Exact solutions to the nonlinear dynamics of learning in deep linear neural networks

Andrew M. Saxe (asaxe@stanford.edu)
Department of Electrical Engineering

James L. McClelland (mcclelland@stanford.edu)
Department of Psychology

Surya Ganguli (sganguli@stanford.edu)
Department of Applied Physics
Stanford University, Stanford, CA 94305 USA

Abstract

Despite the widespread practical success of deep learning methods, our theoretical understanding of the dynamics of learning in deep neural networks remains quite sparse. We attempt to bridge the gap between the theory and practice of deep learning by systematically analyzing learning dynamics for the restricted case of deep linear neural networks. Despite the linearity of their input-output map, such networks have nonlinear gradient descent dynamics on weights that change with the addition of each new hidden layer. We show that deep linear networks exhibit nonlinear learning phenomena similar to those seen in simulations of nonlinear networks, including long plateaus followed by rapid transitions to lower error solutions, and faster convergence from greedy unsupervised pretraining initial conditions than from random initial conditions. We provide an analytical description of these phenomena by finding new exact solutions to the nonlinear dynamics of deep learning. Our theoretical analysis also reveals the surprising finding that as the depth of a network approaches infinity, learning speed can nevertheless remain finite: for a special class of initial conditions on the weights, very deep networks incur only a finite, depth independent, delay in learning speed relative to shallow networks. We show that, under certain conditions on the training data, unsupervised pretraining can find this special class of initial conditions, while scaled random Gaussian initilizations cannot. We further exhibit a new class of random orthogonal initial conditions on weights that, like unsupervised pre-training, enjoys depth independent learning times, and we show that these initial conditions also lead to faithful propagation of gradients even in deep nonlinear networks, as long as they operate just beyond the edge of chaos.

Deep learning methods have realized impressive performance in a range of applications, from visual object classification [1, 2, 3] to speech recognition [4] and natural language processing [5, 6]. These successes have been achieved despite the noted difficulty of training such deep architectures [7, 8, 9, 10, 11]. Indeed, many explanations for the difficulty of deep learning have been advanced in the literature, including the presence of many local minima, low curvature regions due to saturating nonlinearities, and exponential growth or decay of back-propagated gradients [12, 13, 14, 15]. Furthermore, many neural network simulations have observed

strikingly nonlinear learning dynamics, including long plateaus of little apparent improvement followed by almost stage-like transitions to better performance. However, a quantitative, analytical understanding of the rich dynamics of deep learning remains elusive. For example, what determines the time scales over which deep learning unfolds? How does training speed retard with depth? Under what conditions will greedy unsupervised pretraining speed up learning? And how do the final learned internal representations depend on the statistical regularities inherent in the training data?

Here we provide an exact analytical theory of learning in deep linear neural networks that quantitatively answers these questions for this restricted setting. Because of its linearity, the input-output map of a deep linear network can always be rewritten as a shallow network. In this sense, a linear network does not gain expressive power from depth, and hence will underfit and perform poorly on complex real world problems. But while it lacks this important aspect of practical deep learning systems, a deep linear network can nonetheless exhibit highly nonlinear learning dynamics, and these dynamics change with increasing depth. Indeed, the training error, as a function of the network weights, is non-convex, and gradient descent dynamics on this non-convex error surface exhibits a subtle interplay between different weights across multiple layers of the network. Hence deep linear networks provide an important starting point for understanding deep learning dynamics.

To answer these questions, we derive and analyze a set of nonlinear coupled differential equations describing learning dynamics on weight space as a function of the statistical structure of the inputs and outputs. We find exact time-dependent solutions to these nonlinear equations, as well as find conserved quantities in the weight dynamics arising from symmetries in the error function. These solutions provide intuition into how a deep network successively builds up information about the statistical structure of the training data and embeds this information into its weights and internal representations. Moreover, we compare our analytical solutions of learning dynamics in deep linear networks to numerical simulations of learning dynamics in deep non-linear networks, and find that our analytical solutions provide a reasonable approximation. Our solutions also reflect nonlinear phenomena seen in simulations, including alternating plateaus and sharp periods of rapid improvement. Indeed, it has been shown previously [16] that this nonlinear learning dynamics in deep linear networks is sufficient to qualitatively capture aspects of the progressive, hierarchical differentiation of conceptual structure seen in infant development. Next, we apply these solutions to investigate the commonly used greedy layer-wise pretraining strategy for training deep networks [17, 18], and recover conditions under which such pretraining speeds learning. We show that these conditions are approximately satisfied for the MNIST dataset, and that unsupervised pretraining therefore confers an optimization advantage for deep linear networks applied to MNIST. Finally, we exhibit a new class of random orthogonal initial conditions on weights that, in linear networks, provide depth independent learning times, and we show that these initial conditions also lead to faithful propagation of gradients even in deep nonlinear networks, as long as they operate just beyond the edge of chaos.

1 General learning dynamics of gradient descent

We begin by analyzing learning in a three layer network (input, hidden, and output) with linear activation functions (Fig 1). We let N_i be the number of neurons in layer i. The input-output map of the network is $y = W^{32}W^{21}x$. We wish to train the network to learn a particular input-output map from a set of P training examples $\{x^\mu, y^\mu\}, \mu = 1, \ldots, P$. Training is accomplished via gradient descent on the squared error $\sum_{\mu=1}^{P} \left\| y^\mu - W^{32}W^{21}x^\mu \right\|^2$ between the desired feature output, and the network's feature output. This gradient

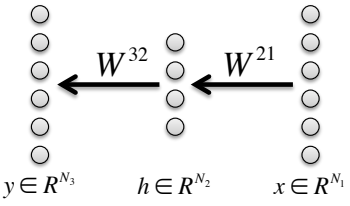

Figure 1: The three layer network analyzed in this section.

descent procedure yields the batch learning rule

$$\Delta W^{21} = \lambda \sum_{\mu=1}^{P} {W^{32}}^T \left(y^\mu x^{\mu T} - W^{32} W^{21} x^\mu x^{\mu T} \right), \qquad \Delta W^{32} = \lambda \sum_{\mu=1}^{P} \left(y^\mu x^{\mu T} - W^{32} W^{21} x^\mu x^{\mu T} \right) {W^{21}}^T, \tag{1}$$

where λ is a small learning rate. As long as λ is sufficiently small, we can take a continuous time limit to obtain the dynamics,

$$\tau \frac{d}{dt} W^{21} = {W^{32}}^T \left(\Sigma^{31} - W^{32} W^{21} \Sigma^{11} \right), \qquad \tau \frac{d}{dt} W^{32} = \left(\Sigma^{31} - W^{32} W^{21} \Sigma^{11} \right) {W^{21}}^T, \tag{2}$$

where $\Sigma^{11} \equiv \sum_{\mu=1}^{P} x^\mu x^{\mu T}$ is an $N_1 \times N_1$ input correlation matrix, $\Sigma^{31} \equiv \sum_{\mu=1}^{P} y^\mu x^{\mu T}$ is an $N_3 \times N_1$ input-output correlation matrix, and $\tau \equiv \frac{1}{\lambda}$. Here t measures time in units of iterations; as t varies from 0 to 1, the network has seen P examples corresponding to one iteration. Despite the linearity of the network's input-output map, the gradient descent learning dynamics given in Eqn (2) constitutes a complex set of coupled nonlinear differential equations with up to cubic interactions in the weights.

1.1 Learning dynamics with orthogonal inputs

Our fundamental goal is to understand the dynamics of learning in (2) as a function of the input statistics Σ^{11} and input-output statistics Σ^{31}. In general, the outcome of learning will reflect an interplay between input correlations, described by Σ^{11}, and the input-output correlations described by Σ^{31}. To begin, though, we further simplify the analysis by focusing on the case of orthogonal input representations where $\Sigma^{11} = I$. This assumption will hold exactly for whitened input data, a widely used preprocessing step.

Because we have assumed orthogonal input representations ($\Sigma^{11} = I$), the input-output correlation matrix contains all of the information about the dataset used in learning, and it plays a pivotal role in the learning dynamics. We consider its singular value decomposition (SVD)

$$\Sigma^{31} = U^{33} S^{31} {V^{11}}^T = \textstyle\sum_{\alpha=1}^{N_1} s_\alpha u_\alpha v_\alpha^T, \tag{3}$$

which will be central in our analysis. Here V^{11} is an $N_1 \times N_1$ orthogonal matrix whose columns contain *input-analyzing* singular vectors v_α that reflect independent modes of variation in the input, U^{33} is an $N_3 \times N_3$ orthogonal matrix whose columns contain *output-analyzing* singular vectors u_α that reflect independent modes of variation in the output, and S^{31} is an $N_3 \times N_1$ matrix whose only nonzero elements are on the diagonal; these elements are the singular values $s_\alpha, \alpha = 1, \ldots, N_1$ ordered so that $s_1 \geq s_2 \geq \cdots \geq s_{N_1}$.

Now, performing the change of variables on synaptic weight space, $W^{21} = \overline{W}^{21} {V^{11}}^T$, $W^{32} = U^{33} \overline{W}^{32}$, the dynamics in (2) simplify to

$$\tau \frac{d}{dt} \overline{W}^{21} = {\overline{W}^{32}}^T (S^{31} - \overline{W}^{32} \overline{W}^{21}), \qquad \tau \frac{d}{dt} \overline{W}^{32} = (S^{31} - \overline{W}^{32} \overline{W}^{21}) {\overline{W}^{21}}^T. \tag{4}$$

To gain intuition for these equations, note that while the matrix elements of W^{21} and W^{32} connected neurons in one layer to neurons in the next layer, we can think of the matrix element ${\overline{W}^{21}}_{i\alpha}$ as connecting input mode v_α to hidden neuron i, and the matrix element ${\overline{W}^{32}}_{\alpha i}$ as connecting hidden neuron i to output mode u_α. Let a^α be the α^{th} column of W^{21}, and let $b^{\alpha T}$ be the α^{th} row of $\overline{W}^{32}$. Intuitively, a^α is a column vector of N_2 synaptic weights presynaptic to the hidden layer coming from input mode α, and b^α is a column vector of N_2 synaptic weights postsynaptic to the hidden layer going to output mode α. In terms of these variables, or connectivity modes, the learning dynamics in (4) become

$$\tau \frac{d}{dt} a^\alpha = (s_\alpha - a^\alpha \cdot b^\alpha)\, b^\alpha - \sum_{\gamma \neq \alpha} b^\gamma \, (a^\alpha \cdot b^\gamma), \qquad \tau \frac{d}{dt} b^\alpha = (s_\alpha - a^\alpha \cdot b^\alpha)\, a^\alpha - \sum_{\gamma \neq \alpha} a^\gamma \, (b^\alpha \cdot a^\gamma). \tag{5}$$

Note that $s_\alpha = 0$ for $\alpha > N_1$. These dynamics arise from gradient descent on the energy function

$$E = \frac{1}{2\tau} \sum_\alpha (s_\alpha - a^\alpha \cdot b^\alpha)^2 + \frac{1}{2\tau} \sum_{\alpha \neq \beta} (a^\alpha \cdot b^\beta)^2, \tag{6}$$

and display an interesting combination of cooperative and competitive interactions. Consider the first terms in each equation. In these terms, the connectivity modes from the two layers, a^α and b^α associated with the same input-output mode of strength s^α, cooperate with each other to drive each other to larger magnitudes as well as point in similar directions in the space of hidden units; in this fashion these terms drive the product of connectivity modes $a^\alpha \cdot b^\alpha$ to reflect the input-output mode strength s^α. The second terms describe competition between the connectivity modes in the first (a^α) and second (b^β) layers associated with different input modes α and β. This yields a symmetric, pairwise repulsive force between all distinct pairs of first and second layer connectivity modes, driving the network to a decoupled regime in which the different connectivity modes become orthogonal.

1.2 The final outcome of learning

The fixed point structure of gradient descent learning was worked out in [19]. In the language of the connectivity modes, a necessary condition for a fixed point is $a^\alpha \cdot b^\beta = s_\alpha \delta_{\alpha\beta}$, while a^α and b^α are zero whenever $s_\alpha = 0$. To satisfy these relations for undercomplete hidden layers ($N_2 < N_1, N_2 < N_3$), a^α and b^α can be nonzero for at most N_2 values of α. Since there are $\text{rank}(\Sigma^{31}) \equiv r$ nonzero values of s_α, there are $\binom{r}{N_2}$ families of fixed points. However, all of these fixed points are unstable, except for the one in which only the first N_2 strongest modes, i.e. a^α and b^α for $\alpha = 1, \ldots, N_2$ are active. Thus remarkably, the dynamics in (5) has only saddle points and no non-global local minima [19]. In terms of the original synaptic variables W^{21} and W^{32}, all globally stable fixed points satisfy

$$W^{32} W^{21} = \sum_{\alpha=1}^{N_2} s_\alpha u_\alpha v_\alpha^T. \tag{7}$$

Hence when learning has converged, the network will represent the closest rank N_2 approximation to the true input-output correlation matrix. In this work, we are interested in understanding the dynamical weight trajectories and learning time scales that lead to this final fixed point.

1.3 The time course of learning

It is difficult though to exactly solve (5) starting from arbitrary initial conditions because of the competitive interactions between different input-output modes. Therefore, to gain intuition for the general dynamics, we restrict our attention to a special class of initial conditions of the form a^α and $b^\alpha \propto r^\alpha$ for $\alpha = 1, \ldots, N_2$, where $r^\alpha \cdot r^\beta = \delta_{\alpha\beta}$, with all other connectivity modes a^α and b^α set to zero (see [20] for solutions to a partially overlapping but distinct set of initial conditions, further discussed in Supplementary Appendix A). Here r^α is a fixed collection of N_2 vectors that form an orthonormal basis for synaptic connections from an input or output mode onto the set of hidden units. Thus for this set of initial conditions, a^α and b^α point in the same direction for each alpha and differ only in their scalar magnitudes, and are orthogonal to all other connectivity modes. Such an initialization can be obtained by computing the SVD of Σ^{31} and taking $W^{32} = U^{33} D_a R^T, W^{21} = R D_b V^{11^T}$ where D_a, D_b are diagonal, and R is an arbitrary orthogonal matrix; however, as we show

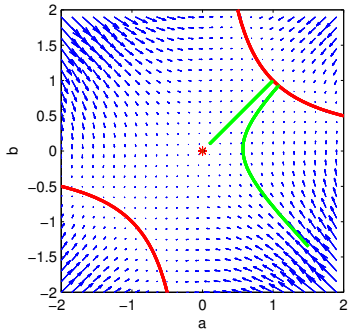

Figure 2: Vector field (blue), stable manifold (red) and two solution trajectories (green) for the two dimensional dynamics of a and b in (8), with $\tau = 1, s = 1$.

in subsequent experiments, the solutions we find are also excellent approximations to trajectories from small random initial conditions.
It is straightforward to verify that starting from these initial conditions, a^α and b^α will remain parallel to r^α for all future time. Furthermore, because the different active modes are orthogonal to each other, they do not

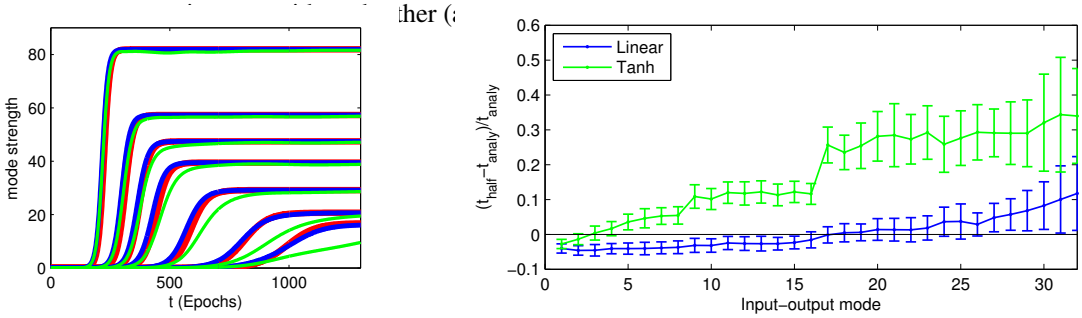

Figure 3: **Left:** Dynamics of learning in a three layer neural network. Curves show the strength of the network's representation of seven modes of the input-output correlation matrix over the course of learning. Red traces show analytical curves from Eqn. 12. Blue traces show simulation of full dynamics of a linear network (Eqn. (2)) from small random initial conditions. Green traces show simulation of a nonlinear three layer network with tanh activation functions. To generate mode strengths for the nonlinear network, we computed the nonlinear network's evolving input-output correlation matrix, and plotted the diagonal elements of ${U^{33}}^T \Sigma^{31}{}_{tanh} V^{11}$ over time. The training set consists of 32 orthogonal input patterns, each associated with a 1000-dimensional feature vector generated by a hierarchical diffusion process described in [16] with a five level binary tree and flip probability of 0.1. Modes 1, 2, 3, 5, 12, 18, and 31 are plotted with the rest excluded for clarity. Network training parameters were $\lambda = 0.5e^{-3}, N_2 = 32, u_0 = 1e^{-6}$. **Right:** Delay in learning due to competitive dynamics and sigmoidal nonlinearities. Vertical axis shows the difference between simulated time of half learning and the analytical time of half learning, as a fraction of the analytical time of half learning. Error bars show standard deviation from 100 simulations with random initializations.

Thus this class of conditions defines an invariant manifold in weight space where the modes evolve independently of each other.

If we let $a = a^\alpha \cdot r^\alpha$, $b = b^\alpha \cdot r^\alpha$, and $s = s^\alpha$, then the dynamics of the scalar projections (a, b) obeys,

$$\tau \frac{d}{dt} a = b\,(s - ab), \qquad \tau \frac{d}{dt} b = a\,(s - ab). \tag{8}$$

Thus our ability to decouple the connectivity modes yields a dramatically simplified two dimensional nonlinear system. These equations can by solved by noting that they arise from gradient descent on the error,

$$E(a, b) = \tfrac{1}{2\tau}(s - ab)^2. \tag{9}$$

This implies that the product ab monotonically approaches the fixed point s from its initial value. Moreover, $E(a, b)$ satisfies a symmetry under the one parameter family of scaling transformations $a \to \lambda a$, $b \to \frac{b}{\lambda}$. This symmetry implies, through Noether's theorem, the existence of a conserved quantity, namely $a^2 - b^2$, which is a constant of motion. Thus the dynamics simply follows hyperbolas of constant $a^2 - b^2$ in the (a, b) plane until it approaches the hyperbolic manifold of fixed points, $ab = s$. The origin $a = 0, b = 0$ is also a fixed point, but is unstable. Fig. 2 shows a typical phase portrait for these dynamics.

As a measure of the timescale of learning, we are interested in how long it takes for ab to approach s from any given initial condition. The case of unequal a and b is treated in the Supplementary Appendix A due to space constraints. Here we pursue an explicit solution with the assumption that $a = b$, a reasonable limit

when starting with small random initial conditions. We can then track the dynamics of $u \equiv ab$, which from (8) obeys

$$\tau \frac{d}{dt} u = 2u(s-u). \tag{10}$$

This equation is separable and can be integrated to yield

$$t = \tau \int_{u_0}^{u_f} \frac{du}{2u(s-u)} = \frac{\tau}{2s} \ln \frac{u_f(s-u_0)}{u_0(s-u_f)}. \tag{11}$$

Here t is the time it takes for u to travel from u_0 to u_f. If we assume a small initial condition $u_0 = \epsilon$, and ask when u_f is within ϵ of the fixed point s, i.e. $u_f = s - \epsilon$, then the learning timescale in the limit $\epsilon \to 0$ is $t = \tau/s \ln(s/\epsilon) = O(\tau/s)$ (with a weak logarithmic dependence on the cutoff). This yields a key result: the timescale of learning of each input-output mode α of the correlation matrix Σ^{31} is inversely proportional to the correlation strength s_α of the mode. Thus the stronger an input-output relationship, the quicker it is learned.

We can also find the entire time course of learning by inverting (11) to obtain

$$u_f(t) = \frac{s e^{2st/\tau}}{e^{2st/\tau} - 1 + s/u_0}. \tag{12}$$

This time course describes the temporal evolution of the product of the magnitudes of all weights from an input mode (with correlation strength s) into the hidden layers, and from the hidden layers to the same output mode. If this product starts at a small value $u_0 < s$, then it displays a sigmoidal rise which asymptotes to s as $t \to \infty$. This sigmoid can exhibit sharp transitions from a state of no learning to full learning. This analytical sigmoid learning curve is shown in Fig. 3 to yield a reasonable approximation to learning curves in linear networks that start from random initial conditions that are not on the orthogonal, decoupled invariant manifold–and that therefore exhibit competitive dynamics between connectivity modes–as well as in nonlinear networks solving the same task. We note that though the nonlinear networks behaved similarly to the linear case for this particular task, this is likely to be problem dependent.

2 Deeper multilayer dynamics

The network analyzed in Section 1 is the minimal example of a multilayer net, with just a single layer of hidden units. How does gradient descent act in much deeper networks? We make an initial attempt in this direction based on initial conditions that yield particularly simple gradient descent dynamics.

In a linear neural network with N_l layers and hence $N_l - 1$ weight matrices indexed by $W^l, l = 1, \cdots, N_l - 1$, the gradient descent dynamics can be written as

$$\tau \frac{d}{dt} W^l = \left(\prod_{i=l+1}^{N_l - 1} W^i \right)^T \left[\Sigma^{31} - \left(\prod_{i=1}^{N_l - 1} W^i \right) \Sigma^{11} \right] \left(\prod_{i=1}^{l-1} W^i \right)^T, \tag{13}$$

where $\prod_{i=a}^{b} W^i = W^b W^{(b-1)} \cdots W^{(a-1)} W^a$ with the caveat that $\prod_{i=a}^{b} W^i = I$, the identity, if $a > b$.

To describe the initial conditions, we suppose that there are N_l orthogonal matrices R_l that diagonalize the starting weight matrices, that is, $R_{l+1}^T W_l(0) R_l = D_l$ for all l, with the caveat that $R_1 = V^{11}$ and $R_{N_l} = U^{33}$. This requirement essentially demands that the output singular vectors of layer l be the input singular vectors of the next layer $l+1$, so that a change in mode strength at any layer propagates to the output without mixing into other modes. We note that this formulation does not restrict hidden layer size; each hidden layer can be of a different size, and may be undercomplete or overcomplete. Making the

change of variables $W_l = R_{l+1}\overline{W}_l R_l^T$ along with the assumption that $\Sigma^{11} = I$ leads to a set of decoupled connectivity modes that evolve independently of each other. In analogy to the simplification occurring in the three layer network from (2) to (8), each connectivity mode in the N_l layered network can be described by $N_l - 1$ scalars $a^1, \ldots, a^{N_l-1}$, whose dynamics obeys gradient descent on the energy function (the analog of (9)),

$$E(a_1, \cdots, a_{N_l-1}) = \frac{1}{2\tau}\left(s - \prod_{i=1}^{N_l-1} a_i\right)^2. \tag{14}$$

This dynamics also has a set of conserved quantities $a_i^2 - a_j^2$ arising from the energetic symmetry w.r.t. the transformation $a_i \to \lambda a_i$, $a_j \to \frac{a_j}{\lambda}$, and hence can be solved exactly. We focus on the invariant submanifold in which $a_i(t=0) = a_0$ for all i, and track the dynamics of $u = \prod_{i=1}^{N_l-1} a_i$, the overall strength of this mode, which obeys (i.e. the generalization of (10)),

$$\tau \frac{d}{dt} u = (N_l - 1) u^{2-2/(N_l-1)}(s - u). \tag{15}$$

This can be integrated for any positive integer N_l, though the expression is complicated. Once the overall strength increases sufficiently, learning explodes rapidly.

Eqn. (15) lets us study the dynamics of learning as depth limits to infinity. In particular, as $N_l \to \infty$ we have the dynamics

$$\tau \frac{d}{dt} u = N_l u^2 (s - u) \tag{16}$$

which can be integrated to obtain

$$t = \frac{\tau}{N_l}\left[\frac{1}{s^2}\log\left(\frac{u_f(u_0 - s)}{u_0(u_f - s)}\right) + \frac{1}{s u_0} - \frac{1}{s u_f}\right]. \tag{17}$$

Remarkably this implies that, for a fixed learning rate, the learning time tends to zero as N_l goes to infinity. This result depends on the continuous time formulation, however. Any implementation will operate in discrete time and must choose a finite learning rate that yields stable dynamics. An estimate of the optimal learning rate can be derived from the maximum eigenvalue of the Hessian over the region of interest. For linear networks with $a_i = a_j = a$, this optimal learning rate α_{opt} decays with depth as $O\left(\frac{1}{N_l s^2}\right)$ for large N_l (see Supplementary Appendix B). Incorporating this dependence of the learning rate on depth, the learning time as depth approaches infinity still surprisingly remains finite: with the optimal learning rate, the difference between learning times for an $N_l = 3$ network and an $N_l = \infty$ network is $t_\infty - t_3 \sim O(s/\epsilon)$ for small ϵ (see Supplementary Appendix B.1).

To verify these predictions, we trained deep linear networks on the MNIST dataset with depths ranging from $N_l = 3$ to $N_l = 100$. We used hidden layers of size 1000, and calculated the iteration at which training error fell below a fixed threshold corresponding to nearly complete learning. We optimized the learning rate separately for each depth by training each network with twenty

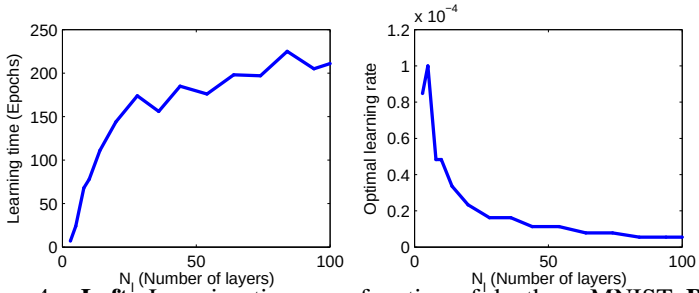

Figure 4: **Left:** Learning time as a function of depth on MNIST. **Right:** Empirically optimal learning rates as a function of depth.

rates logarithmically spaced
between 10^{-4} and 10^{-7} and picking the fastest (see Supplementary Appendix C for details). Networks were initialized with decoupled initial conditions and starting initial mode strength $u_0 = 0.001$. Fig. 4 shows the resulting learning times, which saturate, and the empirically optimal learning rates, which scale like $O(1/N_l)$ as predicted.

Thus learning times in deep linear networks that start with decoupled initial conditions are only a finite amount slower than a shallow network regardless of depth. Moreover, the delay incurred by depth scales inversely with the size of the initial strength of the association. Hence finding a way to initialize the mode strengths to large values is crucial for fast deep learning.

3 Finding good weight initializations: on greediness and randomness

The previous subsection revealed the existence of a decoupled submanifold in weight space in which connectivity modes evolve independently of each other during learning, and learning times can be independent of depth, even for arbitrarily deep networks, as long as the initial composite, end to end mode strength, denoted by u above, of every connectivity mode is $O(1)$. What numerical weight initilization procedures can get us close to this weight manifold, so that we can exploit its rapid learning properties?

A breakthrough in training deep neural networks started with the discovery that greedy layer-wise unsupervised pretraining could substantially speed up and improve the generalization performance of standard gradient descent [17, 18]. Unsupervised pretraining has been shown to speed the optimization of deep networks, and also to act as a special regularizer towards solutions with better generalization performance [18, 12, 13, 14]. At the same time, recent results have obtained excellent performance starting from carefully-scaled random initializations, though interestingly, pretrained initializations still exhibit faster convergence [21, 13, 22, 3, 4, 1, 23] (see Supplementary Appendix D for discussion). Here we examine analytically how unsupervised pretraining achieves an optimization advantage, at least in deep linear networks, by finding the special class of orthogonalized, decoupled initial conditions in the previous section that allow for rapid supervised deep learning, for input-output tasks with a certain precise structure. Subsequently, we analyze the properties of random initilizations.

We consider the effect of using autoencoders as the unsupervised pretraining module [18, 12], for which the input-output correlation matrix Σ^{31} is simply the input correlation matrix Σ^{11}. Hence the SVD of Σ^{31} is PCA on the input correlation matrix, since $\Sigma^{31} = \Sigma^{11} = Q\Lambda Q^T$, where Q are eigenvectors of Σ^{11} and Λ is a diagonal matrix of variances. After pretraining, the weights thus converge to $W^{32}W^{21} = \Sigma^{31}(\Sigma^{31})^{-1} = QQ^T$. Recall that a^α denotes the strength of mode α in W^{21}, and b^α denotes the strength in W^{32}. If we further take the assumption that $a^\alpha \approx b^\alpha$, as is typical when starting from small random weights, then the input-to-hidden mapping will be $W^{21} = R_2 Q^T$ where R_2 is an arbitrary orthogonal matrix. Now consider fine-tuning on a task with input-output correlations $\Sigma^{31} = U^{33}S^{31}V^{11}$. The pretrained initial condition $W^{21} = R_2 Q^T$ will be a decoupled initial condition for the task, $W^{21} = R_2 D_1 {V^{11}}^T$, provided

$$Q = V^{11}. \tag{18}$$

Hence we can state the underlying condition required for successful greedy pretraining in deep linear networks: the right singular vectors of the ultimate input-ouput task of interest V^{11} must be similar to the principal components of the input data Q. This is a quantitatively precise instantiation of the intuitive idea that unsupervised pretraining can help in a subsequent supervised learning task if (and only if) the statistical structure of the input is consistent with the structure of input-output map to be learned. Moreover, this quantitative instantiation of this intuitive idea gives a simple empirical criterion that can be evaluated on any new dataset: given the input-output correlation Σ^{31} and input correlation Σ^{11}, compute the right singular vectors V^{11} of Σ^{31} and check that $V^{11}\Sigma^{11}{V^{11}}^T$ is approximately diagonal. If the condition in Eqn. (18) holds,

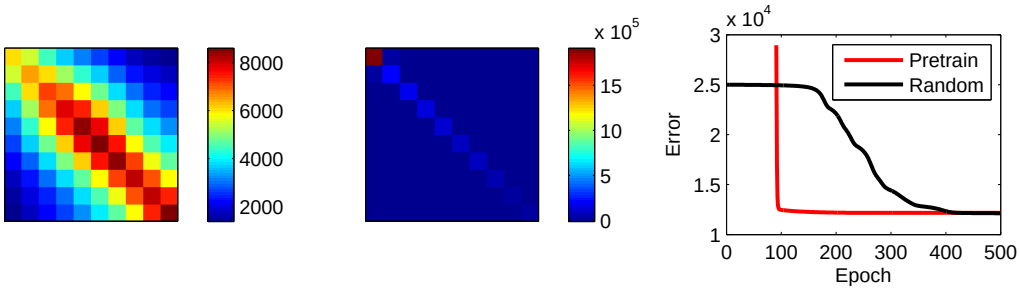

Figure 5: MNIST satisfies the consistency condition for greedy pretraining. **Left:** Submatrix from the raw MNIST input correlation matrix Σ^{11}. **Center:** Submatrix of $V^{11}\Sigma^{11}V^{11^T}$ which is approximately diagonal as required. **Right:** Learning curves on MNIST for a five layer linear network starting from random (black) and pretrained (red) initial conditions. Pretrained curve starts with a delay due to pretraining time. The small random initial conditions correspond to all weights chosen i.i.d. from a zero mean Gaussian with standard deviation 0.01.

autoencoder pretraining will have properly set up decoupled initial conditions for W^{21}, with an appreciable initial association strength near 1. This argument also goes through straightforwardly for layer-wise pretraining of deeper networks. Fig. 5 shows that this consistency condition empirically holds on MNIST, and that a pretrained deep linear neural network learns faster than one started from small random initial conditions, even accounting for pretraining time (see Supplementary Appendix E for experimental details).

As an alternative to greedy layerwise pre-training, [13] proposed choosing appropriately scaled initial conditions on weights that would preserve the norm of typical error vectors as they were backpropagated through the deep network. In our context, the appropriate norm-preserving scaling for the initial condition of an N by N connectivity matrix W between any two layers corresponds to choosing each weight i.i.d. from a zero mean Gaussian with standard deviation $1/\sqrt{N}$. With this choice, $\langle v^T W^T W v \rangle_W = v^T v$, where $\langle \cdot \rangle_W$ denotes an average over distribution of the random matrix W. Moreover, the distribution of $v^T W^T W v$ concentrates about its mean for large N. Thus with this scaling, in linear networks, both the forward propagation of activity, and backpropagation of gradients is typically norm-preserving. However, with this initialization, the learning time with depth on linear networks trained on MNIST grows with depth (Fig. 6A, left, blue). This growth is in distinct contradiction with the theoretical prediction, made above, of depth independent learning times starting from the decoupled submanifold of weights with composite mode strength $O(1)$. This suggests that the scaled random initialization scheme, despite its norm-preserving nature, does not find this submanifold in weight space. In contrast, learning times with greedy layerwise pre-training do not grow with depth (Fig. 6A, left, green curve hiding under red curve), consistent with the predictions of our theory (as a technical point: note that learning times under greedy pre-training initialization in Fig. 6A are faster than those obtained in Fig. 4 by explicitly choosing a point on the decoupled submanifold, because there the initial mode strength was chosen to be small ($u = 0.001$) whereas greedy pre-training finds a composite mode strength closer to 1).

Is there a simple random initialization scheme that does enjoy the rapid learning properties of greedy-layerwise pre-training? We show (Fig. 6A, left, red curve) that if we choose the initial weights in each layer to be a random orthogonal matrix (satisifying $W^T W = I$), instead of a scaled random Gaussian matrix, then this orthogonal random initialization condition yields depth independent learning times just like greedy layerwise pre-training (indeed the red and green curves are indistinguishable). Theoretically, why do random orthogonal initializations yield depth independent learning times, but not scaled random Gaussian initializations, despite their norm preserving nature?

The answer lies in the eigenvalue and singular value spectra of products of Gaussian versus orthgonal random matrices. While a single random orthogonal matrix has eigenvalue spectra lying exactly on the unit circle in the complex plane (Fig. 6A right), the eigenvalue spectra of random Gaussian matrices, whose elements have variance $1/N$, form a uniform distribution on a solid disk of radius 1 the complex plane (Fig. 6C left). Moreover the singular values of an orthogonal matrix are all exactly 1, while the squared singular values

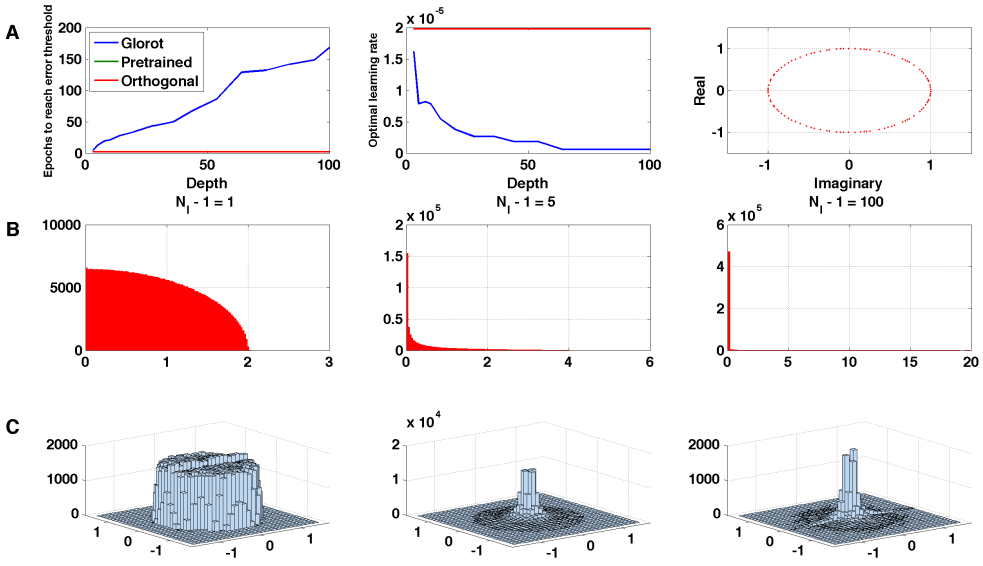

Figure 6: **A Left:** Learning time (on MNIST using the same architecture and parameters as in Fig. 4) as a function of depth for different initial conditions on weights (scaled i.i.d. random gaussian weights chosen to preserve the norm of propagated gradients (blue), greedy unsupervised pre-training (green) and random orthogonal matrices (red). The red curve lies on top of the green curve. **Middle:** Optimal learning rates as a function of depth for different weight initilizations. **Right:** The eigenvalue spectrum, in the complex plane, of a random 100 by 100 orthogonal matrix. **A** Histograms of the singular values of products of $N_l - 1$ independent random Gaussian N by N matrices whose elements themselves are chosen i.i.d. from a zero mean Gaussian with standard deviation $1/\sqrt{N}$. In all cases, $N = 1000$, and histograms are taken over 500 realizations of such random product matrices, yielding a total $5 \cdot 10^5$ singular values in each histogram. **C** Histograms of the eigenvalue distributions on the complex plane of the same product matrices in **B**. The bin width is 0.1, and, for visualization purposes, the bin containing the origin has been removed in each case; this bin would otherwise dominate the histogram in the middle and right plots, as it contains 32% and 94% of the eigenvalues respectively.

of a scaled Gaussian random matrix have the well known Marcenko-Pasteur distribution, with a nontrivial spread even as $N \to \infty$, (Fig. 6B left shows the distribution of singular values themselves). Now consider a product of these matrices across all N_l layers, representing the total end to end propagation of activity across a deep linear network:

$$W_{\text{Tot}} = \prod_{i=1}^{N_l - 1} W^{(i+1,i)}. \tag{19}$$

Due to the random choice of weights in each layer, W_{Tot} is itself a random matrix. On average, it preserves the norm of a typical vector v no matter whether the matrices in each layer are Gaussian or orthogonal. However, the singular value spectra of W_{Tot} differ markedly in the two cases. Under random orthogonal initilization in each layer, W_{Tot} is itself an orthogonal matrix and therefore has *all* singular values equal to 1. However, under random Gaussian initialization in each layer, there is as of yet no complete theoretical characterization of the singular value distribution of W_{Tot}. We have computed it numerically as a function of different depths in Fig. 6B, and we find that it develops a highly kurtotic nature as the depth increases. Most of the singular values become vanishingly small, while a long tail of very large singular values remain.

Thus W_{Tot} preserves the norm of a typical, randomly chosen vector v, but in a highly anisotropic manner, by strongly amplifying the projection of v onto a very small subset of singular vectors and attenuating v in all other directions. Intuitively W_{Tot}, as well as the linear operator W_{Tot}^T that would be closely related to backpropagation of gradients to early layers, act as amplifying projection operators at large depth N_l. In contrast, all of the eigenvalues of W_{Tot} in the scaled Gaussian case concentrate closer to the origin as depth increases. This discrepancy between the behavior of the eigenvalues and singular values of W_{Tot}, a phenomenon that could occur only if the eigenvectors of W_{Tot} are highly non-orthogonal, reflects the highly non-normal nature of products of random Gaussian matrices (a non-normal matrix is by definition a matrix whose eigenvectors are non-orthogonal).

While the combination of amplification and projection in W_{Tot} can preserve norm, it is clear that it is not a good way to backpropagate errors; the projection of error vectors onto a high dimensional subspace corresponding to small singular values would be strongly attenuated, yielding vanishingly small gradient signals corresponding to these directions in the early layers. This effect, which is not present for random orthogonal initializations or greedy pretraining, would naturally explain the long learning times starting from scaled random Gaussian initial conditions relative to the other initilizations in Fig. 6A left. For both linear and nonlinear networks, a more likely appropriate condition on weights for generating fast learning times would be that of *dynamical isometry*. By this we mean that the product of Jacobians associated with error signal backpropagation should act as a near isometry, up to some overall global $O(1)$ scaling, on a subspace of as high a dimension as possible. This is equivalent to having as many singular values of the product of Jacobians as possible within a small range around an $O(1)$ constant, and is closely related to the notion of restricted isometry in compressed sensing and random projections. Preserving norms is a necessary but not sufficient condition for achieving dynamical isometry at large depths, as demonstrated in Fig. 6B, and we have shown that for linear netoworks, orthogonal initializations achieve exact dynamical isometry with all singular values at 1, while greedy pre-training achieves it approximately.

We note that the discrepancy in learning times between the scaled Gaussian initialization and the orthogonal or pre-training initializations is modest for the depths of around 6 used in large scale applications, but is magnified at larger depths (Fig. 6A left). This may explain the modest improvement in learning times with greedy pre-training versus random scaled Gaussian initializations observed in applications (see discussion in Supplementary Appendix D). We predict that this modest improvement will be magnified at higher depths, even in nonlinear networks. Finally, we note that in recurrent networks, which can be thought of as infinitely deep feed-forward networks with tied weights, a very promising approach is a modification to the training objective that partially promotes dynamical isometry for the set of gradients currently being back-propagated [24].

4 Achieving approximate dynamical isometry in nonlinear networks

We have shown above that deep random orthogonal linear networks achieve perfect dynamical isometry. Here we show that nonlinear versions of these networks can also achieve good dynamical isometry properties. Consider the nonlinear feedforward dynamics

$$x_i^{l+1} = \sum_j g\, W_{ij}^{(l+1,l)}\, \phi(x_j^l), \tag{20}$$

where x_i^l denotes the activity of neuron i in layer l, $W_{ij}^{(l+1,l)}$ is a random orthogonal connectivity matrix from layer l to $l+1$, g is a scalar gain factor, and $\phi(x)$ is any nonlinearity that saturates as $x \to \pm\infty$. We show in Supplementary appendix F that there exists a critical value g_c of the gain g such that if $g < g_c$, activity will decay away to zero as it propagates through the layers, while if $g > g_c$, the strong linear positive gain will combat the damping due to the saturating nonlinearity, and activity will propagate indefinitely without decay, no matter how deep the network is. These dynamical properties can be quantitatively captured by the

neural population variance in layer l,

$$q^l \equiv \frac{1}{N} \sum_{i=1}^{N} (x_i^l)^2. \tag{21}$$

Thus $\lim_{l\to\infty} q^l \to 0$ for $g < g_c$ and $\lim_{l\to\infty} q^l \to q^\infty(g) > 0$ for $g > g_c$. When $\phi(x) = \tanh(x)$, we compute $g_c = 1$ and numerically compute $q^\infty(g)$ in Fig. 8 in Supplementary appendix F. Thus these nonlinear feedforward networks exhibit a phase-transition at the critical gain; above the critical gain, infinitely deep networks exhibit chaotic percolating activity propagation, so we call the critical gain g_c the edge of chaos, in analogy with terminology for recurrent networks.

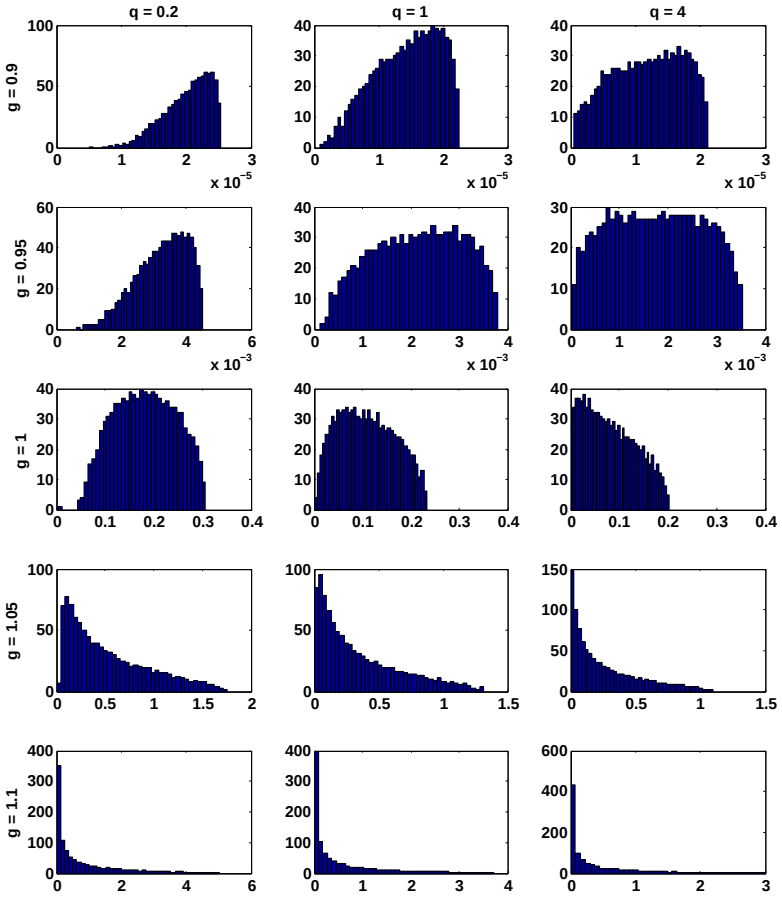

Figure 7: Singular value distribution of the end to end Jacobian, defined in (22), for various values of the gain g in (20) and the input layer population variance $q = q^1$ in (21). The network architecture consists of $N_l = 100$ layers with $N = 1000$ neurons per layer, as in the linear case in Fig. 6B.

Now we are interested in how errors at the final layer N_l backpropagate back to earlier layers, and whether or not these gradients explode or decay with depth. To quantify this, for simplicity we consider the end to

end Jacobian

$$J_{ij}^{N_l,1}(x^{N_l}) \equiv \left. \frac{\partial x_i^{N_l}}{\partial x_j^1} \right|_{x^{N_l}}, \tag{22}$$

which captures how input perturbations propagate to the output. If the singular value distribution of this Jacobian is well-behaved, with few extremely large or small singular values, then the backpropagation of gradients will also be well-behaved, and exhibit little explosion or decay. The Jacobian is evaluated at a particular point x^{N_l} in the space of output layer activations, and this point is in turn obtained by iterating (20) starting from an initial input layer activation vector x^1. Thus the singular value distribution of the Jacobian will depend not only on the gain g, but also on the initial condition x^1. By rotational symmetry, we expect this distribution to depend on x^1, only through its population variance q^1. Thus for large N, the singular value distribution of the end-to-end Jacobian in (22) (the analog of W_{Tot} in (19) in the linear case), depends on only two parameters: gain g and input population variance q^1.

We have numerically computed this singular value distribution as a function of these two parameters in Fig. 7, for a single random orthogonal nonlinear network with $N = 1000$ and $N_l = 100$. These results are typical; replotting the results for different random networks and different initial conditions (with the same input variance) yield very similar results. We see that below the edge of chaos, when $g < 1$, the linear dampening over many layers yields extremely small singular values. Above the edge of chaos, when $g > 1$, the combination of positive linear amplification, and saturating nonlinear dampening yields an anisotropic distribution of singular values. At the edge of chaos, $g = 1$, an $O(1)$ fraction of the singular value distribution is concentrated in a range that remains $O(1)$ despite 100 layers of propagation, reflecting appoximate dynamical isometry. Moreover, this nice property at $g = 1$ remains valid even as the input variance q^1 is increased far beyond 1, where the $\tanh$ function enters its nonlinear regime. Thus the right column of Fig. 7 at g near 1 indicates that the useful dynamical isometry properties of random orthogonal linear networks described above survives in nonlinear networks, even when activity patterns enter deeply into the nonlinear regime in the input layers. Interestingly, the singular value spectrum is more robust to perturbations that increase g from 1 relative to those that decrease g. Indeed, the anisotropy in the singular value distribution at $g = 1.1$ is relatively mild compared to that of random linear networks with scaled Gaussian initial conditions (compare the bottom row of Fig. 7 with the right column of panel B in Fig. 6). Thus overall, these numerical results suggest that being just beyond the edge of orthogonal chaos may be a good regime for learning in deep nonlinear networks.

5 Discussion

In summary, despite the simplicity of their input-output map, the dynamics of learning in deep linear networks reveals a surprising amount of rich mathematical structure, including nonlinear hyperbolic dynamics, plateaus and sudden performance transitions, a proliferation of saddle points, symmetries and conserved quantities, invariant submanifolds of independently evolving connectivity modes subserving rapid learning, and most importantly, a sensitive but computable dependence of learning time scales on input statistics, initial weight conditions, and network depth. With the right initial conditions, deep linear networks can be only a finite amount slower than shallow networks, and unsupervised pretraining can find these initial conditions for tasks with the right structure. Moreover, we introduce a mathematical condition for faithful backpropagation of error signals, namely dynamical isometry, and show, surprisingly that random scaled Gaussian initializations cannot achieve this condition despite their norm-preserving nature, while greedy pre-training and random orthogonal initialization can, thereby achieving depth independent learning times. Finally, we show that the propery of dynamial isometry survives to good approximation even in extremely deep nonlinear random orthogonal networks operating just beyond the edge of chaos. At the cost of expressivity, deep linear networks gain theoretical tractability and may prove fertile for addressing other phenomena in deep learning, such as the impact of carefully-scaled initializations [13, 23], momentum [23], dropout regularization [1], and sparsity constraints [2]. While a full analytical treatment of learning in deep nonlinear networks

currently remains open, one cannot reasonably hope to move towards such a theory without first completely understanding the linear case. In this sense, our work fulfills an essential pre-requisite for progress towards a general, quantitative theory of deep learning.

References

[1] A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, 2012.

[2] Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A.Y. Ng. Building high-level features using large scale unsupervised learning. In *29th International Conference on Machine Learning*, 2012.

[3] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

[4] A. Mohamed, G.E. Dahl, and G. Hinton. Acoustic Modeling Using Deep Belief Networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, January 2012.

[5] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[6] R. Socher, J. Bauer, C.D. Manning, and A.Y. Ng. Parsing with Compositional Vector Grammars. In *Association for Computational Linguistics Conference*, 2013.

[7] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen.* PhD thesis, TU Munich, 1991.

[8] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[9] Y. LeCun, L. Bottou, G.B. Orr, and K.R. Müller. Efficient BackProp. *Neural networks: Tricks of the trade*, 1998.

[10] Y. Bengio and Y. LeCun. Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*, number 1, pages 1–41. MIT Press, 2007.

[11] D. Erhan, P.A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. In *12th International Conference on Artificial Intelligence and Statistics*, volume 5, 2009.

[12] Y. Bengio. Learning Deep Architectures for AI. 2009.

[13] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *13th International Conference on Artificial Intelligence and Statistics*, 2010.

[14] D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, and P. Vincent. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[15] Y.N. Dauphin and Y. Bengio. Big Neural Networks Waste Capacity. In *International Conference on Learning Representations*, 2013.

[16] A.M. Saxe, J.L. McClelland, and S. Ganguli. Learning hierarchical category structure in deep neural networks. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, 2013.

[17] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–7, July 2006.

[18] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems 20*, 2007.

[19] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, January 1989.

[20] K. Fukumizu. Effect of Batch Learning In Multilayer Neural Networks. In *Proceedings of the 5th International Conference on Neural Information Processing*, pages 67–70, 1998.

[21] J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[22] O. Chapelle and D. Erhan. Improved Preconditioner for Hessian Free Optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[23] I. Sutskever, J. Martens, G. Dahl, and G.E. Hinton. On the importance of initialization and momentum in deep learning. In *30th International Conference on Machine Learning*, 2013.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. Technical report, Universite de Montreal, 2012.

[25] P. Lamblin and Y. Bengio. Important gains from supervised fine-tuning of deep architectures on large labeled sets. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

Supplementary Material

A Hyperbolic dynamics of learning

In Section 1.3 of the main text we treat the dynamics of learning in three layer networks where mode strengths in each layer are equal, i.e, $a = b$, a reasonable limit when starting with small random initial conditions. More generally, though, we are interested in how long it takes for ab to approach s from any given initial condition. To access this, given the hyperbolic nature of the dynamics, it is useful to make the hyperbolic change of coordinates,

$$a = \sqrt{c_0}\cosh\frac{\theta}{2} \quad b = \sqrt{c_0}\sinh\frac{\theta}{2} \quad \text{for } a^2 > b^2 \tag{23}$$

$$a = \sqrt{c_0}\sinh\frac{\theta}{2} \quad b = \sqrt{c_0}\cosh\frac{\theta}{2} \quad \text{for } a^2 < b^2. \tag{24}$$

Thus θ parametrizes the dynamically invariant manifolds $a^2 - b^2 = \pm c_0$. For any c_0 and θ, this coordinate system covers the region $a + b > 0$, which is the basin of attraction of the upper right component of the hyperbola $ab = s$. A symmetric situation exists for $a + b < 0$, which is attracted to the lower left component of $ab = s$. We use θ as a coordinate to follow the dynamics of the product ab, and using the relations $ab = c_0 \sinh\theta$ and $a^2 + b^2 = c_0 \cosh\theta$, we obtain

$$\tau\frac{d\theta}{dt} = s - c_0\sinh\theta. \tag{25}$$

This differential equation is separable in θ and t and can be integrated to yield

$$t = \tau\int_{\theta_0}^{\theta_f}\frac{d\theta}{s - c_0\sinh\theta} = \frac{\tau}{\sqrt{c_0^2 + s^2}}\left[\ln\frac{\sqrt{c_0^2+s^2} + c_0 + s\tanh\frac{\theta}{2}}{\sqrt{c_0^2+s^2} - c_0 - s\tanh\frac{\theta}{2}}\right]_{\theta_0}^{\theta_f}. \tag{26}$$

Here t is the amount of time it takes to travel from θ_0 to θ_f along the hyperbola $a^2 - b^2 = \pm c_0$. The fixed point lies at $\theta = \sinh^{-1} s/c_0$, but the dynamics cannot reach the fixed point in finite time. Therefore we introduce a cutoff ϵ to mark the endpoint of learning, so that θ_f obeys $\sinh\theta_f = (1-\epsilon)s/c_0$ (i.e. ab is

close to s by a factor $1-\epsilon$). We can then average over the initial conditions c_0 and θ_0 to obtain the expected learning time of an input-output relation that has a correlation strength s. Rather than doing this, it is easier to obtain a rough estimate of the timescale of learning under the assumption that the initial weights are small, so that c_0 and θ_0 are close to 0. In this case $t = O(\tau/s)$ (with a weak logarithmic dependence on the cutoff (i.e. $\ln(1/\epsilon)$). This modestly generalizes the result given in the main text: the timescale of learning of each input-output mode α of the correlation matrix Σ^{31} is inversely proportional to the correlation strength s_α of the mode even when a and b differ slightly, i.e., c_0 small. This is not an unreasonable limit for random initial conditions because $|c_0| = |a \cdot a - b \cdot b|$ where a and b are random vectors of N_2 synaptic weights into and out of the hidden units. Thus we expect the lengths of the two random vectors to be approximately equal and therefore c_0 will be small relative to the length of each vector.

These solutions are distinctly different from solutions for learning dynamics in three layer networks found in [20]. In our notation, in [20], it was shown that if the initial vectors a^α and b^α satisfy the matrix identity $\sum_\alpha a^\alpha a^{\alpha^T} = \sum_\alpha b^\alpha b^{\alpha^T}$ then the dynamics of learning becomes equivalent to a matrix Riccatti equation. However, the hyperbolic dynamics derived here arises from a set of initial conditions that do not satisfy the restrictions of [20] and therefore do not arise through a solution to a matrix Ricatti equation. Moreover, in going beyond a statement of the matrix Riccatti solution, our analysis provides intuition about the timescales over which the learning dynamics unfolds, and crucially, our methods extend beyond the three layer case to the arbitrary N_l layer case, not studied in [20].

B Optimal discrete time learning rates

In Section 2 we state results on the optimal learning rate as a function of depth in a deep linear network, which we derive here. Starting from the decoupled initial conditions given in the main text, the dynamics arise from gradient descent on

$$E(a_1, \cdots, a_{N_l-1}) = \frac{1}{2\tau}\left(s - \prod_{k=1}^{N_l-1} a_k\right). \tag{27}$$

Hence for each a_i we have

$$\frac{\partial E}{\partial a_i} = -\frac{1}{\tau}\left(s - \prod_{k=1}^{N_l-1} a_k\right)\left(\prod_{k\neq i}^{N_l-1} a_k\right) \equiv f(a_i) \tag{28}$$

The elements of the Hessian are thus

$$\frac{\partial^2 E}{\partial a_i a_j} = \frac{1}{\tau}\left(\prod_{k\neq j}^{N_l-1} a_k\right)\left(\prod_{k\neq i}^{N_l-1} a_k\right) - \frac{1}{\tau}\left(s - \prod_{k=1}^{N_l-1} a_k\right)\left(\prod_{k\neq i,j}^{N_l-1} a_k\right) \tag{29}$$

$$\equiv g(a_i, a_j) \tag{30}$$

for $i \neq j$, and

$$\frac{\partial^2 E}{\partial a_i^2} = \frac{1}{\tau}\left(\prod_{k\neq i}^{N_l-1} a_k\right)^2 \equiv h(a_i) \tag{31}$$

for $i = j$.

We now assume that we start on the symmetric manifold, such that $a_i = a_j = a$ for all i, j. Thus we have

$$E(a) = \frac{1}{2\tau}\left(s - a^{N_l-1}\right), \tag{32}$$

$$f(a) = -\frac{1}{\tau}\left(s - a^{N_l-1}\right)a^{N_l-2}, \tag{33}$$

$$g(a) = \frac{2}{\tau}a^{2N_l-4} - \frac{1}{\tau}sa^{N_l-3} \tag{34}$$

$$h(a) = \frac{1}{\tau}a^{2N_l-4} \tag{35}$$

The Hessian is

$$H(a) = \begin{bmatrix} h & g & \cdots & g & g \\ g & h & \cdots & g & g \\ \vdots & & \ddots & & \vdots \\ g & g & \cdots & h & g \\ g & g & \cdots & g & h \end{bmatrix}. \tag{36}$$

One eigenvector is $v_1 = [11\cdots1]^T$ with eigenvalue $\lambda_1 = h + (N_l - 2)g$, or

$$\lambda_1 = (2N_l - 3)\frac{1}{\tau}a^{2N_l-4} - (N_l - 2)\frac{1}{\tau}sa^{N_l-3}. \tag{37}$$

Now consider the second order update (Newton-Raphson) (here we use 1 to denote a vector of ones)

$$a^{t+1}1 = a^t 1 - H^{-1}f(a^t)1 \tag{38}$$

$$= a^t 1 - f(a^t)H^{-1}1 \tag{39}$$

$$a^{t+1} = a^t - f(a^t)/\lambda_1(a^t) \tag{40}$$

Note that the basin of attraction does not include small initial conditions, because for small a the Hessian is not positive definite.

To determine the optimal learning rate for first order gradient descent, we compute the maximum of λ_1 over the range of mode strengths that can be visited during learning, i.e., $a \in [0, s^{1/(N_l-1)}]$. This occurs at the optimum, $a_{opt} = s^{1/(N_l-1)}$. Hence substituting this into (37) we have

$$\lambda_1(a_{opt}) = (N_l - 1)\frac{1}{\tau}s^{\frac{2N_l-4}{N_l-1}}. \tag{41}$$

The optimal learning rate α is proportional to $1/\lambda_1(a_{opt})$, and hence scales as

$$\alpha \sim O\left(\frac{1}{N_l s^2}\right) \tag{42}$$

for large N_l.

B.1 Learning speeds with optimized learning rate

How does the optimal learning rate impact learning speeds? We compare the three layer learning time to the infinite depth limit learning time, with learning rate set inversely proportional to Eqn. (41) with proportionality constant c.

This yields a three layer learning time t_3 of

$$t_3 = c\ln\frac{u_f(s - u_0)}{u_0(s - u_f)} \tag{43}$$

and an infinite layer learning time t_∞ of

$$t_\infty = c\left[\log\left(\frac{u_f(u_0-s)}{u_0(u_f-s)}\right)+\frac{s}{u_0}-\frac{s}{u_f}\right], \tag{44}$$

Hence the difference is

$$t_\infty - t_3 = \frac{cs}{u_0}-\frac{cs}{u_f}\approx\frac{cs}{\epsilon} \tag{45}$$

where the final approximation is for $u_0=\epsilon$, $u_f=s-\epsilon$, and ϵ small. Thus very deep networks incur only a finite delay relative to shallow networks.

C Experimental setup for MNIST depth experiment

We trained deep linear networks on the MNIST dataset with fifteen different depths $N_l = \{3,5,8,10,14,20,28,36,44,54,64,74,84,94,100\}$. Given a 784-dimensional input example, the network tried to predict a 10-dimensional output vector containing a 1 in the index for the correct class, and zeros elsewhere. The network was trained using batch gradient descent via Eqn. (13) on the 50,000 sample MNIST training dataset. We note that Eqn. (13) makes use of the linearity of the network to speed training and reduce memory requirements. Instead of forward propagating all 50,000 training examples, we precompute Σ^{31} and forward propagate only it. This enables experiments on very deep networks that otherwise would be computationally infeasible. Experiments were accelerated on GPU hardware using the GPUmat package. We used overcomplete hidden layers of size 1000. Here the overcompleteness is simply to demonstrate the applicability of the theory to this case; overcompleteness does not improve the representational power of the network. Networks were initialized with decoupled initial conditions and starting initial mode strength $u_0 = 0.001$, as described in the text. The random orthogonal matrices R_l were selected by generating random Gaussian matrices and computing a QR decomposition to obtain an orthogonal matrix. Learning times were calculated as the iteration at which training error fell below a fixed threshold of 1.3×10^4 corresponding to nearly complete learning. Note that this level of performance is grossly inferior to what can be obtained using nonlinear networks, which reflects the limited capacity of a linear network. We optimized the learning rate λ separately for each depth by training each network with twenty rates logarithmically spaced between 10^{-4} and 10^{-7} and picking the one that yielded the minimum learning time according to our threshold criterion. The range 10^{-4} and 10^{-7} was selected via preliminary experiments to ensure that the optimal learning rate always lay in the interior of the range for all depths.

D Efficacy of unsupervised pretraining

Recently high performance has been demonstrated in deep networks trained from random initial conditions [21, 13, 22, 3, 4, 1, 23], suggesting that deep networks may not be as hard to train as previously thought. These results show that pretraining is not necessary to obtain state-of-the-art performance, and to achieve this they make use of a variety of techniques including carefully-scaled random initializations, more sophisticated second order or momentum-based optimization methods, and specialized convolutional architectures. It is therefore important to evaluate whether unsupervised pretraining is still useful, even if it is no longer necessary, for training deep networks. In particular, does pretraining still confer an optimization advantage and generalization advantage when used in conjunction with these new techniques? Here we review results from a variety of papers, which collectively show that unsupervised pretraining still confers an optimization advantage and a generalization advantage.

D.1 Optimization advantage

The optimization advatage of pretraining refers to faster convergence to the local optimum (i.e., faster learning speeds) when starting from pretrained initializations as compared to random initializations. Faster learning speeds starting from pretrained initial conditions have been consistently found with Hessian free optimization [21, 22]. This finding holds for two carefully-chosen random initialization schemes, the sparse connectivity scheme of [21], and the dense scaled scheme of [13] (as used by [22]). Hence pretraining still confers a convergence speed advantage with second order methods. Pretrained initial conditions also result in faster convergence than carefully-chosen random initializations when optimizing with stochastic gradient descent [22, 13]. In light of this, it appears that pretrained initial conditions confer an optimization advantage beyond what can be obtained currently with carefully-scaled random initializations, regardless of optimization technique. If run to convergence, second order methods and well-chosen scalings can erase the discrepancy between the final objective value obtained on the training set for pretrained relative to random initializations [21, 22]. The optimization advantage is thus purely one of convergence speed, not of finding a better local minimum. This coincides with the situation in linear networks, where all methods will eventually attain the same global minimum, but the rate of convergence can vary. Our analysis shows why this optimization advantage due to pretraining persists over well-chosen random initializations.

Finally, we note that Sutskever et al. show that careful random initialization paired with carefully-tuned momentum can achieve excellent performance [23], but these experiments did not try pretrained initial conditions. Krizhevsky et al. used convolutional architectures and did not attempt pretraining [1]. Thus the possible utility of pretraining in combination with momentum, and in combination with convolutional architectures, dropout, and large supervised datasets, remains unclear.

D.2 Generalization advantage

Pretraining can also act as a special regularizer, improving generalization error in certain instances. This generalization advantage appears to persist with new second order methods [21, 22], and in comparison to gradient descent with careful random initializations [13, 22, 25, 4]. An analysis of this effect in deep linear networks is out of the scope of this work, though promising tools have been developed for the three layer linear case [20].

E MNIST pretraining experiment

We trained networks of depth 5 on the MNIST classification task with 200 hidden units per layer, starting either from small random initial conditions with each weight drawn independently from a Gaussian distribution with standard deviation 0.01, or from greedy layerwise pretrained initial conditions. For the pretrained network, each layer was trained to reconstruct the output of the next lower layer. In the finetuning stage, the network tried to predict a 10-dimensional output vector containing a 1 in the index for the correct class, and zeros elsewhere. The network was trained using batch gradient descent via Eqn. (13) on the 50,000 sample MNIST training dataset. Since the network is linear, pretraining initializes the network with principal components of the input data, and, to the extent that the consistency condition of Eqn. (18) holds, decouples these modes throughout the deep network, as described in the main text.

F Analysis of Neural Dynamics in Nonlinear Orthogonal Networks

We can derive a simple, analytical recursion relation for the propagation of neural population variance q^l, defined in (21), across layers l under the nonlinear dynamics (20). We have

$$q^{l+1} = \frac{1}{N}\sum_{i=1}^{N}(x_i^{l+1})^2 = g^2\frac{1}{N}\sum_{i=1}^{N}\phi(x_i^l)^2, \tag{46}$$

due to the dynamics in (20) and the orthogonality of $W^{(l+1,l)}$. Now we know that by definition, the layer l population x_i^l has normalized variance q^l. If we further assume that the distribution of activity across neurons in layer l is well approximated by a Gaussian distribution, we can replace the sum over neurons i with an integral over a zero mean unit variance Gaussian variable z:

$$q^{l+1} = g^2\int \mathcal{D}z\,\phi\big(\sqrt{q^l}z\big)^2, \tag{47}$$

where $\mathcal{D}z \equiv \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}z^2}\,dz$ is the standard Gaussian measure. This map from input to output variance is numerically computed for $g = 1$ and $\phi(x) = \tanh(x)$ in Fig. 8, left (other values of g yield a simple multiplicative scaling of this map). This recursion relation has a stable fixed point $q^\infty(g)$ obtained by solving the nonlinear fixed point equation

$$q^\infty = g^2\int \mathcal{D}z\,\phi\big(\sqrt{q^\infty}z\big)^2. \tag{48}$$

Graphically, solving this equation corresponds to scaling the curve in Fig. 8 left by g^2 and looking for intersections with the line of unity. For $g < 1$, the only solution is $q^\infty = 0$. For $g > 1$, this solution remains, but it is unstable under the recurrence (47). Instead, for $g > 1$, a new stable solution appears for some nonzero value of q^∞. The entire set of stable solutions as a function of g is shown as the red curve in Fig. 8 right. It constitutes a theoretical prediction of the population variance at the deepest layers of a nonlinear network as the depth goes to infinity. It matches well for example, the empirical population variance obtained from numerical simulations of nonlinear networks of depth 30 (blue points in Fig. 8 right).

Overall, these results indicate a dynamical phase transition in neural activity propagation through the nonlinear network as g crosses the critical value $g_c = 1$. When $g > 1$, activity propagates in a chaotic manner, and so $g = 1$ constitutes the edge of chaos.

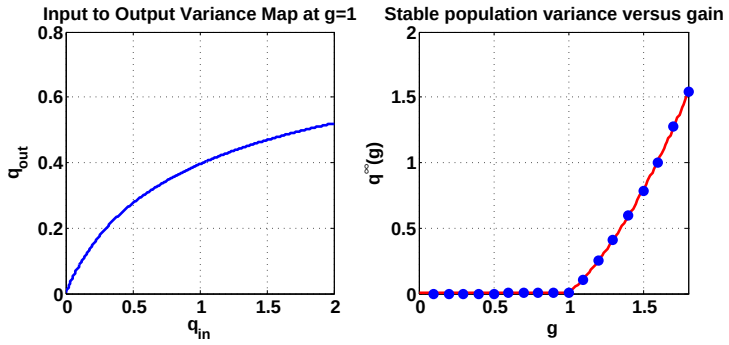

Figure 8: **Left**: The map from variance in the input layer $q^{in} = q^l$ to variance in the output layer $q^{out} = q^{l+1}$ in (47) for $g = 1$ and $\phi(x) = \tanh(x)$. **Right**: The stable fixed points of this map, $q^{\infty}(g)$, as a function of the gain g. The red curve is the analytic theory obtained by numerically solving (48). The blue points are obtained via numerical simulations of the dynamics in (20) for networks of depth $N_l = 30$ with $N = 1000$ neurons per layer. The asymptotic population variance q^{∞} is obtained by averaging the population variance in the last 5 layers.