TRAINING INVARIANCES AND THE LOW-RANK PHENOMENON: BEYOND LINEAR NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The implicit bias induced by the training of neural networks has become a topic of rigorous study. In the limit of gradient flow and gradient descent with appropriate step size, it has been shown that when one trains a deep linear network with logistic or exponential loss on linearly separable data, the weights converge to rank-1 matrices. In this paper, we extend this theoretical result to the much wider class of nonlinear ReLU-activated feedforward networks containing fully-connected layers and skip connections. To the best of our knowledge, this is the first time a low-rank phenomenon is proven rigorously for these architectures, and it reflects empirical results in the literature. The proof relies on specific local training invariances, sometimes referred to as alignment, which we show to hold for a wide set of ReLU architectures. Our proof relies on a specific decomposition of the network into a multilinear (up to sign) function and another ReLU network whose weights are constant under a certain parameter directional convergence.

1 INTRODUCTION

Recently, great progress has been made in understanding the trajectory of gradient flow (GF) (Ji & Telgarsky, 2019; 2020; Lyu & Li, 2020), gradient descent (GD) (Ji & Telgarsky, 2019; Arora et al., 2018) and stochastic gradient descent (SGD) (Neyshabur et al., 2015; 2017) in the training of neural networks. While good theory has been developed for deep linear networks (Zhou & Liang, 2018; Arora et al., 2018), practical architectures such as ReLU fully-connected networks or ResNets are highly non-linear. This causes the underlying optimization problem (usually empirical risk minimization) to be highly non-smooth (e.g. for ReLus) and non-convex, necessitating special tools such as the Clarke subdifferential (Clarke, 1983).

One of the many exciting results of this body of literature is that gradient-based algorithms exhibit some form of implicit regularization: the optimization algorithm prefers some stationary points to others. In particular, a wide range of implicit biases has been shown in practice (Huh et al., 2021) and proven for deep linear networks (Arora et al., 2018; Ji & Telgarsky, 2019), convolutional neural networks (Gunasekar et al., 2018) and homogeneous networks (Ji & Telgarsky, 2020). One well known result for linearly separable data is that in various practical settings, linear networks converge to the solution of the hard SVM problem, i.e., a max-margin classifier (Ji & Telgarsky, 2019), while a relaxed version is true for CNNs (Gunasekar et al., 2018). This holds even when the margin does not explicitly appear in the optimization objective - hence the name implicit regularization.

Another strong form of implicit regularization relates to the structure of weight matrices of fully connected networks. In particular, Ji & Telgarsky (2019) prove that for deep *linear* networks for binary classification, weight matrices tend to rank-1 matrices in Frobenius norm as a result of GF/GD training, and that adjacent layers' singular vectors align. The max-margin phenomenon follows as a result. In practice, Huh et al. (2021) empirically document the low rank bias across different non-linear architectures. However, their results include ReLu fully-connected networks, CNNs and ReNet, which are not all covered by the existing theory. Beyond linear fully connected networks, Du et al. (2018) show vertex-wise invariances for fully-connected ReLu networks and invariances in Frobenius norm differences between layers for CNNs. Yet, despite the evidence in (Huh et al., 2021), it has been an open theoretical question how more detailed structural relations between layers, which, e.g., imply the low-rank result, generalize to other, structured or local nonlinear and possibly non-homogeneous architectures, and how to even characterize these.

Hence, in this work, we take steps to addressing a wider set of architectures and invariances. First, we show a class of vertex and edge-wise quantities that remain invariant during gradient flow training. Applying these invariances to architectures containing fully connected, convolutional and residual blocks, arranged appropriately, we obtain invariances of the singular values in adjacent (weight) matrices. Second, we prove how the controlled singular values imply low-rank results for fully connected ReLu networks. Third, we obtain low-rank results for arbitrary non-homogeneous networks whose last few layers contain fully-connected and ResNet blocks. To the best of our knowledge, this is the first time a low-rank phenomenon is proven rigorously for these architectures.

Our theoretical results offer explanations for empirical observations on more general architectures, and apply to the experiments in (Huh et al., 2021) for ResNet and CNNs. They also include the squared loss used there, in addition to the exponential or logistic loss used in most theoretical low-rank results. Moreover, our Theorem 2 gives an explanation for the "reduced alignment" phenomenon observed by Ji & Telgarsky (2019), where in experiments on AlexNet over CIFAR-10 the ratio $\|W\|_2/\|W\|_F$ converges to a value strictly less than 1 for some fully-connected layer W towards the end of the network.

One challenge in the analysis is the non-smoothness of the networks and ensuring an "operational" chain rule. To cope with this setting, we use a specific decomposition of an arbitrary ReLu architecture into a multilinear (up to sign) function and a ReLu network with +1/-1 weights. This reduction holds in the *stable sign regime*, a certain convergence setting of the parameters. This regime is different from stable activations, and is implied, e.g., by directional convergence of the parameters to a vector with non-zero entries (Lemma 2). This construction may be of independent interest.

In short, we make the following contributions to analyzing implicit biases of general architectures:

- We show vertex and edge-wise weight invariances during training with gradient flow. Using these invariances, we prove that for architectures containing fully-connected layers, convolutional layers and residual blocks, when appropriately organized into matrices, the absolute value of adjacent matrices have the same singular value, up to initialization (Theorem 1).
- We prove that in the stable sign regime, the absolute value of weight matrices in a fully-connected ReLU-activated network tend to rank-1 matrices (Corollary 1). We also show a low-rank bias for arbitrary nonhomogeneous feedforward networks whose last few layers are a composition of fully-connected and ResNet blocks (Theorem 2). In particular, if the Frobenius norms of these layers diverge, then the ratio between their operator norm and Frobenius norm is bounded nontrivially by an expression C fully specified by the architecture. To the best of our knowledge, this is the first time this type of bias is shown for nonlinear, nonhomogeneous networks.
- We prove our results via a decomposition that reduces arbitrarily structured deep feedforward networks with positively-homogeneous activation (e.g., ReLU or linear) to a multilinear (up to sign) structure (Lemma 2).

1.1 Related works

Decomposition of fully-connected neural networks into a multilinear part and a 0-1 part has been utilized by Choromanska et al. (2015); Kawaguchi (2016) but their formulation does not apply for trajectory studies. In Section 3, we give a detailed comparison between their approach and ours. Our decomposition makes use of the construction of a tree net that has been used by Khim & Loh (2019) in analyzing generalization of fully-connected networks. We describe their approach in 2 and show how we extend their construction to arbitrary feedforward neural nets. This construction makes up the first part of the proof of our decomposition lemma. The paper also makes use of path enumeration of neural nets, which overlaps with the path-norm literature of Neyshabur et al. (2015; 2017). The distinction is that we are studying classical gradient flow, as opposed to SGD (Neyshabur et al., 2015) or its variants (Neyshabur et al., 2017).

For linear networks, low-rank bias is proven for separable data and exponential-tailed loss in Ji & Telgarsky (2019). Du et al. (2018) give certain vertex-wise invariances for fully-connected ReLU networks and Frobenius norm difference invariances for CNNs. Compared to their results, ours are slightly stronger since we prove that our invariances hold for almost every time t on the gradient flow trajectory, thus allowing for the use of a Fundamental Theorem of Calculus and downstream analysis. Moreover, the set of invariances we show is strictly larger. Radhakrishnan et al. (2020)

show negative results when generalizing the low-rank bias from (Ji & Telgarsky, 2019) to vector-valued neural networks. In our work, we only consider scalar-valued neural networks performing binary classification.

2 PRELIMINARIES AND NOTATION

For an integer $k \in \mathbb{N}$, we write the set $[k] := \{1, 2, \ldots, k\}$. For vectors, we extend the sign function $\text{sgn} : \mathbb{R} \to \{-1, 1\}$ coordinatewise as $\text{sgn} : (x_i)_{i\in[d]} \mapsto (\text{sgn}(x_i))_{i\in[d]}$. For some (usually the canonical) basis $(e_i)_{i\in[n]}$ in some vector space $\mathbb{R}^n$, for all $x \in \mathbb{R}^n$ we use the notation $[x]_i = \langle x, e_i\rangle$ to denote the i-th coordinate of x.

Clarke subdifferential, definability and nonsmooth analysis. The analysis of non-smooth functions is central to our results. For a locally Lipschitz function $f : D \to \mathbb{R}$ with open domain D, there exists a set $D^c \subseteq D$ of full Lebesgue measure on which the derivative ∇f exists everywhere by Rademacher's theorem. As a result, calculus can usually be done over the *Clarke subdifferential* defined as the set

$$\partial f(x) := \text{CONV}\left\{\lim_{i\to\infty} \nabla f(x_i) \mid x_i \in D^c, x_i \to x\right\}$$

where CONV denotes the convex hull.

The Clarke subdifferential generalizes both the smooth derivative when $f \in C^1$ (continuously differentiable) and the convex subdifferential when f is convex. However, it only admits a chain rule with an inclusion and not equality, which is though necessary for backpropagation in deep learning.

If one is willing to work with a slightly restricted set of functions - formally known as tame, Whitney stratifiable, path differentiable or definable in an o-minimal structure (Davis et al., 2020; Ji & Telgarsky, 2020; Bolte & Pauwels, 2020), the Clarke subdifferential admits an *operational* chain rule. All architectures considered in this paper are definable (Ji & Telgarsky, 2020). Here, we use the form from Ji & Telgarsky (2020) Lemma B.9, which extends Theorem 5.2 of Davis et al. (2020).

Lemma 1 (Lemma B.9 in Ji & Telgarsky (2020))**.** *If $f : D \to \mathbb{R}$ is locally Lipschitz and defineable with D open, then for any interval I and arc $z : I \to D$, for a.e. $t \in I$, $\frac{\mathrm{d}f(z(t))}{\mathrm{d}t} = \left\langle y, \frac{\mathrm{d}z_t}{\mathrm{d}t}\right\rangle$, for all $y \in \partial f(z(t))$.*

Neural networks. Consider a neural network $\nu : \mathbb{R}^d \to \mathbb{R}$. The computation graph of ν is a weighted directed graph $G = (V, E, w)$ with weight function $w : E \to \mathbb{R}$. For each neuron $v \in V$, let $\text{IN}_v := \{u \in V : uv \in E\}$ and $\text{OUT}_v := \{w \in V : vw \in E\}$ be the input and output neurons of v. Let $\{i_1, i_2, \ldots, i_d\} =: I \subset V$ and $O := \{o\} \subset V$ be the set of input and output neurons defined as $\text{IN}(i) = \emptyset = \text{OUT}(o), \forall i \in I$. Each neuron $v \in V\backslash I$ is equipped with a positively-1-homogeneous activation function σ_v (such as the ReLU $x \mapsto \max(x, 0)$, leaky ReLU $x \mapsto \max(x, \alpha x)$ for some small positive α , or the linear activation).

To avoid unnecessary brackets, we write $w_e := w(e)$ for some $e \in E$. We will also abuse notation and write $w \in \mathbb{R}^{\overline{E}}$ where $\overline{E}$ is the set of learnable weights as the vector of learnable parameters. Let P be a path in G, i.e., a set of edges in E that forms a path. We will abuse notation and write $v \in P$ for some $v \in V$ if there exists $u \in V$ such that $uv \in P$ or $vu \in P$. Let ρ be the number of distinct paths from any $i \in I$ to $o \in O$. Let $\mathcal{P} := \left\{p_1, \ldots, p_\rho\right\}$ be the enumeration of these paths. For a path $p \in \mathcal{P}$ and an input x to the neural net, denote x_p the coordinate of x used in p.

Given a binary classification dataset $\left\{(x_i, y_i)\right\}_{i\in[n]}$ with $x_i \in \mathbb{R}^d$, $\|x_i\| \leq 1$ and $y_i \in \{-1, 1\}$, we minimize the empirical risk

$$\mathcal{R}(w) = \frac{1}{n}\sum\nolimits_{i=1}^{n} \ell(y_i\nu(x_i)) = \frac{1}{n}\sum\nolimits_{i=1}^{n} \ell(\nu(y_i x_i))$$

with loss $\ell : \mathbb{R} \to \mathbb{R}$, using gradient flow $\frac{\mathrm{d}w(t)}{\mathrm{d}t} \in -\partial\mathcal{R}(w(t))$. More theorem-specific assumptions will be given as we state the theorems.

As we detail the architectures used in this paper, we recall that the activation of each neuron is still positively-homogeneous. The networks considered here are assumed to be bias-free unless stated otherwise.

Definition 1 (Feedforward networks)**.** *A neural net ν with graph G is a* feedforward network *if G is a directed acyclic graph (DAG).*

Definition 2 (Fully-connected networks)**.** *A feedforward network ν with graph G is a* fully-connected network *if there exists partition of V into $V = (I \equiv V_1) \sqcup V_2 \sqcup \ldots \sqcup (V_{L+1} \equiv O)$ such that for all $u, v \in V, uv \in E$ iff there exists $i \in [L]$ such that $u \in V_i$ and $v \in V_{i+1}$.*

Definition 3 (Tree networks)**.** *A feedforward network ν with graph G is a* tree network *if the underlying undirected graph of G is a tree (undirected acyclic graph).*

Examples of feedforwards networks include ResNet (He et al., 2016), DenseNet (Huang et al., 2017), CNNs (Fukushima, 1980; LeCun et al., 2015) and other fully-connected ReLU architectures.

For a fully-connected network ν with layer partition $V =: V_1 \sqcup \ldots \sqcup V_{L+1}$ where L is the number of (hidden) layers, let $n_i := |V_i|$ be the number of neurons in the i-th layer and enumerate $V_i = \{v_{i,j}\}_{j \in [n_i]}$. Weights in this architecture can be organized into matrices $W^{[1]}, W^{[2]}, \ldots, W^{[L]}$ where $\mathbb{R}^{n_{i+1} \times n_i} \ni W^{[i]} = ((w_{v_{i,j} v_{i+1,k}}))_{j \in [n_i], k \in [n_{i+1}]}$, for all $i \in [L]$.

Tree networks. Most practical architectures are not tree networks, but they have been used to prove generalization bounds for adversarial risks. In particular, for fully-connected neural networks f whose activations are monotonically increasing and 1-Lipschitz, Khim & Loh (2019) defined the *tree transform* as the tree network $Tf(x; w) = \sum_{p_L=1}^{n_L} W^{[L]}_{1,p_L} \sigma \left(\ldots \sum_{p_2=1}^{n_2} W^{[2]}_{p_3,p_2} \sigma \left(w_{p_2..p_L} + \sum_{p_1=1}^{n_1} W^{[1]}_{p_2,p_1} x_{p_1} \right) \right)$ for vectors w with $\prod_{j=2}^{L} n_j$ entries, indexed by an L-tuple $(p_2, \ldots, p_L)$. We extend this idea in the next section.

3 Structural lemma: decomposition of deep network

We begin with a decomposition of a neural network into a multilinear (up to sign) and a non-weighted nonlinear part, which will greatly facilitate the chain rule that we need to apply in the analysis. Before stating the decomposition, we need the following definition of a path enumeration function, which computes the product of all weights and inputs on each path of a neural network.

Definition 4 (Path enumeration function)**.** *Let ν be a feedforward neural network with graph G and paths $\mathcal{P} = \{p_1, \ldots, p_\rho\}$. The path enumeration function h is defined for this network as $h : (x_1, x_2, \ldots, x_d) \mapsto \left(x_p \prod_{e \in p} |w_e| \right)_{p \in \mathcal{P}}$ where $x_p := x_k$ such that $i_k \in p$.*

We first state the main result of this section, proven in Appendix B.

Lemma 2 (Decomposition)**.** *Let $\nu : \mathbb{R}^d \to \mathbb{R}$ be a feedforward network with computation graph G, and ρ is the number of distinct maximal path in G. Then there exists a tree network $\mu : \mathbb{R}^\rho \to \mathbb{R}$ such that $\nu = \mu \circ h$ where $h : \mathbb{R}^d \to \mathbb{R}^\rho$ is the path enumeration function of G. Furthermore, all weights in μ are either -1 or $+1$ and fully determined by the signs of the weights in ν.*

Path activation of ReLU networks in the literature. The viewpoint that for every feedforward network ν there exists a tree network μ such that $\nu = \mu \circ h$ is not new and our emphasis here is on the fact that the description of $\mu : \mathbb{R}^\rho \to \mathbb{R}$ is fully determined by the signs of the weights. Indeed, in analyses of the loss landscape (Choromanska et al., 2015; Kawaguchi, 2016), ReLU networks are described as a sum over paths:

$$\nu(x; w) = \sum_{p \in \mathcal{P}} Z_p(x; w) \prod_{e \in p} w_e = \left\langle (Z_p(x; w))_{p \in \mathcal{P}}, h(x; w) \right\rangle_{\mathbb{R}^\rho}, \tag{1}$$

where $Z_p(x; w) = 1$ iff all ReLUs on path p are active (have nonnegative preactivation) and 0 otherwise. One can then take μ as a tree network with no hidden layer, ρ input neurons all connected to a single output neuron. However, this formulation complicates analyses of gradient trajectories, because of the explicit dependence of Z_p on numerical values of w. In our lemma, μ is a tree network whose description depends only on the signs of the weights. This assumption is realistic, in the sense that it is implied by directional parameter convergence (Section 4.1). If the weight signs (not necessarily the Relu activation pattern!) are constant, μ is independent of w, allowing for a

chain rule to differentiate through it. To see this, compare the partial derivative with respect to some w_e (when it exists) between the two approaches, in the limit where weight signs are constant

$$\text{(using Lemma 2)} \qquad \partial\nu/\partial w_e = \sum_{p\in\mathcal{P}|e\in p} [\nabla_w \mu(x;w)]_p \cdot \text{sgn}(w_e) x_p \prod_{f\in p, f\neq e} |w_f|, \tag{2}$$

$$\text{(using } Z_p \text{ in Eqn 1)} \qquad \partial\nu/\partial w_e = \sum_{p\in\mathcal{P}} \left(w_e \partial Z_p(x;w)/\partial w_e + Z_p(x;w)\right) \prod_{f\in p, \neq e} w_f. \tag{3}$$

In particular, the dependence of Equation 2 on w_e is extremely simple. This usefulness of this fact will be made precise in the next section when we study invariances.

Proof sketch. The proof contains two main steps. First, we 'unroll' the feedforward network into a tree network that computes the same function by adding extra vertices, edges and enable weight sharing. This step is part of the tree transform in Khim & Loh (2019) if the neural network is a fully-connected network; we generalize it to work with arbitrary feedforward networks. Second, we "pull back" the weights towards the input nodes using positive homogeneity of the activations: $a \cdot \sigma(x) = \text{sgn}(a) \cdot \sigma(x|a|)$. This operation is done on vertices closest (in number of edges on the unique path between any two vertices in a tree) to the output vertex first and continue until all vertices have been processed. We give a quick illustration of the two steps described above for a fully-connected ReLU-activated network with 1 hidden layer in the Appendix A.

4 Main Theorem: Training invariances

In this section, we put the previous decomposition lemma to use in proving an implicit regularization property of gradient flow in training deep neural networks.

4.1 Stable sign regime: a consequence of directional convergence

Recall the gradient flow curve $\{w(t)\}_{t\in[0,\infty)}$ defined by the differential inclusion $\frac{\mathrm{d}w(t)}{\mathrm{d}t} \in -\partial\mathcal{R}(w(t))$. We first state the main assumption in this section.

Assumption 1 (Stable sign regime)**.** *We assume that for all $t > 0, \text{sgn}(w(t)) = \text{sgn}(w(0))$. If this holds, we say that gradient flow is in a* stable sign regime.

In fact, the following assumption, shown in the literature, is sufficient.

Assumption 2 (Directional convergence to non-vanishing limit in each entry)**.** *We assume that $\frac{w(t)}{\|w(t)\|_2} \xrightarrow{t\to\infty} \overline{w}$ exists, is finite in each entry and furthermore, for all $e \in E, \overline{w}_e \neq 0$.*

Motivation and justification. It is straightforward to see that Assumption 1 follows from Assumption 2 but we provide a proof in the Appendix (Claim 1). Directional convergence was proven by Ji & Telgarsky (2020) using the exponential/logistic loss for the class of homogeneous networks, under additional mild assumptions. This fact justifies the first part of Assumption 2 for these architectures. For more general architectures, directional convergence, though not proven in theory, is widely used in practice: most practical stopping criteria based on parameter convergence imply at least directional convergence. The second part of Assumption 2 is pathological for our case, in the sense that directional convergence alone does not imply stable signs (for example, a weight that converges to 0 can change sign a infinite number of time).

Pointwise convergence is too strong in general. Note also that assuming pointwise convergence of the weights (i.e. $\lim_{t\to\infty} w(t)$ exists and is finite) is a much stronger statement, which is not true for the case of exponential/logistic loss and homogeneous networks (since $\|w(t)\|_2$ diverges, see for example Lyu & Li (2020), Ji & Telgarsky (2020), Ji & Telgarsky (2019)). Even when pointwise convergence holds, it would immediately reduce statements on asymptotic properties of gradient flow on ReLU activated architectures to that of linearly activated architectures. One may want to assume that gradient flow starts in the final affine piece prior to its pointwise convergence and thus activation patterns are fixed throughout training and teh behavior (multi)linear. In contrast, directional convergence of the weights *does not* imply such reduction from the ReLU-activation to the linear case (since, for example, linear pieces of ReLU networks are not necessarily convex (Hanin & Rolnick, 2019) or see a counter-example in Claim 2).

Stable sign implication. The motivation for Assumption 1 is the fact that weights in the tree network μ in the statement of Lemma 2 are fully determined by the *signs* of the weights in the original feedforward network ν. Thus, under Assumption 1, one can completely fix the weights of μ - it has no learnable parameters. Since we have the decomposition $\nu = \mu \circ h$ where h is the path enumeration function, dynamics of μ are fully determined by dynamics of h in the stable sign regime. To complete the picture, observe that h is highly multilinear in structure: the degree of a particular edge weight w_e in each entry of h is at most 1 for all edge e by definition of a path; and if ν is fully-connected, then h is a tensor in $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_L}$.

4.2 Training invariances

First, we state an assumption on the loss function, which holds for most losses used in practice, such as the logistic loss, exponential loss or squared loss.

Assumption 3 (Differentiable loss)**.** *The loss function $\ell : \mathbb{R} \to \mathbb{R}$ is differentiable everywhere.*

Lemma 3 (Vertex-wise invariance)**.** *Under Assumption 1, and 3, for all $v \in V \backslash \{I \cup O\}$ such that all edges incident to v have learnable weights, for a.e. time $t \geq 0$,*

$$\sum_{u \in \text{IN}_v} w_{uv}^2(t) - \sum_{b \in \text{OUT}_v} w_{vb}^2(t) = \sum_{u \in \text{IN}_v} w_{uv}^2(0) - \sum_{b \in \text{OUT}_v} w_{vb}^2(0). \tag{4}$$

If we also have $\text{IN}_u = \text{IN}_v = \text{IN}$ and $\text{OUT}_u = \text{OUT}_V = \text{OUT}$ then for a.e. time $t \geq 0$,

$$\sum_{a \in \text{IN}} |w_{au}(t) w_{av}(t)| - \sum_{b \in \text{OUT}} |w_{ub}(t) w_{vb}(t)| = \sum_{a \in \text{IN}} |w_{au}(0) w_{av}(0)| - \sum_{b \in \text{OUT}} |w_{ub}(0) w_{vb}(0)|. \tag{5}$$

Comparison to Du et al. (2018) A closely related form of Equation 4 in Lemma 3 has appeared in Du et al. (2018) (Theorem 2.1) for fully-connected ReLU/leaky-ReLU networks. In particular, the authors showed that the rate of change of difference between incoming and outgoing weights is 0. Invoking the Fundamental Theorem of Calculus (FTC) over this statement will return ours. However, their proof may not hold over a nonnegligible set of time t due to the use of the operational chain rule (Lemma 1) that holds only for almost all w_e. Thus the FTC can fail. The stronger form we showed here is useful in proving downstream algorithmic consequences, such as the low rank phenomenon. Furthermore, our result also holds for arbitrary feedforward architectures and not just the fully-connected case.

Before we put Lemma 3 to use, we list definitions of some ResNet variants.

Definition 5. *Denote ResNetIdentity, ResNetDiagonal and ResNetFree to be the version of ResNet described in He et al. (2016) where the residual block is defined respectively as*

1. $r(x; U, Y) = \sigma(U\sigma(Yx) + Ix)$ *where* $x \in \mathbb{R}^a, Y, U^\top \in \mathbb{R}^{b \times a}$, *and I is the identity,*
2. $r(x; U, Y, D) = \sigma(U\sigma(Yx) + Dx)$ *where* $x \in \mathbb{R}^a, Y, U^\top \in \mathbb{R}^{b \times a}$, *and D is diagonal,*
3. $r(x; U, Y, Z) = \sigma(U\sigma(Yx) + Zx)$ *where* $x \in \mathbb{R}^a, Y \in \mathbb{R}^{b \times a}, U \in \mathbb{R}^{c \times b}$ *and* $Z \in \mathbb{R}^{c \times a}$.

ResNetIdentity is the most common version of ResNet used in practice and is described in Equation 1 of (He et al., 2016). ResNetFree is described in Equation 2 of (He et al., 2016) but is less commonly used. Note that although ResNetIdentity is a special case of ResNetDiagonal, which is in turn a special case of ResNetFree. However, theorems for ResNetFree, for example, do not generalize trivially to the remaining variants, due to the restriction of Lemma 3 and Lemma 4 to vertices adjacent to all learnable weights and layers containing all learnable weights.

When applying Lemma 3 to specific architectures, we obtain the following:

Theorem 1 (Matrix-wise invariances)**.** *Recall that a convolutional layer with number of input kernels a, kernel size b and number of output kernels c and is a tensor in $\mathbb{R}^{a \times b \times c}$. Under Assumptions 1 and 3, we have the following matrix-wise invariance for a.e. time $t \geq 0$:*

$$|W_2(t)|^\top |W_2(t)| - |W_1(t)||W_1(t)|^\top = |W_2(0)|^\top |W_2(0)| - |W_1(0)||W_1(0)|^\top \tag{6}$$

for:

1. *(Fully-connected layers)* $W_1 \in \mathbb{R}^{b\times a}$ *and* $W_2 \in \mathbb{R}^{c\times b}$ *consecutive fully-connected layers,*
2. *(Convolutional layers)* W_1 *is convolutional, viewed as a flattening to a matrix* $\mathbb{R}^{c\times(a\times b)}$*, and* W_2 *adjacent convolutional, viewed as a flattening to a matrix* $\mathbb{R}^{(d\times e)\times c}$*,*
3. *(Within residual block of ResNet)* $W_1 = Y$ *and* $W_2 = U$ *where* $r(x; U, Y, Z)$ *is a residual block of ResNetIdentity, ResNetDiagonal or ResNetFree,*
4. *(Between residual blocks of ResNet)* $W_1 = \begin{bmatrix} U_1 & Z_1 \end{bmatrix}$*,* $W_2 = \begin{bmatrix} Y_2 \\ Z_2 \end{bmatrix}$ *where* $r(x; U_j, Y_j, Z_j)$*,* $j \in \{1, 2\}$ *are consecutive ResNetFree blocks,*
5. *(Convolutional-fully-connected layers)* W_1 *convolutional, viewed as a flattening to a matrix* $\mathbb{R}^{c\times(a\times b)}$ *and* W_2 *adjacent fully-connected layer, viewed as an rearrangement to* $\mathbb{R}^{d\times c}$*,*
6. *(Convolutional-ResNetFree block)* W_1 *convolutional, viewed as a flattening to a matrix* $\mathbb{R}^{c\times(a\times b)}$ *and* W_2 *is a rearrangement of* $\begin{bmatrix} U & Z \end{bmatrix}$ *into an element of* $\mathbb{R}^{d\times c}$*, where* $r(x; U, Y, Z)$ *is an adjacent ResNetFree block,*
7. *(ResNetFree block-fully-connected layers and vice versa)* $W_1 = \begin{bmatrix} U & Z \end{bmatrix} \in \mathbb{R}^{b\times a}$ *and* $W_2 \in \mathbb{R}^{c\times b}$ *adjacent fully-connected or* $W_1 \in \mathbb{R}^{b\times a}$ *fully-connected and* $W_2 = \begin{bmatrix} Y \\ Z \end{bmatrix} \in \mathbb{R}^{c\times b}$ *adjacent ResNet block where* $r(x; U, Y, Z)$ *is the ResNetFree block.*

We emphasize that the above theorem only makes *local* requirements on the neural network, to have local parts that are either fully-connected, convolutional or a residual block. The only global architecture requirement is feedforward-ness. The first point of Theorem 1 admits an extremely simple proof for the linear fully-connected network case in Arora et al. (2018) (Theorem 1) but has not been generalized to other architectures.

Significance of Theorem 1. Assume for simplicity that the right hand side in Equation 6 is 0, then the singular values of W_1 and W_2 are the same for each of the cases listed in Theorem 1. If we can form a chain of matrices whose singular values are the same by iteratively invoking Theorem 1 then all matrices considered have the same singular values as the final fully-connected layer that connects to the output. Recall that our networks are scalar-valued, so the final layer is a row vector, which is rank 1 and thus all layers considered in the chain have rank 1, which is useful in the next section.

Proof sketch of Theorem 1. Given Lemma 3, we demonstrate the proof for the first point. The remaining points admit the exact same proof technique but on different matrices, which can get messy and require some bookkeeping. Let $W_1 \in \mathbb{R}^{b,a}$ and $W_2 \in \mathbb{R}^{c,b}$ be two consecutive fully-connected layers for some $a, b, c \in \mathbb{N}$ number of vertices in these layers. Apply Equation 4 of Lemma 3 to each of the b shared neurons between these two layers, one get the diagonal entries of Equation 6 of the Theorem. Now, apply Equation 5 to each pair among the b shared neurons between these two layers, one get the off-diagonal entries of Equation 6.

We give a definition of layer for architectures where weights are not necessarily organized into matrices - say ResNet or DenseNet.

Definition 6 (Layer)**.** *Let* $F \subset E$ *be such that 1) for all* $e \neq f \in F$*, there is no path that contains both* e *and* f*; and 2) the graph* $(V, E\backslash F, w)$ *is disconnected. Then* F *is called a* layer *of* G*.*

For this definition, we have the following invariance:

Lemma 4 (Edge-wise invariance)**.** *Under Assumption 1, and 3, for all layers* F *and* F' *that contains all learnable weights, for a.e. time* $t \geq 0$*,*

$$\sum_{e\in F} w_e^2 - \sum_{f\in F'} w_f^2 = \sum_{e\in F} w_e^2(0) - \sum_{f\in F'} w_f^2(0). \tag{7}$$

Since IN_o *forms a layer, we also have*

$$\sum_{e\in F} w_e^2 - \sum_{v\in \text{IN}_o} w_{vo}^2 = \sum_{e\in F} w_e^2(0) - \sum_{v\in \text{IN}_o} w_{vo}^2(0). \tag{8}$$

Significance of Lemma 4. A flattening of a convolutional parameter tensor and a stacking of matrices in a ResNetDiagonal and ResNetFree block forms a layer. This lemma implies that the squared Frobenius norm of these matrices in the same network differs by a value that is fixed at initialization. This lemma also gives a direct implicit regularization for networks with biases, by treating neurons with bias as having an extra in-edge whose weight is the bias, from an extra in-vertex which is an input vertex with fixed input value 1.

4.3 Proof sketch of Lemma 3 and Lemma 4

All proofs in Lemma 3, Lemma 4 and Theorem 1 share the same technique of double counting paths, which we explain next. For simplicity, let us here assume that we are working with a network that is differentiable everywhere in some domain that we are considering – we give a full general proof in the Appendix. The main proof idea has been used in (Arora et al., 2018) and involves simply writing down the partial derivative of the risk. We have for some particular weight $w_e, e \in E$, via the smooth chain rule

$$\frac{\partial \mathcal{R}(w)}{\partial w_e} = \frac{1}{n}\sum_{i=1}^{n} l'(y_i \nu(x_i; w)) \cdot y_i \cdot \frac{\partial \nu(w)}{\partial w_e} \tag{9}$$

$$= \frac{1}{n}\sum_{i=1}^{n} l'(y_i \nu(x_i; w)) \cdot y_i \cdot \sum_{p \in \mathcal{P}, p \ni e} \left[\mu'(x_i; w)\right]_p \cdot \mathrm{sgn}(w_e) \cdot (x_i)_p \prod_{f \in p, f \neq e} |w_f| \tag{10}$$

where in the second line, we invoke the decomposition Lemma 2 and emphasize that μ has no learnable parameters in the stable sign regime. Now multiply w_e to the above expression to get

$$w_e \frac{\partial \mathcal{R}(w)}{\partial w_e} = \sum_{p \in \mathcal{P}, p \ni e} \frac{1}{n}\sum_{i=1}^{n} A_{i,p}(w) \tag{11}$$

where $A_{i,p}(w) = l'(y_i \nu(x_i; w)) \cdot y_i \cdot \left[\mu'(x_i; w)\right]_p \cdot (x_i)_p \prod_{f \in p} |w_f|$. Notice that $A_{i,p}(w)$ does not depend explicitly on the edge e we are taking the partial derivative over (only through w). Thus, we can sum over in-edges and out-edges of a particular v satisfying the assumption of Lemma 3 to get

$$\sum_{u \in \mathrm{IN}_v} w_{uv} \frac{\partial \mathcal{R}(w)}{\partial w_{uv}} = \sum_{p \in \mathcal{P}, p \ni v} \frac{1}{n}\sum_{i=1}^{n} A_{i,p}(w) = \sum_{b \in \mathrm{OUT}_v} w_{vb} \frac{\partial \mathcal{R}(w)}{\partial w_{vb}}. \tag{12}$$

Note that the only difference between Equations 11 and 12 is the set of paths that it is summing over, and we double count this set of paths.

We use the definition of gradient flow to get $\partial \mathcal{R}(w)/\partial w_e = \mathrm{d}w_e(t)/\mathrm{d}t$ and integrate with respect to time using a FTC to get the first part of Lemma 3. To get the second part of Lemma 3, we double count the set of jagged paths defined as all paths from I to v together with all paths from u to O and vice versa. Finally, to get Lemma 4, we double count the set of all paths $\mathcal{P}$.

5 Consequences: Low rank phenonmenon for nonlinear nonhomogeneous deep feedforward net

We apply the results from previous parts to prove a low-rank bias result for a large class of feed-forward networks. To the best of our knowledge, this is the first time such a result is shown for this class of deep networks, although the linear network analog has been known for some time. In particular, Equation 13 is the *reduced alignment* observed in (Ji & Telgarsky, 2019).

In light of Theorem 1, we have a definition of matrix representations of a layer:

Definition 7 (Matrix representation of a layer)**.** *The matrix representation for a ResNetFree block $r(x; U, Y, Z)$ is $\left|\begin{bmatrix} U & Z \end{bmatrix}\right|$; for a $T^{a,b,c} \in \mathbb{R}^{a,b,c}$ convolutional tensor is the absolute value of the flattening to a member of $\mathbb{R}^{c \times (a \times b)}$; and for a fully-connected layer is just the absolute value of its weight matrix.*

Theorem 2 (Reduced alignment for non-homogeneous networks)**.** *Under Assumption 1 and 3, let ν consist of an arbitrary feedforward neural network η, followed by $K \geq 0$ convolutional layers $(T^{a_k,b_k,c_k})_{k\in[K]}$, followed by $M \geq 0$ layers that are either ResNetFree blocks or fully-connected layers; and finally ending with a linear fully-connected layer Fin. For $j \in [K+M]$, denote $W^{[j]}$ the matrix representation of the j-th layer after η, $N_r(j)$ the number of ResNetFree blocks between j and Fin exclusively and $V_c(j) := \max\dim W^{[M+1]} \cdot \prod_{j<k\leq M} a_k b_k / \max(a_k, b_k)$ if $j \leq M$ and 1 otherwise. Then there exists a constant $D \geq 0$ fixed at initialization such that for a.e. time $t > 0$,*

$$\frac{1}{8^{N_r(j)} V_c(j)} \|W^{[j]}(t)\|_F^2 - \|W^{[j]}(t)\|_2^2 \leq D, \ \forall j \in [K+M]. \tag{13}$$

Furthermore, assume that $\|W^{[j]}\|_F \to \infty$ for some $j \in [K+M]$, then we have:

$$1/\min\Big(\textit{rank}(W^{[k]}), 8^{N_r(j)} V_c(j)\Big) \leq \|W^{[k]}(t)\|_2^2 / \|W^{[k]}(t)\|_F^2 \leq 1, \tag{14}$$

where rank returns the number of nonzero singular value of a matrix. In particular, for the last few fully-connected layers j with $N_r(j) = 0$ and $V_c(j) = 1$, we have:

$$\left\| \frac{W^{[j]}(t)}{\|W^{[j]}(t)\|_F} - u_j(t) v_j^\top(t) \right\|_F \xrightarrow{t\to\infty} 0, \forall k \in [M], \qquad |\left\langle v_{j+1}, u_j \right\rangle| \xrightarrow{t\to\infty} 1, \tag{15}$$

where u_j and v_j is the left and right principal singular vector of $W^{[j]}$.

Corollary 1. *For fully-connected networks with ReLU activations, trained with linearly separable data under logistic loss ℓ, Equation 15 holds for the absolute value of weight matrix in each layer with additional assumptions that $\mathcal{R}(w(0)) < \ell(0)$ and the weight vector converges in direction to a limiting vector with no 0 entries.*

5.1 Proof sketch

The proof is similar to that of Lemma 2.6 in (Ji & Telgarsky, 2019). Fix a $j \in [M+1]$ and invoke Lemma 4 over the layer $W^{[j]}$ to get $\|W^{[j]}\|_F^2 - \|Fin\|_F^2 = D_1$ for some D_1 fixed at initialization. But since our network is scalar-valued, Fin is a vector and thus $\|Fin\|_F^2 = \|Fin\|_2^2$. Now invoking Theorem 1 for consecutive pairs of layers, we can prove that $\|W^{[k]}\|_2^2 \geq C\|W^{[k+1]}\|_2^2 - D_{2,k}$ for $C = 1/8$ if the $k+1$ layer is a ResNetFree block (Lemma 6), $C = \max(a_{k+1}, b_{k+1})/(a_{k+1} b_{k+1})$ if the $k+1$ layer is a convolutional layer, $C = \max\dim W^{[M+1]}$ if $k = M$ (Lemma 5) and $C = 1$ otherwise, for some $D_{2,k}$ fixed at initialization, for all k from j to $M+K$. Telescope and subtract from $\|W^{[j]}\|_F^2$ to get the final result. The rest follows from Lemma 2.6 in (Ji & Telgarsky, 2019).

6 Concluding remarks and Future directions

In this paper, we extend the proof of the low rank phenonmenon, which has been widely observed in practice, beyond the linear network case. In particular, we address a variety of nonlinear architectural structures, homogeneous and non-homogeneous, which in this context have not been addressed theoretically before. To this end, we decomposed a feedforward ReLU/linear activated network into a composition of a multilinear (up to sign) function with a tree net. If the weights converge in direction to a vector with non-zero entries, the tree net is eventually fixed, allowing for chain rules to differentiate through. This leads to various matrix-wise invariances between fully-connected, convolution layers and ResNet blocks, enabling us to control the singular values of consecutive layers. In the end, we obtain a low-rank theorem for said local architectures.

Proving directional convergence for non-homogeneous networks will strengthen Theorem 2. Moreover, it would be interesting to adapt the same techniques for gradient descent with appropriate step size, in place of gradient flow. Another direction is to connect our low-rank bias results to the max-margin implicit regularization literature, which has been shown for linear networks and, more recently, certain 2-homogeneous architecture (Ji & Telgarsky, 2020). Another direction is to extend alignment results to vector-valued neural nets, where Radhakrishnan et al. (2020) gave a negative result for the straightforward generalization.

REFERENCES

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In Jennifer G. Dy and Andreas Krause (eds.), *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 244–253. PMLR, 2018. URL http://dblp.uni-trier.de/db/conf/icml/icml2018.html#AroraCH18.

Jérôme Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10809–10819. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/7a674153c63cff1ad7f0e261c369ab2c-Paper.pdf.

Anna Choromanska, MIkael Henaff, Michael Mathieu, Gerard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In Guy Lebanon and S. V. N. Vishwanathan (eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pp. 192–204, San Diego, California, USA, 09–12 May 2015. PMLR. URL https://proceedings.mlr.press/v38/choromanska15.html.

F.H. Clarke. *Optimization and Nonsmooth Analysis*. Wiley New York, 1983.

Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D. Lee. Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, 20(1): 119–154, 2020. doi: 10.1007/s10208-018-09409-5. URL https://doi.org/10.1007/s10208-018-09409-5.

Simon S. Du, Wei Hu, and Jason D. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced, 2018.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. doi: 10.1007/BF00344251. URL https://doi.org/10.1007/BF00344251.

Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional 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/2018/file/0e98aeeb54acf612b9eb4e48a269814c-Paper.pdf.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/9766527f2b5d3e95d4a733fcfb77bd7e-Paper.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Christopher Heil. Absolute continuity and the fundamental theorem of calculus, 2019. URL https://doi.org/10.1007/978-3-030-26903-6_6.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *arXiv*, 2021.

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

Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17176–17186. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c76e4b2fa54f8506719a5c0dc14c2eb9-Paper.pdf.

Kenji Kawaguchi. Deep learning without poor local minima. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/f2fc990265c712c49d51a18a32b39f0c-Paper.pdf.

Justin Khim and Po-Ling Loh. Adversarial risk bounds via function transformation, 2019.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeLIgBKPS.

Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/eaa32c96f620053cf442ad32258076b9-Paper.pdf.

Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning, 2017.

Adityanarayanan Radhakrishnan, Eshaan Nichani, Daniel Bernstein, and Caroline Uhler. On alignment in deep linear neural networks, 2020.

Miaoyan Wang, Khanh Dao Duc, Jonathan Fischer, and Yun S. Song. Operator norm inequalities between tensor unfoldings on the partition lattice, May 2017. ISSN 0024-3795. URL http://dx.doi.org/10.1016/j.laa.2017.01.017.

Yi Zhou and Yingbin Liang. Critical points of linear neural networks: Analytical forms and landscape properties. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SysEexbRb.

A Illustration of Lemma 2

We give a quick illustration of the two steps described in Section 3 for a fully-connected ReLU-activated network with 1 hidden layer. Figure 1 describes the unrolling of the neural network (Figure 1a) into a tree network (Figure 1b). Figure 2a describes the weight pull-back in the hidden layer and Figure 2b describes the weight pull-back in the input layer. It is clear that the inputs of the tree net in Figure 2b are coordinates of the path enumeration function $h(x; w)$ in this example. Furthermore, weights in the tree net depend entirely on the signs of the original weights. The rest of the proof argues this intuition for general feed-forward neural nets. As a remark, in general, ρ is a very large number - exponential in the number of layers for a fully-connected net with fixed width.

B Proof of Lemma 2

Proof of Theorem 2. For clarity, assume that activations are the same for all neurons and is denoted σ. It is clear afterwards how to specify individual activations.

We will use induction on $\alpha := \max_{p \in \mathcal{P}} |p|$. In the base case, $\alpha = 0$ and we have $I = 0 = V = \{o\}$. There are exactly one neural network with such a computation graph and $\nu(x) = \text{POST}_o(x) = x$.

Let us assume that the proposition is true for all neural net whose α is at least $t \geq 0$. Consider a neural net ν with computation graph G whose $\alpha = t + 1$. Let $G' = G \backslash I$ be the subgraph of G

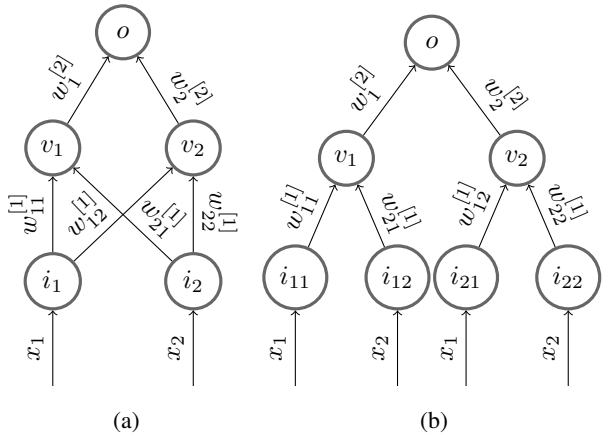

Figure 1: Transformation of a feedforward network into a tree net. All nodes apart from the input nodes use ReLU activation. The two neural nets drawn here compute the same function. This idea has been used in Khim & Loh (2019) to prove generalization bounds for adversarial risk.
.

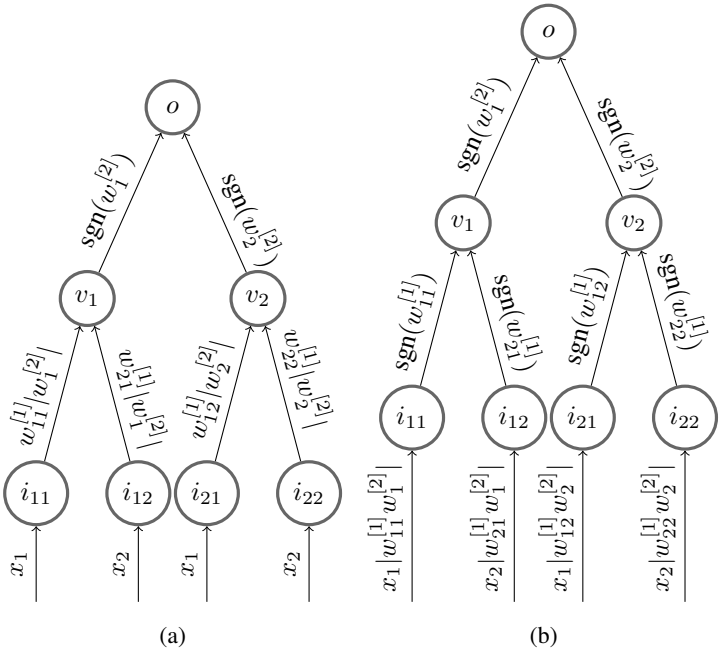

Figure 2: Pulling back of weights in a tree net. All nodes apart from the input nodes use the ReLU activation. The original net was drawn in Figure 1a and 1b. This is possible due to the positive-homogeneity of the activation function. Figure 2b is the final tree net in Theorem 2 with weights from $\{-1, +1\}$ and input from the path enumeration function of the neural net.

formed by removing all vertices in I and all edges incident to any vertex in I. The idea is then trying to apply inductive hypothesis on G' and reconnect I to the input vertices of the resulting graph.

However, this idea does not work immediately on G'. The problem is as followed: let $U = \text{OUT}_I \backslash I'$ where I' be the set of vertices in G' with indegree 0 and $\text{OUT}_I = \bigcup_{i \in I} \text{OUT}_i$ and U maybe non-empty. In other words, there may be a skip connection from some $i \in I$ to an inner vertex j of G'. When we remove $i \in I$ to form G', j is not an input vertex of G'. Thus, when we do path enumeration on G' in the inductive hypothesis, paths from j to output node o of G' is not accounted for and there is no way to reconnect i to j to prove the statement for G. It is not hard to see that U

contains all potentially troublesome inner vertices j that I connects to. The fix is simple: we add an artificial vertex k with linear activation between $i \in I$ and j inner vertex of G' and let the weight of ik be 1. This way, paths from i to o through j in G can be recorded as the path from k to o in G' and we can reconnect the paths afterwards.

Formally, we have the construction: construct another graph $G'' = (V'', E'', w'')$ as followed:

- $V'' = V \cup \{v_{a,u} | a \in I, u \in U, au \in E\}$.
- $E'' = E \backslash \left\{au | a \in I, u \in U, au \in E\right\} \cup \{av_{a,u}, v_{a,u}u | a \in I, u \in U, au \in E\}$
- $w''(av_{a,u}) = 1, w''(v_{a,u}u) = w(au)$ for all $a \in I, u \in U$ such that $au \in E$.

It is easy to verify the following facts about G'':

- The neural net computed by G'' is identical to that computed by G (in particular the set of input neurons I is the same).
- The path enumeration $\mathcal{P}$ of G'' and G is in bijection and the path enumeration function h of G'' and G is the same.
- The max path length α of G'' and G is the same.

We will identify G with G'' and prove the proposition for G'' instead. Denote $H = G'' \backslash I$ the subgraph of G'' formed by removing all vertices in I and all edges incident to any vertex in I. It is easy to verify the following facts about H:

- OUT_I is in bijection with the set of vertices with indegree 0 in H (note that G' does not have this property and this is the reason we need to construct G'').
- The max path length α of H is t.

Let the neural network computed by H be κ. Let $\mathcal{P}_H$ be the set of paths from any input neuron of H to the output neuron of H. Since H has max path length $\alpha = t$, we can invoke induction hypothesis to get $\kappa = \mu_\kappa \circ h_\kappa$. Furthermore, since the input neurons of κ is in bijection with OUT_I, we can write

$$\nu(x_1, \ldots, x_d) = \kappa\left(\left(\sigma\left(\sum_{i_k \in \text{IN}_j \subset I} w(i_k j) x_k\right)\right)_{j \in \text{OUT}_I}\right) \tag{16}$$

$$= \mu_\kappa\left(h_\kappa\left(\left(\sigma\left(\sum_{i_k \in \text{IN}_j \subset I} w(i_k j) x_k\right)\right)_{j \in \text{OUT}_I}\right)\right) \tag{17}$$

$$= \mu_\kappa\left(\left(\sigma\left(\sum_{i_k \in \text{IN}_j \subset I} w(i_k j) x_k\right) \prod_{e \in q} |w(e)|\right)_{q \in \mathcal{P}_H}\right) \quad \text{where } j \in \text{OUT}_I \cap q \tag{18}$$

$$= \mu_\kappa\left(\left(\sigma\left(\sum_{i_k \in \text{IN}_j \subset I} w(i_k j) x_k \prod_{e \in q} |w(e)|\right)\right)_{q \in \mathcal{P}_H}\right) \quad \text{where } j \in \text{OUT}_I \cap q. \tag{19}$$

where the three lines are definition of ν, κ and h_κ respectively and the last line is because σ is positively homogeneous. Denote $y : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{P}_H|}$ as

$$y(x_1, \ldots, x_d) = \left(\sigma \left(\sum_{i_k \in \text{IN}_j \subset I} w(i_k j) x_k \prod_{e \in q} |w(e)| \right) \right)_{q \in \mathcal{P}_H} \qquad \text{where } j \in \text{OUT}_I \cap q.$$

Then $\nu = \mu_k \circ y$

We now construct a tree network $\eta : \mathbb{R}^{|\mathcal{P}|} \to \mathbb{R}^{|\mathcal{P}|_H}$ and attempt to show that $\mu_\kappa \circ \eta$ is a tree network and $y = \eta \circ h$. If this is done then we can set $\mu = \mu_\kappa \circ \eta$ and get $\nu = \mu_\kappa \circ y = \mu_\kappa \circ \eta \circ h = \mu \circ h$ where h is path enumeration function of G (or G'').

Index a path (whether in $\mathcal{P}$ or in $\mathcal{P}_H$) by the set of edges that it contains, define

$$\eta : (a_p)_{p \in \mathcal{P}} \to \left(z_q := \sigma \left(\sum_{s \in \mathcal{P} \text{ s.t. } s \supset q} a_s \text{sgn}\left(w(q \backslash s)\right) \right) \right)_{q \in \mathcal{P}_H}.$$

then by definition, we have

$$\eta \circ h : x = (x_1, \ldots, x_d) \mapsto \left(\sigma \left(\sum_{s \in \mathcal{P} \text{ s.t. } s \supset q} x_s \text{sgn}\left(w(q \backslash s)\right) \prod_{e \in s} |w(e)| \right) \right)_{q \in \mathcal{P}_H}.$$

Fix $q \in \mathcal{P}_H$ and let $j \in \text{OUT}_I \cup q$. For all $s \in \mathcal{P}$, $s \supset q$ iff $I \cap s \subset \text{IN}_j$ and thus the sum in the definition of $\eta \circ h$ and that in the definition of y sums over the same set of indices. Now fix $s \in \mathcal{P}$ such that $s \supset q$. We have

$$\text{sgn}(w(q \backslash s)) \prod_{e \in s} |w(e)| = \text{sgn}(w(q \backslash s)) |w(q \backslash s)| \prod_{e \in q} |w(e)| = w(q \backslash s) \prod_{e \in q} |w(e)|$$

so the summands are also equal in the two expressions. Thus we have $\eta \circ h = y$. That $\mu_k \circ \eta$ is a tree network is also easy to see due to the fact that inclusions in the definition of η forms a partial order and thus the proof is done.

We clean up the construction to remove artificially introduced vertices as followed: if there is a vertex that is linearly activated, has one in-edge with weight 1 and one out-edge with weight w, then remove that vertex, connect the in-vertex with the out-vertex by an edge of weight w. It is not hard to see that the recursive construction of the final network ν has a nice description: let $\mathcal{C}$ be the set of paths from any vertex to o, construct a graph with vertices $\mathcal{C}$ and edge pq iff $q \subset p$ and the length of p is exactly one more than q. Furthermore, the weight of pq edge is $\text{sgn}(w(p \backslash q)$.

Consider the case when each neuron in ν may have different activations. Each neuron in μ is on a distinct path g from some input neuron of μ to its output neuron. This path corresponds to a specific path p in ν of the same length. Thus we can assign a neuron in g the activation of the corresponding neuron in p. □

C Proof of Training Invariances

Claim 1 (Directional convergence to non-vanishing point implies stable sign)**.** *Assumption 2 implies Assumption 1.*

Proof. Let v be the direction that $\frac{w(t)}{\|w(t)\|_2}$ converges to. Let $\mathcal{O}$ be the orthant that v lies in. Since v does not have a 0 entry, the ball $\mathcal{B}$ with radius $\min_i |v_i|/2$ and center v is a subset of the interior of $\mathcal{O}$. Since $\frac{w(t)}{\|w(t)\|_2}$ converges to v, there exists a time T such that for all $s > T$, $\frac{w(s)}{\|w(s)\|_2} \in \mathcal{B}$. Thus, eventually, $w(s) \in \mathcal{B}$ where its signs stay constant. □

Claim 2 (Directional convergence does not imply stable activation)**.** *There exists a function $w : \mathbb{R} \to \mathbb{R}^d$ such that $w(t)$ converges in direction to some vector v but for all $u \in w(\mathbb{R})$, $w^{-1}(u)$ is infinite.*

Proof. Fix a vector $v \in \mathbb{R}^d$. Consider the function $t \mapsto v|t\sin(t)|$. □

Proof of Lemma 3. Recall that our real weight vector $w(t)$ is an arc Ji & Telgarsky (2020) which, by definition, is absolutely continuous. It then follows from real analysis that its component $w_e(t)$ is also absolutely continuous for any $e \in E$. For absolutely continuous functions $w_e(t)$ for some $e \in E$, invoke the Fundamental Theorem of Calculus (FTC) (Chapter 6, (Heil, 2019)), to get

$$\sum_{u\in \text{IN}_v} w_{uv}^2(t) - \sum_{u\in \text{IN}_v} w_{uv}^2(0) = 2\sum_{u\in \text{IN}_v}\int_{[0,t]} w_{uv}(s)\frac{\mathrm{d}w_{uv}}{\mathrm{d}t}(s)\mathrm{d}s \tag{20}$$

$$= 2\sum_{u\in \text{IN}_v}\int_{[0,t]} w_{uv}(s)\left[\overline{\partial}\mathcal{R}(w)\right]_{uv}\mathrm{d}s. \tag{21}$$

We now proceed to compute $\frac{\mathrm{d}w}{\mathrm{d}t}(s)$. Since $w(t)$ is absolutely continuous and we are taking the integral via FTC, we only need to compute $\frac{\mathrm{d}w}{\mathrm{d}t}(s)$ for a.e. time s. By chain rule (see for example, the first line of the proof of Lemma C.8 in Lyu & Li (2020)), there exists functions $(g_j)_{j=1}^n$ such that $g_j \in \partial\nu_{x_j}(w) \subseteq \mathbb{R}^{|E|}$ for all $j \in [n]$ where $\nu_{x_j}(w) = \nu(x_j; w)$, and for a.e. time $s \geq 0$,

$$\frac{\mathrm{d}w}{\mathrm{d}t}(s) = \frac{1}{n}\sum_{j=1}^n l'(y_i\nu(x_i; w(s)))\cdot y_i \cdot g_i \tag{22}$$

Fix $j \in [n]$, by the inclusion chain rule, since $\nu_{x_j} = \mu \circ h_{x_j}$, with h and μ also locally Lipschitz, we have by Theorem I.1 of Lyu & Li (2020),

$$\partial\nu_{x_j}(w) = \partial(\mu\circ h_{x_j})(w) \subseteq \text{CONV}\left\{\sum_{p\in\rho}[\alpha]_\rho\beta_{j,\rho} \mid \alpha \in \partial\mu(h(w)), \beta_{j,\rho} \in \partial[h_{x_j}]_p(w)\right\} \tag{23}$$

Thus there exists $(\gamma_a)_{a=1}^A \geq 0, \sum_{a=1}^A \gamma_a = 1$ and $(\alpha^a \in \partial\mu(h(w)), \beta_{j,p}^a \in \partial[h_{x_j}]_p(w))_{a=1}^A$ such that

$$g_j = \sum_{a=1}^A \gamma_a \sum_{p\in\rho}[\alpha]_p^a\beta_{j,p}^a.$$

Here we use Assumption 1 to deduce that eventually, all weights are non-zero in gradient flow trajectory to compute

$$\partial[h_{x_j}]_p(w) = \left\{\frac{\mathrm{d}[h_{x_j}]_p(w)}{\mathrm{d}(w)}\right\} = \left\{\left(\mathbb{1}_{e\in p}\cdot \text{sgn}(w_e)\cdot(x_j)_p\prod_{f\in p, f\neq e}|w_f|\right)_{e\in E}\right\}.$$

Plug this back into g_j to get

$$g_j = \left(\sum_{a=1}^A \gamma_a \sum_{p\in\rho|e\in p}[\alpha]_p^a \text{sgn}(w_e)\cdot(x_j)_p\prod_{f\in p, f\neq e}|w_f|\right)_{e\in E}.$$

Plug this back into $\frac{\mathrm{d}w}{\mathrm{d}t}(s)$ and to get - coordinate-wise, for a.e. $s \geq 0$,

$$\frac{\mathrm{d}w_e}{\mathrm{d}t}(s) = \frac{1}{n}\sum_{j=1}^n l'(y_i\nu(x_i; w(s)))\cdot y_i\cdot\sum_{a=1}^A\gamma_a\sum_{p\in\rho|e\in p}[\alpha]_p^a\text{sgn}(w_e(s))\cdot(x_j)_p\prod_{f\in p, f\neq e}|w_f(s)|. \tag{24}$$

Multiply both sides with w_e gives

$$w_e(s)\frac{\mathrm{d}w_e}{\mathrm{d}t}(s) = \sum_{p\in\mathcal{P}|e\in p}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s)) \tag{25}$$

where

$$d_{j,p}(w) = l'(y_i\nu(x_i;w))\cdot y_i\cdot\sum_{a=1}^{A}\gamma_a[\alpha]_p^a\cdot(x_j)_p\cdot\prod_{f\in p}|w_f(s)|. \tag{26}$$

Note that d does not depend on the specific edge e used in Equation 24.

Plugging back into the FTC to get

$$\sum_{u\in\mathrm{IN}_v} w_{uv}^2(t) - \sum_{u\in\mathrm{IN}_v} w_{uv}^2(0) = 2\sum_{u\in\mathrm{IN}_v}\int_{[0,t]}\sum_{p\in\mathcal{P}|uv\in p}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s))\mathrm{d}s \tag{27}$$

$$= 2\int_{[0,t]}\sum_{p\in\mathcal{P}|v\in p}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s))\mathrm{d}s \tag{28}$$

Finally, by an identical argument but applied to the set of edges $b\in\mathrm{OUT}_v$, we have

$$\sum_{b\in\mathrm{OUT}_v} w_{vb}^2(t) - \sum_{b\in\mathrm{OUT}_v} w_{vb}^2(0) = 2\int_{[0,t]}\sum_{p\in\mathcal{P}|v\in p}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s))\mathrm{d}s \tag{29}$$

$$= \sum_{u\in\mathrm{IN}_v} w_{uv}^2(t) - \sum_{u\in\mathrm{IN}_v} w_{uv}^2(0) \tag{30}$$

which completes the proof of the first part of the lemma.

For the second part, recall that we have 2 vertices u, v such that $\mathrm{IN}_v = \mathrm{IN}_u = \mathrm{IN}$ and $\mathrm{OUT}_v = \mathrm{OUT}_u = \mathrm{OUT}$. Use 24 to get, for notations set up previously, for almost every time $s > 0$,

$$\sum_{a\in\mathrm{IN}}\mathrm{sgn}(w_{au}(s)w_{av}(s))\left(w_{au}(s)\frac{\mathrm{d}w_{av}}{\mathrm{d}t}(s) + \frac{\mathrm{d}w_{au}}{\mathrm{d}t}(t)w_{av}(t)\right) = \sum_{p\in\mathcal{A}}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s)) \tag{31}$$

where $\mathcal{A}$ is a set of jagged paths defined as

$$\mathcal{A} = \left\{p_1\cup p_2 | p_1 \text{ is a path from } I \text{ to } v \text{ or } u; p_2 \text{ is a of path from } u \text{ or } v, \text{ respectively, to } O\right\}.$$

Then by a double counting argument, we also have

$$\sum_{b\in\mathrm{OUT}}\mathrm{sgn}(w_{ub}(s)w_{vb}(s))\left(w_{ub}(t)\frac{\mathrm{d}w_{vb}}{\mathrm{d}t}(t) + \frac{\mathrm{d}w_{ub}}{\mathrm{d}t}(t)w_{vb}(t)\right) \tag{32}$$

$$= \sum_{p\in\mathcal{A}}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s)) \tag{33}$$

$$= \sum_{a\in\mathrm{IN}}\mathrm{sgn}(w_{au}(s)w_{av}(s))\left(w_{au}(t)\frac{\mathrm{d}w_{av}}{\mathrm{d}t}(t) + \frac{\mathrm{d}w_{au}}{\mathrm{d}t}(t)w_{av}(t)\right) \tag{34}$$

Integrate both sides over $[0,t]$ as we did in the first part to get the conclusion of the second part. Note that the second part is special case of the first part, which is obtainable by setting $u = v$. We point out that proof of the first and second part of this lemma differs only in the choice of the set $\mathcal{A}$ (in the first part, $\mathcal{A}$ is the set of all paths that goes through v). □

Proof of Lemma 4. The proof is identical to that of Lemma 3, with the only difference being the set $\mathcal{A}$ that we double count. Here, set $\mathcal{A}$ to be $\mathcal{P}$. Then by the definition of layer (see Definition 6), one can double count $\mathcal{P}$ by counting paths that goes through any element of a particular layer. The proof completes by considering (using the same notation as the proof of Lemma 3) for a.e. time $s \geq 0$,

$$\sum_{e\in F} w_e(s)\frac{\mathrm{d}w_e}{\mathrm{d}t}(s) = \sum_{p\in\mathcal{P}}\frac{1}{n}\sum_{j=1}^{n} d_{j,p}(w(s))\cdot(x_j)_p\cdot\prod_{f\in p}|w_f(s)| \tag{35}$$

for any layer F. □

Proof of Theorem 1. All points in this theorem admit the same proof technique, where we form the matrices as instructed. Let $X = \{v_1, v_2, \ldots, v_m\} \subseteq V$ be the neurons shared between the two layers. Invoke the first part of Lemma 3 on each element of X to get the diagonal of Equation 6. To get the off-diagonal, check the conditions of the second part of Lemma 3 and invoke the lemma for each pair $v_j \neq v_k \in X$.

For the first point, let $W_1 \in \mathbb{R}^{b\times a}$ be a fully-connected layer from vertices $v_1, \ldots, v_a$ to vertices $U = u_1, \ldots, u_b$ and let $W_2 \in \mathbb{R}^{c\times b}$ be a fully-connected layer from $u_1, \ldots, u_b$ to $w_1, \ldots, w_c$. Then we have $\text{IN}_{u_1} = \ldots = \text{IN}_{u_b} = \{v_1, \ldots, v_a\} =: \text{IN}$ and $\text{OUT}_{u_1} = \ldots = \text{OUT}_{u_b} = \{w_1, \ldots, w_c\} =: \text{OUT}$. Furthermore, all weights around any u_j's are learnable. Thus the condition of both parts of Lemma 3 is satisfied for vertices in U. Invoke Lemma 3 to get the proof.

For the second point, let $T_1 \in \mathbb{R}^{a\times b\times c}$ and $T_2 \in \mathbb{R}^{c\times d\times e}$ be the convolutional tensor. Form the matrix representation W_1 and W_2 as per the theorem statement. Let each kernel be use l times, then the output vertices of the first convolutional (and the input vertices of the second) can be enumerated as $\left\{\left\{v_i^j\right\}_{i\in[l]}\right\}_{j\in[c]}$. We use double path counting to show the diagonal entries of the matrix-wise invariance. Fix a $j \in [c]$. Let $\mathcal{A}_j$ be the set of paths that goes through $\bigcup_{i\in[l]}\{v_i^j\}$. Then using the proof technique of Lemma 3, the sum of squared weights of the j-th row of W_1 count paths going through in-edges of $\bigcup_{i\in[l]}\{v_i^j\}$. Similarly, the sum of the squared weights of the j-column of W_2 count paths going through out-edges of $\bigcup_{i\in[l]}\{v_i^j\}$. It is not hard to see that we are just double counting $\mathcal{A}$. To see the off-diagonals, notice that for $j \neq k \in [c]$, $\bigcup_{i\in[l]} \text{IN}_{v_i^j} = \bigcup_{i\in[l]} \text{IN}_{v_i^k}$ since they both are the input vertices of the first convolutional layer. Also, $\bigcup_{i\in[l]} \text{OUT}_{v_i^j} = \bigcup_{i\in[l]} \text{OUT}_{v_i^k}$ since they both count the output vertices of the second convolutional layer. Invoke the second part of Lemma 3 to get the proof.

For the third point, let $r(x; U, Y, Z)$ be a residual block of either ResNetIdentity, ResNetDiagonal or ResNetFree. In all variants, this block consists of vertices $v_1, \ldots, v_a$ that is the size of the input ($x \in \mathbb{R}^a$). Then $Y \in \mathbb{R}^{b\times a}$ is a fully connected layer between $v_1, \ldots, v_a$ and $S = \{u_1, \ldots, u_b\}$ and $U \in \mathbb{R}^{c\times b}$ is a fully connected layer between $u_1, \ldots, u_b$ and $w_1, \ldots, w_c$. Note that skip connection does not affect these U and Y. Then for each $u \in S, \text{IN}_u = \{v_1, \ldots, v_a\} =: \text{IN}$ and $\text{OUT}_u = \{w_1, \ldots, w_c\} =: \text{OUT}$ and thus conditions of Lemma 3 is satisfied in both parts. Invoke this Lemma to get the matrix invariance.

For the fourth point, let $r_i(x; U_i, Y_i, Z_i), i \in \{1, 2\}$ be consecutive ResNetFree block. Let $v_1, \ldots, v_a$ be the input layer of the first block. Let $Y_1 \in \mathbb{R}^{b\times a}$ fully-connect the input layer to $u_1, \ldots, u_b$ be the. Let $U_1 \in \mathbb{R}^{c\times b}$ fully-connect the $u_1, \ldots, u_b$ to $w_1, \ldots, w_c$. Then $Z_1 \in \mathbb{R}^{c\times a}$ fully-connect the v's to the w's. Thus the matrix $\begin{bmatrix} U_1 & Z_1 \end{bmatrix} \in \mathbb{R}^{c\times(b+a)}$ fully-connects $w_1, \ldots, w_c$ to $v_1, \ldots, v_a, u_1, \ldots, u_b$. Let the remaining node in the ResNetFree bloxk r_2 (not including $w_1, \ldots, w_c$) be $t_1, \ldots, t_d$. Then by an identical argument, $\begin{bmatrix} Y_2 \\ Z_2 \end{bmatrix}$ fully-connects $w_1, \ldots, w_c$ to $t_1, \ldots, t_d$. Invoke the first point to complete the proof.

For the fifth point, let the convolutional tensor be $T \in \mathbb{R}^{a,b,c}$. For each output kernel, assume that it is used l times over the input. Then there are $l \times c$ output neurons for the convolutional layer, enumerated as $\{\{v_i^j\}_{i\in[l]}\}_{j\in[c]}$ which are the input neurons to the adjacent fully connected layer. Let the there output neurons of the fully-connected layer be $u_1, \ldots, u_d$. Now, $W_1 \in \mathbb{R}^{c\times(a\times b)}$ and $W_2 \in \mathbb{R}^{(l\times d)\times c}$ is a rearrangement of the fully-connected layer to expose the dimension c. We will use path double counting to prove matrix invariance. For the diagonals of the matrix invariance, pick $j \in [c]$, then the j-th entry of the diagonal counts the paths that goes through exactly one of the v_i^j for some $i \in [l]$. Call this set $\mathcal{A}_j$. There are $a \times b$ in-edges to each v_i^j for each $i \in [l]$ and among different i's the weights are shared. Thus using the proof of Lemma 3, the sum over the squared of weights in the j-th row of W_1 counts $\mathcal{A}$. On the other hand, we can count $\mathcal{A}_j$ by counting all the paths that uses exactly one of the edges going out of any v_i^j for some $i \in [l]$, which is the sum of the squared of weights in the j-th column of W_2. To show the statement for the off-diagonals, for a fix $j, k \in [c]$, consider $\bigcup_{i\in[l]} \text{IN}_{v_i^j} = \bigcup_{i\in[l]} \text{IN}_{v_i^k}$ since they both are the input neurons to

the convolutional layer. Also $\bigcup_{i\in[l]} \text{OUT}_{v_i^j} = \bigcup_{i\in[l]} \text{OUT}_{v_i^k} = \{u_1, \ldots, u_d\}$. Thus we can invoke Lemma 3 to get the off-diagonals.

For the sixth point, let the input vertices to the ResNetFree block be $v_1, \ldots, v_a$ and the remaining vertices in the block be $u_1, \ldots, u_b$. Then we have $W_2 = \begin{bmatrix} U & Z \end{bmatrix}$ fully-connect $v_1, \ldots, v_a$ to $u_1, \ldots, u_b$. Thus,we invoke the fifth point to get the desired results.

For the seventh point, first consider the case where the ResNetFree block is followed by the fully-connected layer. Let $r(x; U, Y, Z)$ be the ResNetFree block with input neurons $v_1, \ldots, v_a$, $Y \in \mathbb{R}^{b\times a}$ fully-connect $v_1, \ldots, v_a$ to $u_1, \ldots, u_b$ and $U \in \mathbb{R}^{c\times b}$ fully-connect $u_1, \ldots, u_b$ to $w_1, \ldots, w_c$. Then by the same argument as the fourth point, we have $\begin{bmatrix} U & Z \end{bmatrix}$ fully-connects $v_1, \ldots, v_a, u_1, \ldots, u_b$ to $w_1, \ldots, w_c$ and W_2 fully connects $w_1, \ldots, w_c$ to the output vertices of the fully-connected layer. Invoke the first point to complete the proof.

For the case where the fully-connected layer $W_1 \in \mathbb{R}^{b\times a}$ is followed by the ResNetFree block with input vertices $v_1, \ldots, v_b$ and remaining vertices $w_1, \ldots, w_c$. Then by the same argument as the fourth point, we have $\begin{bmatrix} Y \\ Z \end{bmatrix}$ fully connects $v_1, \ldots, v_b$ to $w_1, \ldots w_c$. Invoke the first point to complete the proof. □

D PROOF OF THEOREM 2

First we state a helper lemma

Lemma 5 (Largest singular value of different flattening of the same tensor is close)**.** *Let $T \in \mathbb{R}^{a,b,c}$ be an order 3 tensor (say a convolutional weight tensor). Let T_1, T_2 be the standard flattening of this tensor into an element in $\mathbb{R}^{c\times(a\times b)}$ and $\mathbb{R}^{(b\times c)\times a}$ respectively. Then*

$$\frac{\max(a,b)}{ab}\|T_1\|_2^2 \le \|T_2\|_2^2 \le \frac{\max(b,c)}{bc}\|T_1\|_2^2.$$

Proof. Invoke Theorem 4.8 of Wang et al. (2017). □

Lemma 6 (Shuffling layer in ResNetFree block preserves largest singular value up to multiple by 8)**.** *Recall that for a ResNetFree block $r(U, Y, Z)$ with $Y \in \mathbb{R}^{b\times a}$, $U \in \mathbb{R}^{c\times b}$ and $Z \in \mathbb{R}^{c\times a}$, there are two possible rearrangement of the weights $A = \begin{bmatrix} U & Z \end{bmatrix}$ and $B = \begin{bmatrix} Y \\ Z \end{bmatrix}$. We have $\|B\|_2^2 \ge \frac{1}{8}\|A\|_2^2 - D'$ where $D' \ge 0$ is fixed at inialization.*

Proof. Recall that by point three of Theorem 1, we have matrix invariance

$$U^\top(t)U(t) - Y(t)Y^\top(t) = U^\top(0)U(0) - Y(0)Y^\top(0).$$

By line $B.2$ in Ji & Telgarsky (2019),

$$\|Y\|_2^2 \ge \|U\|_2^2 - D \tag{36}$$

where $D = \|U^\top(0)U(0) - Y(0)Y^\top(0)\|_2^2$ fixed at initialization.

For positive semidefinite matrix X, denote $\lambda(X)$ to be the function that returns the maximum eigenvalue of X. We have

$$\|B\|_2^2 = (\lambda(B^\top B))^2 \tag{37}$$

$$= \left(\lambda\left(Y^\top Y + Z^\top Z\right)\right)^2 \tag{38}$$

$$\geq \frac{1}{4}\left(\lambda\left(Y^\top Y\right) + \lambda\left(Z^\top Z\right)\right)^2 \tag{39}$$

$$\geq \frac{1}{4}\left(\lambda\left(Y^\top Y\right)\right)^2 + \frac{1}{4}\left(\lambda\left(Z^\top Z\right)\right)^2 \tag{40}$$

$$\geq \frac{1}{4}\left(\lambda\left(UU^\top\right)\right)^2 + \frac{1}{4}\left(\lambda\left(ZZ^\top\right)\right)^2 - \frac{D}{4} \tag{41}$$

$$\geq \frac{1}{8}\left(\lambda\left(UU^\top\right) + \lambda\left(ZZ^\top\right)\right)^2 - \frac{D}{4} \tag{42}$$

$$\geq \frac{1}{8}\left(\lambda\left(UU^\top + ZZ^\top\right)\right)^2 - \frac{D}{4} \tag{43}$$

$$= \frac{1}{8}\left(\lambda\left(AA^\top\right)\right)^2 - \frac{D}{4} = \frac{1}{8}\|A\|_2^2 - \frac{D}{2} \tag{44}$$

where 39 is by an application of Weyl's inequality for Hermitian matrices which states that $\lambda(C + D) \geq \lambda(C) + t$ where t is is the smallest eigenvalue of D, which is nonnegative since matrices here are all positive semidefinite; 41 is a consequence of 36 and 43 is the application of the inequality $\lambda(C + D) \leq \lambda(C) + \lambda(D)$ which is another of Weyl's inequality for Hermitian matrices. □

Proof of Theorem 2. Fix $j \in [K + M]$, similar to the proof of Theorem 1, we first invoke Lemma 4 to bound the Frobenius norm of each of the final $K + M + 1$ layers (counting the last layer Fin) via the last layer. Let W_j be the matrix representation of the j-th layer among the last $M + 1$ layer as described in Theorem 1. Note that even if a layer has more than one matrix representation in Theorem 1, their Frobenius norm is still the same because different representation merely reorganize the weights. Thus, we can pick an arbitrary representation. Thus, for each $j \in [K + M]$, we have

$$\|W_j(t)\|_F^2 - \|Fin(t)\|_F^2 = D_0, \tag{45}$$

where $D_0 = \|W_j(0)\|_F^2 - \|Fin(0)\|_F^2$ fixed at initialization.

Now, similar to the proof of Theorem 1, we bound the difference between the operator norm of $W_j(t)$ and $Fin(t)$ by a telescoping argument. By Lemma 6, switching from one matrix representation to the other for ResNet incurs at most a multiplicative factor of 2 and an additive factor cost that depends only on the initialization. In each adjacent layer, the maximum number of switch between representation is one (so that it fits the form prescribed in Theorem 1). By matrix invariance between adjacent layers of the $K + M$ pairs of adjacent layers,

$$\|W_l(t)\|_2^2 \geq C\|W_{l+1}(t)\|_2^2 - D_l \tag{46}$$

for $C = 1/8$ if the $k+1$ layer is a ResNetFree bl ck (Lemma 6), $C = \max(a_{k+1}, b_{k+1})/(a_{k+1}b_{k+1})$ if the $k + 1$ layer is a convolutional layer, $C = \max \dim W^{[M+1]}$ if $k = M$ (Lemma 5) and $C = 1$ otherwise; for $D_l = \|W_{l+1}^\top(0)W_{l+1}(0) - W_l(0)W_l^\top(0)\|_2^2$.

Telescope the sum and subtract from 45 to get the statement of the theorem. When Frobenius norm diverges, divide both sides by the Frobenius norm to get the ratio statement. Note that the bound $\|W\|_F^2/n_\sigma(W) \leq \|W\|_2^2$ is trivial since the largest singular value squared is at least the average singular value squared.

When the lower bound of 13 is 1, it matches the upper bound and thus the largest singular value dominates all other singular values. Alignment follows from the proof of Lemma 2.6 second point in Ji & Telgarsky (2019). □

E PROOF OF COROLLARY 1

Proof. Let L be the number of layers in the network. Under the conditions stated in Corollary 1, Lyu & Li (2020) and Ji & Telgarsky (2020) showed that $\|w(t)\|_2$ diverges. Invoke Lemma 4 for all $k \in [L-1]$ and sum up the results, we have $L\|W_L(t)\|_F^2 = D'' + \sum_{j=1}^{L} \|W_j(t)\|_F^2 \; \|W_k\|_F^2 = D'' + \|w(t)\|_2^2$ where D'' is constant in t. Thus $\|W_j(t)\|_F^2$ diverges for all $j \in [L]$. Since the sum of all but the largest singular value is bounded, but the sum of all singular values diverge, we conclude that the largest singular value eventually dominates the remaining eigenvalue. Together with convergence in direction for these architecture Ji & Telgarsky (2020), we have each matrix converges to its rank-1 approximation.

□