Reexamining the principle of mean-variance preservation for neural network initialization

Kyle Luther^{1,2,*} and H. Sebastian Seung^{2,3}¹*Department of Physics, Princeton University, Princeton, New Jersey 08544, USA*²*Neuroscience Institute, Princeton University, Princeton, New Jersey 08544, USA*³*Department of Computer Science, Princeton University, Princeton, New Jersey 08544, USA*

(Received 30 January 2020; accepted 23 June 2020; published 24 July 2020)

Before backpropagation training, it is common to randomly initialize a neural network so that mean and variance of activity are uniform across neurons. Classically these statistics were defined over an ensemble of random networks. Alternatively, they can be defined over a random sample of inputs to the network. We show analytically and numerically that these two formulations of the principle of mean-variance preservation are very different in deep networks using rectification nonlinearity (ReLU). We numerically investigate training speed after data-dependent initialization of networks to preserve sample mean and variance.

DOI: [10.1103/PhysRevResearch.2.033135](https://doi.org/10.1103/PhysRevResearch.2.033135)

I. INTRODUCTION

The procedure used to initialize the weights and biases of a neural network has a large impact on backpropagation training dynamics, in some cases determining whether the network will train at all [1]. One rule of thumb, dating back to the 1990s, is to set the biases to zero and randomly initialize the weights so that neural activities (or preactivities) have the same mean and variance throughout the network [2]. This is known as “Kaiming Initialization” for networks using rectification nonlinearity [1]. This nonlinearity is also known as the “ReLU” nonlinearity and is defined as:

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases}$$

Recently, the principle has been extended to handle residual connections [3] and networks with weight normalization [4].

In this Letter, we reexamine the principle, focusing on a subtlety in its formulation. In the classical formulation (the basis of Kaiming initialization), the mean and variance of the preactivity are defined over an ensemble of random networks. This means the statistics are computed by fixing a single input and randomizing over weight configurations. Since training involves a single network but multiple examples, it might be more appropriate to reformulate the principle in terms of the *sample* mean and variance of the preactivity, defined over an ensemble of inputs for a fixed network.

A numerical experiment illustrates that these two formulations of mean-variance preservation can be very different for deep networks. We apply Kaiming initialization to a deep wide ReLU network, provide the net with Gaussian white

noise input vectors, and visualize the preactivity distributions for single neurons in various layers (Fig. 1). The sample mean of the preactivity for a neuron can be far from zero. The sample variance of the preactivity is similar for neurons in a layer but becomes smaller as we move deeper into the network. In short, Kaiming initialization does *not* preserve sample mean and variance. However, if we now randomize over networks, the preactivity mean is zero for all neurons, and the preactivity variance is the same across layers (Fig. 1).

In the following, we will elucidate the phenomena observed in Fig. 1 by applying the formalism of “signal propagation” [5–7]. The formalism was invented for a superficially different purpose but can be applied to compute variance of preactivity in the limit of an infinitely wide network. We will compare the analytical result to numerical simulations in finite width networks.

We explore one method to implement the sample mean-variance preservation principle: Using a data-dependent initialization to, at each layer, initialize the bias to subtract out the sample mean of preactivity and scale the weights to make the sample variance of preactivity equal unity. We empirically show that this initialization scheme, similar to that of Krähenbühl *et al.* [8], can improve training speed of two moderately deep networks (10 and 20 layers) on an object recognition and image segmentation task. This may account for part of the success of batch normalization and related schemes [9–13] that can be interpreted as extensions of the principle of mean-variance preservation to hold throughout training rather than only at the beginning. Note that most of these results have appeared in preprint form [14].

II. NETWORK ENSEMBLE AND DEFINITIONS

We study fully connected, feedforward networks of uniform width N ,

$$y_i^l = \sum_{j=1}^N w_{ij}^l x_j^{l-1} + b_i^l \quad x_i^l = [y_i^l]^+, \quad (1)$$

*kluther@princeton.edu

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

FIG. 1. Numerical experiments on preactivity distributions in (a) layer 1, (b) layer 5, (c) layer 10, and (d) layer 50. Using Kaiming initialization, we randomly generate a 50-layer-deep, 1024-neuron-wide network with ReLU nonlinearity. Gaussian white noise is input to the network. We display as a solid histogram the distribution of a single preactivation over white noise inputs. This process is repeated for three random networks (green, blue, and orange). We then generate 1024 networks using the same procedure and visualize the distribution of a preactivation over the joint distribution of networks and white noise inputs (red histogram).

where l is the layer, $i = 1, 2, \dots, N$ is the neuron index and $[u]^+ = \max(u, 0)$ is (half-wave) rectification. We call $\mathbf{y}^l$ the vector of preactivities and $\mathbf{x}^l$ the vector of activities at layer l .

We distinguish between two kinds of randomness. First, the input $\mathbf{x}^0$ to the network is drawn randomly from some distribution. Second, the weights of the network are drawn independently at random from a zero-mean Gaussian distribution with variance $2/N$, and biases are initialized to zero (Kaiming initialization). Averages with respect to the random input will be denoted by angle brackets $\langle \cdot \rangle$ and will be called “sample averages.” Averages with respect to the random configuration of weights will be denoted by double angle brackets $\langle \langle \cdot \rangle \rangle$ and called “quenched averages.”

The preactivity fluctuates about the sample mean, and the sample mean in turn fluctuates about zero, $y_i^l = (y_i^l - \langle y_i^l \rangle) + \langle y_i^l \rangle$. This leads to the variance decomposition

$$(\sigma^l)^2 = (v^l)^2 + (m^l)^2, \quad (2)$$

where

$$(\sigma^l)^2 = \langle \langle (y_i^l)^2 \rangle \rangle \quad (m^l)^2 = \langle \langle (y_i^l)^2 \rangle \rangle \quad (3)$$

are the “total variance” and the variance of the sample mean. By symmetry these are the same for all neurons in a layer, so the index i is suppressed on the left-hand side. Note that the “total mean” of the preactivity y_i^l vanishes for all neurons in all layers, $\langle \langle y_i^l \rangle \rangle = \langle \langle y_i^l \rangle \rangle = 0$, and the quenched average also vanishes, $\langle \langle y_i^l \rangle \rangle = 0$. In contrast, the “sample mean” $\langle y_i^l \rangle$ is typically nonzero, is different for each neuron in a layer (Fig. 1), and depends on the configuration of weights.

In practice, we are interested in the sample variance $\langle (y_i^l)^2 \rangle - \langle y_i^l \rangle^2$ for a frozen or “quenched” configuration of weights. For wide nets, Fig. 1 suggests that the sample variance can be approximated by its quenched average $(v^l)^2$.

In Fig. 1, σ^l is the width of the red line histogram, v^l is the average width of a solid histogram, and m^l characterizes the variability in the locations of the solid histograms. Under Kaiming initialization, the total variance $(\sigma^l)^2$ is the same across all layers l ; all red line histograms have the same width. In the first layer, the total variance is entirely sample variance $(\sigma^1)^2 = (v^1)^2$ and $m^1 = 0$. With increasing layer l , the sample variance becomes a smaller and smaller fraction

of total variance, until in the limit $l \rightarrow \infty$ the total variance becomes entirely variance of the sample mean $(m^l)^2$.

Before showing the preceding with mathematical formalism, we provide an intuitive explanation. It is straightforward to show that rectifying y increases its sample mean and decreases its sample variance (assuming that y has nonzero probability of being negative). Therefore rectification is expected to increase m/v . The other part of Eq. (1) is multiplication by a random matrix. We can show that this operation tends to preserve m/v because $\langle \langle (y_i^l)^2 \rangle \rangle = 2 \langle \langle (x_i^l)^2 \rangle \rangle$ and $\langle \langle (y_i^l)^2 \rangle \rangle = 2 \langle \langle (x_i^l)^2 \rangle \rangle$, assuming that W_{ij}^l are independent and identically distributed with zero mean and $2/N$ variance. The essential point is that $y_i^l = \sum_j W_{ij}^l x_j^l$ has zero mean when averaged over weights but is typically nonzero when sample averaged, which is why $\langle \langle (y_i^l)^2 \rangle \rangle$ is nonvanishing.

Because the networks we examine are repeated application of random matrix multiplication followed by half-wave rectification, we expect that m^2 should grow relative to v^2 in deeper layers. Note that this claim is specific to ReLU and may not hold for other nonlinear activation functions.

III. CALCULATING m^l, v^l IN WIDE NETWORKS

In Eq. (3), the quenched averages are easier to calculate because y_i^l is a sum of Gaussian random weights by Eq. (1). The sample averages are difficult because l nonlinearities intervene between y_i^l and the random input. Therefore it is advantageous to perform the quenched average before the sample average, as in $(\sigma^l)^2 = \langle \langle (y_i^l)^2 \rangle \rangle$. To switch the order of averaging for $(m^l)^2$, we use y_{i1}^l and y_{i2}^l to refer to the preactivities when the network is presented with two different input vectors drawn independent and identically distributed. Then

$$(m^l)^2 = \langle \langle y_{i1}^l y_{i2}^l \rangle \rangle = \langle \langle y_{i1}^l y_{i2}^l \rangle \rangle. \quad (4)$$

This identity relates the variance preservation principle to a body of work on “signal propagation” [5–7], which addresses the following question: If two different inputs are given to the same random net, do their preactivity vectors become more or less similar as we move deeper into the net? According to Eq. (4), the variance of the sample mean $(m^l)^2$ is a measure of similarity between the two preactivity vectors in layer l .

We can write $(\sigma^l)^2$ and $(m^l)^2$ as the diagonal and off-diagonal elements of the 2×2 matrix $\langle Q \rangle$, where Q is the covariance matrix,

$$Q_{\alpha\beta}^l = \langle y_{i\alpha}^l y_{i\beta}^l \rangle. \quad (5)$$

The index i is suppressed on the left-hand side because the right-hand side is independent of i by symmetry. It will also be convenient to define a normalized version of Q^l , the matrix of Pearson correlation coefficients,

$$C_{\alpha\beta}^l = Q_{\alpha\beta}^l / \sqrt{Q_{\alpha\alpha}^l Q_{\beta\beta}^l}. \quad (6)$$

Substituting $y_{i\alpha}^{l+1} = \sum_{j=1}^N W_{ij}^{l+1} [y_{j\alpha}^l]^+$ into Eq. (5) and using the fact that the weights are uncorrelated with variance $2/N$, we obtain $Q_{\alpha\beta}^{l+1} = 2 \langle \langle [y_{\alpha}^l]^+ [y_{\beta}^l]^+ \rangle \rangle$. In the limit of an infinitely wide network, Poole *et al.* [5], introduced a mean-field approximation in which preactivities on the right-hand side are Gaussian random variables with covariance matrix

Q^l . Evaluating the Gaussian integral [6,7] yields the recursion relation

$$Q_{\alpha\beta}^{l+1} = \sqrt{Q_{\alpha\alpha}^l Q_{\beta\beta}^l} K(C_{\alpha\beta}^l), \quad (7)$$

where

$$K(c) = c \left\{ 1 + \frac{1}{\pi} \left[\sqrt{1/c^2 - 1} - \cos^{-1}(c) \right] \right\}. \quad (8)$$

Since the diagonal elements of C^l are unity [see definition in Eq. (6)], Eq. (7) implies $Q_{\alpha\alpha}^{l+1} = Q_{\alpha\alpha}^l$, i.e., the diagonal elements of Q^l remain constant with l . Then the recursion of Eq. (7) simplifies to

$$C_{\alpha\beta}^{l+1} = K(C_{\alpha\beta}^l),$$

which is solved by the mapping K iterated l times, $C_{\alpha\beta}^l = K^l(C_{\alpha\beta}^0)$ [6,7]. The correlation in the first layer is proportional to the inner product of the input vectors, $Q_{\alpha\beta}^1 = (2/N) \mathbf{x}_\alpha^0 \cdot \mathbf{x}_\beta^0$, and $C_{\alpha\beta}^0$ is the cosine similarity between the same input vectors. Recursion yields the correlations in layer l ,

$$Q_{\alpha\beta}^l = (2/N) \|\mathbf{x}_\alpha^0\| \|\mathbf{x}_\beta^0\| K^l(C_{\alpha\beta}^0). \quad (9)$$

Averaging over inputs, the diagonal terms yield:

$$(\sigma^l)^2 = \langle Q_{\alpha\alpha} \rangle = (2/N) \langle \|\mathbf{x}^0\|^2 \rangle \quad (10)$$

and the off-diagonal terms yield:

$$(m^l)^2 = \langle Q_{12} \rangle = (2/N) \langle \|\mathbf{x}_\alpha^0\| \|\mathbf{x}_\beta^0\| K^l(C_{12}^0) \rangle, \quad (11)$$

where the average is over independently chosen sample pairs α, β . In deeper layers, we can simplify this equation using the fact that $K^l(C_{12}^0) \rightarrow 1$ for large l , so as long as the input pair are not perfectly antiparallel ($C_{12}^0 = -1$):

$$(m^l)^2 \rightarrow (2/N) \langle \|\mathbf{x}^0\|^2 \rangle. \quad (12)$$

Using the identity $(\sigma^l)^2 = (m^l)^2 + (v^l)^2$, we can derive behavior of v^l in the large- l regime:

$$(v^l)^2 \rightarrow (2/N) (\langle \|\mathbf{x}^0\|^2 \rangle - \langle \|\mathbf{x}^0\|^2 \rangle). \quad (13)$$

In a wide ReLU network, preactivity sample variance is twice the variance per element of the length of the input vector. When the input length has zero variance per element, which occurs when inputs are explicitly normalized to be uniform length, or when their elements are independent and identically distributed, the sample variance actually decays to zero as l increases.

IV. WHITE NOISE INPUTS

Suppose that elements of the input vector are independent and identically distributed with zero mean and variance equal to $1/2$. Then $N^{-1} \|\mathbf{x}^0\|^2 \rightarrow 1/2$ and $C_{\alpha\beta}^0 \rightarrow 0$ for $\alpha \neq \beta$ almost surely as $N \rightarrow \infty$. If we further assume the number of samples is large, so that $\alpha \neq \beta$ almost surely for any randomly chosen input pair, we can find that Eqs. (2), (10), and (11) yield:

$$(\sigma^l)^2 = 1 \quad (m^l)^2 = K^l(0) \quad (v^l)^2 = 1 - K^l(0). \quad (14)$$

We plot m^l, v^l as a function of l (Fig. 2) in this white noise, large sample number regime. The ratio m^l/v^l actually

FIG. 2. Preactivation statistics in random ReLU networks with white noise inputs. This provides a quantitative description of the phenomenon observed in Fig. 1. (a) Predicted sample standard deviation (v), standard deviation of sample mean (m), and total standard deviation (σ) of preactivations in wide net limit. (b) Numerically simulated ratio of m/v in finite width networks. Solid line indicates average and shaded regions indicate standard deviation of m/v over 30 random configurations of networks.

grows linearly with depth. This can be shown analytically by combining a result from Hayou *et al.* [7] that computed behavior of $K^l(c)$ for large l and using Eq. (11) to write m^l/v^l in terms of K^l . The result is that $m^l/v^l \approx \sqrt{2/(9\pi^2)} l$ for large l .

We test the validity of the infinite width approximation via numerical simulations in finite width networks with white Gaussian noise inputs. We sample 30 random networks of depth 50 with uniform width, N , per layer. We compare $N = 30, 100, 300, 1000, 3000$. We sample 100 input vectors with elements chosen independent and identically distributed from an $\mathcal{N}(0, 1/2)$ distribution. These are then propagated through each network and the empirical ratio of squared sample mean to average sample variance for each network is computed (Fig. 2). This ratio grows with network width and appears to asymptote to the value calculated in the infinite width regime.

V. DATA-DEPENDENT INITIALIZATION

Real-world neural network training often involves some form of data-dependent normalization, either via a one-time reinitialization [8,15] or via a dynamic normalization implemented at each training step [9–13,16].

One of the most widely used normalization schemes is batch normalization [9], which normalizes every preactivation by subtracting its mean and dividing by its standard deviation, computed over some random subset of samples. Because preactivations can be far from zero mean and unit variance at initialization, normalization can have a dramatic impact on the network's initialization. We now show that *normalized initialization can modify training dynamics*.

We do so by comparing training dynamics for two data-dependent initialization schemes, along with batch normalized networks. The first scheme, *total init*, implements the total mean-variance preservation principle and the second, *sample init*, roughly implements the sample mean-variance preservation principle. We do not use Kaiming initialization

FIG. 3. Training curves for (a) ALLCNN-C object recognition network and (b) UNet segmentation network. *sample init*, which implements the sample mean-variance preservation scheme, improved training speed compared to *total init*, which implemented total mean-variance preservation. Batch normalization yielded faster training networks than either initialization.

because we train nets containing pooling and residual connections that modify the total variance of units. In practice, one it may acceptable to ignore this complication; however, for fair comparison we will implement a data-dependent scheme to preserve the total variance at each layer.

Total init first estimates the total variance of a preactivation in each layer by estimating the variance over all n^l preactivations in the layer and B inputs. The weights are then rescaled so this empirical joint preactivation distribution has unit variance. No extra work is needed to preserve the total mean of a preactivation (it is zero because $y_i^l = \sum_j W_{ij}^l x_j^l$ so $\langle y_i^l \rangle = 0$). This algorithm is similar to Algorithm 1 from Mishkin and Matas [15].

Algorithm 1: Total Init

Sample B inputs $\{\mathbf{x}_1^0, \mathbf{x}_2^0, \dots, \mathbf{x}_B^0\}$ from dataset;
for each layer l do
 Initialize parameters: $W_{ij}^l \sim \mathcal{N}(0, 1)$ and $b_i^l = 0$;
 Compute preactivations: $\{\mathbf{y}_1^l, \mathbf{y}_2^l, \dots, \mathbf{y}_B^l\}$;
 Estimate variance: $(\sigma^l)^2 = \frac{1}{n^l B} \sum_{i=1}^{n^l} \sum_{b=1}^B (y_{ib}^l)^2$;
 Normalize weights: $W_{ij}^l \leftarrow W_{ij}^l / \sqrt{(\sigma^l)^2 + \epsilon}$;
end

Sample init first reinitializes the biases, *per-feature*, to subtract out each preactivation's estimated sample mean. Rescaling is then done per layer using the estimated sample variance of each unit. This is similar to Algorithm 1 from Krähenbühl *et al.* [8].

We compare both initialization schemes, along with batch normalization, for two *convolutional* networks trained on two popular computer vision tasks: object recognition with ALL-CNN-C [17] on the CIFAR10 dataset [18] and image segmentation with UNet [19] on the ISBI2012 dataset [20].

Note that these networks are more complicated than the fully connected ones we have so far discussed. The ALL-CNN-C network contains nine convolutional layers, a global pooling layer, and a single fully connected output layer. UNet contains 23 convolutional layers, four layers of max pooling, and additionally contains skip connections from earlier

to higher layers. We empirically confirm that the sample variance decays in both nets when initialized with *total init*. See the Supplemental Material [21] for figures showing the empirical sample mean-to-standard deviation ratio in these nets.

Algorithm 2: Sample Init

Sample B inputs $\{\mathbf{x}_1^0, \mathbf{x}_2^0, \dots, \mathbf{x}_B^0\}$ from dataset;
for each layer l do
 Initialize parameters: $W_{ij}^l \sim \mathcal{N}(0, 1)$ and $b_i^l = 0$;
 Compute preactivations: $\{\mathbf{y}_1^l, \mathbf{y}_2^l, \dots, \mathbf{y}_B^l\}$;
 Estimate mean: $\mu_i^l = \frac{1}{B} \sum_{b=1}^B y_{bi}^l$;
 Set biases: $b_i^l \leftarrow -\mu_i^l$;
 Estimate variance: $(\sigma^l)^2 = \frac{1}{n^l B} \sum_{i=1}^{n^l} \sum_{b=1}^B (y_{ib}^l - \mu_i^l)^2$;
 Normalize weights: $W_{ij}^l \leftarrow W_{ij}^l / \sqrt{(\sigma^l)^2 + \epsilon}$;
end

We train networks using stochastic gradient descent with momentum 0.9. We use a minibatch of 32 images for ALL-CNN-C and 1 image for UNet. We choose the optimal learning rate for each setting. See the Supplemental Material [21] for the learning rates. The optimal learning rate ended up being larger for sample init nets and batch norm nets than for total init nets. The learning curves and their standard deviation over three randomly initialized runs are shown in Fig. 3. The standard deviation is generally too small to see.

In both tasks, initializing the biases to subtract out each preactivation's sample mean results in improved training speed (Fig. 3). Using batch normalization (which both reinitializes and reparameterizes networks) sped up training more than just reinitializing networks. This suggests that part of the training speedup that is frequently associated with batch normalization may be a result of the impact it has on a network's initialization.

VI. DISCUSSION

We have thus far focused on the rectified linear activation, but our formalism applies to other activations as well. In sigmoid networks (initialized instead with Xavier initialization) we observe the sample mean to standard deviation ratio grows *exponentially* with depth. It is long been argued that the tanh nonlinearity, which is simply a rescaled and centered sigmoid: $\tanh(y) = 2 \text{sigmoid}(y) - 1$, should be used instead [2]. With this modification, the nonlinearity is antisymmetric around the origin, and we see no growth of the sample means.

We have studied feedforward networks, but our research bears similarity to studies of random recurrent networks [22–24]. Our feedforward nets can be viewed as a discrete time dynamical system with different couplings between states at each layer.

Our research is mathematically similar to studies of random feedforward neural nets in the physics literature but conceptually distinct because those studies have been motivated by explaining generalization performance. Li and Saad [25] investigated the sensitivity of neural activities in each layer to small perturbations of a randomly initialized net. De Palma *et al.* [26] showed that ReLU network output becomes

less sensitive to perturbations of binary-valued inputs with increasing network depth. Mingard *et al.* [27] applied an information theoretic argument to show that typical functions implemented by deep random ReLU networks are “low entropy.” This observation was used in Ref. [28] to hypothesize that this relative invariance to input perturbations could be a reason for neural network generalization performance.

In the computer science literature, Hayou *et al.* [7] applied the signal propagation formalism of Poole *et al.* [5] to show that neural activations from two different inputs become increasingly similar in higher layers. Hanin and Rolnick [29] used a combinatorial argument to show that deep random ReLU networks have “surprisingly few activation patterns.”

Our formalism provides a simple interpretation of the above phenomena: samples are mapped to increasingly small fluctuations around an increasingly large mean as they propagate through layers of a deep ReLU network. This suggests two peculiar aspects of neural activities in higher layers of these networks. One, half the neurons are “nearly dead”; half of the preactivations y are negative for most samples so the rectified activity $[y]^+$ is zero for most samples. Two, the other

half are “nearly linear”; half of the preactivations y are positive for most samples so $[y]^+ = y$. Centering each preactivation, thus ensures the nonlinearity for every unit is effectively used at initialization.

ACKNOWLEDGMENTS

This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) Contract No. D16PC0005, NIH/NIMH (U01MH114824, U01MH117072, RF1MH117815), NIH/NINDS (U19NS104648, R01NS104926), NIH/NEI (R01EY027036), and ARO (W911NF-12-1-0594). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government. We are grateful for assistance from Google, Amazon, and Intel.

-
- [1] K. He, X. Zhang, S. Ren, and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, Piscataway, NJ, 2015), pp. 1026–1034.
 - [2] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, Efficient backprop, in *Neural Networks: Tricks of the Trade*, edited by G. B. Orr and K.-R. Müller (Springer, Berlin, 1998), pp. 9–50.
 - [3] H. Zhang, Y. N. Dauphin, and T. Ma, Fixup initialization: Residual learning without normalization, Residual learning without normalization via better initialization, *International Conference on Learning Representations* (2019), <https://openreview.net/forum?id=H1gsz30cKX>.
 - [4] D. Arpit, V. Campos, and Y. Bengio, How to initialize your network? Robust initialization for weightnorm & resnets, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2019), pp. 10900–10909.
 - [5] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, Exponential expressivity in deep neural networks through transient chaos, in *Advances in Neural Information Processing Systems* 29, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., Red Hook, NY, 2016), pp. 3360–3368.
 - [6] Y. Cho and L. K. Saul, Kernel methods for deep learning, in *Advances in Neural Information Processing Systems* 22, edited by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (Curran Associates, Inc., Red Hook, NY, 2009), pp. 342–350.
 - [7] S. Hayou, A. Doucet, and J. Rousseau, On the selection of initialization and activation function for deep neural networks, [arXiv:1805.08266](https://arxiv.org/abs/1805.08266).
 - [8] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell, Data-dependent initializations of convolutional neural networks, [arXiv:1511.06856](https://arxiv.org/abs/1511.06856).
 - [9] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *Proceedings of the 32nd International Conference on Machine Learning, ICML'15*, Vol. 37 (JMLR.org, Lille, France), pp. 448–456.
 - [10] Y. Wu and K. He, Group normalization, in *Proceedings of the European Conference on Computer Vision (ECCV'18)* (Springer, Cham, Switzerland, 2018), pp. 3–19.
 - [11] S. Ioffe, Batch renormalization: Towards reducing minibatch dependence in batch-normalized models, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2017), pp. 1945–1953.
 - [12] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, Instance normalization: The missing ingredient for fast stylization, [arXiv:1607.08022](https://arxiv.org/abs/1607.08022).
 - [13] H. Nam and H.-E. Kim, Batch-instance normalization for adaptively style-invariant neural networks, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2018), pp. 2558–2567.
 - [14] K. Luther and H. S. Seung, Sample variance decay in randomly initialized ReLU networks, [arXiv:1902.04942](https://arxiv.org/abs/1902.04942).
 - [15] D. Mishkin and J. Matas, All you need is a good init, in *Proceedings of the 4th International Conference on Learning Representations, {ICLR} 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (2016), [arXiv:1511.06422](https://arxiv.org/abs/1511.06422).
 - [16] J. L. Ba, J. R. Kiros, and G. E. Hinton, Layer normalization, [arXiv:1607.06450](https://arxiv.org/abs/1607.06450).
 - [17] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, Striving for simplicity: The all convolutional net, ICLR (workshop track), (2015), <http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a>.
 - [18] A. Krizhevsky *et al.*, *Learning multiple layers of features from tiny images*, Tech. Rep. (Citeseer, Toronto, Canada, 2009).

- [19] O. Ronneberger, P. Fischer, and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, edited by N. Navab, J. Hornegger, W. Wells, and A. Frangi (Springer International Publishing, Cham, 2015), pp. 234–241.
- [20] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamensky, R. Burget, V. Uher, X. Tan, C. Sun, T. D. Pham, E. Bas, M. G. Uzunbas, A. Cardona, J. Schindelin, and H. S. Seung, Crowdsourcing the creation of image segmentation algorithms for connectomics, *Front. Neuroanat.* **9**, 142 (2015).
- [21] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/PhysRevResearch.2.033135> for further discussion on the datasets used to train nets, validation performance of networks, and sample variance decay in these real world nets.
- [22] H. Sompolinsky, A. Crisanti, and H. J. Sommers, Chaos in Random Neural Networks, *Phys. Rev. Lett.* **61**, 259 (1988).
- [23] R. Meir and E. Domany, Exact Solution of a Layered Neural Network Model, *Phys. Rev. Lett.* **59**, 359 (1987).
- [24] J. Kadmon and H. Sompolinsky, Transition to Chaos in Random Neuronal Networks, *Phys. Rev. X* **5**, 041030 (2015).
- [25] B. Li and D. Saad, Exploring the Function Space of Deep-Learning Machines, *Phys. Rev. Lett.* **120**, 248301 (2018).
- [26] G. De Palma, B. Kiani, and S. Lloyd, Random deep neural networks are biased towards simple functions, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., Red Hook, NY, 2019), pp. 1962–1974.
- [27] C. Mingard, J. Skalse, G. Valle-Pérez, D. Martínez-Rubio, V. Mikulik, and A. A. Louis, Neural networks are a priori biased towards boolean functions with low entropy, [arXiv:1909.11522](https://arxiv.org/abs/1909.11522).
- [28] G. Valle-Pérez, C. Q. Camargo, and A. A. Louis, Deep learning generalizes because the parameter-function map is biased towards simple functions, *International Conference on Learning Representations* (2019), <https://openreview.net/forum?id=rye4g3AqFm>.
- [29] B. Hanin and D. Rolnick, Deep ReLU networks have surprisingly few activation patterns, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., Red Hook, NY, 2019), pp. 361–370.