Skip Connections as Effective Symmetry-Breaking

Emin Orhan
aeminorhan@gmail.com
Rice University & Baylor College of Medicine

February 14, 2017

Abstract

Skip connections made the training of very deep neural networks possible and have become an indispendable component in a variety of neural architectures. A completely satisfactory explanation for their success remains elusive. Here, we present a novel explanation for the benefits of skip connections in training very deep neural networks. We argue that skip connections help break symmetries inherent in the loss landscapes of deep networks, leading to drastically simplified landscapes. In particular, skip connections between adjacent layers in a multilayer network break the permutation symmetry of nodes in a given layer, and the recently proposed DenseNet architecture, where each layer projects skip connections to every layer above it, also breaks the rescaling symmetry of connectivity matrices between different layers. This hypothesis is supported by evidence from a toy model with binary weights and from experiments with fully-connected networks suggesting (i) that skip connections do not necessarily improve training unless they help break symmetries and (ii) that alternative ways of breaking the symmetries also lead to significant performance improvements in training deep networks, hence there is nothing special about skip connections in this respect. We find, however, that skip connections confer additional benefits over and above symmetry-breaking, such as the ability to deal effectively with the vanishing gradients problem.

Contents

1 Introduction

Introduction of skip (or residual) connections has substantially improved the training of very deep neural networks [6, 7, 9]. Despite informal intuitions and sometimes worryingly non-rigorous metaphors ("keeping a "clean" information path" [7], "to ... improve the information flow between layers" [9]) put forward to motivate skip connections, a clear understanding of how these connections improve training has been lacking. Such understanding is invaluable both in its own right and for the possibilities it might offer for further improvements in training very deep neural networks. A number of recent papers addressed different aspects

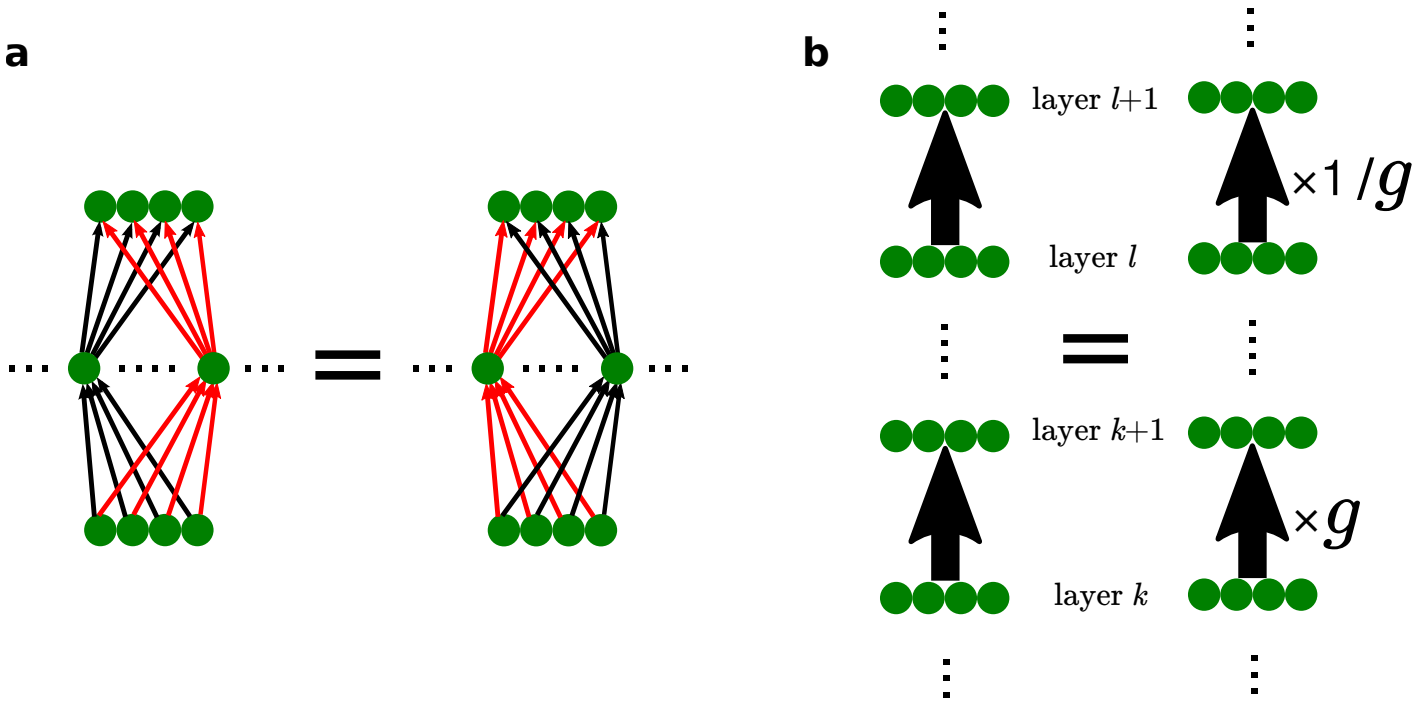

Figure 1: Symmetries of a fully connected multilayer network. (a) Permutation symmetry of hidden units at the same layer. (b) Rescaling symmetry of weight matrices between different layers.

of this question [4,12,14]. In this paper, we attempt to shed further light on this question. We argue that skip connections help break symmetries inherent in the loss landscapes of deep neural networks. Symmetries lead to saddle structures in the landscape, causing problems for gradient-based optimization methods [1, 17, 19]. Hence, symmetry breaking is useful to the extent that it eliminates such saddle structures. We show that skip connections between adjacent layers break the permutation symmetry of nodes at a given layer, whereas the more recently introduced DenseNet architecture [9], where each layer projects skip connections to every layer above it, breaks the rescaling symmetry of connectivity matrices between different layers. Breaking of these symmetries leads to drastically more regular landscapes than those of networks with no skip connections.

2 Results

2.1 Symmetries in fully-connected networks

Loss landscapes of fully-connected multilayer networks are riddled with several exact and approximate symmetries. We will focus on two particular symmetries in this paper. The first one is the permutation symmetry of nodes within a given layer: nodes within the same layer can be permuted without changing the function computed by the network (Figure 1a). This is a rather general symmetry that does not require any restrictive assumptions on the network.

The second symmetry is the invariance of the computed function to a rescaling of the connectivity matrices between different layers (Figure 1b): the connectivity matrix $\mathbf{W}_k$ can be scaled by a constant g and $\mathbf{W}_l$ by the constant $1/g$ without changing the function computed by the network (if biases are included, all the biases between the layers k and l also have to be scaled by g). This symmetry holds exactly in networks using the ReLU nonlinearity as the activation function (as well as in linear networks). A similar rescaling symmetry has been discussed before [15], where it has been noted that the incoming weights of any given node can be scaled up by a constant and its outgoing weights scaled down by the same constant without changing the function computed by the network and a modified stochastic gradient descent (SGD) algorithm has been proposed to effectively break this rescaling symmetry.

How do skip connections break these symmetries? Skip connections between adjacent layers break the permutation symmetry of hidden units at a given layer by ordering them according to the ordering of the hidden units at the previous layer (Figure 2a). Note that the input units are already ordered unambiguously. Additional skip connections between each layer and all layers above it, as in the DenseNet architecture introduced in [9], break the rescaling symmetry of the weight matrices by adding distinct sets of skip vectors

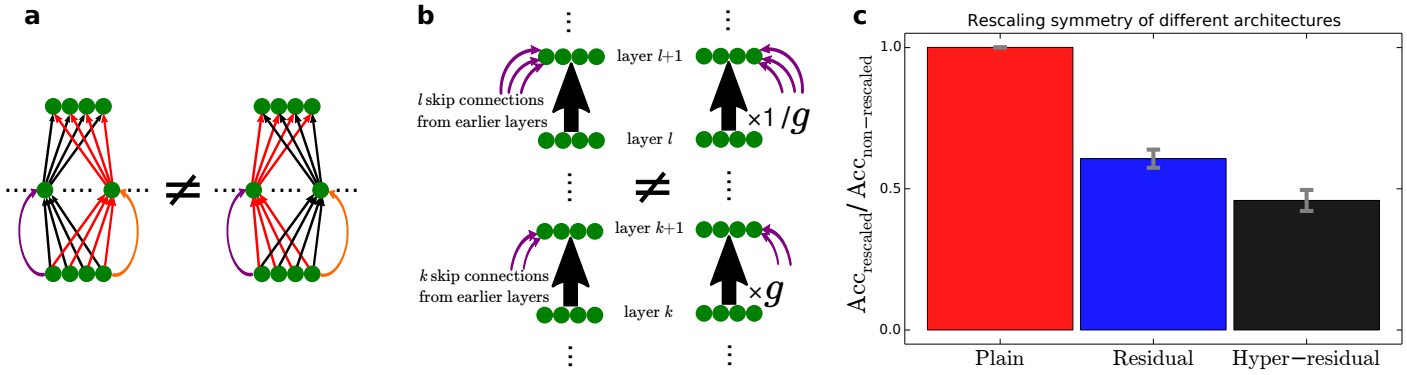

Figure 2: Skip connections break symmetries in fully-connected multilayer networks. (a) Skip connections between adjacent layers break the permutation symmetry of nodes at the same layer. (b) Skip connections from all earlier layers break the rescaling symmetry of weight matrices between different layers. (c) Rescaling symmetry of three different architectures. The rescaling symmetry is measured by the ratio of the training accuracy of the rescaled network to that of the non-rescaled network. The rescaling is done by first randomly choosing two different connectivity matrices $\mathbf{W}_k$ and $\mathbf{W}_l$ ($k < l$), then multiplying $\mathbf{W}_k$ by 10 and $\mathbf{W}_l$ by 0.1 (all bias vectors between layers k and l are also multiplied by 10). The networks are all fully-connected 20-layer feedforward networks with ReLU activation. Error bars represent standard errors over 40 independent runs of the experiment. The difference between the residual and hyper-residual architectures is significant (two-sided t-test, $p < .01$).

to each layer (Figure 2b).

To illustrate the second point, we consider the simple linear case: the composite function computed by a linear L-layer plain network can be expressed as $\mathbf{W}_{L-1}\mathbf{W}_{L-2}\dots\mathbf{W}_1\mathbf{x}_1$, where $\mathbf{x}_1$ is the input (ignoring the biases for simplicity). This expression clearly displays the rescaling symmetry as $\mathbf{W}_{L-1}\dots\mathbf{W}_l\dots\mathbf{W}_k\dots\mathbf{W}_1 = \mathbf{W}_{L-1}\dots g^{-1}\mathbf{W}_l\dots g\mathbf{W}_k\dots\mathbf{W}_1$. In linear networks with skip connections between each layer and all layers above it (we call this architecture "hyper-residual" henceforth) and assuming all skip connectivity matrices to be the identity, the composite function computed by the network becomes $(\mathbf{W}_{L-1}+(L-1)\mathbf{I})(\mathbf{W}_{L-2}+(L-2)\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I})\mathbf{x}_1$ and the rescaling symmetry is broken, since $(\mathbf{W}_{L-1}+(L-1)\mathbf{I})\dots(\mathbf{W}_l+l\mathbf{I})\dots(\mathbf{W}_k+k\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I}) \neq (\mathbf{W}_{L-1}+(L-1)\mathbf{I})\dots(g^{-1}\mathbf{W}_l+l\mathbf{I})\dots(g\mathbf{W}_k+k\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I})$.

We note that the rescaling symmetry is also broken in the "residual" architecture with skip connections between adjacent layers only. In this case, the function computed by the network can be expressed as $(\mathbf{W}_{L-1}+\mathbf{I})(\mathbf{W}_{L-2}+\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I})\mathbf{x}_1$ and it is easy to see that $(\mathbf{W}_{L-1}+\mathbf{I})\dots(\mathbf{W}_l+\mathbf{I})\dots(\mathbf{W}_k+\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I}) \neq (\mathbf{W}_{L-1}+\mathbf{I})\dots(g^{-1}\mathbf{W}_l+\mathbf{I})\dots(g\mathbf{W}_k+\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I})$. Although both residual and hyper-residual architectures break the rescaling symmetry, we find empirically that the hyper-residual architecture is more effective in this respect, presumably because it sets the scales of each connectivity matrix less ambiguously than the residual architecture by adding more distinct skip connectivity matrices at each layer. We demonstrate this effect empirically in Figure 2c, where the training accuracy ratios of rescaled to non-rescaled networks are shown for the three architectures in 20-layer nonlinear fully-connected networks trained on the CIFAR-100 dataset. A ratio of 1 indicates perfect rescaling symmetry and smaller ratios indicate less rescaling symmetry.

2.2 Landscapes of small networks with binary weights

To illustrate how skip connections change the loss landscapes of multilayer networks, we first consider a toy model with seven layers and two nodes at each layer, except for the final layer which has a single node (Figure 3). To be able to characterize the landscape completely, all the weights in the network are restricted to be $+1$ or -1. Biases of the units are set to zero (assigning random biases to the units leads to qualitatively similar results; supplementary Figure S1). This gives rise to a model with 23 binary parameters and a landscape with 2^{23} (~ 8.4M) possible parameter configurations, which we evaluate exhaustively on a two-dimensional regression problem with a random target function.

We considered the four architectures shown in Figure 3: (a) the plain architecture is a fully connected feedforward network with no skip connections (Figure 3a), described by the equation:

$$\mathbf{x}_{l+1} = f(\mathbf{W}_l\mathbf{x}_l) \tag{1}$$

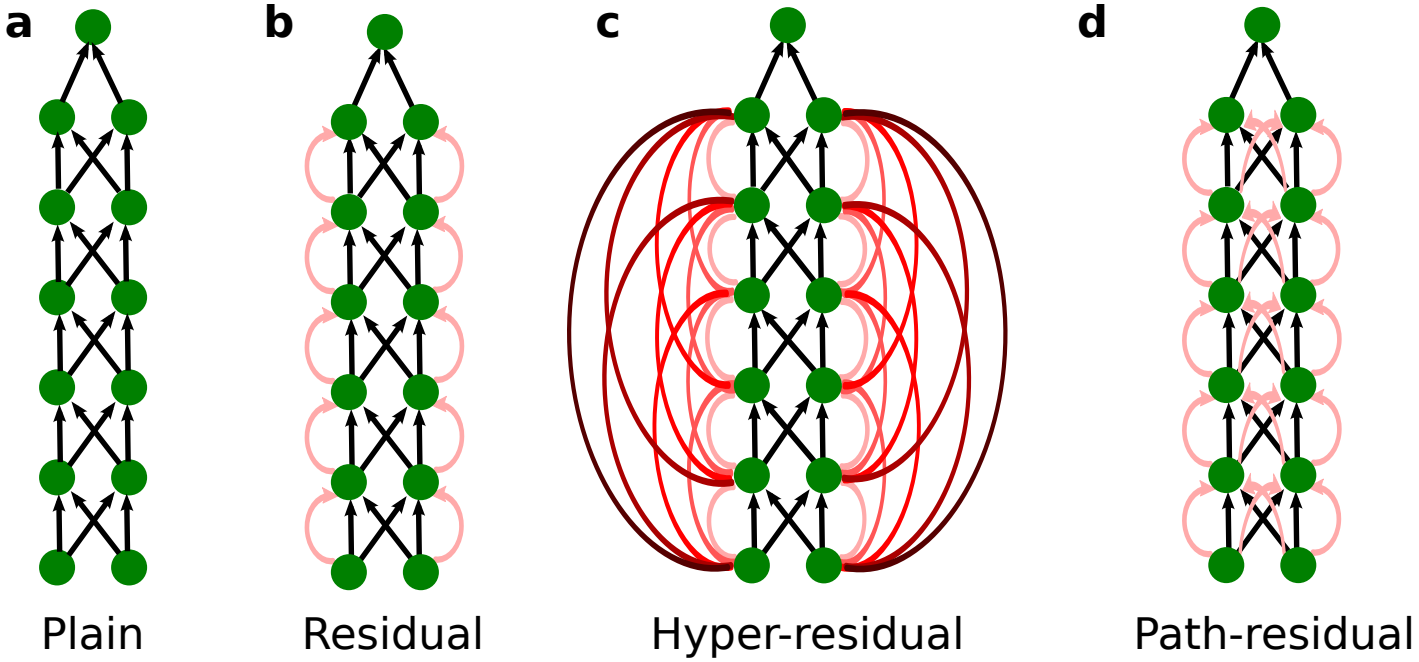

Figure 3: Schematic diagrams of toy networks. (a) Plain feedforward network. (b) Residual network. (c) Hyper-residual network. Skip connections are represented by undirected lines here for clarity, but all connections should be interpreted as feedforward. (d) Path-residual network. Standard feedforward connections (black arrows) are applied before the nonlinearity, whereas the skip connections are applied after the nonlinearity (see the text for details).

for $l = 1, \ldots, 5$. $f(\cdot)$ is chosen to be the ReLU nonlinearity and $\mathbf{x}_1$ denotes the input layer.

(b) The residual architecture introduces identity skip connections between adjacent layers (Figure 3b). To avoid any trivial effects due to differences in scaling of the activities in different architectures, we normalized the unit activities by the number of incoming connections they receive and equalized this between all four models (note that the number of incoming connections for each hidden unit is 2 in the plain architecture). Hence, the equations describing the residual architecture are given by:

$$\mathbf{x}_{l+1} = \frac{2}{3}\Big[f(\mathbf{W}_l \mathbf{x}_l) + \mathbf{x}_l\Big] \tag{2}$$

for $l = 1, \ldots, 5$.

(c) The hyper-residual architecture adds identity skip connections between each layer and all the layers above it (Figure 3c):

$$\mathbf{x}_{l+1} = \frac{2}{l+2}\Big[f(\mathbf{W}_l \mathbf{x}_l) + \mathbf{x}_l + \mathbf{x}_{l-1} + \ldots + \mathbf{x}_1\Big] \tag{3}$$

for $l = 1, \ldots, 5$. This architecture is inspired by the DenseNet architecture introduced in [9]. In both architectures, each layer projects skip connections to layers above it, but the specific way in which these projections are combined at a given layer differs between the two architectures.

(d) In the path-residual architecture, each individual connection has its own identity skip connection (Figure 3d). This is equivalent to using a matrix of ones, instead of an identity matrix, as the skip connectivity matrix:

$$\mathbf{x}_{l+1} = \frac{2}{4}\Big[f(\mathbf{W}_l \mathbf{x}_l) + \mathbf{1}\mathbf{x}_l\Big] \tag{4}$$

where $\mathbf{1}$ denotes a matrix of ones. Because each hidden unit receives the same skip connections from both units at the previous layer, this architecture does not break the permutation symmetry of the hidden units and therefore we expect its landscape to be more rugged than the landscapes of the residual and hyper-residual architectures.

We define the neighbors of a given configuration to be the configurations that can be reached from it by one bit flips, i.e. by flipping a single parameter from $+1$ to -1 or vice versa.

For each configuration, we computed the number of better neighbors of that configuration, i.e. neighbors

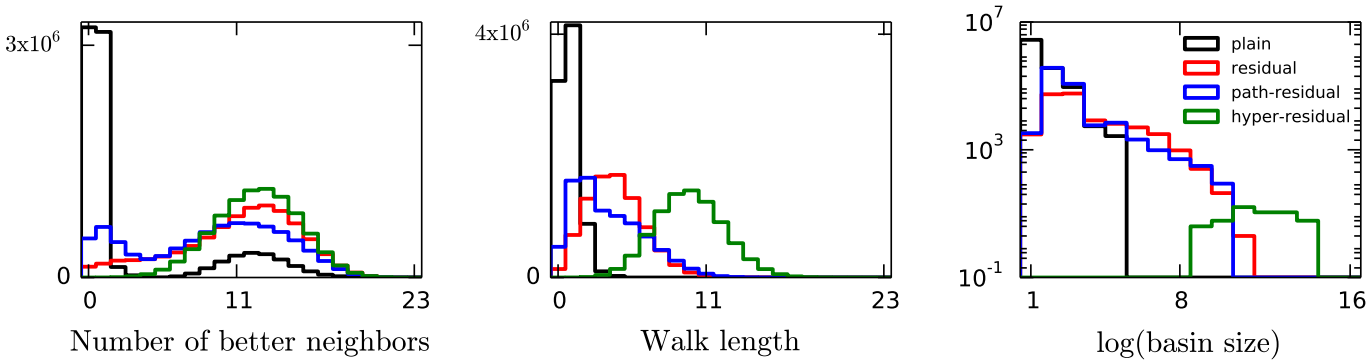

Figure 4: Characterization of the loss landscapes of toy networks with binary parameters. (Left) Distribution of the number of better neighbors; (middle) lengths of adaptive walks to a local minimum; (right) distribution of the basin sizes of local minima.

that have a lower loss value. Figure 4 (left) shows the distribution of the number of better neighbors for the four architectures. Note that zero corresponds to local minima and 23 to local maxima in this plot. The plain architecture has a large number of local minima ($\sim$ 3.2M) and the distribution is dominated by configurations with one or zero better neighbors, hinting at a very rugged landscape. The residual architecture drastically reduces the number of local minima to $\sim$ 140K and the hyper-residual architecture reduces it further to a mere 54 local minima. As predicted from a symmetry-breaking consideration, the path-residual architecture has a more rugged landscape than the residual one, having $\sim$ 505K local minima (the fact that its landscape is less rugged than the plain network's is consistent with the observation that the rescaling symmetry is broken in the path-residual architecture).

Statistics of random or adaptive walks are also commonly used to describe various features of combinatorial landscapes [16]. We performed adaptive walks in all landscapes by starting from each configuration and moving to the best neighbor at each step until a local minimum is reached. Such walks take shorter in rugged landscapes than in more regular landscapes. Similarly, basin sizes of local minima should be smaller in rugged landscapes. We found that the plain architecture has the shortest walks to a local minimum and the smallest basin sizes. The hyper-residual architecture, on the other hand, has the longest walks and largest basin sizes (Figure 4 middle and right), consistent with the distributions of the number of better neighbors for the different architectures.

Although we do not expect features of the discrete landscapes to carry over to their continuous counterparts exactly, we expect that the ruggedness of the discrete landscapes should give a good indication of the difficulty of navigating their continuous counterparts with adaptive methods such as gradient descent.

2.3 Dynamics of learning in linear networks with skip connections

We next investigate how skip connections affect the learning dynamics in linear networks. We recall that in an L-layer linear plain network, the input-output mapping is given by:

$$\mathbf{x}_L = \mathbf{W}_{L-1}\mathbf{W}_{L-2}\dots\mathbf{W}_1\mathbf{x}_1 \tag{5}$$

where $\mathbf{x}_1$ and $\mathbf{x}_L$ are the input and output vectors, respectively. In linear residual networks with identity skip connections between adjacent layers, the input-output mapping becomes:

$$\mathbf{x}_L = (\mathbf{W}_{L-1}+\mathbf{I})(\mathbf{W}_{L-2}+\mathbf{I})\dots(\mathbf{W}_1+\mathbf{I})\mathbf{x}_1 \tag{6}$$

Finally, in hyper-residual linear networks, the input-output mapping is given by:

$$\mathbf{x}_L = \Big(\mathbf{W}_{L-1}+(L-1)\mathbf{I}\Big)\Big(\mathbf{W}_{L-2}+(L-2)\mathbf{I}\Big)\dots\Big(\mathbf{W}_1+\mathbf{I}\Big)\mathbf{x}_1 \tag{7}$$

In the derivations to follow, we do not have to assume that the connectivity matrices are square matrices. If they are rectangular matrices, the identity matrix $\mathbf{I}$ should be interpreted to be a rectangular identity matrix of the appropriate size. This corresponds to zero-padding the layers when they are not the same size, as is usually done in practice.

2.3.1 Three-layer networks

Dynamics of learning in plain linear networks with no skip connections was analyzed in [18]. For a three-layer network ($L = 3$), the learning dynamics can be expressed by the following differential equations [18]:

$$\tau \frac{d}{dt} a^\alpha = (s_\alpha - a^\alpha \cdot b^\alpha) b^\alpha - \sum_{\gamma \neq \alpha} (a^\alpha \cdot b^\gamma) b^\gamma \tag{8}$$

$$\tau \frac{d}{dt} b^\alpha = (s_\alpha - a^\alpha \cdot b^\alpha) a^\alpha - \sum_{\gamma \neq \alpha} (a^\gamma \cdot b^\alpha) a^\gamma \tag{9}$$

Here a^α and b^α are n-dimensional column vectors (where n is the number of hidden units) connecting the hidden layer to the α-th input and output modes, respectively, of the input-output correlation matrix and s_α is the corresponding singular value (see [18] for further details). The first term on the right-hand side of Equations 8-9 facilitates cooperation between a^α and b^α corresponding to the same input-output mode, while the second term encourages competition between vectors corresponding to different modes. These dynamics can be interpreted as gradient descent on the following energy function:

$$E = \frac{1}{2\tau} \sum_\alpha (s_\alpha - a^\alpha \cdot b^\alpha)^2 + \frac{1}{2\tau} \sum_{\alpha \neq \beta} (a^\alpha \cdot b^\beta)^2 \tag{10}$$

This energy function has several symmetries. The one that concerns us in connection with skip connections between adjacent layers is the permutation symmetry of hidden units which reveals itself as the invariance of the energy function to a (simultaneous) permutation of the elements of the vectors a^α and b^α for all α. This causes saddle structures in the landscape that slow down learning. Specifically, for the permutation symmetry of hidden units, these saddle structures are the hyperplanes $a_i^\alpha = a_j^\alpha \ \forall \alpha$, for each pair of hidden units i, j (similarly, the hyperplanes $b_i^\alpha = b_j^\alpha \ \forall \alpha$) that make the model non-identifiable. Formally, these correspond to the singularities of the Hessian or the Fisher information matrix [1]. Indeed, it is easy to check that when $a_i^\alpha = a_j^\alpha \ \forall \alpha$ for any pair of hidden units i, j, the Hessian becomes singular in the plain network, but not in the residual network due to the broken permutation symmetry (Supplementary Note 2). The Hessian also has additional singularities at the hyper-planes $a_i^\alpha = 0 \ \forall \alpha$ for any i and at $b_i^\alpha = 0$ for any i and α, which can be expected to affect the dynamics if, for instance, the parameters are initialized to small random values. However, these additional singularities are not caused by the permutation symmetry of the hidden units. The effect of such saddle structures on the learning dynamics has previously been analyzed in shallow non-linear feedforward networks [1, 17, 19]: in particular, it has been shown that they significantly slow down learning. Once on any of these manifolds, the dynamics never leaves it. In practice, the variables are initialized randomly and hence they eventually escape the vicinity of the slow manifolds, but the manifolds can exert their effect for a long time because they can be attractive under gradient descent dynamics [1,17,19].

In the simplest scenario where there are only two input and output modes, the learning dynamics of Equations 8, 9 reduces to:

$$\frac{d}{dt} a^1 = (s_1 - a^1 \cdot b^1) b^1 - (a^1 \cdot b^2) b^2 \tag{11}$$

$$\frac{d}{dt} a^2 = (s_2 - a^2 \cdot b^2) b^2 - (a^2 \cdot b^1) b^1 \tag{12}$$

$$\frac{d}{dt} b^1 = (s_1 - a^1 \cdot b^1) a^1 - (a^1 \cdot b^2) a^2 \tag{13}$$

$$\frac{d}{dt} b^2 = (s_2 - a^2 \cdot b^2) a^2 - (a^2 \cdot b^1) a^1 \tag{14}$$

How does adding skip connections between adjacent layers change the learning dynamics? Considering again a three-layer network ($L = 3$) with only two input and output modes, a straightforward extension of

Equations 11-14 shows that the learning dynamics changes as follows:

$$\frac{d}{dt}a^1 = \Big[s_1 - (a^1+v^1)\cdot(b^1+u^1)\Big](b^1+u^1) - \Big[(a^1+v^1)\cdot(b^2+u^2)\Big](b^2+u^2) \quad (15)$$

$$\frac{d}{dt}a^2 = \Big[s_2 - (a^2+v^2)\cdot(b^2+u^2)\Big](b^2+u^2) - \Big[(a^2+v^2)\cdot(b^1+u^1)\Big](b^1+u^1) \quad (16)$$

$$\frac{d}{dt}b^1 = \Big[s_1 - (a^1+v^1)\cdot(b^1+u^1)\Big](a^1+v^1) - \Big[(a^1+v^1)\cdot(b^2+u^2)\Big](a^2+v^2) \quad (17)$$

$$\frac{d}{dt}b^2 = \Big[s_2 - (a^2+v^2)\cdot(b^2+u^2)\Big](a^2+v^2) - \Big[(a^2+v^2)\cdot(b^1+u^1)\Big](a^1+v^1) \quad (18)$$

where u^1 and u^2 are orthonormal vectors (similarly for v^1 and v^2). The derivation proceeds essentially identically to the corresponding derivation for plain networks in [18]. The only differences are: (i) we substitute the plain weight matrices $\mathbf{W}_l$ with their residual counterparts $\mathbf{W}_l + \mathbf{I}$ and (ii) when changing the basis from the canonical basis for the weight matrices $\mathbf{W}_1$, $\mathbf{W}_2$ to the input and output modes of the input-output correlation matrix, $\mathbf{U}$ and $\mathbf{V}$, we note that:

$$\mathbf{W}_2 + \mathbf{I} = \mathbf{U}\overline{\mathbf{W}}_2 + \mathbf{U}\mathbf{U}^\top = \mathbf{U}(\overline{\mathbf{W}}_2 + \mathbf{U}^\top) \quad (19)$$

$$\mathbf{W}_1 + \mathbf{I} = \overline{\mathbf{W}}_1\mathbf{V}^\top + \mathbf{V}\mathbf{V}^\top = (\overline{\mathbf{W}}_1 + \mathbf{V})\mathbf{V}^\top \quad (20)$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and the vectors a^α, b^α, u^α and v^α in Equations 15-18 correspond to the α-th columns of the matrices $\overline{\mathbf{W}}_1$, $\overline{\mathbf{W}}_2^\top$, $\mathbf{U}$ and $\mathbf{V}$, respectively.

Figure 5 shows, for two different initializations, the evolution of the variables a^α and b^α in plain and residual networks with input and output modes and two hidden units. When the variables are initialized to small random values, the dynamics in the plain network initially evolves slowly (Figure 5a, blue); whereas it is much faster in the residual network (Figure 5a, red). This effect is attributable to two factors. First, the added orthonormal vectors u^α and v^α increase the initial velocity of the variables in the residual network. Second, even when we equalize the initial norms of the vectors, a^α and $a^\alpha + v^\alpha$ (and those of the vectors b^α and $b^\alpha + u^\alpha$) in the plain and the residual networks, respectively, we still observe an advantage for the residual network (Figure 5b), because the cooperative and competitive terms are orthogonal to each other in the residual network (or close to orthogonal, depending on the initialization of a^α and b^α; see right-hand side of Equations 15-18), whereas in the plain network they are not necessarily orthogonal and hence can cancel each other (Equations 11-14), thus slowing down convergence.

2.3.2 Networks with more than three-layers

As shown in [18], in linear networks with more than a single hidden layer, assuming that there are orthogonal matrices $\mathbf{R}_l$ and $\mathbf{R}_{l+1}$ for each layer l that diagonalize the initial weight matrix of the corresponding layer (i.e. $\mathbf{R}_{l+1}^\top \mathbf{W}_l(0)\mathbf{R}_l = \mathbf{D}_l$ is a diagonal matrix), dynamics of different singular modes decouple from each other and each mode α evolves according to gradient descent dynamics in an energy landscape described by [18]:

$$E_{plain} = \frac{1}{2\tau}\Big(s_\alpha - \prod_{l=1}^{N_l-1} a_l^\alpha\Big)^2 \quad (21)$$

where a_l^α can be interpreted as the strength of mode α at layer l and N_l is the total number of layers. In residual networks, assuming further that the orthogonal matrices $\mathbf{R}_l$ satisfy $\mathbf{R}_{l+1}^\top \mathbf{R}_l = \mathbf{I}$, the energy function changes to:

$$E_{hyperres} = \frac{1}{2\tau}\Big(s_\alpha - \prod_{l=1}^{N_l-1} (a_l^\alpha + 1)\Big)^2 \quad (22)$$

and in hyper-residual networks, it is:

$$E_{res} = \frac{1}{2\tau}\Big(s_\alpha - \prod_{l=1}^{N_l-1} (a_l^\alpha + l)\Big)^2 \quad (23)$$

Comparing these energy functions, we see that the plain network possesses a rescaling symmetry between mode strengths at different layers manifested in the rescaling symmetry of the product term in the right-hand side of Equation 21. The residual and the hyper-residual models eliminate the rescaling symmetry by adding constants to the mode strength variables. As discussed before, we empirically find the symmetry-breaking

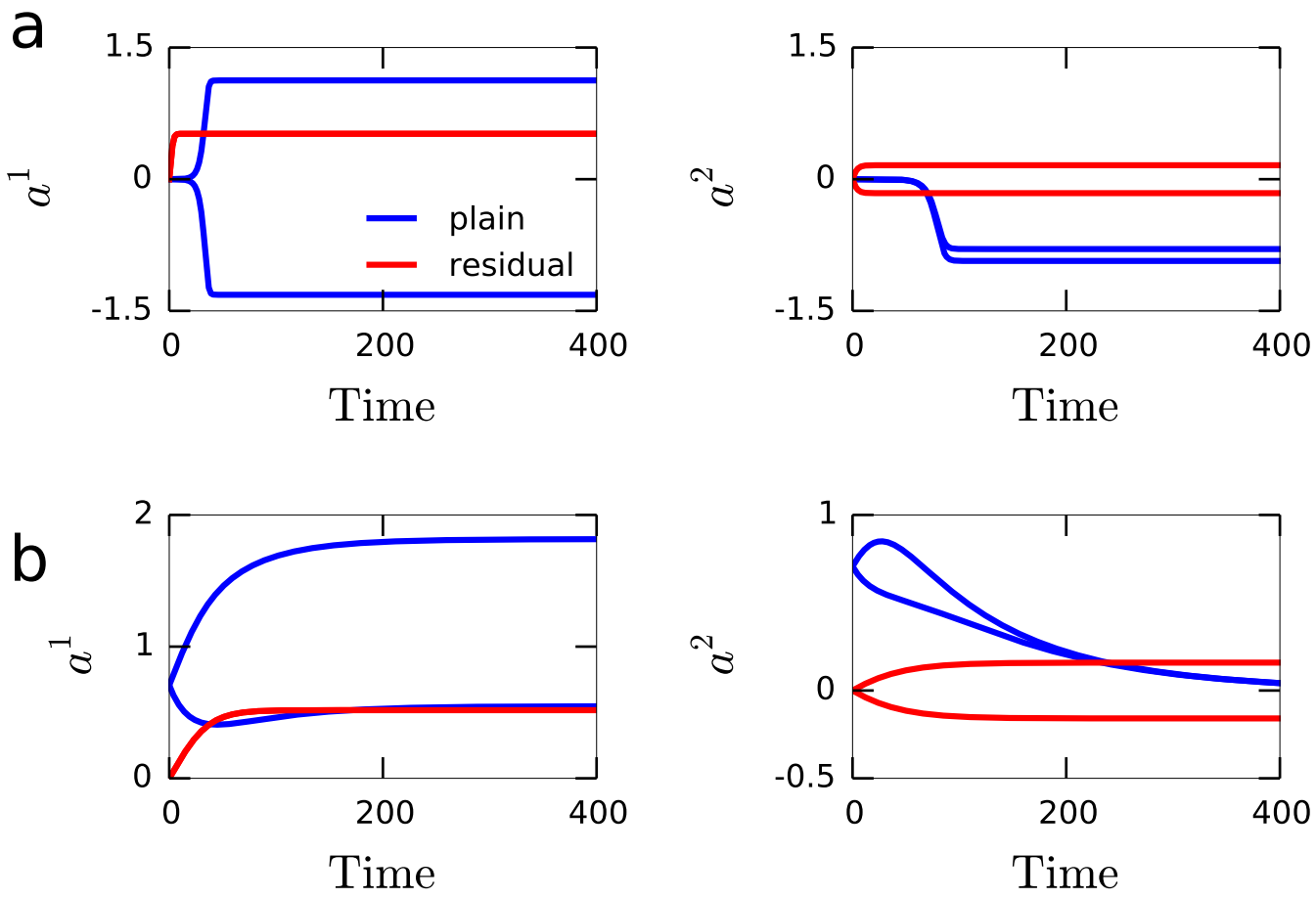

Figure 5: Evolution of a^1 and a^2 in linear plain and residual networks (evolution of b^1 and b^2 proceeds similarly). The weights converge faster in residual networks. Simulation details are as follows: the number of hidden units is 2 (the two solid lines for each color represent the weights associated with the two hidden nodes, e.g. a_1^1 and a_2^1 on the left), the singular values are $s_1 = 3.0$, $s_2 = 1.5$. For the residual network, $u_1 = v_1 = [1/\sqrt{2}, 1/\sqrt{2}]^\top$ and $u_2 = v_2 = [1/\sqrt{2}, -1/\sqrt{2}]^\top$. In (a), the weights of both plain and residual networks are initialized to random values drawn from a Gaussian with zero mean and standard deviation of 0.0001. The learning rate was set to 0.1. In (b), the weights of the plain network are initialized as follows: the vectors a^1 and a^2 are initialized to $[1/\sqrt{2}, 1/\sqrt{2}]^\top$ and the vectors b^1 and b^2 are initialized to $[1/\sqrt{2}, -1/\sqrt{2}]^\top$; the weights of the residual network are all initialized to zero, thus equalizing the initial norms of the vectors a^α and $a^\alpha + v^\alpha$ (and those of the vectors b^α and $b^\alpha + u^\alpha$) between the plain and residual networks. The residual network still converges faster than the plain network. In (b), the learning rate was set to 0.01 to make the different convergence rates of the two networks more visible.

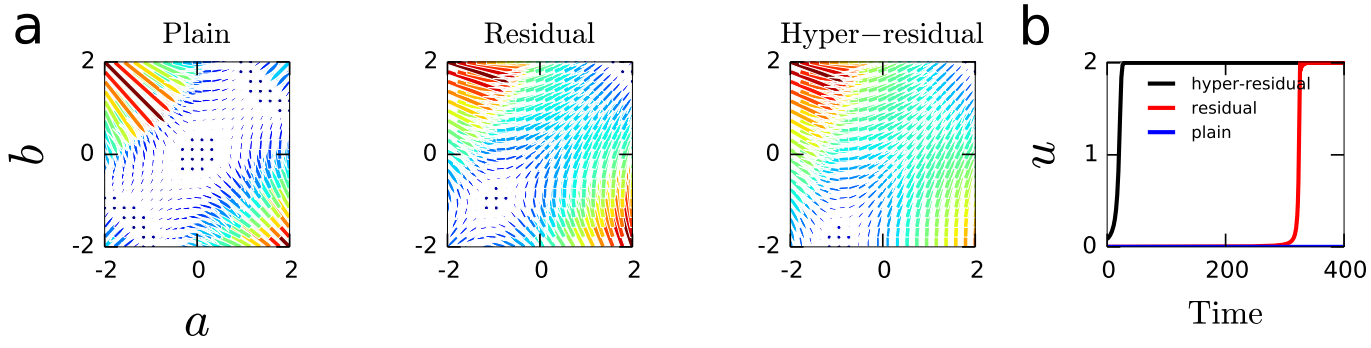

Figure 6: (a) Phase portraits for three-layer plain, residual and hyper-residual linear networks. (b) Evolution of $u = \prod_{l=1}^{N_l-1} a_l$ for 10-layer plain, residual and hyper-residual linear networks. In the plain network, u did not converge to its asymptotic value s within the simulated time window.

to be more effective in the hyper-residual model (Figure 2c). Moreover, the residual energy function (Equation 22) possesses an additional symmetry: the permutation symmetry between mode strength variables a_l^α. This symmetry causes a singularity in the Hessian whenever $a_i^\alpha = a_j^\alpha$ for any pair of layers i, j (Supplementary Note 3). The hyper-residual model eliminates this additional permutation symmetry as well by adding distinct constants to mode strength variables at different layers (Equation 23).

Figure 6a illustrates the effect of skip connections on the phase portrait of a three layer network. The two axes, a and b, represent the mode strength variables for $l = 1$ and $l = 2$, respectively: i.e. $a \equiv a_1^\alpha$ and $b \equiv a_2^\alpha$. The plain network has a saddle point at $(0, 0)$ (Figure 6a; left). The dynamics around this point is slow, hence starting from small random values causes initially very slow learning. The network funnels the dynamics through the unstable manifold $a = b$ to the stable hyperbolic solution corresponding to $ab = s$. Identity skip connections between adjacent layers in the residual architecture move the saddle point to $(-1, -1)$ (Figure 6a; middle). This speeds up the dynamics around the origin, but not as much as in the hyper-residual architecture where the saddle point is moved further away from the origin and the main diagonal to $(-1, -2)$ (Figure 6a; right). We found these effects to be more pronounced in deeper networks. Figure 6b shows the dynamics of learning in 10-layer linear networks, demonstrating a clear advantage for the residual architecture over the plain architecture and for the hyper-residual architecture over the residual architecture.

2.4 Experiments with fully-connected networks

To test the symmetry-breaking hypothesis in more realistic networks, we conducted several experiments with deep fully-connected feedforward networks. In this section, we present the results of these experiments. We recall that the equations describing the plain networks are given by:

$$\mathbf{x}_{l+1} = f(\mathbf{W}_l \mathbf{x}_l + \mathbf{b}_{l+1}) \tag{24}$$

The equations for the residual networks are given by:

$$\mathbf{x}_{l+1} = f(\mathbf{W}_l \mathbf{x}_l + \mathbf{b}_{l+1}) + \mathbf{Q}_l \mathbf{x}_l \tag{25}$$

where $\mathbf{Q}_l$ denotes the skip connectivity matrix, which can be different from the identity matrix, and the equations describing the hyper-residual networks are given by:

$$\mathbf{x}_{l+1} = f(\mathbf{W}_l \mathbf{x}_l + \mathbf{b}_{l+1}) + \mathbf{Q}_l \mathbf{x}_l + \frac{1}{l-1}\Big[\mathbf{Q}_{l-1}\mathbf{x}_{l-1} + \ldots + \mathbf{Q}_1 \mathbf{x}_1\Big] \tag{26}$$

where every layer projects to all layers above itself. We divided the contribution from the non-adjacent layers by $l - 1$, because we found that this performed better than the non-normalized version. As usual, $f(\cdot)$ is chosen to be the ReLU nonlinearity. The networks all have 30 fully-connected hidden layers (20 layers in Figure 11) with $n = 128$ hidden units in each hidden layer. We used the Adam optimizer for training [11], with learning rate 0.0005 and a batch size of 500.

2.4.1 Alternative ways of breaking the permutation symmetry of hidden units

If the success of the residual network architecture can be attributed, at least partly, to symmetry breaking, then alternative ways of breaking the permutation symmetry of the hidden units at the same layer should also improve training. We tested this hypothesis by introducing a particularly simple way of breaking the permutation symmetry of the hidden units. For each layer in the network, we drew random biases from a Gaussian distribution, $\mathcal{N}(0, \sigma^2)$, for each hidden unit in that layer. We used these random values as soft targets for the biases of the hidden units. In particular, we put an l_2-norm penalty on deviations from those bias values. This imposes a particular order on the hidden units according to their target biases and hence breaks their permutation symmetry. Note that setting $\sigma = 0$ corresponds to the standard l_2-norm regularization of the biases, which does not break the symmetry of the hidden units. Hence, we expect the performance to be worse in this case than in cases with properly broken symmetry. On the other hand, although larger values of σ correspond to greater symmetry-breaking, the network also has to perform well in the classification task and very large σ values might be inconsistent with the latter requirement. Therefore, we expect the performance to be optimal for intermediate values of σ. In the experiments reported below, we found the values of σ, the standard deviation of the target bias distribution, and λ, the strength of the bias regularization term, that achieved the best average training accuracy through grid search.

Putting a prior over the biases can be considered as indirectly putting a prior over the activities of the units. More complicated joint priors over hidden unit responses that favor decorrelated [3] or clustered [13] responses have been proposed before. Although the primary motivation for these regularization schemes was to improve the generalizability or interpretability of the learned representations, they can potentially be understood from a symmetry-breaking perspective as well. For example, a prior that favors decorrelated responses can help facilitate the breaking of permutation symmetries between hidden units, even though it does not directly break those symmetries itself (unlike the bias regularizer we have used).

We trained 30-layer feedforward networks on CIFAR-10 and CIFAR-100 (with coarse labels) datasets. Because we are mainly interested in understanding how symmetry-breaking changes the shape of the loss landscape and consequently affects the optimization difficulty, we primarily monitor the training accuracy. Thus, unless otherwise noted, all the results reported below are training accuracies. Figure 7 shows the training accuracy of different models on CIFAR-10 and CIFAR-100 datasets.

For both datasets, the residual network performs the best and the plain network the worst. Symmetry-breaking through bias regularization (BiasSymmBreak, black) leads to a significant improvement over the plain network. Importantly, just putting an l_2-norm penalty on the biases (BiasL2Reg ($\sigma = 0$)) does not improve performance over the plain network. These results are consistent with the symmetry-breaking hypothesis. In all these models, we initialized the biases to zero. We also tested a plain network where the biases were initialized to random values drawn from the same target bias distribution as in the BiasSymmBreak model, but the regularization was not enforced throughout training. We call this network "BiasInit" (green in Figure 7). The BiasInit network performed significantly worse than the BiasSymmBreak network, suggesting that enforcing the symmetry-breaking bias regularization through the entire training period is more beneficial than breaking the symmetry through initialization only.

It is important to note that the BiasSymmBreak network does not perform nearly as well as the residual network. We conjecture that this is due to additional benefits of the residual architecture over and above the symmetry-breaking it enables. One of those advantages is its ability to deal effectively with the vanishing gradients problem encountered in training deep networks [2, 8]. Indeed, we found that the gradient norms with respect to the layer activities do not diminish in earlier layers of the residual network (Figure 8a), demonstrating that it effectively solves the vanishing gradients problem. On the other hand, both in the plain network and in the BiasSymmBreak network, the gradient norms decay quickly as one descends from the top of the network. Adding a single batch normalization layer [10] in the middle of the BiasSymmBreak network alleviates the vanishing gradients problem and brings its performance close to that of the residual network (Figure 8a-b; BiasSymmBreak+BN).

2.4.2 Non-identity skip connections

If the symmetry-breaking hypothesis is correct, there should be nothing special about identity skip connections. Skip connections other than identity should lead to training improvements as well, as long as they break the permutation symmetries. The crucial condition for symmetry-breaking is that the skip connection vector for each unit should disambiguate that unit optimally from all other nodes in that layer. Mathematically, this corresponds to an orthogonality condition on the skip connectivity matrix. We therefore tested random dense orthogonal matrices as skip connectivity matrices. Random dense orthogonal matrices performed at least as well as the identity skip connections (Figure 9, black vs. red). In fact, these dense orthogonal matrices

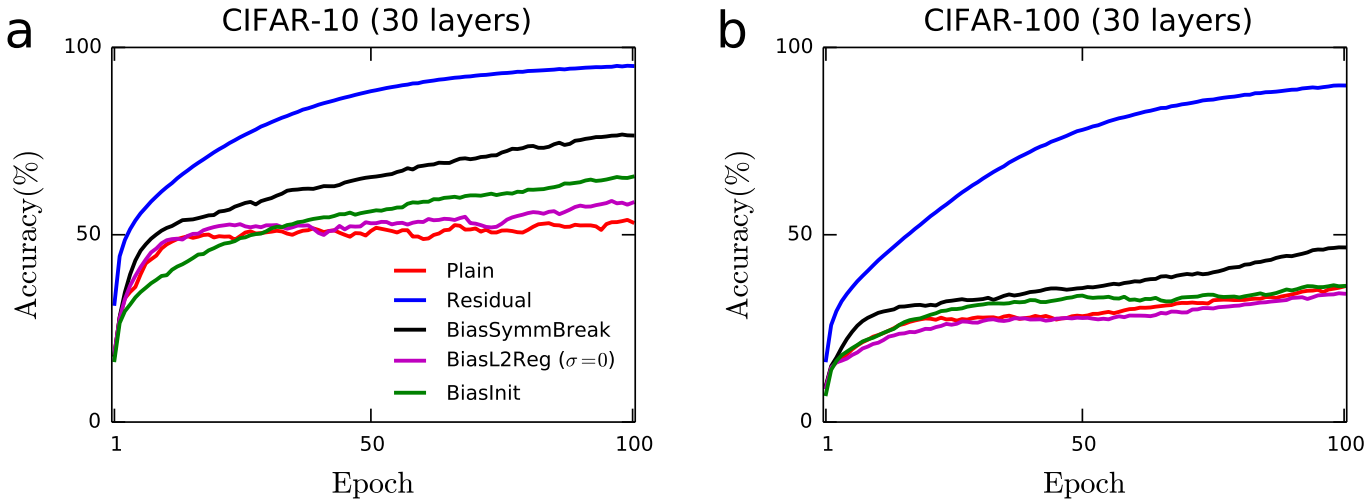

Figure 7: Training accuracy of 30 layer networks on the CIFAR-10 and CIFAR-100 benchmarks. The results shown are average accuracies over 20 independent runs.

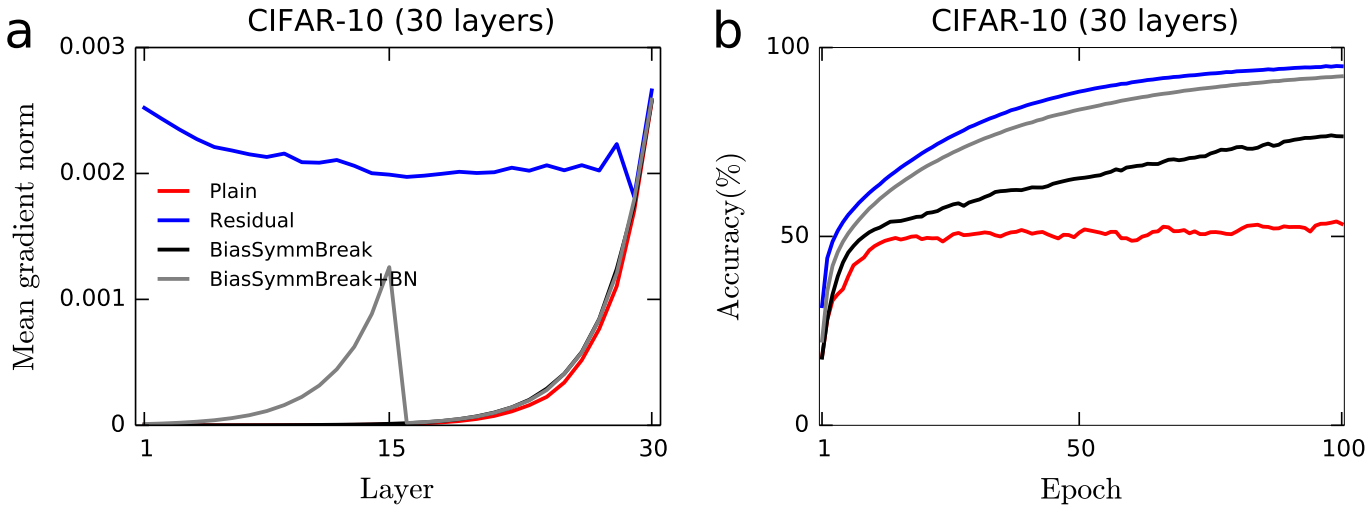

Figure 8: (a) Mean gradient norms with respect to layer activities at different layers of the networks. (b) Training accuracy of the networks on the CIFAR-10 dataset. The performance of the bias symmetry-breaking network with a single batch normalization layer in the middle of the network (BiasSymmBreak+BN) approaches the performance of the residual network.

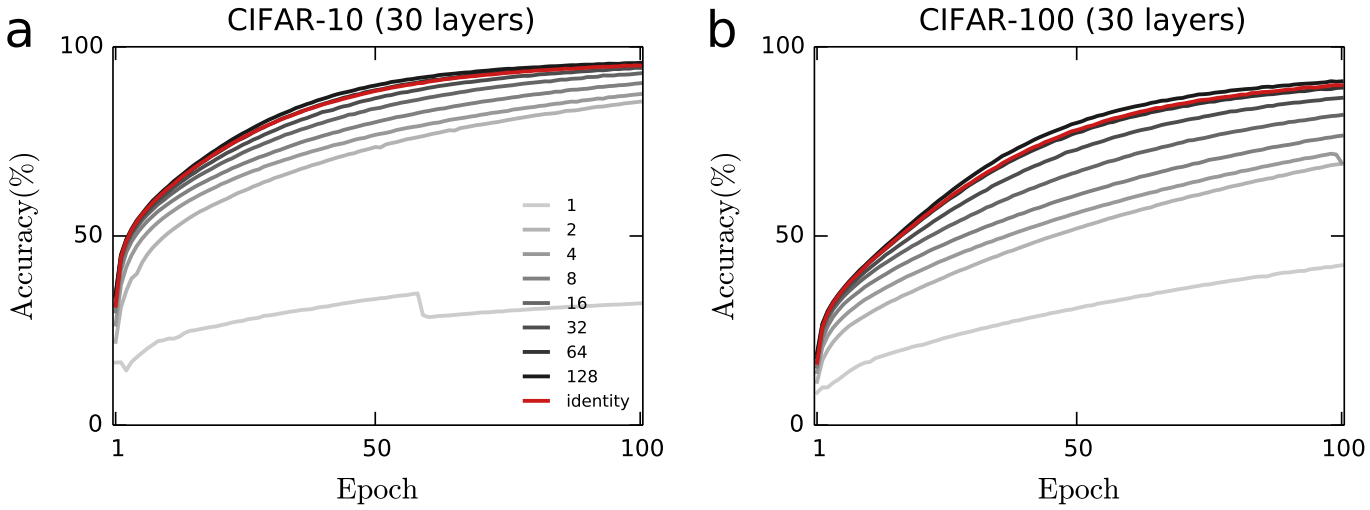

Figure 9: Random dense orthogonal skip connectivity matrices work at least as well as identity skip connections. Increasing the non-orthogonality of the skip connectivity matrix reduces the performance (represented by lighter shades of gray). The results shown are averages over 10 independent runs of the simulations.

performed slightly better than identity skip connections in both CIFAR-10 and CIFAR-100 datasets. We suspect that the reason for this is that sometimes units go silent because of the ReLU nonlinearity. When that happens to two distinct units at layer l, the identity skips cannot disambiguate the corresponding units at the next layer; whereas with dense orthogonal skips, all units at layer l are made use of, so even if some of them go silent, the units at layer $l+1$ can still be disambiguated with the remaining active units.

Next, we gradually decreased the degree of "orthogonality" of the skip connectivity matrix to see how the orthogonality of the matrix affects the performance. Starting from a random dense orthogonal matrix, we first divided the matrix into two halves and copied the first half to the second half. Starting from n orthonormal vectors, this reduces the number of orthonormal vectors to $n/2$. We continued on like this until the columns of the matrix were repeats of a single unit vector. We predict that as the number of orthonormal vectors in the skip connectivity matrix is decreased, the performance should deteriorate, because the symmetry-breaking capacity of the skip connectivity matrix is reduced. Figure 9 shows the results for $n = 128$ hidden units. Darker colors correspond to "more orthogonal" matrices (e.g. 128 means all 128 vectors are orthonormal to each other etc.) The red line is the identity skip connectivity. Clearly more orthogonal skip connectivity matrices yield better performance, consistent with the symmetry breaking hypothesis.

The less orthogonal skip matrices also suffer from the vanishing gradients problem. So, their failure could be partly attributed to the vanishing gradients problem. To control for this effect, we also designed skip connectivity matrices with eigenvalues on the unit circle (hence with eigenvalue spectra equivalent to an orthogonal matrix), but with varying degrees of orthogonality (Supplementary Note 4). More specifically, the columns (or rows) of an orthogonal matrix are orthonormal to each other, hence the covariance matrix of these vectors is the identity matrix. We designed matrices where this covariance matrix was allowed to have non-zero off-diagonal values, reflecting the fact that the vectors are not orthogonal any more. By controlling the magnitude of the correlations between the vectors, we manipulated the degree of orthogonality of the vectors. We achieved this by setting the eigenvalue spectrum of the covariance matrix to be given by $\lambda_i = \exp(-\tau(i-1))$ where λ_i denotes the i-th eigenvalue of the covariance matrix and τ is the parameter that controls the degree of orthogonality: $\tau = 0$ corresponds to the identity covariance matrix, hence to an orthonormal set of vectors, whereas larger values of τ correspond to gradually more correlated vectors. This orthogonality manipulation was done while fixing the eigenvalue spectrum of the skip connectivity matrix to be on the unit circle (see supplementary Note 4 for details). Hence, the effects of this manipulation cannot be attributed to any change in the eigenvalue spectrum, but only to the degree of orthogonality of the skip vectors.

The results of this experiment are shown in Figure 10. Clearly, more orthogonal skip connectivity matrices still perform better than less orthogonal ones, even when their eigenvalue spectrum is fixed, suggesting that the results of the earlier experiment (Figure 9) cannot be explained solely by the vanishing gradients problem.

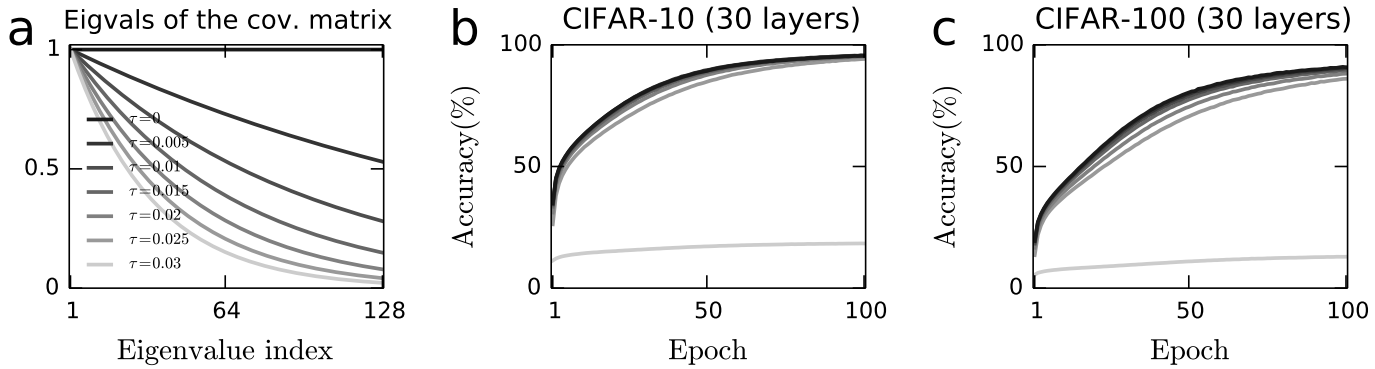

Figure 10: (a) Eigenvalues of the covariance matrices with different τ values: $\tau = 0$ corresponds to orthogonal skip connectivity matrices, larger τ values correspond to less orthogonal matrices, i.e. where skip connectivity vectors for different hidden units are more similar to each other. Note that these eigenvalues are the eigenvalues of the covariance matrix of the skip connectivity vectors. The eigenvalue spectra of the skip connectivity matrices are always fixed to be on the unit circle, hence equivalent to that of an orthogonal matrix, in this experiment. (b) Results on CIFAR-10. (c) Results on CIFAR-100. More orthogonal skip connectivity matrices perform better even when the eigenvalue spectrum is fixed.

2.4.3 Hyper-residual networks and breaking the rescaling symmetry

The results for the hyper-residual networks are shown in Figure 11. In these networks, we chose the skip connectivity matrix between adjacent layers ($\mathbf{Q}_l$ in Equation 26) to be the identity matrix. For the skip connectivity matrices between non-adjacent layers, we used dense random matrices that were twice folded starting from an orthogonal matrix (corresponding to matrices labeled "32" in Figure 9). The results show that these hyper-residual networks perform better than residual networks with identity skip connections only between adjacent layers (represented by the solid red lines in Figure 11).

To provide evidence that the breaking of the rescaling symmetry contributes to the performance improvement in the hyper-residual architecture, we introduced an alternative way of breaking the rescaling symmetry and tested its effectiveness. A particularly simple way of breaking the rescaling symmetry in a plain network is to add constant, untrained biases to each layer in addition to the trainable biases that they possess. This can be thought of as adding skip connections to each layer with a constant input of 1. Specifically, we added biases of the form $al + b$ to each layer, where l denotes the layer index and a and b are hyper-parameters. All units in a given layer received the same constant bias, therefore this manipulation does not break the permutation symmetry of hidden units. Also, note that setting $a = b = 0$ corresponds to a plain network. We performed a grid search over the hyper-parameters a and b. We predicted that the networks where the rescaling symmetry is broken should generally perform better than the plain network. Moreover, adding distinct biases to each layer, as opposed to adding the same bias to each layer, should be more effective at breaking the rescaling symmetry (analogous to how the hyper-residual architecture is more effective at breaking the rescaling symmetry than the residual architecture: Figure 2c), and hence, is predicted to perform better. Both of these predictions were borne out: the best-performing network had a decreasing bias profile as a function of layer index l (black line in Figure 12a). The best-performing network with a constant bias profile is shown by the blue line in Figure 12a. The network with the decreasing bias profile displayed less rescaling-symmetry than the best-performing network with a constant bias profile, as measured by the rescaling experiment discussed in connection with Figure 2c above (compare black and blue bars in Figure 12b). Both of these networks had less rescaling-symmetry, and performed better, than the plain network (Figure 12b-c).

2.5 Symmetries in recurrent neural networks

It is well-known that a recurrent neural network unfolded in time is equivalent to a feedforward network with shared weights between successive layers. In the unfolded network, the permutation symmetry of hidden units does not hold in general due to weight sharing across layers, i.e. across time; the permutation symmetry only holds if the recurrent connectivity matrix is symmetric. The rescaling symmetry does not hold either, since the connectivities between different "layers" cannot be rescaled independently, again due to weight sharing across time. This suggests that recurrent networks are less symmetric than feedforward networks of

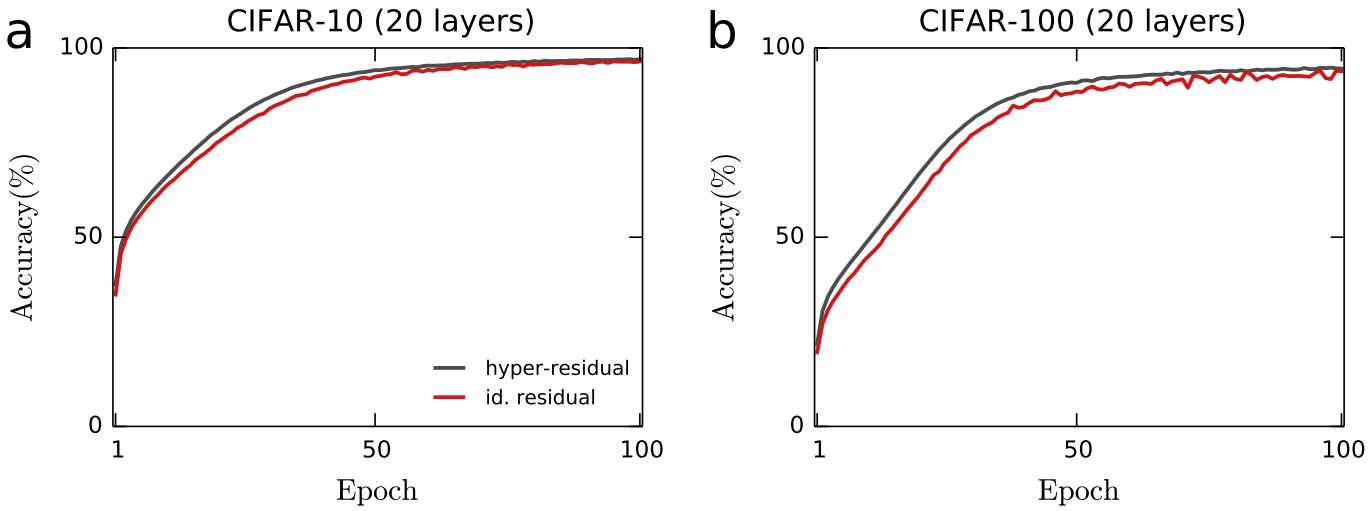

Figure 11: Performance of the hyper-residual architecture and comparison with the residual architecture. (a) Results on CIFAR-10. (b) Results on CIFAR-100. Results shown are the average performances over 10 independent runs of the simulations.

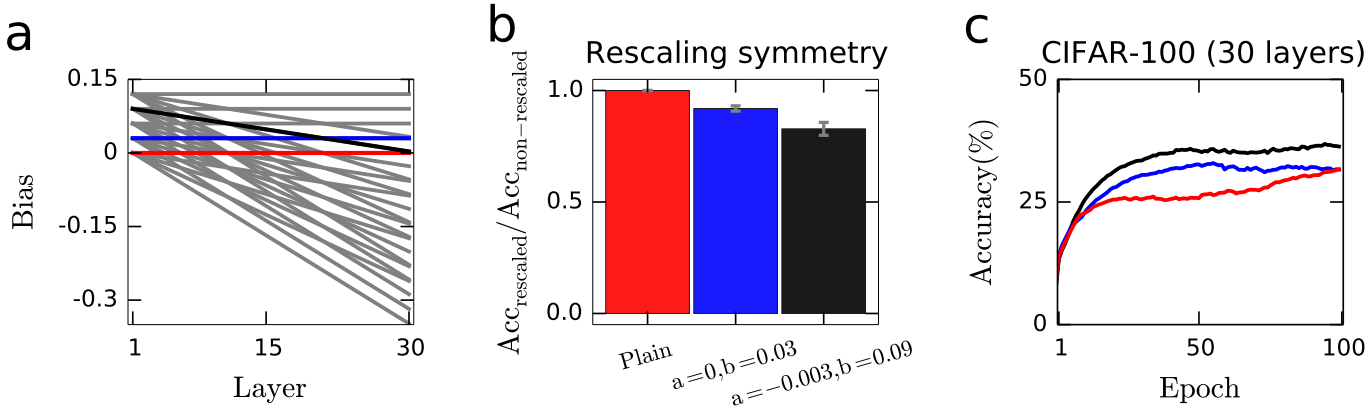

Figure 12: (a) Tested constant bias profiles as a function of hidden layer index. Biases were a linear function of the layer index l: $al + b$ where $a \in \{0, -0.003, -0.006, -0.009, -0.012\}$ and $b \in \{0, 0.03, 0.06, 0.09, 0.12\}$, giving a total of 25 linear functions, depicted by the different lines in the figure. We tested positive slopes a as well, but found that they perform worse than negative slopes. Note that the plain network corresponds to setting $a = b = 0$. (b) Rescaling symmetry of networks with different choices for a and b, highlighted by the corresponding colors in (a). Analogous to Figure 2c, rescaling symmetry is measured by the ratio of the training accuracy of the rescaled network to that of the non-rescaled network. As in Figure 2c, the rescaling is done by first randomly choosing two different connectivity matrices $\mathbf{W}_k$ and $\mathbf{W}_l$ $(k < l)$, then multiplying $\mathbf{W}_k$ by 10 and $\mathbf{W}_l$ by 0.1 (all trained bias vectors between layers k and l are also multiplied by 10). Error bars represent standard errors over 50 independent runs. The difference between the blue and black bars are significant (two-sided t-test, $p < .01$). (c) Performance of the networks on the CIFAR-100 dataset. Results are averages over 50 independent runs of the simulations.

similar size and that the difficulties of training recurrent networks might have more to do with the previously identified vanishing/exploding gradients problem [2, 8] than anything else.

3 Discussion

In this paper, we proposed a novel explanation for the benefits of skip connections in terms of symmetry-breaking. We argued that skip connections between adjacent layers in a multilayer network help break the permutation symmetry between the hidden units and skip connections between a layer and all layers above it break the rescaling symmetry of connectivity matrices between different layers.

We found that dense orthogonal skip connectivity matrices perform slightly better than the typically used identity skip connections. This result can be understood from a symmetry-breaking perspective as identity skip connections use a single hidden unit from the previous layer to disambiguate the hidden units at the next layer, whereas a dense orthogonal matrix uses all the hidden units at the previous layer, and is more robust to the hidden units becoming silent (hence becoming less disambiguating) due to the ReLU nonlinearity. On the other hand, sparse orthogonal matrices such as the identity matrix are computationally more efficient than dense orthogonal matrices.

Our results suggest that symmetry-breaking contributes at least partly to the success of skip connections. However, we emphasize that symmetry-breaking is not the sole explanation for the benefits of skip connections. We presented evidence suggesting that skip connections are also quite effective at dealing with the problem of vanishing/exploding gradients and not every form of symmetry-breaking can be expected to be equally effective at dealing with such additional problems that beset the training of deep networks.

As an intriguing observation, we finally note, from a symmetry-breaking viewpoint, a particular benefit of using differentiated types or classes of neurons with distinct connectivity patterns, a strategy seemingly employed by the brain [5], over using undifferentiated neurons, as is the common practice in artificial neural networks. Specifically, if one divides n undifferentiated neurons into distinct classes, one can reduce their permutation symmetry. For instance, for K classes with d neurons in each, the number of valid permutations that respect class membership is $K!(d!)^K$ which is smaller than the number of permutations of $n = Kd$ $(1 < K < n)$ undifferentiated neurons, which is $n!$.

Symmetry is a remarkably rich and powerful idea in the physical sciences [20]. The results reported in this paper suggest that it could be useful for neural network researchers to pay closer attention to the symmetries inherent in their models as well. As a general design principle, we recommend reducing the symmetries in a model as much as possible, but without reducing the model's expressive capacity at the same time.

Acknowledgments

I would like to thank Xaq Pitkow and Guangyu Robert Yang for helpful discussions and comments on the paper, and the HPC facilities at NYU for making the experiments reported in this paper possible to run. I would like to thank Guangyu Robert Yang further for pointing out an error in an earlier version of the paper regarding symmetries in recurrent neural networks.

References

[1] Amari S, Park H, Ozeki T (2006) Singularities affect dynamics of learning in neuromanifolds. Neural Comput 18(5):1007-65.

[2] Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 5(2):157-66.

[3] Cogswell M, Ahmed F, Girshick R, Zitnick L, Batra D (2015) Reducing overfitting in deep networks by decorrelating representations. arxiv:1511.06068.

[4] Hardt M, Ma T (2016) Identity matters in deep learning. arXiv:1611.04231.

[5] Harris KD, Shepherd GMG (2015) The neocortical circuit: themes and variations. Nat Neurosci 18:170-81.

[6] He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv:1512.03385.

[7] He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. arXiv:1603.05027.

[8] Hochreiter S (1991) Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut f. Informatik, Technische Univ. Munich.

[9] Huang G, Liu Z, Weinberger KQ, van der Maaten L (2016) Densely connected convolutional networks. arXiv:1608.06993.

[10] Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.

[11] Kingma DP, Ba JL (2014) Adam: a method for stochastic optimization. arXiv:1412.6980.

[12] Li S, Jiao J, Han Y, Weissman T (2016) Demystifying ResNet. arXiv:1611.01186.

[13] Liao R, Schwing AG, Zemel RS, Urtasun R (2016) Learning deep parsimonious representations. Advances in Neural Information Processing Systems, 2016, 5076-5084.

[14] Littwin E, Wolf L (2016) The loss surface of residual networks: ensembles and the role of batch normalization. arXiv:1611.02525.

[15] Neyshabur B, Salakhutdinov R, Srebro N (2015) Path-SGD: Path-normalized optimization in deep neural networks. arXiv:1506.02617.

[16] Reidys CM, Stadler PF (2002) Combinatorial landscapes. SIAM Review 44:3-54.

[17] Saad D, Solla SA (1995) On-line learning in soft committee machines. Phys Rev E 52:4225.

[18] Saxe AM, McClelland JM, Ganguli S (2013) Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv:1312.6120.

[19] Wei H, Zhang J, Cousseau F, Ozeki T, Amari S (2008) Dynamics of learning near singularities in layered networks. Neural Comput 20(3):813-43.

[20] Wilczek FA (2015) *A Beautiful Question: Finding Nature's Deep Design.* Allen Lane.

Supplementary Materials

Supplementary Note 1: Singularity of the Hessian in multilayer networks

Because the cost function can be expressed as a sum over training examples, it is enough to consider the cost for a single example: $E = \frac{1}{2}||\mathbf{y} - \mathbf{x}_L||^2 \equiv \frac{1}{2}\mathbf{e}^\top\mathbf{e}$, where $\mathbf{x}_l$ are defined recursively as $\mathbf{x}_l = f(\mathbf{W}_{l-1}\mathbf{x}_{l-1})$ for $l = 2, \ldots, L$. We denote the inputs to units at layer l by the vector $\mathbf{h}_l$: $\mathbf{h}_l = \mathbf{W}_{l-1}\mathbf{x}_{l-1}$. We ignore the biases for simplicity. The derivative of the energy function with respect to a single weight $\mathbf{W}_{l,ij}$ between layers l and $l+1$ is given by:

$$\frac{\partial E}{\partial \mathbf{W}_{l,ij}} = -\begin{bmatrix} 0 \\ \vdots \\ f'(\mathbf{h}_{l+1,i})\mathbf{x}_{l,j} \\ \vdots \\ 0 \end{bmatrix}^\top \mathbf{W}_{l+1}^\top \mathrm{diag}(\mathbf{f}'_{l+2})\mathbf{W}_{l+2}^\top \mathrm{diag}(\mathbf{f}'_{l+3}) \cdots \mathbf{W}_{L-1}^\top \mathrm{diag}(\mathbf{f}'_L)\mathbf{e} \tag{27}$$

Now, consider a different connection between the same output unit i at layer $l+1$ and a different input unit j' at layer l. The crucial thing to note is that if the units j and j' have the same set of incoming weights, then the derivative of the cost function with respect to $\mathbf{W}_{l,ij}$ becomes identical to its derivative with respect to $\mathbf{W}_{l,ij'}$: $\partial E/\partial \mathbf{W}_{l,ij} = \partial E/\partial \mathbf{W}_{l,ij'}$. This is because in this condition $\mathbf{x}_{l,j'} = \mathbf{x}_{l,j}$ for all possible inputs and all the remaining terms in Equation 27 are independent of the input index j. Thus, the columns (or rows) corresponding to the connections $\mathbf{W}_{l,ij}$ and $\mathbf{W}_{l,ij'}$ in the Hessian become identical, making the Hessian degenerate. This is a re-statement of the simple observation that when the units j and j' have the same set of incoming weights, the parameters $\mathbf{W}_{l,ij}$ and $\mathbf{W}_{l,ij'}$ become non-identifiable (only their sum is identifiable).

Supplementary Note 2: Singularity of the Hessian in linear three-layer networks

We start from Equation 10 which gives the energy function of a plain linear three-layer network. Taking the derivative with respect to a single input-to-hidden layer weight, a_i^α:

$$\frac{\partial E}{\partial a_i^\alpha} = -(s_\alpha - a^\alpha \cdot b^\alpha)b_i^\alpha + \sum_{\beta \neq \alpha}(a^\alpha \cdot b^\beta)b_i^\beta \tag{28}$$

and the second derivatives are as follows:

$$\frac{\partial^2 E}{\partial (a_i^\alpha)^2} = (b_i^\alpha)^2 + \sum_{\beta \neq \alpha} (b_i^\beta)^2 = \sum_\beta (b_i^\beta)^2 \tag{29}$$

$$\frac{\partial^2 E}{\partial a_i^\alpha \partial a_j^\alpha} = b_j^\alpha b_i^\alpha + \sum_{\beta \neq \alpha} b_j^\beta b_i^\beta = \sum_\beta b_i^\beta b_j^\beta \tag{30}$$

Note that the second derivatives are independent of mode index α, reflecting the fact that the energy function is invariant to a permutation of the mode indices. Furthermore, when $b_i^\beta = b_j^\beta$ for all β, the columns in the Hessian corresponding to a_i^α and a_j^α become identical, causing an additional degeneracy reflecting the non-identifiability of a_i^α and a_j^α. A similar derivation establishes that $a_i^\beta = a_j^\beta$ for all β also leads to a degeneracy in the Hessian, this time reflecting the non-identifiability of b_i^α and b_j^α.

When we add skip connections between adjacent layers, i.e. in the residual architecture, the energy function changes as follows:

$$E = \frac{1}{2} \sum_\alpha (s_\alpha - (a^\alpha + v^\alpha) \cdot (b^\alpha + u^\alpha))^2 + \frac{1}{2} \sum_{\alpha \neq \beta} ((a^\alpha + v^\alpha) \cdot (b^\beta + u^\beta))^2 \tag{31}$$

and straightforward algebra yields the following second derivatives:

$$\frac{\partial^2 E}{\partial (a_i^\alpha)^2} = \sum_\beta (b_i^\beta + u_i^\beta)^2 \tag{32}$$

$$\frac{\partial^2 E}{\partial a_i^\alpha \partial a_j^\alpha} = \sum_\beta (b_i^\beta + u_i^\beta)(b_j^\beta + u_j^\beta) \tag{33}$$

Unlike in the plain network, setting $b_i^\beta = b_j^\beta$ for all β (or $a_i^\beta = a_j^\beta$ for all β) does not lead to a degeneracy here, thanks to the orthogonal vectors u^β.

Supplementary Note 3: Singularity of the Hessian in reduced linear multilayer networks with skip connections

Under the conditions required for the dynamics of different singular modes to decouple from each other, as worked out in [18], for each singular mode, the cost function of a linear multilayer network with skip connections between adjacent layers can be written as:

$$E = \frac{1}{2} \Big[s - \prod_{l=1}^{L} (a_l + 1) \Big]^2 \equiv \frac{1}{2} (s - u)^2 \tag{34}$$

The derivative with respect to a_i is given by:

$$\frac{\partial E}{\partial a_i} = -(s - u) \prod_{l \neq i} (a_l + 1) \tag{35}$$

and the second derivatives are:

$$\frac{\partial^2 E}{\partial a_i^2} = \Big[\prod_{l \neq i} (a_l + 1) \Big]^2 \tag{36}$$

$$\frac{\partial^2 E}{\partial a_i \partial a_k} = \Big[2 \prod_l (a_l + 1) - s \Big] \prod_{l \neq i,k} (a_l + 1) \tag{37}$$

It is easy to check that the columns (or rows) corresponding to a_i and a_j in the Hessian become identical when $a_i = a_j$, making the Hessian degenerate. The hyper-residual architecture breaks this degeneracy by adding distinct constants to a_i and a_j (and to all other variables).

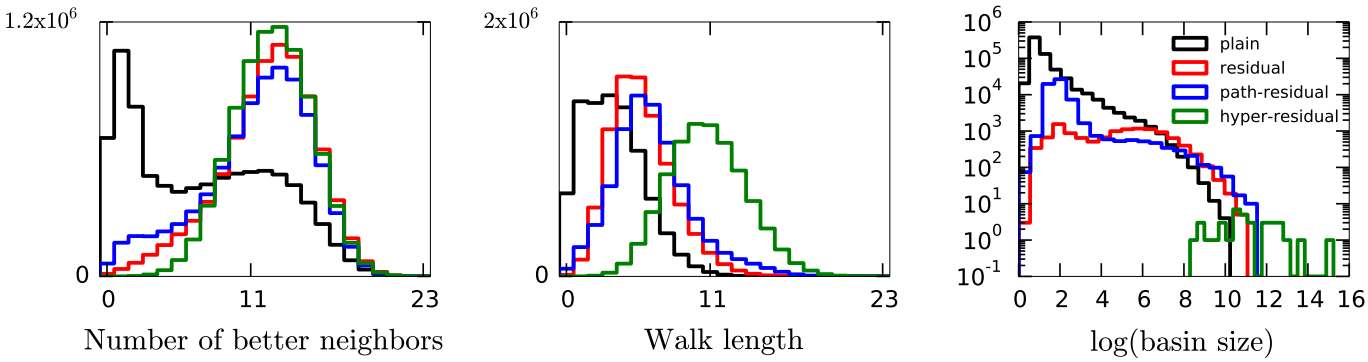

Figure S1: Characterization of the loss landscapes of toy networks with binary parameters. Similar to Figure 4, but with biases of the units randomly and independently drawn from a standard normal distribution.

Supplementary Note 4: Designing skip connectivity matrices with a varying degree of orthogonality and with eigenvalues on the unit circle

We generated the covariance matrix of the eigenvectors by $\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, where $\mathbf{Q}$ is a random orthogonal matrix and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues, $\mathbf{\Lambda}_{ii} = \exp(-\tau(i-1))$, as explained in the main text. We find the correlation matrix through $\mathbf{R} = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2}$ where $\mathbf{D}$ is the diagonal matrix of the variances: i.e. $\mathbf{D}_{ii} = \mathbf{S}_{ii}$. We take the Cholesky decomposition of the correlation matrix, $\mathbf{R} = \mathbf{T}\mathbf{T}^\top$. Then the designed skip connectivity matrix is given by $\mathbf{\Sigma} = \mathbf{T}\mathbf{U}\mathbf{L}\mathbf{U}^{-1}\mathbf{T}^{-1}$, where $\mathbf{L}$ and $\mathbf{U}$ are the matrices of eigenvalues and eigenvectors of another randomly generated orthogonal matrix, $\mathbf{O}$: i.e. $\mathbf{O} = \mathbf{U}\mathbf{L}\mathbf{U}^\top$. With this construction, $\mathbf{\Sigma}$ has the same eigenvalue spectrum as $\mathbf{O}$, however the eigenvectors of $\mathbf{\Sigma}$ are linear combinations of the eigenvectors of $\mathbf{O}$ such that their correlation matrix is given by $\mathbf{R}$. Thus, the eigenvectors of $\mathbf{\Sigma}$ are not orthogonal to each other unless $\tau = 0$. Larger values of τ yield more correlated, hence less orthogonal, eigenvectors.