Beyond Signal Propagation: Is Feature Diversity Necessary in Deep Neural Network Initialization?

Yaniv Blumenfeld [1] **Dar Gilboa** [2] **Daniel Soudry** [1]

Abstract

Deep neural networks are typically initialized with random weights, with variances chosen to facilitate signal propagation and stable gradients. It is also believed that diversity of features is an important property of these initializations. We construct a deep convolutional network with identical features by initializing almost all the weights to 0. The architecture also enables perfect signal propagation and stable gradients, and achieves high accuracy on standard benchmarks. This indicates that random, diverse initializations are *not* necessary for training neural networks. An essential element in training this network is a mechanism of symmetry breaking; we study this phenomenon and find that standard GPU operations, which are non-deterministic, can serve as a sufficient source of symmetry breaking to enable training.

1. Introduction

Random, independent initialization of weights in deep neural networks is a common practice across numerous architectures and machine learning tasks (He et al., 2016; Vaswani et al., 2017; Szegedy et al., 2015). When backpropagation was first proposed (Rumelhart et al., 1986), neural networks were initialized randomly with the goal of breaking symmetry between the learned features. Empirically, it had been established that the method used to initialize the weights of each layer can have a significant effect on the accuracy of the trained model, and common initialization schemes (Glorot & Bengio, 2010; He et al., 2015) are generally motivated by ensuring that the variance of the neurons does not grow rapidly with depth at initialization.

Neural networks with random features (Rahimi & Recht, 2008) have been thoroughly studied, and such models admit a detailed analysis of their dynamics and generalization properties. Unlike the features in a neural network, the randomly initialized features in these models are not trained. While random features models are known to be limited compared to ones with learned or hand-designed features (Yehudai & Shamir, 2019), numerous studies have shown that classifiers can be trained to good accuracy relying solely on random, untrained features (Louart et al., 2018; Mei & Montanari, 2019).

A possible explanation for the role of random initialization was implied by the recently proposed "lottery ticket hypothesis" (Frankle & Carbin, 2018). After showing that sparse networks could be trained as effectively as dense networks if certain subsets of the weights were initialized with the same values, the authors suggest that neural networks contain trainable sub-networks, characterized by their unique architecture and initialization. These sparse sub-network can be extracted by first training dense neural networks and selecting weights with large magnitudes. Additionally, (Ramanujan et al., 2019) has shown that randomly initialized neural network contain sub-networks that achieve relatively high accuracy without any training. However, more recent studies of this topic (Frankle et al., 2019b;a) has put the significance of the initial sub-networks back into question. It was suggested that in the case of deeper models, the Lottery Ticket method is more effective when relying on the values of the weights in a dense network at advanced training epochs, and not the randomly initialization values.

While this may indicate that feature diversity is an important property of the network initialization, there is evidence to the contrary as well. Calculations of signal propagation in random neural networks (Poole et al., 2016; Schoenholz et al., 2016), applied in (Xiao et al., 2018) to convolutional networks, led the authors to suggest the Delta Orthogonal initialization, which they have used to successfully train a 10,000 layer convolutional network without skip connections on the CIFAR-10 dataset. The resulting initialization is relatively sparse, with only a single non-zero entry per convolution filter. When studying residual neural networks with a similar approach, as was done in (Yang & Schoenholz, 2017), it is apparent that signal propagation is optimized when the entire signal passes through the residual

[1]Technion, Israel [2]Columbia University. Correspondence to: Yaniv Blumenfeld <yanivblm6@gmail.com>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

connection at initialization, which can be achieved by simply initializing all weights that can be bypassed to zero. This approach is supported, to some extent by (Zhang et al., 2019), where the authors suggest the Fixup initialization in which the final layer of the residual block (a block which can be bypassed by a single skip connection) is initialized to zero.

The inevitable side-effect of these "sparse" initializations is that the variety of the sub-networks at initialization is limited, as zero-initialized parameters do not contribute new sub-networks to the grand total.

The initialization schemes proposed above suggest that feature diversity may not be necessary. In this work, we address the fundamental question: **Does deviating from the standard of independent, random initialization of weights have negative effects on training?** We do so by taking the idea of feature diversity to the extreme, and design networks where all the initial features are identical, while the requirements of signal propagation are maintained. We present surprising evidence that some networks are capable of fully recovering from these naive initializations during training, given some trivial requirements. We characterize the process where the features' symmetry is broken, distinguish between different levels of symmetry, and suggest criteria for the overall feature diversity in the network, which is shown to be tied with the success of the model.

2. Feature Diversity

When considering the function implemented by a neural network, the hidden state at every layer can be seen as a collection of *features*, extracted from the input by the preceding computational logic in the network. In a classification task, the subsequent logic in the network uses these features to classify the input to the target label. Features are a function of the inputs and therefore two features will be considered identical only if their respected neurons are equal for **all** possible inputs. As mentioned in the introduction, there are reasons to believe that identical features at initialization will be detrimental to training. Symmetries between parameters will, in general, induce symmetries on the features at initialization, though the manner in which the two are related will depend on the architecture. In a fully-connected layer for example, a sufficient condition for equality between two features is equality of the two corresponding rows of the weight matrix.

2.1. Shallow Networks with Identical Features

Before delving into deep models, it worthwhile to start by examining the effect of identical features in a simple toy model. For this task, we use a fully-connected single hidden layer with ReLU activations. Additional details are provided in Appendix E.

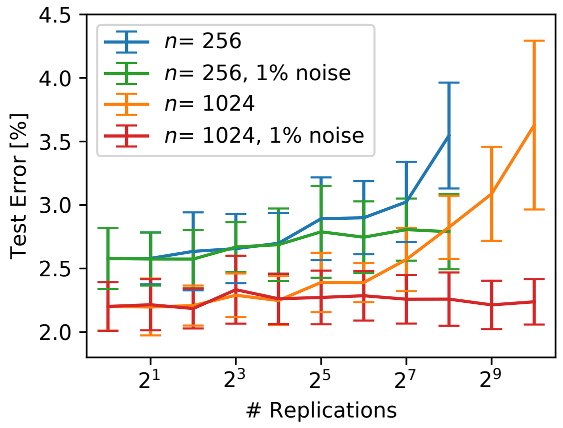

Figure 1. Test error of a 2-layers fully connected network, for networks with hidden layer of width n initialized with replicated features. The network performance degrades as the number of **unique** features declines, unless a small amount of noise is introduced during initialization.

The initial symmetry between features in this model can theoretically be broken by back-propagation alone. We test it by training the neural network over the MNIST dataset, while changing the width of the network n and the number of copies (replicas) we initialized for each unique row/feature.

As shown in figure 1, the test accuracy of the network degrades as the number of unique features at initialization decreases. Additional results, shown in the Appendix in figure 5, even suggest that initializing features with replicas of existing features may result in worse accuracy than removing those features altogether. Nevertheless, the negative effect of feature replication can be ameliorated by the addition of a small random independent 'noise' to the initial values of each replica. This suggests that the diversity of features at initialization can have long-term implications on the success of training, yet this effect appears to vanish quickly when minor stochastic elements are introduced.

2.2. ConstNet - A Convolutional Network with Identical Features at Initialization

In order to explore whether networks with identical features can be trained on more challenging tasks, we introduce ConstNet, an architecture based on the Wide-ResNet model (Zagoruyko & Komodakis, 2016). Apart of standard computation operations, residual networks contain *skip connections*, which can be described as an addition of identity operators to the operation performed by one or more layers. We provide full details of the design of ConstNet, in Appendix A. The ConstNet network, as used in the experiments, includes:

- An initial convolutional layer, initialized to average the input channels to identical copies.
- A variable number of "skip-able" convolutions, with 3 layers where the number of features is increased (referred to as *widening layers*). See figure 2 for illustration.
- All skip connections bypass a single ReLU + convolution block, *with its weights initialized to* 0.
- Values of the residual convolutions, used for network widening, are initialized to a constant.
- Skip section ends with a 2D pooling operation, followed by a fully connected layer initialized to zero.
- Optional batch-norm before each activation layer.

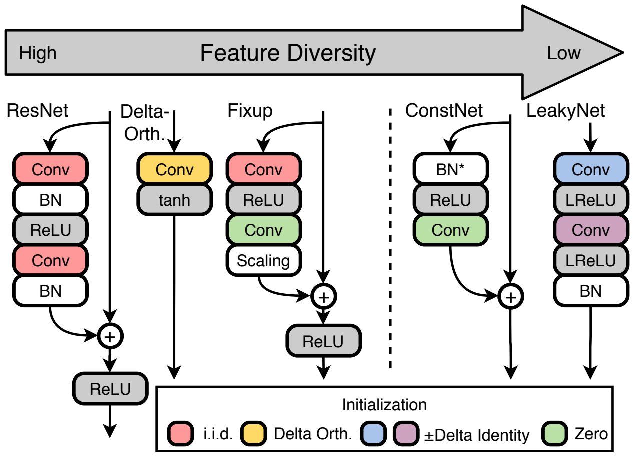

Figure 2. Convolutional architectures with reduced feature diversity. While a standard ResNet (He et al., 2016) is initialized with i.i.d. random weights, the Fixup initialization (Zhang et al., 2019) initializes one convolutional layer at 0 while the Delta-Orthogonal initialization (Xiao et al., 2018) factorized the weight tensors, initializing the filter matrix deterministically and hidden state matrix with a random orthogonal initialization. ConstNet and LeakyNet by constrast are *completely deterministic* at initialization with *symmetrical features* as a result. In ConstNet all convolutional layers are initialized at 0 while the LeakyNet initialization is identical to Delta-Orthogonal, with the random orthogonal matrix replaced by an identity that is multiplied by a negative factor at every odd layer. Both ConstNet and LeakyNet blocks implement an identity function at initialization. Batch-norm in ConstNet is optional, and we omit additional layers in Fixup that have no effect at initialization.

In order to motivate this architecture, we first show that due to the structure of the blocks in ConstNet, signals can propagate in a depth-independent manner from inputs to outputs, and gradients do not grow exponentially with depth. These are generally considered necessary conditions for trainabiliy of a neural network.

3. Signal Propagation at Initialization

Signal propagation in wide neural networks has been the subject of recent work for fully-connected (Poole et al., 2016; Schoenholz et al., 2016; Pennington et al., 2017; Yang & Schoenholz, 2017), convolutional (Xiao et al., 2018) and recurrent architectures (Chen et al., 2018; Gilboa et al., 2019). These works study the evolution of covariances between the hidden states of the network and the stability of the gradients. At the wide-networks limit, the covariance evolution depends only on the leading moments of the weight distributions and the nonlinearities at the infinite width limit, greatly simplifying analysis. They identify critical initialization schemes that allow training of very deep networks (or recurrent networks on long time sequence tasks) without performing costly hyperparameter searches.

We briefly review standard approaches to the study of signal propagation in neural networks, before specializing to ConstNet. To begin, we consider a fully-connected feed-forward network $f(x)$ given by

$$\begin{gathered}\alpha^{(0)}(x) = x\\ \tilde{\alpha}^{(\ell)}(x) = W^{(\ell)}\alpha^{(\ell-1)}(x) + b^{(\ell)} \quad l = 1,\dots,L-1\\ \alpha^{(\ell)}(x) = \phi(\tilde{\alpha}^{(\ell)}(x)) \quad l = 1,\dots,L-1\\ f(x) = W^{(L)}\alpha^{(L-1)}(x) + b^{(L-1)}.\end{gathered}$$

with $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, b^{(\ell)} \in \mathbb{R}^{n_l}$ initialized according to $W_{ij}^{(\ell)} \sim \mathcal{N}(0, \sigma_\ell^2)$ i.i.d., $b_i^{(\ell)} = 0$, and ϕ is a nonlinearity that acts element-wise on vectors. Here $\alpha^{(\ell)}(x) \in \mathbb{R}^{n_\ell}$ is a vector of *forward features* at layer ℓ. We denote the scalar loss by $\mathscr{L}(f(x))$.

Common initialization schemes are motivated by guaranteeing stability of signals propagating from the inputs to the outputs by ensuring that the variance of the forward features does not change with depth (He et al., 2016; Vaswani et al., 2017; Szegedy et al., 2015). For example, if we take ϕ to be the ReLU activation function, since $\sum_j W_{ij}^{(\ell)}\alpha_j^{(\ell-1)}|\alpha^{(\ell-1)}$ is a Gaussian with variance $\sigma_\ell^2 \left\|\alpha^{(\ell-1)}\right\|_2^2$ we obtain

$$\underset{W^{(\ell)}}{\mathbb{E}}\left(\alpha_i^{(\ell)}\right)^2 |\alpha^{(\ell-1)} = \underset{W^{(\ell)}}{\mathbb{E}}\,\phi\left(\sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)}\alpha_j^{(\ell-1)}\right)^2 |\alpha^{(\ell-1)}$$

$$= \sigma_\ell^2 \left\|\alpha^{(\ell-1)}\right\|_2^2 \int_0^\infty g^2 \mathscr{D}g = \frac{\sigma_\ell^2 \left\|\alpha^{(\ell-1)}\right\|_2^2}{2}$$

where $\mathscr{D}g$ is a standard Gaussian measure. The variance is preserved if $\mathbb{E}\left(\alpha_i^{(\ell)}\right)^2 |\alpha^{(\ell-1)} = \frac{\left\|\alpha^{(\ell-1)}\right\|_2^2}{n_{\ell-1}}$ which is guaranteed by choosing $\sigma_\ell^2 = \frac{2}{n_{\ell-1}}$.

Additionally, one can ensure that not only the variances of features are insensitive to depth but also the covariances between features given different inputs (Poole et al., 2016;

Schoenholz et al., 2016; Xiao et al., 2018). For the forward features, these are given by

$$\Sigma_j^{(\ell)}(x,x') = \mathop{\mathbb{E}}_{\substack{\{W^{(i)},\\ b^{(i)}\}}} \begin{pmatrix} \left(\alpha_j^{(\ell)}(x)\right)^2 & \alpha_j^{(\ell)}(x)\alpha_j^{(\ell)}(x') \\ \alpha_j^{(\ell)}(x)\alpha_j^{(\ell)}(x') & \left(\alpha_j^{(\ell)}\right)^2 \end{pmatrix} \quad (1)$$

where the expectations are taken over the weight distribution. At the wide network limit with random weights under weak moment assumptions, the above equation reduces to a deterministic dynamical system since the pre-activations are jointly Gaussian as a consequence of the Central Limit Theorem (Poole et al., 2016; Schoenholz et al., 2016; Matthews et al., 2018). As a result, at this limit the signals propagated through the network can be described completely in terms of these covariances, which are also independent of the neuron index j. Additionally, for a sufficiently wide network (typically once the width is few hundred neurons) the ensemble average above has proven to be predictive of the behavior of individual networks.

The covariances evolve according to

$$\Sigma^{(\ell+1)}(x,x') = \mathop{\mathbb{E}}_{\substack{(u_1,u_2)\sim\\ \mathcal{N}(0,n_\ell\sigma_\ell^2\Sigma^{(\ell)}(x,x'))}} \begin{pmatrix} \phi(u_1)^2 & \phi(u_1)\phi(u_2) \\ \phi(u_1)\phi(u_2) & \phi(u_2)^2 \end{pmatrix}.$$

By studying this dynamical system one can obtain initialization schemes that allow signals to propagate stably, even in very deep networks, enabling them to be trained (Schoenholz et al., 2016; Xiao et al., 2018)[1].

In order to facilitate trainability with gradient descent, one can also study the variance of the *backward features*

$$\beta_j^{(\ell)}(x) = \frac{\partial\mathcal{L}}{\partial\tilde{\alpha}_j^{(\ell)}(x)}$$

which are of interest, since the gradients take the form

$$\frac{\partial\mathcal{L}(x)}{\partial W_{ij}^{(\ell)}} = \beta_i^{(\ell)}(x)\alpha_j^{(\ell-1)}(x). \quad (2)$$

In the case of convolutional networks, where $W^{(\ell)} \in \mathbb{R}^{K\times n_\ell\times n_{\ell-1}}, b^{(\ell)} \in \mathbb{R}^{n_\ell}$, the backward and forwards features are tensors $\alpha^{(\ell)}(x) \in \mathbb{R}^{S^{(\ell)}\times n_\ell}, \beta^{(\ell)}(x) \in \mathbb{R}^{S^{(\ell)}\times n_\ell}$ where $S^{(\ell)}$ is the space of spatial dimensions (pixels) at layer ℓ. We will denote by γ (or other Greek letters) a vector denoting the spatial location and K denotes the dimensions of the kernel. We denote the convolution with respect to the kernel and a summation over the feature index by $\hat{*}$. The features are updated according to

$$\begin{gathered} \alpha^{(0)}(x) = x, \\ \tilde{\alpha}_{\gamma j}^{(\ell)}(x) = \left[W^{(\ell)} \hat{*} \alpha^{(\ell-1)}(x)\right]_{\gamma j} + b_j^{(\ell)} \\ = \sum_{i=1}^{n_{\ell-1}} \sum_{\kappa\in K} W_{\kappa ij}^{(\ell)} \alpha_{\gamma+\kappa,i}^{(\ell-1)}(x) + b_j^{(\ell)}, \\ \alpha^{(\ell)}(x) = \phi(\tilde{\alpha}^{(\ell)}(x)), \\ f(x) = P(\alpha^{(L-1)}(x)). \end{gathered}$$

where P is a function independent of depth (typically a composition of a pooling operation and an affine map). For simplicity of exposition we also assume periodic boundary conditions in the spatial dimensions. The covariances above can then be generalized to a tensor $\Sigma_{\gamma\gamma' jj'}^{(\ell)}(x,x')$ that can be analyzed in a similar manner to the fully-connected case (Xiao et al., 2018). Analogously to eq. 2, the gradients are related to the features by

$$\frac{\partial\mathcal{L}(x)}{\partial W_{\kappa ij}^{(\ell)}} = \sum_\gamma \beta_{\gamma i}^{(\ell)}(x)\alpha_{\gamma+\kappa,j}^{(\ell-1)}(x). \quad (3)$$

At the infinite width it was shown that using smooth activations, stable gradients at initialization can be obtained with careful hyper-parameter tuning, inducing a dependence between the weight variance and the depth (Pennington et al., 2017). In (Burkholz & Dubatovka, 2019) it was recently shown that weight sharing at initialization enables signal propagation in feed-forward networks with ReLU activations. In Appendix B we show that this approach can also be extended to convolutional networks by constructing a random initialization that guarantees signal propagation surely (and not just in expectation over the weights). As a consequence, the result is applicable to networks of arbitrary width.

We also note that the FixUp initialization (Zhang et al., 2019) does not exhibit stable backwards signal propagation since the gradients to certain layers within each block are zero at initialization (see figure 2 for details).

3.1. Depth-independent signal propagation with constant weights and skip connections

We now consider a class of neural networks that exhibit perfect forward signal propagation to arbitrary depth at initialization, and depth-independent backwards signal propagation by dispensing with randomness at initialization and utilizing skip connections. ConstNet is a member of this class.

Given a hidden state tensor $\alpha \in \mathbb{R}^{h\times w\times n}$ where h, w are spatial dimensions and n is the number of filters, we define for some integer s such that $h \mod s = w \mod s = 0$ a *widening block* by

$$\mathrm{WB}_{W,b,s,n} : \mathbb{R}^{h\times w\times n} \to \mathbb{R}^{h/s\times w/s\times sn},$$

[1]These works in fact study closely related covariances defined for the pre-activations instead since these are jointly Gaussian at the infinite width limit and thus the evolution of the covariances obeys a simple closed form equation. See Appendix B of (Blumenfeld et al., 2019) for details of the relation between the two.

$$\left[\mathrm{WB}_{W,b,s,n}(\tilde{\alpha})\right]_{\gamma i} = \frac{1}{n}\sum_{j=1}^{n} \tilde{\alpha}_{s\gamma,j} + \left[W \circledast g(\tilde{\alpha})\right]_{\gamma i} + b_i.$$

where $g : \mathbb{R}^{h\times w\times n} \to \mathbb{R}^{h\times w\times n}$ is a differentiable function such as a composition of a non-linearity and batch-normalization, and $W \in \mathbb{R}^{K\times sn\times n}, b \in \mathbb{R}^{sn}$*are initialized as* 0. This is equivalent at initialization to a convolution with a 1×1 identity filter, stride s and a constant matrix acting on the channels.

We define a *ConstNet block function* as a map

$$\mathrm{CB}_{W,b,n} : \mathbb{R}^{h\times w\times n} \to \mathbb{R}^{h\times w\times n},$$

$$\left[\mathrm{CB}_{W,b,n}(\tilde{\alpha})\right]_{\gamma i} = \tilde{\alpha}_{\gamma i} + \left[W \circledast g(\tilde{\alpha})\right]_{\gamma i} + b_i$$

where $W \in \mathbb{R}^{K\times n\times n}, b \in \mathbb{R}^{n}$ *are initialized as* 0.

Considering inputs $x \in \mathbb{R}^{S^{(0)}\times n_d}$, we define a depth L *ConstNet function* by

$$f(x,B) = P(O_{L-1}(O_{L-2}(\ldots O_1(\tilde{\alpha}^{(0)}(x))\ldots))) \qquad (4)$$

where O_i is either a ConstNet block or a widening block, P is a differentiable operation (typically a composition of pooling and an affine map) and B is a batch of datapoints containing x. We define $\tilde{\alpha}^{(0)}(x,B) \in \mathbb{R}^{S^{(0)}\times n_0}$ to be a function of B (but dropping the B dependence to lighten notation), that obeys $\tilde{\alpha}^{(0)}_{\gamma i}(x) = \tilde{\alpha}^{(0)}_{\gamma j}(x)$ and normalized so that $\sum\limits_{\gamma\in S^{(0)}}\sum\limits_{j\in B} \tilde{\alpha}^{(0)}_{\gamma i}(x_j) = 0$ and $\sum\limits_{\gamma\in S^{(0)}}\sum\limits_{j\in B}\left(\tilde{\alpha}^{(0)}_{\gamma i}(x_j)\right)^2 = 1$ (which can be achieved by applying a convolution and batch normalization operation to x for instance). We assume the stride parameter s and $S^{(0)}$ are chosen such that the spatial dimension remains larger than 0 throughout.

We consider translation invariant inputs, meaning for any $x, x' \in \mathbb{R}^{S^{(0)}\times n_d}$ drawn from the data distribution we have

$$x_{\gamma k} \overset{d}{=} x'_{\gamma' k}$$

where $\overset{d}{=}$ denotes equality in distribution and γ, γ' are vectors denoting spatial location. A local invariance to translation is a well-studied property of natural images and is believed to be key to the widespread use of convolutional networks in image classification tasks.

Claim. *Let f be an L-layer ConstNet function as in eq. 4 and denote the scalar loss by $\mathscr{L}$. Then for any $0 \le \ell \le L-1$ we have*

$$\frac{\left\langle \beta^{(\ell)}(x), \beta^{(\ell)}(x')\right\rangle}{n_\ell} = \frac{\left\langle \beta^{(L-1)}(x), \beta^{(L-1)}(x')\right\rangle}{n_{L-1}}$$

$$\frac{\partial \mathscr{L}(x)}{\partial W^{(\ell)}_{\kappa i j}} = C_{ij}\cos(\theta_{\kappa,\ell})$$

$$\frac{\partial \mathscr{L}(x)}{\partial b^{(\ell)}_{i}} = C'_i$$

where $C_{ij}, C'_i > 0$ are constants that are independent of L (but depend on the functions P, g in the definition of the ConstNet function and on $\mathscr{L}$). $\theta_{\kappa,\ell}$ are constants that can depend on L.

Additionally, for translation invariant inputs, if we denote the spatial dimensions at layer ℓ by $S^{(\ell)}$ we have for any $\gamma, \gamma' \in S^{(\ell)}$

$$\tilde{\alpha}^{(\ell)}_{\gamma i}(x) \overset{d}{=} \tilde{\alpha}^{(0)}_{\gamma' 1}(x)$$

$$\frac{\left\langle \tilde{\alpha}^{(\ell)}(x), \tilde{\alpha}^{(\ell)}(x')\right\rangle}{\left|S^{(\ell)}\right| n_\ell} \overset{d}{=} \frac{\left\langle \tilde{\alpha}^{(0)}(x), \tilde{\alpha}^{(0)}(x')\right\rangle}{\left|S^{(0)}\right| n_0}$$

Proof: See Appendix C

The claim shows that angles between forward and backward features are preserved by ConstNet (the former only in distribution), and that the gradients cannot grow exponentially with depth. The stability of angles between inputs is known to be predictive of trainability and generalization in many architectures (Schoenholz et al., 2016; Pennington et al., 2017; Xiao et al., 2018). In networks where angles are not preserved, training tends to fail. Note that the stability conditions in the above claim hold for arbitrary width, unlike similar results that only apply to wide networks.

One can ask whether the depth-independent signal propagation in ConstNet is a sufficient condition for it to be trainable. Indeed, this property is often considered a necessary but insufficient condition for trainability. The answer turns out to be negative due to the symmetry between features at all layers (which is also not surprising, given the results in Section 2.1), yet there happens to be a simple solution to this issue.

4. Feature Symmetry and Symmetry Breaking

Two different features of the same hidden layer, which were initialized to present an identical function of the input, are still expected to diverge during back-propagation if their connection to the output is weighted differently. However, in the case when the their respective connection to the following layer is also symmetrical, the two features become interchangeable, and are expected to get the same updates in a deterministic process. This is the case for all the initial features in ConstNet as well. It stands to reason that some form of stochasticity must be introduced into the training process, in order for the initial symmetry to be broken. This could be achieved by a deliberate injection of noise to the parameters or gradients, or by relying on existing training mechanisms, such as Dropout.

One additional source of stochasticity that is often overlooked is the computation process itself. For example, the order of execution when parallelizing GEMM (General Matrix Multiplication) operations is often non-deterministic,

Model	Init	Variant	Accuracy [%]
Wide-ResNet	He	–	95.77 ± 0.05
ConstNet	He	–	95.40 ± 0.07
ConstNet	0	1% Dropout	95.46 ± 0.13
ConstNet	0	–	95.37 ± 0.06
ConstNet	0	Deterministic	24.79 ± 0.58

Table 1. Stochastic GPU computations enables training with identical features. Test accuracy on CIFAR-10, with 300 epochs. A minimal mechanism of symmetry breaking (e.g small dropout or non-deterministic GPU operations) is sufficient for ConstNet with almost all weights initialized at 0 to train, matching the test accuracy achieved with standard random initialization. Training fails without such a mechanism (the "Deterministic" variant).

resulting a stochastic output for non-commutative operations (including the addition of floating point values). It is therefore one of the mechanisms that may enable training with gradient descent to break the symmetry between the features at initialization. In our experiments, we used Nvidia GTX-1080 GPUs and the cudNN library, which is non-deterministic by default[2].

Results of training ConstNet with different symmetry breaking mechanisms, as well as comparison with the baseline of He random initialization, can be seen in figure 1. Surprisingly, the effect of identical features at initialization appears to be minor whenever any form of symmetry breaking was allowed, and even negligible when non-deterministic computation noise is allowed. Not only was the training process capable of breaking features symmetry, but the overall test accuracy was not affected when compared to runs where the initial features were diverse (when 'He' initialization was used). Our immediate conclusion is that the fixed point where networks features are identical is extremely unstable. When no mechanism of symmetry breaking is present (the deterministic run with no Dropout) training fails.

It should be noted that '0' initialization does have its flaws, which were not captured in this experiment. The loss/accuracy curves of training when '0'-init was used, were typically few epochs behind the curves of its random initialization counterparts. On some occasions, depending on the hardware specifications, we have identified cases where the '0'-init runs relying only on hardware noise were stuck on the initial stages of training (10% accuracy) for a random number of epochs. This phenomenon can be avoided by adding dropout as a complementary symmetry breaking mechanism, and did not significantly affect the final accuracy, due to the high number of epochs in this experiment. More results regarding the effect of different symmetry breaking mechanisms are detailed in appendix I and appendix I.1. Furthermore, 0-initialized network were less robust to learning rate changes, and would fail completely for high learning rates, which could be handled by 'He' initialized networks.

[2]See https://github.com/NVIDIA/tensorflow-determinism for additional details.

4.1. Evolution of Weight Correlations

To analyze the phenomena of symmetry breaking, we will first define macro parameters which can represent the degree of symmetry in our network. Given a convolution with the weights $W \in \mathbb{R}^{k_x \times k_y \times n_{\text{out}} \times n_{\text{in}}}$, we can split the weight tensor W to either n_{out} tensors that defines each of our output channels as $W_{\text{f}}^i \in \mathbb{R}^{k_x \times k_y \times n_{\text{in}}}, 0 \leq i < n_{\text{out}}$, or split it to n_{in} tensors, where each tensor, $W_{\text{b}}^i \in \mathbb{R}^{k_x \times k_y \times n_{\text{out}}}, 0 \leq i < n_{\text{in}}$, contain the weights that operate on a single input channel. For our macro-parameter, we will look at the mean *correlation* between the different tensors (W_{b} or W_{f}), and denote them as the *forward correlations* C_f and *backward correlations* C_b as:

$$C_f \equiv \frac{1}{n_{\text{out}}\left(n_{\text{out}}-1\right)} \sum_{i}^{n_{\text{out}}} \sum_{i \neq j}^{n_{\text{out}}} \frac{W_{\text{f}}^i \cdot W_{\text{f}}^j}{\left\|W_{\text{f}}^i\right\|_F \left\|W_{\text{f}}^j\right\|_F}, \tag{5}$$

$$C_b \equiv \frac{1}{n_{\text{in}}\left(n_{\text{in}}-1\right)} \sum_{i}^{n_{\text{in}}} \sum_{i \neq j}^{n_{\text{in}}} \frac{W_{\text{b}}^i \cdot W_{\text{b}}^j}{\left\|W_{\text{b}}^i\right\|_F \left\|W_{\text{b}}^j\right\|_F} \tag{6}$$

where $A \cdot B$ is an element-wise dot product operations, and $\|\cdot\|_F$ is Frobenius norm. For the case of a fully-connected layer, the operation can be viewed as a convolution where each of the input's and output's channels is of size 1×1, with a kernel of the same size.

The definitions of backward and forward correlations allow us to go back and examine the different initializations presented in section 4:

- For a weight tensor initialized with random, unbiased initialization, $C_f = 0, C_b = 0$ in the limit where the number of input/output channels or kernel dimensions goes to infinity.
- For an orthogonal initialized weight tensor in the channel dimensions, $C_f = 0, C_b = 0$.
- When the identity operation is expanded to also replicate n_{in} channels to $n_{\text{out}} = P \times n_{\text{in}}$, $C_f = \frac{P-1}{n_{\text{out}}-1}, C_b = 0$.
- When all the filters are initialized to a constant $C_f = 1, C_b = 1$.
- Replicating features affects the forward correlation only. For d sized output and R replication per feature, $C_f = \frac{R-1}{d-1}, C_b = 0$.

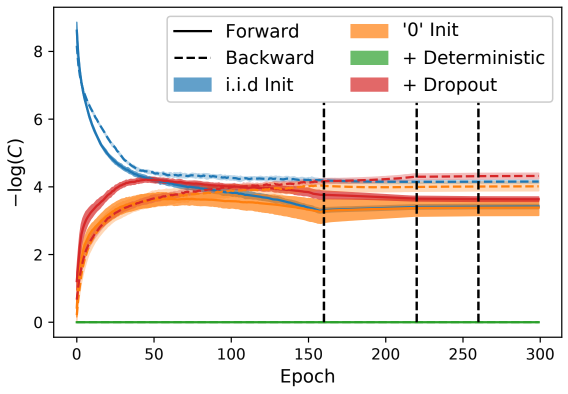

Figure 3. Weight correlations converge to similar values despite different initializations. The forward correlation (solid) of a model with zero initialization and full feature symmetry eventually matches that of a model initialized without feature symmetry (He). Backward correlations (dashed) also converge to relatively similar values. When no mechanism for breaking symmetry was introduced (Green), the tensor remained maximally correlated, and training fails. Similar behavior is observed in almost all convolutional layers.

- For zero initialization, the fraction is undefined, but will be considered as fully correlated, $C_f = 1, C_b = 1$.

In the case of ConstNet with '0' init, all layers had been initialized with $C_f = 1, C_b = 1$. In contrast, standard random initialization will result in near-zero correlations. Correlations at a specific layer will be denoted by $C_f(\ell), C_b(\ell)$.

A characteristic example for the evolution of the forward and backward correlations of a single tensor in ConstNet, during training, can be seen in figure 3, and the full results can be see in figure 7. An interesting finding when comparing random and constant initialization is that in almost all layers, the forward correlations appear to converge to similar values for both initializations (He or ConstNet). Due to the highly nonconvex optimization landscape, one might expect the properties of the solutions that gradient descent finds to depend on the initialization in some way. At least with respect to feature correlations this appears to not be the case.

Since perfectly correlated features during inference are equivalent to a single neuron, it is tempting to interpret the dissimilarity $1 + (n_\ell - 1)\sqrt{1 - C_f(\ell)^2}$ as an "effective width" of a layer. The convergence of the correlations in figure 3 despite different initializations indicate that gradient descent is biased towards solutions with a certain effective width regardless of the initial degree of feature diversity, as long as there is a symmetry breaking mechanism present.

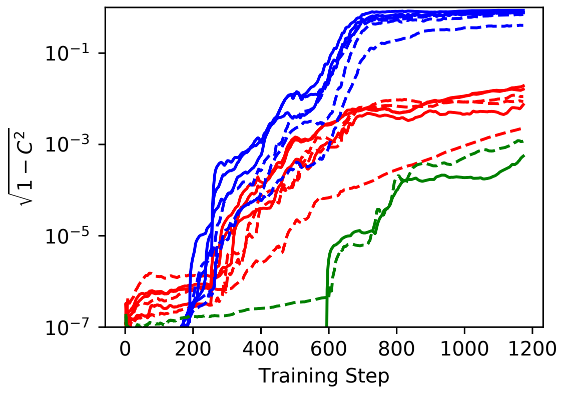

Figure 4. The early evolution of backward and forward correlations is group dependent (larger values are less correlated). This figure describe the forward correlations (solid) and backward correlations (dashed) members of different groups in ConstNet with '0' init, at the first 3 epochs of training (1 Epoch = 391 Steps). Different colors indicate different groups, as explained in section 4.2. A more detailed view of the individual correlations is available on figure 6 in the Appendix.

4.2. Group behaviour of symmetry breaking

When observing the the behaviour of the correlations in different layers in ConstNet at the initial stages of training, we identify a group behaviour. The forward and backward correlations of the members of each group appears to behave in a similar manner. We identify several distinct groups in ConstNet, corresponding to the different widths of the hidden layers. We hypothesize that the layers between the the different widths, where we have a highly correlated averaging operation, are responsible for this phenomena. An example of this can be seen in figure 4. While the number of those groups in ConstNet is limited and depth independent (such operations are done only when the network width is increased), we suspect that an intensive use of this initialization, and thus a greater amount of individual groups required to break symmetry, could result a failure to train, despite the naive conservation of signal in depth.

5. Propagation of Signals that Break Symmetry

In the previous sections, we have shown that networks with identical features can be trained if they i) enable signal propagation and ii) possess a mechanism for symmetry breaking. It appears that even when these conditions are satisfied, *symmetry breaking can still be hindered if the signals that break symmetry cannot propagate* through the network. This is a novel type of signal that is not considered in standard analyses of signal propagation. We illustrate this phenomenon by designing a convolutional network that satisfies the above conditions (as shown in Appendix H), yet still cannot be trained without additionally ensuring symmetry breaking signal propagation.

5.1. LeakyNet

Rather than using the skip connection as a means of signal propagation, we can suggest alternative methods to initialize networks to represent identity at initialization, for the signal to be preserved. The main obstacle is the activation functions, being non-linear. We can overcome this limitation by replacing the *ReLU* activation with the *Leaky ReLU*: $\sigma^{\rho}(x) \equiv \mathrm{LReLU}(x, \rho) = \rho x + (1-\rho)\mathrm{ReLU}(x)$. In the limit $\rho \to 0$, both activations are identical, and we can use invertibility of leaky ReLU to form an identity. More specifically, we will use the equality: $\sigma^{\rho}(-\frac{1}{\rho} I \sigma^{\rho}(Ix)) = -x$. Simply put, we initialize all layers so that every neuron swaps sign after each activation, making the composition of two nonlinear computation blocks linear (see figure 2).

To enforce identical features, it is sufficient to ensure that the first layer's features are identical, which we do by averaging the incoming channels, as in ConstNet. On instances when the network is widened, we map each feature to several identical copies. Apart of the computational block, all other components of network are similar to these presented in ConstNet. We train LeakyNet with the same configuration used for training ConstNet, and a default parameter $\rho = 0.01$ for the leaky ReLUs. In all runs, the final layer was initialized to zero.

The benefit of this architecture is that after factorizing the weight tensors into kernel matrices and matrices that act on the channels, we can choose between initializing these as an identity (denoted by 'I'), and a matrix $\frac{1}{n}\mathbf{1}\mathbf{1}^T$ which we denote by '$\mathbb{1}$'. As seen in section 4.1, the two different initializations will have a dramatically different effect on feature correlations, even though they both enable perfect signal propagation at initialization. Since the '$\mathbb{1}$' initialization averages over all channels, its output will be multiple copies of a single feature, hence they will be perfectly correlated. During training, as the symmetry between features is gradually broken due to some source of non-determinism, these symmetry breaking signals will decay when passing through a layer with a '$\mathbb{1}$' initialization, but not when passing through a layer with 'I' initialization. We discuss this further in Appendix G.

5.2. Controlling the Effectiveness of Symmetry Breaking

Consequently, if we hold the total depth fixed and begin with a network that uses the 'I' initialization at all layers, *we can progressively hinder symmetry breaking by initializing more of layers with the '$\mathbb{1}$' initialization* instead. We perform a series of experiments of this nature. For reference, we also train this network with random (He) initialization, adjusted for the gain of the leaky ReLU initialization.

When random initialization was used, the correlations have frequently converged to a **higher value**, indicating some degree of features co-adaptation. The relative success of random initialization, compared with the orthogonal initialization, implies that feature diversity (in the sense of maximal forward correlations) have a negative effect as well, when examining the network in advanced stages of training. It is possible that it was, in fact, the orthogonal initializations that failed in the task of features co-adaptation. The full results regarding forward correlations are presented in figure 8 in the appendix.

Initialization (# Conv Layers)				Test Acc. [%] (4 seeds)	Max C_f	Mean $\sqrt{1-C_f^2}$
Total	He	'I'	'$\mathbb{1}$'			
13	13	-	-	95.57 ± 0.12	0.21	0.97
13	-	13	0	94.05 ± 0.23	0.27	0.97
13	-	12	1	94.34 ± 0.12	0.25	0.97
13	-	10	3	93.47 ± 0.32	0.63	0.94
13	-	8	5	87.85 ± 0.52	1.00	0.79
13	-	6	7	44.89 ± 2.52	1.00	0.52
13	-	0	13	29.77 ± 2.47	1.00	0.21

Table 2. Test Accuracy of LeakyNet is correlated with dissimilarity between features. Results are on the CIFAR-10 dataset. Each network is initialized with a different number of '$\mathbb{1}$' and 'I' initialization. The final performance degrades as more '$\mathbb{1}$' layers are present. The mean dissimilarity between features $\frac{1}{L}\sum_{\ell}\sqrt{1-C_f^2(\ell)}$ is indicative of final test accuracy, and networks where the maximal feature correlation $\max_{\ell} C_f(\ell)$ was close to 1 performed poorly.

6. Discussion

In this work, we have shown that random initialization is not a necessary condition for deep convolutional networks to be trainable. Depth-independent propagation of signals through the network can be achieved without recourse to random initialization through the use of skip connections, or through artificial initialization methods. In one case, we show that the initial symmetry forced on a model is so fragile during training, that non-deterministic computation is sufficient to bring an otherwise untrainable model to a final accuracy of 95% on the CIFAR-10 benchmark.

By experimenting with radically uniform, naive initializations, we identify cases where the lack of feature diversity and overall symmetry in the network lead to failure of deep neural networks at classification tasks. Nonetheless, our ultimate conclusion is that feature diversity should **not** be a major factor, when initializing a deep neural network. The extent of symmetry we had to enforce over the network at initialization to disrupt the training process to a distinguishable degree, indicates that there is no significant penalty from subtle adjustments that sacrifice random initialization to improve the network's dynamics. Any negative effect we did encounter, could be easily negated by a small addition of independent, random values to break the initial symmetry.

While our symmetrically initialized networks are not intended for practical training, their simplicity can prove useful for the purpose of theoretical analysis. ConstNet, for example, implements a trivial mapping at initialization (as shown in Appendix D). On the other hand, there is a growing theoretical literature analyzing the behavior of deep networks by linearizing near initialization (Jacot et al., 2018; Arora et al., 2019; Lee et al., 2019). Our work suggests that understanding the behaviour of typical networks will likely require analysis of the nonlinear process of feature learning from data that happens in later stages of training. This of course is far from a novel conclusion, yet is made more palpable when considering that the features in the ConstNet model at initialization are identical to features of a linear model, as shown in Appendix D. A relatively complex task like CIFAR-10 classification cannot be solved using kernel regression with such features. This is a clear indication that after training, the network is far from the linear regime around initialization.

Acknowledgements

The research of DS was supported by the Israel Science Foundation (grant No. 31/1031), by the Israel Innovation Authority (the Avatar Consortium), and by the Taub Foundation. A Titan Xp used for this research was donated by the NVIDIA Corporation. The work of YB is supported by Leonard and Diane Sherman Interdisciplinary Graduate School Fellowship. This work was partially done while the DS and DG were visiting the Simons Institute for the Theory of Computing. The work of DG was supported by the NSF NeuroNex Award DBI-1707398, and the Gatsby Charitable Foundation.

References

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

Blumenfeld, Y., Gilboa, D., and Soudry, D. A mean field theory of quantized deep networks: The quantization-depth trade-off. In *Advances in Neural Information Processing Systems*, pp. 7036–7046, 2019.

Burkholz, R. and Dubatovka, A. Initialization of relus for dynamical isometry. In *Advances in Neural Information Processing Systems*, pp. 2382–2392, 2019.

Chen, M., Pennington, J., and Schoenholz, S. S. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. *arXiv preprint arXiv:1806.05394*, 2018.

Chizat, L. and Bach, F. A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. *arXiv preprint arXiv:1912.05671*, 2019a.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. The lottery ticket hypothesis at scale. *arXiv preprint arXiv:1903.01611*, 2019b.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, pp. 9108–9118, 2019.

Gilboa, D., Chang, B., Chen, M., Yang, G., Schoenholz, S. S., Chi, E. H., and Pennington, J. Dynamical isometry and a mean field theory of lstms and grus. *arXiv preprint arXiv:1901.08987*, 2019.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. June 2018. URL `http://arxiv.org/abs/1806.07572`.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

Louart, C., Liao, Z., Couillet, R., et al. A random matrix approach to neural networks. *The Annals of Applied Probability*, 28(2):1190–1248, 2018.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Mei, S. and Montanari, A. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355*, 2019.

Pennington, J., Schoenholz, S., and Ganguli, S. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pp. 4785–4795, 2017.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? *arXiv preprint arXiv:1911.13299*, 2019.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. ISSN 00280836. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S. S., and Pennington, J. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018.

Yang, G. and Schoenholz, S. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pp. 7103–7114, 2017.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

Yehudai, G. and Shamir, O. On the power and limitations of random features for understanding neural networks. *arXiv preprint arXiv:1904.00687*, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhang, H., Dauphin, Y. N., and Ma, T. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.