved quantity Attn. ($\lambda = 0.3$).



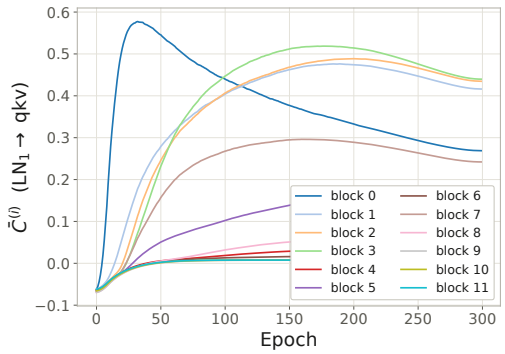
(b) DeiT-Tiny: conserved quantity MLP ($\lambda = 0.3$).



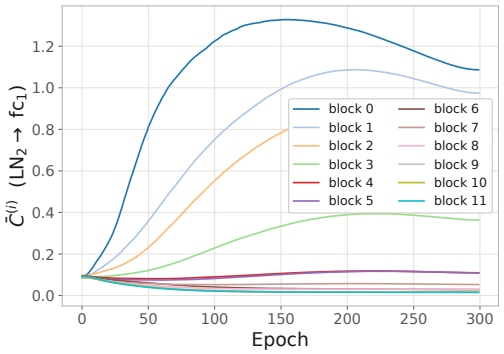
(c) DeiT-Tiny: conserved quantity Attn ($\lambda = 1$).



(d) DeiT-Tiny: conserved quantity MLP ($\lambda = 1$).

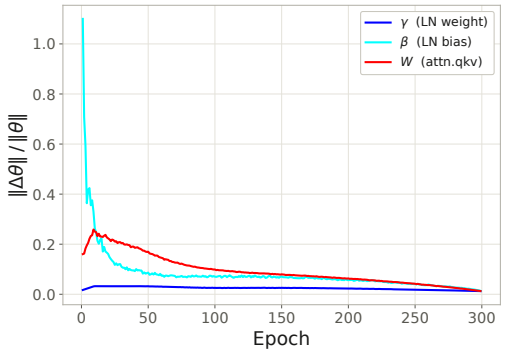


(e) DeiT-Tiny: conserved quantity Attn ($\lambda_{\text{bal}} \approx 1.39$).

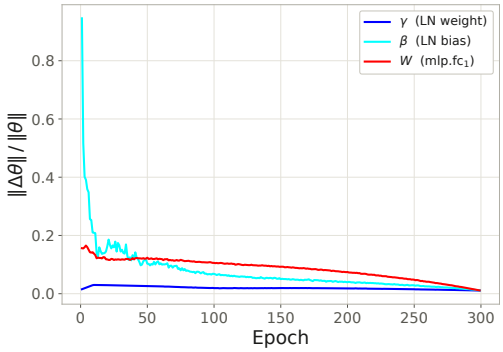


(f) DeiT-Tiny: conserved quantity MLP ($\lambda_{\text{bal}} \approx 1.39$).

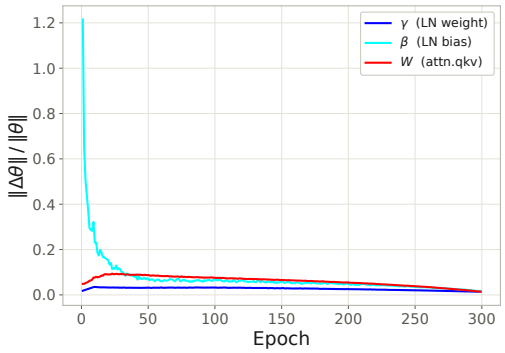
Figure 10: **Conserved quantities for DeiT-Tiny on CIFAR-100 across initialization scalings λ.** Left column: the attention (i.e. "Attn") interface $\mathrm{LN}_1 \to \mathtt{qkv}$. Right column: the corresponding MLP interface $\mathrm{LN}_2 \to \mathtt{fc}_1$. For each Transformer block $i \in \{0,\ldots,11\}$ we plot the feature-wise mean $\bar{C}^{(i)}(t) = \frac{1}{d}\sum_{j=1}^{d} C_j^{(i)}(t)$ with $d = 192$, so each curve is one block. The rescaling sets the initial offset: since $C_j^{\lambda}(t) = \lambda^2 \|W_{\cdot j,t}\|_2^2 - (\gamma_{j,t}^2 + \beta_{j,t}^2)/\lambda^2$ is strictly increasing in $\lambda > 0$, every curve moves monotonically with λ, with $\lambda < 1$ driving $\bar{C}^{(i)}$ negative (γ, β dominant) and $\lambda > 1$ positive (W dominant). The standard initialization $\lambda = 1$ is already γ-dominant, $\bar{C}^{(i)} \approx -0.77$ and -0.69 at the two interfaces.



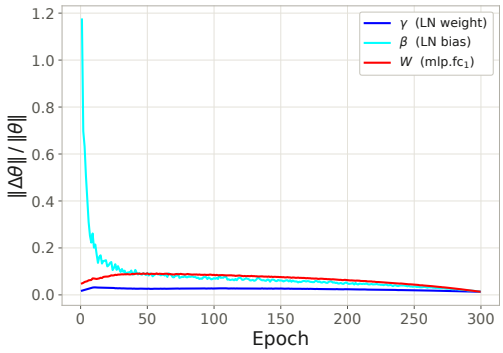
(a) DeiT-Tiny: relative parameter speed Attn ($\lambda = 0.3$).



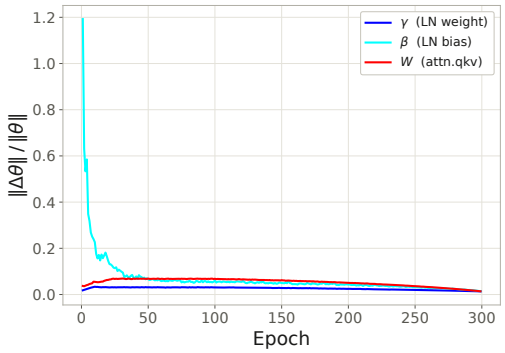
(b) DeiT-Tiny: relative parameter speed MLP ($\lambda = 0.3$).



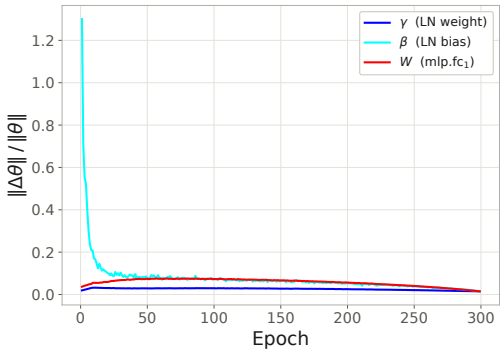
(c) DeiT-Tiny: relative parameter speed Attn ($\lambda = 1$).



(d) DeiT-Tiny: relative parameter speed MLP ($\lambda = 1$).



(e) DeiT-Tiny: relative parameter speed Attn ($\lambda_{\text{bal}} \approx 1.39$).



(f) DeiT-Tiny: relative parameter speed MLP ($\lambda_{\text{bal}} \approx 1.39$).

Figure 11: **Relative parameter speeds for DeiT-Tiny on CIFAR-100 across initialization scalings λ.** We plot $s_\theta(t) = \|\theta_t - \theta_{t-1}\|_2 / \|\theta_{t-1}\|_2$, the per-epoch relative displacement of W, γ and β, averaged over the 12 Norm–Weight interfaces of a given type: the attention (i.e. "Attn") $\mathrm{LN}_1 \to \mathtt{qkv}$ in the left column and the corresponding MLP interface $\mathrm{LN}_2 \to \mathtt{fc}_1$ in the right. The rescaling $W \mapsto \lambda W$, $(\gamma, \beta) \mapsto (\gamma, \beta)/\lambda$ leaves the network function unchanged but not its dynamics: relative speeds transform as $(s_W, s_\gamma, s_\beta) \mapsto (\lambda^{-2} s_W, \lambda^2 s_\gamma, \lambda^2 s_\beta)$. As a rule of thumb, λ scales the norms and moves relative speed the opposite way: $\lambda > 1$ makes W slow and γ, β fast, while $\lambda < 1$ makes W fast and γ, β slow.

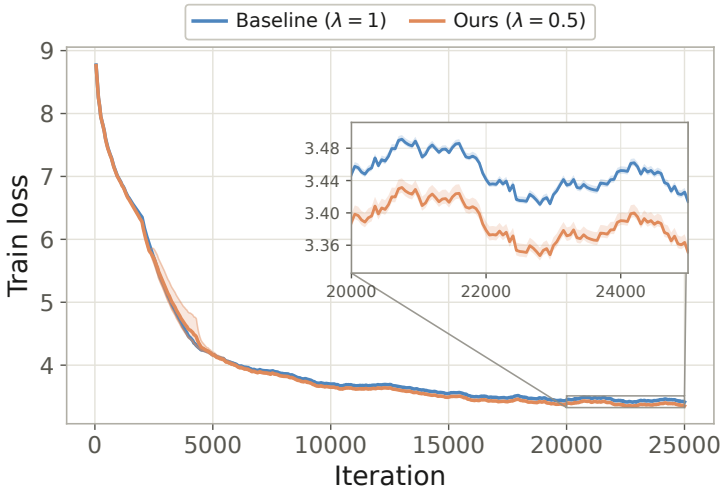


Figure 12: GPT-2 Small on Slimpajama-6B: train loss. Comparing the proposed initialization with $\lambda = 0.5$ vs. the standard initialization ($\lambda = 1$).