Neural Characteristic Activation Value Analysis for Improved ReLU Network Feature Learning

Wenlin Chen WC337@CAM.AC.UK
University of Cambridge, Cambridge, United Kingdom
Max Planck Institute for Intelligent Systems, Tübingen, Germany

Hong Ge HG344@CAM.AC.UK
University of Cambridge, Cambridge, United Kingdom

Abstract

We examine the characteristic activation values of individual ReLU units in neural networks. We refer to the corresponding set for such characteristic activation values in the input space as the characteristic activation set of a ReLU unit. We draw an explicit connection between the characteristic activation set and learned features in ReLU networks. This connection leads to new insights into why various neural network normalization techniques used in modern deep learning architectures regularize and stabilize SGD optimization. Utilizing these insights, we propose a geometric approach to parameterize ReLU networks for improved feature learning. We empirically verify its usefulness with less carefully chosen initialization schemes and larger learning rates. We report improved optimization stability, faster convergence speed, and better generalization performance.

1 Introduction

In a standard neural network, each neuron applies an affine transformation to its input $\mathbf{x} \in \mathbb{R}^n$ followed by an element-wise nonlinear activation function g:

$$z = g(\mathbf{w}^{\mathrm{T}} \mathbf{x} + b), \tag{1}$$

where the affine transformation is parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^n$ and a bias scalar $b \in \mathbb{R}$. Rectified Linear Unit (ReLU) (Glorot et al., 2011) is arguably the most popular activation function used in modern deep learning architectures, which has a cut-off point at $s = 0$:

$$g(s) = \mathrm{ReLU}(s) = \max(0, s). \tag{2}$$

The characteristic activation boundary/set of such a ReLU neuron refers to the set of input locations with zero pre-activations, which, by definition, separates the active region from the inactive region in the input space. Characteristic activation boundaries are the building blocks for the decision boundaries of ReLU networks, which characterize the quality of the learned features.

This paper focuses on a geometric interpretation for learned features in ReLU networks based on characteristic activation analysis. This provides a theoretical justification for why various neural network normalization techniques used in modern deep learning architectures effectively regularize and stabilize SGD optimization. Motivated by these insights, we also propose a novel neural network parameterization technique that smooths the evolution of the characteristic activation boundaries in ReLU networks. We empirically show that our

new parameterization enables faster and more stable SGD optimization and achieves better generalization performance even under less carefully chosen initialization schemes and larger learning rates.

2 Characteristic Activation Value Analysis for ReLU Networks

This section formally defines the characteristic activation sets of individual neurons and introduces a geometric connection between the characteristic activation sets and learned features in ReLU networks. This geometric insight will help understand the stability of neural network optimization and motivate a new neural network parameterization that is provably stable under SGD.

2.1 Characteristic Activation Set for ReLU Unit

By definition, the ReLU activation function (2) is active for positive arguments $s > 0$ and inactive for negative arguments $s < 0$. For a neuron with ReLU activation, the *characteristic activation set* $\mathcal{B}$ is defined by a set of input locations such that $s = 0$:

$$\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0\}. \tag{3}$$

In other words, it forms a *characteristic boundary* for each neuron. This boundary separates the active and inactive regions of a ReLU unit in the input space. We define a representative point $\boldsymbol{\phi}$ that lies on the characteristic boundary $\mathcal{B}$ as

$$\boldsymbol{\phi} = -\frac{b \cdot \mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{w}} = -\frac{b}{\|\mathbf{w}\|_2} \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|_2}. \tag{4}$$

We refer to the point $\boldsymbol{\phi}$ as the *spatial location* of $\mathcal{B}$ and the vector that goes from the origin to the point $\boldsymbol{\phi}$ as the *characteristic vector*[1] of $\mathcal{B}$. The spatial location (or the characteristic vector) $\boldsymbol{\phi}$ uniquely determines the characteristic boundary/set.

2.2 ReLU Characteristic Activation Boundary in Hyperspherical Coordinate

In a high dimensional input space, most data points $\mathbf{x}$ live in a thin shell. This is because the volume of a high dimensional space concentrates near its surface (Blum et al., 2020). Intuitively, we want the spatial locations $\boldsymbol{\phi}$ of characteristic activation boundaries $\mathcal{B}$ to be close to the thin shell where most data points lie, because this spatial affinity between the characteristic activation set and data points will introduce non-linearity at suitable locations in the input space to separate different inputs $\mathbf{x}$ by assigning them different activation values z. This motivates the use of the hyperspherical coordinate to represent the spatial locations of the characteristic activation boundaries.

More concretely, we reparameterize the characteristic activation boundary in terms of its characteristic radius $\lambda \in \mathbb{R}$ and angles $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_{n-1}]^{\mathrm{T}}$ in the hyperspherical coordinate system. Noticing that $\mathbf{w}/\|\mathbf{w}\|_2$ is a unit vector, the radial-angular decomposition of the characteristic vector is given by

$$\boldsymbol{\phi}(\lambda, \boldsymbol{\theta}) = -\lambda \cdot \mathbf{u}(\boldsymbol{\theta}), \quad \text{with the definition } \lambda \coloneqq \frac{b}{\|\mathbf{w}\|_2} \text{ and } \mathbf{u}(\boldsymbol{\theta}) \coloneqq \frac{\mathbf{w}}{\|\mathbf{w}\|_2}, \tag{5}$$

1. We will slightly abuse the notation and unambiguously denote the characteristic vector by $\boldsymbol{\phi}$ as well.

where the direction of the unit vector $\mathbf{u}(\boldsymbol{\theta})$ is determined by the characteristic angle $\boldsymbol{\theta}$:

$$\mathbf{u}(\boldsymbol{\theta}) = \begin{pmatrix} \cos(\theta_1) \\ \sin(\theta_1)\cos(\theta_2) \\ \sin(\theta_1)\sin(\theta_2)\cos(\theta_3) \\ \vdots \\ \sin(\theta_1)\sin(\theta_2)\sin(\theta_3)\cdots\cos(\theta_{n-1}) \\ \sin(\theta_1)\sin(\theta_2)\sin(\theta_3)\cdots\sin(\theta_{n-1}) \end{pmatrix}. \tag{6}$$

In the hyperspherical coordinate system, the characteristic activation boundary can be expressed as

$$\mathcal{B}(\lambda, \boldsymbol{\theta}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}(\boldsymbol{\theta})^{\mathrm{T}}\,\mathbf{x} + \lambda = 0\}. \tag{7}$$

2.3 Geometric Interpretations of ReLU Characteristic Activation Set

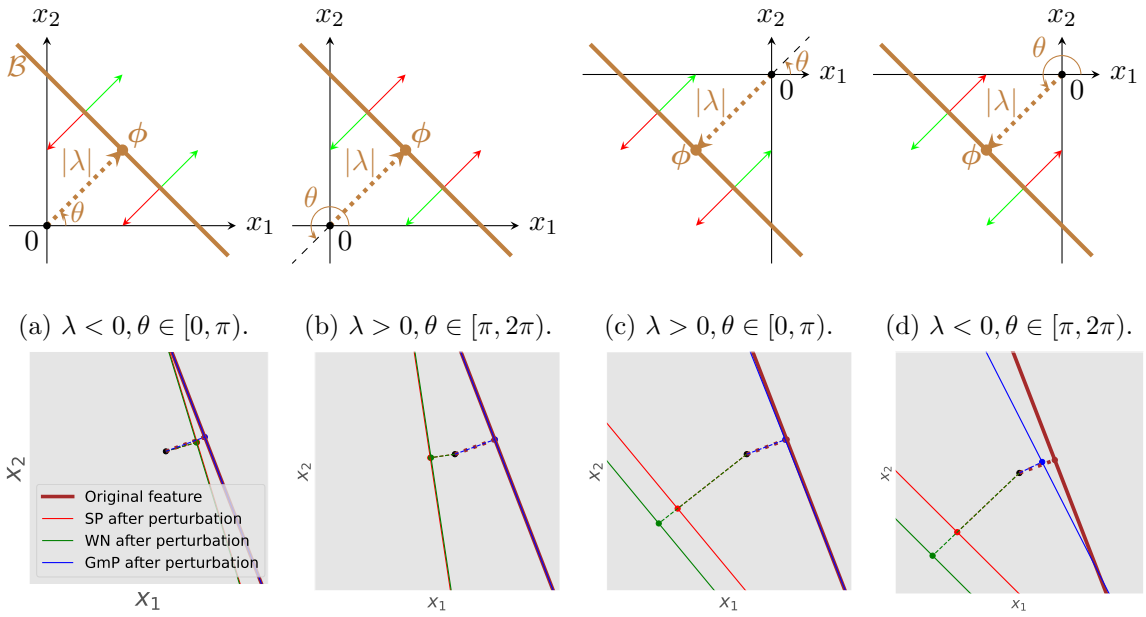
The characteristic activation set of a ReLU Unit forms a line in $\mathbb{R}^2$, as shown by the brown solid line in Figures 1a-1d. More generally, the characteristic activation set $\mathcal{B}$ forms an $(n-1)$-dimensional hyperplane in $\mathbb{R}^n$. The spatial location/characteristic vector $\boldsymbol{\phi} = -\lambda \cdot \mathbf{u}(\boldsymbol{\theta})$ fully specifies the characteristic activation boundary $\mathcal{B}$: it is perpendicular to $\mathcal{B}$, and its endpoint lies on $\mathcal{B}$. The angle $\boldsymbol{\theta}$ controls the direction of the characteristic activation boundary. The radius λ controls the distance between the origin and the characteristic activation boundary. Geometrically speaking, calculating the pre-activation of a ReLU unit for an input $\mathbf{x}$ is equivalent to projecting $\mathbf{x}$ over the unit vector $\mathbf{u}(\boldsymbol{\theta})$ and then adding the radius λ to the signed norm of the projected vector. From this perspective, it is clear the characteristic activation boundary is a set of inputs whose projections over $\mathbf{u}(\boldsymbol{\theta})$ have signed norm $-\lambda$.

2.4 Perturbation Analysis of ReLU Characteristic Activation Boundary

One benefit of defining characteristic activation boundaries in the hyperspherical coordinate system is that the radius λ and angle $\boldsymbol{\theta}$ of the spatial location $\boldsymbol{\phi}$ are disentangled. More concretely, this means that small perturbations to the parameter λ and $\boldsymbol{\theta}$ will only cause small changes in the spatial location of the characteristic activation boundary. To illustrate this, consider a small perturbation $\boldsymbol{\varepsilon}$ (e.g., gradient noise during SGD) to the weight $\mathbf{w}$ in the standard neural network parameterization. This perturbation results in a change in the angular direction of the characteristic activation boundary by

$$\langle \mathbf{w}, \mathbf{w} + \boldsymbol{\varepsilon} \rangle = \arccos\left(\frac{\mathbf{w}^{\mathrm{T}}(\mathbf{w} + \boldsymbol{\varepsilon})}{\|\mathbf{w}\|_2 \cdot \|\mathbf{w} + \boldsymbol{\varepsilon}\|_2}\right), \tag{8}$$

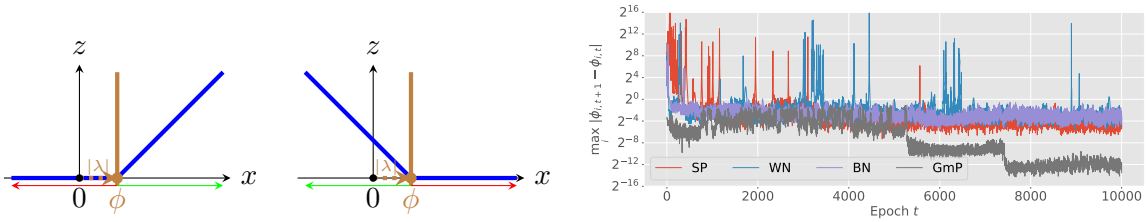
which can take arbitrary values in $[0, \pi]$ even for a small perturbation $\boldsymbol{\varepsilon}$. For example, we could have $\langle \mathbf{w}, \mathbf{w} + \boldsymbol{\varepsilon} \rangle = \pi$ if $\boldsymbol{\varepsilon}$ satisfies the condition $\boldsymbol{\varepsilon} = -(1 + {}^{\|\mathbf{w}+\boldsymbol{\varepsilon}\|_2}/_{\|\mathbf{w}\|_2})\,\mathbf{w}$. This indicates that the characteristic activation boundary is unstable in the sense that it is vulnerable to small perturbations if the weight $\mathbf{w}$ has a small norm. This has the implication that even a small gradient noise could destabilize the evolution of characteristic boundaries during SGD. Such instability is a critical reason that prevents practitioners from using larger



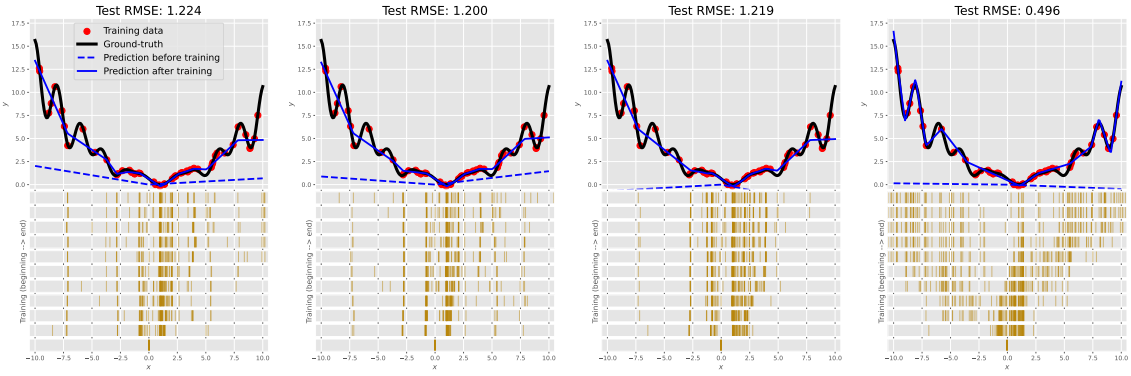
(a) $\lambda < 0, \theta \in [0, \pi)$. (b) $\lambda > 0, \theta \in [\pi, 2\pi)$. (c) $\lambda > 0, \theta \in [0, \pi)$. (d) $\lambda < 0, \theta \in [\pi, 2\pi)$.

(e) $\varepsilon = 0.0001$. (f) $\varepsilon = 0.001$. (g) $\varepsilon = 0.01$. (h) $\varepsilon = 0.1$.

Figure 1: (a)-(d) Characteristic activation boundary $\mathcal{B}$ (brown solid line) and spatial location $\boldsymbol{\phi} = -\lambda \cdot \mathbf{u}(\theta)$ of a ReLU unit $z = \mathrm{ReLU}(\mathbf{u}(\theta)^{\mathrm{T}} \mathbf{x} + \lambda) = \mathrm{ReLU}(\cos(\theta)x_1 + \sin(\theta)x_2 + \lambda)$ for inputs $\mathbf{x} \in \mathbb{R}^2$. The characteristic activation set forms a line in $\mathbb{R}^2$, which acts as a boundary separating inputs into two regions. Green arrows denote the active region, and red arrows denote the inactive region. (e)-(h) Stability of the characteristic activation boundary (set) of a ReLU unit in $\mathbb{R}^2$ under small perturbations $\boldsymbol{\varepsilon} = \varepsilon \cdot \mathbf{1}$ to the parameters of the ReLU unit. Solid lines denote characteristic activation boundaries $\mathcal{B}$, and colored dotted lines connect the origin and spatial locations $\boldsymbol{\phi}$ of $\mathcal{B}$. SP refers to standard parameterization, WN refers to weight normalization, and GmP refers to geometric parameterization. Smaller changes between the perturbed and original boundaries imply higher stability. Geometric parameterization is most stable against perturbations to its parameters.



(a) $\lambda < 0, \theta = 0$. (b) $\lambda > 0, \theta = \pi$. (c) Change in $\mathcal{B}$ and ϕ at each epoch (lr=0.1).



(d) SP (lr=0.01). (e) WN (lr=0.01). (f) BN (lr=0.01). (g) GmP (lr=0.1).

Figure 2: (a)-(b) Characteristic activation point $\mathcal{B}$ (intersection of brown solid lines and the x-axis) and spatial location $\phi = -\lambda u(\theta)$ of a single ReLU unit $z = \text{ReLU}(u(\theta)x + \lambda)$ (blue solid lines) for inputs $x \in \mathbb{R}$. Green arrows denote active regions, and red arrows denote inactive regions. (c) Evolution dynamics of the characteristic points in a one-hidden-layer network with 100 ReLU units for a 1D Levy regression problem under 4 different parameterizations during training. SP refers to standard parameterization, WN refers to weight normalization, BN refers to batch normalization, GmP refers to geometric parameterization. Smaller values are better as they indicate higher stability of the characteristic point during training. (d)-(g): The top row illustrates the experimental setup, including the network's predictions at initialization and after training, as well as the training data and the ground-truth function (Levy). A single-hidden-layer network with 100 ReLU units is used. Bottom row: the evolution of the characteristic activation point for the 100 ReLU units during training. Each horizontal bar shows the spatial location spectrum for a chosen optimization step, moving from the bottom (at initialization) to the top (after training). More spread of the spatial locations adds more useful non-linearity to the model, making prediction more accurate. Regression accuracy is measured by root mean squared error (RMSE) on a separate test set. Smaller RMSE values are better. We use a smaller learning rate for SP, WN, and BN because their training became unstable with higher learning rates as shown in (c).

learning rates. In contrast, the hyperspherical coordinate parameterization, which we will refer to as geometric parameterization (GmP), are much more stable under perturbation: the change in direction $\langle \mathbf{u}(\boldsymbol{\theta}), \mathbf{u}(\boldsymbol{\theta}+\boldsymbol{\varepsilon})\rangle$ of $\mathcal{B}$ under perturbation $\boldsymbol{\varepsilon}$ smoothly increases as the magnitude of $\boldsymbol{\varepsilon}$ becomes larger. For instance, consider a 2D input $\mathbf{x}$ and a characteristic activation plane with angle $\theta \in [0, 2\pi)$. Applying a small perturbation ε to θ results in a change in direction which is exactly ε:

$$\langle \mathbf{u}(\theta), \mathbf{u}(\theta+\varepsilon)\rangle = \arccos\left(\cos(\theta)\cos(\theta+\varepsilon) + \sin(\theta)\sin(\theta+\varepsilon)\right) = \varepsilon. \tag{9}$$

In Figures 1e-1h, we simulate the evolution behavior of characteristic boundaries in $\mathbb{R}^2$ for three different neural network parameterizations: the standard parameterization, weight normalization (Salimans and Kingma, 2016), and geometric parameterization. We apply small perturbations $\boldsymbol{\varepsilon}$ of different scales ε to the network parameters under different parameterizations and show how it affects the spatial location of the characteristic activation plane. We can see that the characteristic activation plane changes smoothly under the geometric parameterization: it gradually moves away from its original spatial location as we increase ε. In sharp contrast, even a small perturbation ε of magnitude 10^{-3} can drastically change the spatial locations of the characteristic activation planes under other parameterizations.

The improved stability of the characteristic activation plane under geometric parameterization is beneficial for speeding up the optimization of neural networks by acting as a pre-conditioner. This is illustrated in Figure 2, where we train a one-hidden-layer network with 100 ReLU units under various parameterizations on the 1D Levy regression dataset. As shown in Figures 2a-2b, $\mathcal{B}$ and ϕ reduce to the same point in $\mathbb{R}$, which we will refer to as the characteristic activation point. The angle θ of the characteristic activation point can only take two values 0 or π, which corresponds to the two directions on the real line. Clearly, the geometric parameterization significantly improves the stability of the characteristic activation point. Figure 2c shows that under the geometric parameterization the maximum change $\max_i |\Delta\phi_{i,t}| = \max_i |\phi_{i,t+1} - \phi_{i,t}|$ at each epoch t is always smaller than 1 throughout training, while under other parameterizations the changes can be up to 2^{16} at some epochs. The stable evolution of the characteristic point under geometric parameterization leads to improved generalization performance on this regression task, as shown in Figures 2d-2g.

3 Geometric Parameterization for ReLU Networks

Motivated by the improved stability of the characteristic activation boundary in the hyperspherical coordinate system, we develop a new parameterization technique for ReLU networks. We will refer to this new neural network parameterization as the *geometric parameterization* (GmP).

3.1 Geometric Parameterization for ReLU Unit

Starting from reparameterizing a single ReLU unit, we replace the weight vector $\mathbf{w} \in \mathbb{R}^n$ and the bias scalar $b \in \mathbb{R}$ in a standard ReLU unit (1) using the radial parameter $\lambda \in \mathbb{R}$ and the angular vector $\boldsymbol{\theta}$ as defined in Equations (5) and (6). We denote the scale of the activation by r and move it to the outside of the ReLU activation function. These changes

lead to the geometric parameterization:

$$z = r \cdot \mathrm{ReLU}(\mathbf{u}(\boldsymbol{\theta})^{\mathrm{T}} \mathbf{x} + \lambda). \tag{10}$$

The geometric parameterization has three learnable parameters: the scaling parameter r, the radial parameter λ, and angular parameter $\boldsymbol{\theta}$. As discussed in Section 2.3, the radial and angular parameters λ and $\boldsymbol{\theta}$ specify the spatial location $\boldsymbol{\phi}$ of the characteristic activation boundary. The scaling parameter r specifies the scale of the activation. As we have seen in Section 2, this geometric parameterization results in several nice properties to feature learning: optimizing these geometric parameters during training directly translates into a smooth evolution of 1) the spatial location of the characteristic activation boundary and 2) the scale of the activation.

We apply the geometric parameterization to all layers except for the output layer. The output layer is a linear layer with an inverse link function (e.g., softmax or identity) for producing the network output. Since the inverse link function involves no feature learning, it does not need to be reparameterized. For multiple-hidden-layer networks, the inputs to immediate layers are outputs from previous layers, potentially suffering from a covariate shift phenomenon (Salimans and Kingma, 2016). In the next section, we propose a simple fix by normalizing the input means to ReLU units under geometric parameterization.

3.2 Input Mean Normalization for Intermediate Layers

One implicit assumption of the characteristic activation set analysis is that the input distribution to a neuron is frozen. This assumption holds for one-hidden-layer networks automatically since the training data is constant. However, this assumption is not necessarily satisfied for the inputs to the intermediate layers in a multiple-hidden-layer network. This is because the inputs to an intermediate layer are transformed by the weights and squashed by the activation function in the previous layer. This could cause optimization difficulties even under the geometric parameterization. For ReLU units in the intermediate layers of a multiple-hidden-layer, we propose a simple fix called mean normalization (MN), which subtracts the input by their empirical mean,

$$z = r \cdot \mathrm{ReLU}(\mathbf{u}(\boldsymbol{\theta})^{\mathrm{T}} (\mathbf{x} - \hat{\mathbb{E}}[\mathbf{x}]) + \lambda). \tag{11}$$

This is a standard parameter-free data pre-processing technique which ensures that the inputs are always centered around the origin, similar to the mean-only batch normalization (MBN) (Salimans and Kingma, 2016) but applied to post-activations instead of pre-activations.

3.3 Layer-Size Independent Parameter Initialization

While existing neural network parameterizations are sensitive to the initialization, geometric parameterization can work with less carefully chosen initialization schemes that are independent of the size of the layer, thanks to an invariant property of the hyperspherical coordinate system. To see this, first we consider the distribution of the angular direction of the characteristic activation boundary under standard parameterization. Under popular initialization methods such as the Glorot initialization (Glorot and Bengio, 2010) and He initialization (He et al., 2015), each element in the initial weight vector $\mathbf{w}$ in the standard

parameterization is independently and identically sampled from a zero mean Gaussian distribution. This always induces a uniform distribution over the unit n-sphere for the direction $\mathbf{u}(\boldsymbol{\theta})$ of the characteristic activation boundary, no matter what variance value is used in that Gaussian distribution. In practice, this means we can initialize the angular parameter $\boldsymbol{\theta}$ uniformly at random. The parameter λ is initialized to zero due to its connection to the standard parameterization $\lambda = {}^{b}/_{\|w\|_2}$ and the common practice to set $b = 0$ at initialization. The scaling parameter r is initialized to one based on the intuition that the scale r roughly corresponds to the total variance of the weight parameters $\mathbf{w}$ in the standard parameterization. Overall, none of the parameters λ, $\boldsymbol{\theta}$, and r in the geometric parameterization require layer-size dependent initialization. Furthermore, the scaling parameter r can be initialized more flexibly without affecting the convergence of the optimization step. Such robustness against a wider range of initialization schemes adds further evidence to the superiority of geometric parameterization.

4 Related work

This section reviews two popular normalization methods, weight normalization (Salimans and Kingma, 2016) and batch normalization (Ioffe and Szegedy, 2015), and discusses how the proposed geometric parameterization relates to them.

4.1 Weight Normalization

Weight normalization (Salimans and Kingma, 2016) is a simple weight reparameterization technique that decouples the length l and the direction ${}^{\mathbf{v}}/_{\|\mathbf{v}\|_2}$ of the weight vector $\mathbf{w}$ in a standard ReLU unit (1):

$$z = \mathrm{ReLU}\left(l \cdot \left(\frac{\mathbf{v}}{\|\mathbf{v}\|_2}\right)^{\mathrm{T}} \mathbf{x} + b\right). \tag{12}$$

The idea behind weight normalization is to make the length l and the direction ${}^{\mathbf{v}}/_{\|\mathbf{v}\|_2}$ of the weight vector $\mathbf{w}$ independent of each other. This has been shown to be effective in improving the conditioning of the gradients of the parameters and speeding up the convergence of optimization.

It might be tempting to think that our proposed geometric parameterization is identical to weight normalization. In fact, it indeed inherits the advantages of weight normalization because the length-directional decomposition in weight normalization is automatically inherent in the geometric parameterization. However, the geometric parameterization possesses an extra nice property that weight normalization lacks: it makes the evolution of the characteristic activation boundaries robust against small perturbations the the parameters (e.g., SGD noise), as shown in Section 2.4. In contrast, the characteristic activation boundary under weight normalization can be unstable, since its change in direction $\left\langle l \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|_2}, (l+\varepsilon') \cdot \frac{\mathbf{v}+\boldsymbol{\varepsilon}}{\|\mathbf{v}+\boldsymbol{\varepsilon}\|_2} \right\rangle = \arccos(\frac{\mathbf{v}^{\mathrm{T}}(\mathbf{v}+\boldsymbol{\varepsilon})}{\|\mathbf{v}\|_2 \cdot \|\mathbf{v}+\boldsymbol{\varepsilon}\|_2})$ under perturbations $\boldsymbol{\varepsilon}$ and ε' has exactly the same form as that for standard parameterization, as shown in Equation 8.

4.2 Batch Normalization

Batch normalization (Ioffe and Szegedy, 2015) is a widely-used neural network normalization layer in modern deep learning architectures such as ResNet (He et al., 2016), which has been shown to be effective to accelerate and stabilize stochastic gradient optimization of neural networks (Kohler et al., 2019). In ReLU networks, batch normalization (BN) is often applied at the pre-activation level in each layer:

$$z = \mathrm{ReLU}(\mathrm{BN}(\mathbf{w}^{\mathrm{T}}\,\mathbf{x} + b)). \tag{13}$$

The BN layer standardizes the pre-activation using the empirical mean and covariance estimated from the current mini-batch:

$$\mathrm{BN}(\mathbf{w}^{\mathrm{T}}\,\mathbf{x} + b) = \gamma \cdot \frac{\mathbf{w}^{\mathrm{T}}\,\mathbf{x} - \hat{\mathbb{E}}_{\mathbf{x}}[\mathbf{w}^{\mathrm{T}}\,\mathbf{x}]}{\sqrt{\hat{\mathrm{Var}}_{\mathbf{x}}[\mathbf{w}^{\mathrm{T}}\,\mathbf{x} + b]}} + \beta, \tag{14}$$

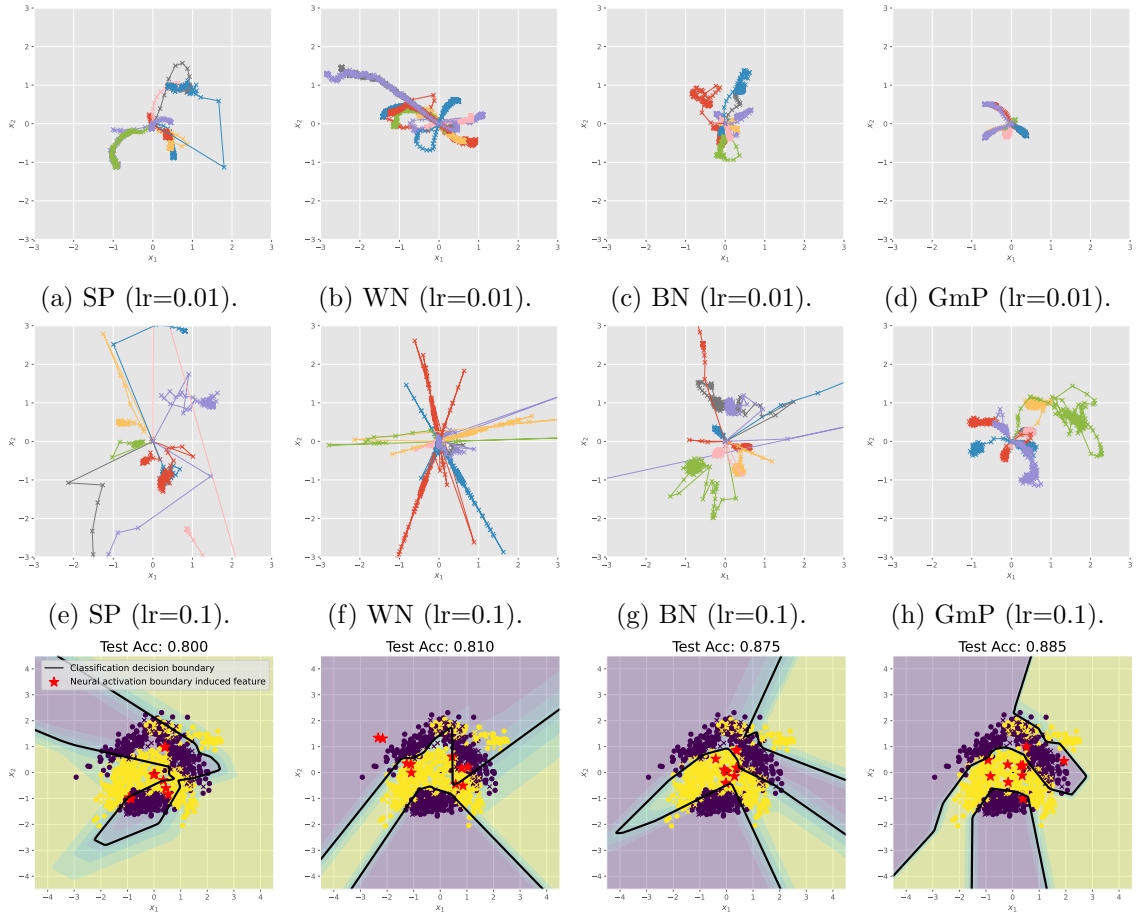
where $\gamma \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are two free parameters to be learned from data, which adjusts the output of the BN layer as needed to increase its expressiveness. Unlike batch normalization, the geometric parameterization and weight normalization do not insert extra layers or add extra parameters into the existing neural network architecture. Despite being simpler, they are, in fact, closely related to one another: assuming that the input $\mathbf{x}$ has zero means, one can show that batch normalization is also a kind of neural network parameterization technique:

$$\mathrm{BN}(\mathbf{w}^{\mathrm{T}}\,\mathbf{x} + b) = \gamma \cdot \frac{\mathbf{w}^{\mathrm{T}}\,\mathbf{x}}{\sqrt{\hat{\mathrm{Var}}_{\mathbf{x}}[\mathbf{w}^{\mathrm{T}}\,\mathbf{x} + b]}} + \beta = \gamma \cdot \frac{\mathbf{w}^{\mathrm{T}}\,\mathbf{x}}{\sqrt{\mathbf{w}^{\mathrm{T}}\hat{\mathbf{\Sigma}}\,\mathbf{w}}} + \beta = \gamma \left(\frac{\mathbf{w}}{\|\mathbf{w}\|_{\hat{\mathbf{\Sigma}}}}\right)^{\mathrm{T}} \mathbf{x} + \beta, \tag{15}$$

where the vector norm $\|\mathbf{w}\|_{\hat{\mathbf{\Sigma}}}$ is calculated with respect to the empirical data covariance matrix $\hat{\mathbf{\Sigma}} = \hat{\mathrm{Var}}[\mathbf{x}]$ estimated from the current mini-batch. This shows that batch normalization is effectively an adaptive, data-dependent parameterization of standard neurons (1) that decouples the correlations in the input $\mathbf{x}$. However, in practice, it is common to estimate only the diagonal elements in the data covariance matrix $\hat{\mathbf{\Sigma}}$ and set all its off-diagonal elements to zero to reduce the amount of extra computation introduced by the BN layers. Under this formalism, both geometric parameterization and weight normalization can be seen as special cases of batch normalization where the covariance matrix is replaced by the identity matrix $\hat{\mathbf{\Sigma}} = \mathbf{I}$ independent of the input $\mathbf{x}$ (since $\|\cdot\|_2 = \|\cdot\|_{\mathbf{I}}$). Moreover, our geometric parameterization operates in the hyperspherical coordinate system to further stabilize feature learning.

5 Empirical Evaluations

We empirically validate our claims about the geometric interpretation of feature learning and evaluate the performance of our proposed GmP parameterization. We compare GmP with standard parameterization (SP), batch normalization (BN) and weight normalization (WN). For neural networks with more than one hidden layer, we also consider the variant of WN that uses mean-only batch normalization (MBN) and the variant of GmP that uses mean normalization (MN). For GmP, we report improved optimization stability, faster convergence speed and better generalization performance.



(a) SP (lr=0.01). (b) WN (lr=0.01). (c) BN (lr=0.01). (d) GmP (lr=0.01).

(e) SP (lr=0.1). (f) WN (lr=0.1). (g) BN (lr=0.1). (h) GmP (lr=0.1).

(i) SP (lr=0.01). (j) WN (lr=0.01). (k) BN (lr=0.01). (l) GmP (lr=0.1).

Figure 3: Performance of a single-hidden-layer neural network with 10 ReLU units on the Banana dataset under four different parameterizations. SP refers to standard parameterization; WN refers to weight normalization; BN refers to batch normalization; GmP refers to geometric parameterization (this work). First two rows: trajectories of the spatial locations of the 10 ReLU units during training. Each color depicts one ReLU unit. Smoother evolution means higher training stability. The evolution under the geometric parameterization is stable so that we can use a 10× larger learning rate. Last row: Network predictions after training. Black bold lines depict the classification boundary between two classes. Classification accuracy is measured on a test set. Higher values are better. The red stars show the spatial locations of 10 ReLU units. Intuitively speaking, more evenly spread out red stars are better for classification accuracy, as they provide more useful non-linearity.

Table 1: Mean test RMSE with standard error for a one-hidden-layer MLP trained on 7 UCI regression benchmarks.

Benchmark	Boston	Concrete	Energy	Naval	Power	Wine	Yacht
SP	3.370 ± 0.145	5.472 ± 0.144	0.898 ± 0.274	0.002 ± 0.000	4.065 ± 0.029	0.623 ± 0.008	0.639 ± 0.063
WN	3.459 ± 0.156	5.952 ± 0.148	2.093 ± 0.789	0.003 ± 0.000	4.073 ± 0.026	0.632 ± 0.008	0.624 ± 0.076
BN	3.469 ± 0.153	5.695 ± 0.160	1.648 ± 0.302	$\mathbf{0.001 \pm 0.000}$	4.164 ± 0.026	0.622 ± 0.011	0.777 ± 0.055
GmP	$\mathbf{3.057 \pm 0.144}$	$\mathbf{5.153 \pm 0.098}$	$\mathbf{0.474 \pm 0.013}$	0.003 ± 0.001	$\mathbf{4.022 \pm 0.025}$	$\mathbf{0.613 \pm 0.006}$	$\mathbf{0.584 \pm 0.046}$

Table 2: Mean top-1 and top-5 validation accuracy (%) with standard error for VGG-6 trained on ImageNet32 with varying batch sizes.

Metric	Top-1 validation accuracy			Top-5 validation accuracy		
Batch size	256	512	1024	256	512	1024
SP	38.31 ± 0.13	36.99 ± 0.11	35.02 ± 0.03	62.48 ± 0.14	60.71 ± 0.18	58.14 ± 0.39
WN	39.13 ± 0.10	37.92 ± 0.12	36.17 ± 0.03	63.28 ± 0.02	61.93 ± 0.09	60.16 ± 0.18
WN+MBN	42.22 ± 0.01	40.96 ± 0.02	39.33 ± 0.07	66.04 ± 0.07	65.08 ± 0.03	63.32 ± 0.08
BN	42.79 ± 0.03	41.90 ± 0.19	41.39 ± 0.02	67.17 ± 0.08	66.50 ± 0.25	65.89 ± 0.06
GmP	40.76 ± 0.09	41.65 ± 0.09	41.29 ± 0.08	65.08 ± 0.08	65.76 ± 0.05	65.49 ± 0.06
GmP+MN	$\mathbf{43.14 \pm 0.05}$	$\mathbf{43.62 \pm 0.08}$	$\mathbf{42.70 \pm 0.15}$	$\mathbf{67.36 \pm 0.05}$	$\mathbf{67.76 \pm 0.09}$	$\mathbf{66.98 \pm 0.18}$

2D Classification. Besides the 1D Levy function regression problem shown in Figures 2c-2g in Section 2.4, we evaluate GmP on another synthetic problem: the classic 2D banana-shaped classification dataset. We train a single-hidden-layer MLP with 10 hidden units using Adam (Kingma and Ba, 2014). We find that GmP allows us to use a $10\times$ larger learning rate while still maintaining a smooth evolution of the characteristic activation boundary. In Figure 3, we show that under GmP, the spatial locations of the characteristic activation boundaries move steadily towards different directions during training and spread over all training data points in different regions, which forms a classification decision boundary with a reasonable shape that achieves the highest test accuracy among all compared methods.

UCI Regression. We evaluate GmP on 7 standard regression benchmarks from the UCI machine learning repository (Dua and Graff, 2017). We train an MLP with one hidden layer and 100 hidden units for 10 different random 80/20 train/test splits. We use Adam (Kingma and Ba, 2014) with a learning rate of 0.1 for GmP and 0.01 for all the other methods. We report the test RMSE results in Table 1: GmP consistently achieves the best performance among all compared methods on all benchmarks except for the Naval benchmark. We note that for Naval, all methods achieve almost zero test RMSE, indicating that this is a relatively simple task where the relative performance under different methods is less indicative of their merits.

ImageNet32. We further evaluate GmP with a medium-sized neural network architecture VGG-6 (Simonyan and Zisserman, 2014) on ImageNet32 (Chrabaszcz et al., 2017), which contains all images from ImageNet (ILSVRC 2012) (Deng et al., 2009) resized to 32×32 and provides a better alternative to CIFAR-10/100. We use SGD with momentum 0.9. The learning rate is 0.1 for GmP and 0.01 for all the other methods. We reduce the learning

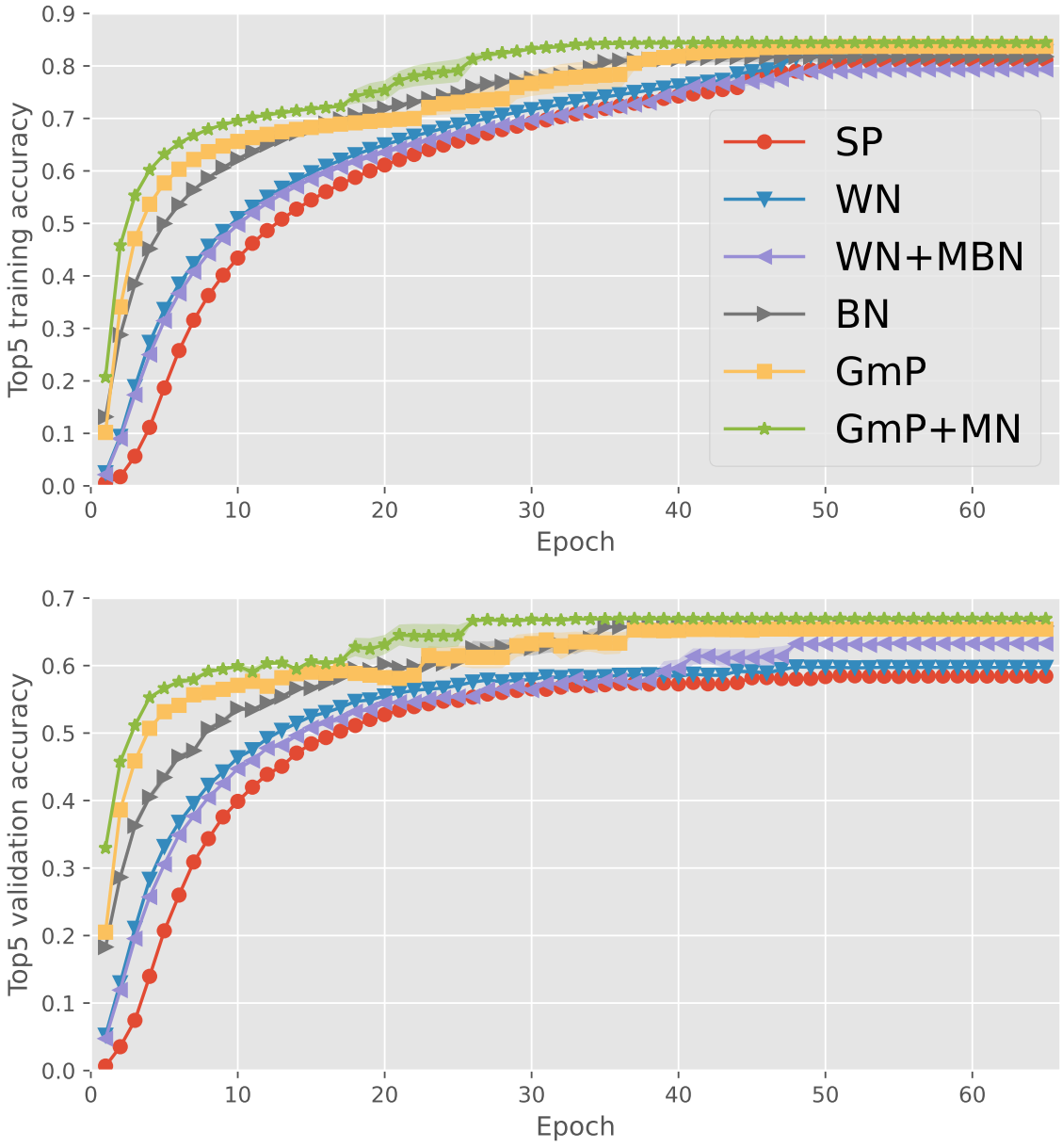


Figure 4: Mean top-5 training and validation accuracy with standard error as a function of epoch for VGG-6 network trained on the ImageNet32 dataset with a batch size of 1024. Top: top-5 training accuracy. Bottom: top-5 validation accuracy.

Table 3: Mean top-1 and top-5 validation accuracy (%) with standard error for ResNet-34 trained on ImageNet (ILSVRC 2012).

Metric	Top-1 valid. acc.	Top-5 valid. acc.
WN+MBN	66.57 ± 0.16	86.69 ± 0.11
BN	66.85 ± 0.05	86.92 ± 0.02
GmP+MN	$\mathbf{67.24 \pm 0.21}$	$\mathbf{87.19 \pm 0.15}$

rate when the top-5 validation accuracy does not improve for 5 epochs and stop training when it plateaus for 10 epochs. We use random horizontal flips for data augmentation, following Chrabaszcz et al. (2017). Table 2 shows that GmP+MN achieves the best top-1 and top-5 validation accuracy for all batch sizes considered. It is worth noting that mean normalization improves the performance of GmP in deep networks as expected. Furthermore, the improvement of our method over other methods becomes larger as the batch size increases. Interestingly, the top-1 validation accuracy for VGG-6 with GmP and batch size 512 (43.62%) is better than that for Wide Residual Networks (WRN-28-2, 43.08%) reported in Chrabaszcz et al. (2017), which is trained with batch size 500 and batch normalization. This indicates that even a small convolutional neural network (CNN) such as VGG-6 with our GmP parameterization can beat large, wide residual networks. Figure 4 shows that GmP also achieves the fastest convergence rate among all compared methods.

ImageNet (ILSVRC 2012). Finally, we evaluate GmP with a large neural network architecture ResNet-34 (He et al., 2016) on the full resolution ImageNet (ILSVRC 2012) dataset (Deng et al., 2009). We do not report the performance of standard parameterization, since ResNet includes batch normalization by default, which, with residual connections, greatly improves the performance of CNNs and enables the use of large learning rates. In our experiments, we use SGD with momentum 0.9, learning rate 0.1, and batch size 512 for all compared methods. We reduce the learning rate when the top-5 validation accuracy does not improve for 5 epochs and stop training when it plateaus for 10 epochs. We use random horizontal flip, random resizing ($L \times L, L \sim U[256, 480]$), and random crop ($224 \times 224$) for data augmentation. Due to a lack of computational resources, we do not preserve the aspect ratio of the images when performing random resizing and do not use color augmentation (Krizhevsky et al., 2017). We resize the validation images to 368×368 without preserving the aspect ratio and then calculate the validation metric based on a single center-crop of size 224×224. Table 3 reports the top-1 and top-5 validation accuracy for all compared methods (standard ResNet uses batch normalization by default), which shows that GmP+MN outperforms all other compared methods.

6 Conclusion

6.1 Summary and Discussion

We have proposed a novel method for understanding various normalization techniques and their roles in neural network feature learning. This method is called characteristic activation

value analysis since it exploits special activation values to characterize ReLU units. We have found the preimage of such characteristic activation values, referred to as the characteristic activation set, and used it to identify ReLU units uniquely. In order to advance the understanding of neural network normalization techniques, we have performed a perturbation analysis of the characteristic activation set and discovered the instabilities of existing approaches. Motivated by the newly gained insights, we have proposed a new parameterization using the hyperspherical coordinate system called geometric parameterization. We have demonstrated its advantages for single-hidden-layer ReLU networks and combined it with mean normalization to handle covariance shifts in multiple-hidden-layer ReLU networks. We have performed theoretical analysis and empirical evaluations to validate its usefulness in improving feature learning. We have shown that it improves training stability, convergence speed, and generalization performance in models of different sizes, including ResNet.

6.2 Limitations and Future work

This work focused on analyzing the ReLU activation function due to its wider adoption. However, the characteristic value analysis technique is quite general and can be extended to other activation functions. In addition, we only performed the characteristic activation analysis for single-hidden-layer ReLU networks and proposed a practical workaround for multiple-hidden-layer ReLU networks by using mean normalization. For future work, this analysis needs to be generalized to examine feature learning in multiple-hidden-layer networks in order to understand the theoretical behavior of deep networks as a whole. One potential difficulty is that the characteristic activation boundary of multiple ReLU units becomes a piecewise linear partition of the input space, which is less straightforward to analyze. A possible solution would be to consider how the assignments of data points to the partitions evolve during training, similar to how we track the characteristic activation sets.

Acknowledgements and Disclosure of Funding

WC acknowledges funding via a Cambridge Trust Scholarship (supported by the Cambridge Trust) and a Cambridge University Engineering Department Studentship (under grant G105682 NMZR/089 supported by Huawei R&D UK). HG acknowledges generous support from Huawei R&D UK.

Part of this work was performed using the SL2 GPU service provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (`https://www.csd3.cam.ac.uk`) provided by Dell EMC and Intel, purchased using grant NMZR/089 supported by Huawei R&D UK.

Broader Impact

This paper studies the theory that underpins deep learning, as such it takes a step towards improving the reliability and robustness of deep learning techniques. We believe that the ethical implications of this work are minimal: this research involves no human subjects, no sensitive data where privacy is a concern, no domains where discrimination/bias/fairness is concerning, and is unlikely to have a noticeable social impact. Optimistically, our hope is that this work can result in improved deep learning training algorithms. However, as with most research in machine learning, new modeling and inference techniques could be used by bad actors to cause harm more effectively, but we do not see how this work is more concerning than any other work in this regard.

References

Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020. doi: 10.1017/9781108755528.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Thomas Hofmann, Ming Zhou, and Klaus Neymeyr. Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 806–815. PMLR, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.