WHY RELU UNITS SOMETIMES DIE: ANALYSIS OF SINGLE-UNIT ERROR BACKPROPAGATION IN NEURAL NETWORKS

Scott C. Douglas and Jiutian Yu

Southern Methodist University
Department of Electrical and Computer Engineering
Dallas, Texas 75275 USA
sdouglas@smu.edu, jiutiany@smu.edu

ABSTRACT

Recently, neural networks in machine learning use rectified linear units (ReLUs) in early processing layers for better performance. Training these structures sometimes results in "dying ReLU units" with near-zero outputs. We first explore this condition via simulation using the CIFAR-10 dataset and variants of two popular convolutive neural network architectures. Our explorations show that the output activation probability $\Pr[y > 0]$ is generally less than 0.5 at system convergence for layers that do not employ skip connections, and this activation probability tends to decrease as one progresses from input layer to output layer. Employing a simplified model of a single ReLU unit trained by a variant of error backpropagation, we then perform a statistical convergence analysis to explore the model's evolutionary behavior. Our analysis describes the potentially-slower convergence speeds of dying ReLU units, and this issue can occur regardless of how the weights are initialized.

Index Terms— adaptive algorithms, adaptation models, algorithm design and analysis, backpropagation algorithms.

1. INTRODUCTION

Machine learning has seen a recent explosion in both research and development due to the availability of large data sets, useful adaptive algorithms and architectures, and fast computational processing for training. One popular class of architectures are multilayer neural networks that uses rectified linear units, or ReLUs, in the early stages of processing, in which a single-unit input-output relation within the network is

$$y = \mathbf{w}^T\mathbf{x} + b \quad (1)$$

$$f(y) = \max(0, y), \quad (2)$$

where $\mathbf{w} = [w_1 \ \cdots \ w_N]^T$ is the weight vector for the particular unit, $\mathbf{x} = [x_1 \ \cdots \ x_N]^T$ is the unit's input vector, b is the bias weight, and $f(y)$ is the ReLU activation function [1, 2, 3, 4]. The ReLU activation function is simple to compute. It also avoids difficulties with regard to output scaling.

Systems employing these units sometime exhibit an issue during training known as "dying ReLU units," in which the condition $y < 0$ is highly likely for almost all training patterns for many units within the network. Little is understood about this condition other than the simple observation that the error backpropagation algorithm employs a function $f'(y)$ that for the ReLU nonlinearity is zero for $y < 0$, such that patterns that cause $y < 0$ do not change the unit's weights. It is suggested in [5] that this condition could be problematic in practice, because units that are not active will likely not be trained, and thus "leaky ReLUs," that make $f'(y) > 0$ for all y are introduced. However, the results in [5] using such modified activation functions are not substantially-different from those using ReLUs. To our knowledge, the issue of "dying ReLUs" has not been extensively explored in the machine learning literature.

This paper explores the issue of "dying ReLUs" in two ways. First, we perform extensive simulations of multiple variants of the VGG and ResNet architectures on the CIFAR-10 image classification dataset to numerically evaluate the output activation probability $\Pr[y > 0]$ across each layer of each structure for both training and test data. Our results produce the following observations:

1. The average value of $\Pr[y > 0]$ generally decreases as one progresses from input layer to output layer for those layers that do not have direct skip connections.
2. The convergence speed of the value of $\Pr[y > 0]$ is generally slower for layers that exhibit small values of $\Pr[y > 0]$ near convergence. Thus, units that are "dying" appear to be slower to converge.

Second, to understand the convergence speed issue associated with varying $\Pr[y > 0]$ unit values, we analytically explore the convergence behavior of a simple single-unit $(L + 1)$-parameter (signal plus bias weights) error backpropagation model given by the update relations

$$\mathbf{w}_{new} = \mathbf{w} + \eta(d - f(y))f'(y)\mathbf{x} \quad (3)$$

$$b_{new} = b + \eta(d - f(y))f'(y) \quad (4)$$

where $\mathbf{w}_{new}$ and b_{new} are the updated weights, d is the target or training signal for the unit, and $u(y) = f'(y)$ is the unit step function, under a simple statistical model for d and $\mathbf{x}$. The approach taken follows that employed for understanding linear adaptive filters with non-quadratic error costs as well as nonlinear adaptive systems such as the single-layer error backpropagation algorithm with saturating nonlinearities [6, 7, 8]. Our statistical analysis shows that the combined mean weight vector $[E\{\mathbf{w}^T\}\ E\{b\}]^T$ converges approximately exponentially near its optimal solution with all but three of its modes of the form $(1 - \eta \Pr[y > 0])^k$. Moreover, these converging modes are likely to dominate the overall convergence rate of the adaptive parameters due to both their multiplicity and their slow rate when $\Pr[y > 0]$ is small. Thus, the issue of "dying ReLUs" is one of a potentially-slow convergence rate due to their sparse output activations, as opposed to the absolute sparsity of the output activations themselves. Moreover, unlike the concerns raised in [5], this issue appears not to be tied to how the units are initialized.

2. NUMERICAL EVALUATIONS

In order to better understand the underlying convergence issues, we first experiment using state-of-the-art architectures from the very deep convolutional networks (VGG) [9] and residual networks (ResNet) [10] families; specifically, VGG-9, VGG-16, ResNet-20, and ResNet-32. VGG-9 is a cropped version of VGG-16 containing 7 convolutional layers followed by two fully-connected layers. A batch normalization layer is added after each convolutional layer and the first fully-connected layer. We apply these systems to a popular image classification benchmark: CIFAR-10, which consists of (32×32)-pixel color images across ten classes, with 50000 images for training and 10000 images for testing [11]. For each architecture, we calculate the probability of non-zero outputs of the ReLU units at each layer at each epoch during training; similar values were also observed on the testing data. To improve classification performance, flipping and translation data augmentation has been used. We employ standard gradient descent for each model with the cross-entropy loss, with the following parameters: a mini-batch size of 128, weight decay of 0.0001, a momentum value of 0.9, and He normal weight initialization. As for learning rates, values were chosen to be initially large for the VGG and ResNet architectures and then reduced by a factor of 1/10 at epochs 150 and 100, respectively, and again at epochs 225 and 200, respectively, except for a short 10-epoch initial training period for ResNet whereby a small step size was used. Averages across ten different training runs have been computed in each case.

Figure 1 shows the evolutions of $\Pr[y > 0]$ for the various layers of VGG-9, where it is seen that the output activation probabilities tends to decrease during training for all of the layers. There is a tendency for layers that have lower $\Pr[y > 0]$ to have a somewhat slower convergence of the observed $\Pr[y > 0]$ values near convergence, which upon further inspection is particularly evident for Layers 5, 6, and 7 prior to the first step size change at epoch 150. All of the layers have a reduced (less than 0.5) output activation probability at convergence, indicating that output sparsity is a desirable trait from the standpoint of classification performance with this architecture.

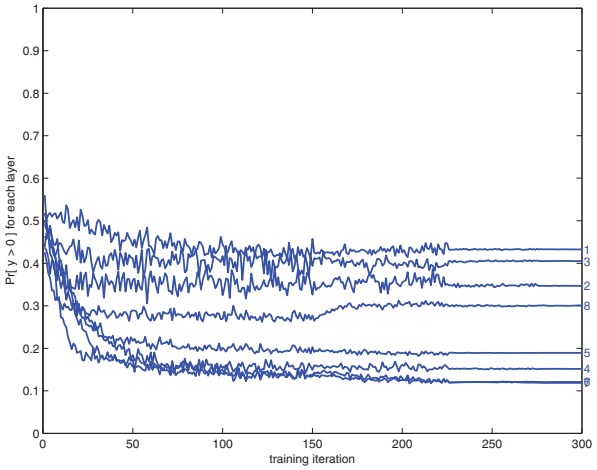

Fig. 1. Evolution of $\Pr[y > 0]$ values for the VGG-9 architecture on CIFAR-10.

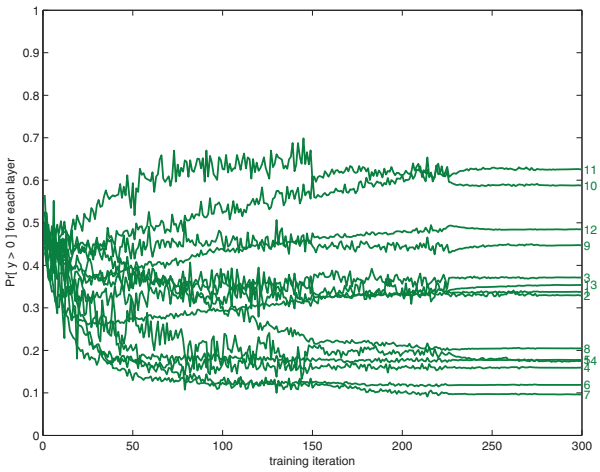

Fig. 2. Evolution of $\Pr[y > 0]$ values for the VGG-16 architecture on CIFAR-10.

Figure 2 shows the evolutions of the output activation probabilities for VGG-16, for which the trends are not as clear. Considering only the first training period up to epoch 150, the slowest-converging layer appears to be Layer 8 in the middle of the architecture, which also has a low $\Pr[y > 0] \approx 0.204$ value at convergence. Layers after this layer, however, also exhibit somewhat slower convergence to higher final $\Pr[y > 0]$ values after having smaller values at intermediate points in training.

Figures 3 and 4 show the evolutions of $\Pr[y > 0]$ val-

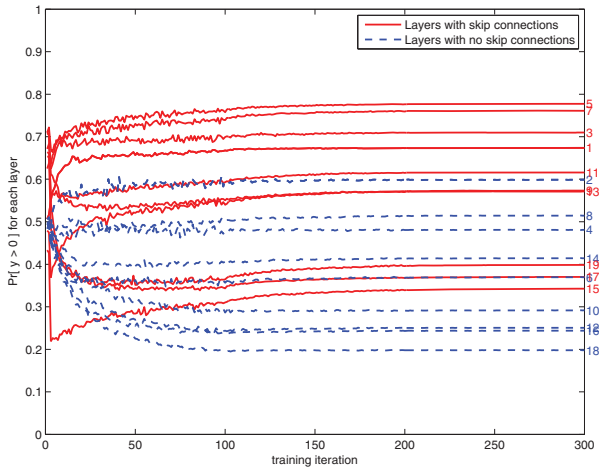

Fig. 3. Evolution of $\Pr[y > 0]$ values for the ResNet-20 architecture on CIFAR-10.

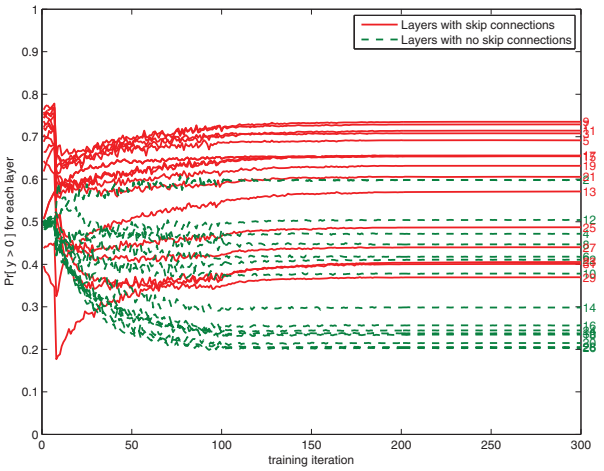

Fig. 4. Evolution of $\Pr[y > 0]$ values for the ResNet-32 architecture on CIFAR-10.

ues for the two ResNet architectures, respectively, where it is seen that layers with skip connections, shown in red, generally have higher $\Pr[y > 0]$ values throughout training. This is to be expected given the nature of skip connections, which are designed to increase the possibility of activation where they are used. The layers that have lower steady-state $\Pr[y > 0]$ values are those without skip connections, shown in blue and green, respectively, which also tend to converge more slowly over the fastest training period from epoch 10 to epoch 100. In this case, Layer 15 in the ResNet-20 architecture also exhibits slow convergence after experiencing a significant change to a low $\Pr[y > 0]$ value in the initial training period.

Figure 5 shows the average values of $\Pr[y > 0]$ obtained for each layer at convergence for the VGG (top) and ResNet (bottom) architectures in our evaluations. An interesting behavior is noted: The output activation probabilities decrease from input to output for VGG-9 and both ResNet architectures when considering the presence or absence of skip connections. This characteristic was not exhibited by VGG-16, which had its lowest $\Pr[y > 0]$ values near the middle of the architecture.

3. STATISTICAL ANALYSIS

In this section, we develop an analytical model to explore features of the convergence behavior observed in the previous section. The model chosen is quite simple and assumes that any particular unit within the network evolves according to a single-unit model of the form in (3)–(4), where the input signal vector follows a Gaussian distribution with mean vector $\boldsymbol{\mu}$ and an identity covariance matrix $\mathbf{I}$, or

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}). \tag{5}$$

Furthermore, we assume that the desired response follows a linear model of the form

$$d = \mathbf{a}^T\mathbf{x} + \bar{c}, \tag{6}$$

where the vector $\mathbf{a}$ and $\bar{c}$ contain unknown parameters.

Our analysis employs a locally-convergent variant of the update in (3)–(4), given by

$$\mathbf{w}_{new} = \mathbf{w} + \eta(d - y)u(d)\mathbf{x} \tag{7}$$

$$b_{new} = b + \eta(d - y)u(d), \tag{8}$$

where $u(y)$ is the unit step function, which can be shown to be equivalent to (3)–(4) when y is in the close vicinity of d. Eqns. (7)–(8) are identical to the well-known least-mean-square (LMS) adaptive filtering algorithm with data-dependent step size $\eta(d) = \eta u(d)$, admitting a straightforward dynamic analysis.

Without loss of generality, let $\mathbf{a}$ take the simplified form

$$\mathbf{a} = \begin{cases} a & i = L \\ 0 & 1 \le i \le L-1. \end{cases} \tag{9}$$

This form is valid because of the covariance structure of $\mathbf{x}$, which is rotation-invariant; hence, we can rotate the axes of the analysis to allow the above form. Then,

$$d = ax_L + c \tag{10}$$

$$x_L = \frac{d - c}{a}. \tag{11}$$

Thus, the desired response signal is also Gaussian with distribution

$$d \sim \mathcal{N}(\mu_d, a^2), \quad \mu_d = a\mu_L + c. \tag{12}$$

This signal model allows us to write the coefficient updates of the algorithm analysis model as

$$\mathbf{w}_{new} = \mathbf{w} + \eta u(d)\left(d - w_L\frac{d-c}{a} - b - \sum_{j=1}^{L-1} w_j x_j\right)\mathbf{x} \tag{13}$$

$$b_{new} = b + \eta u(d)\left(d - w_L\frac{d-c}{a} - b - \sum_{j=1}^{L-1} w_j x_j\right). \tag{14}$$

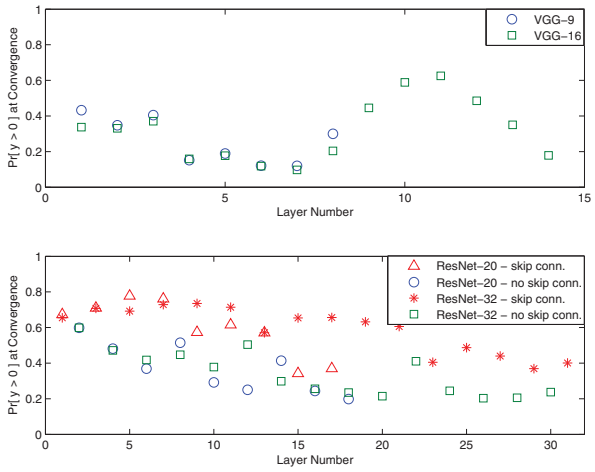

Fig. 5. Final $\Pr[y > 0]$ values for the VGG (top) and ResNet (bottom) architectures on CIFAR-10.

At this point, we are able to determine the evolutionary equations for the mean coefficient vector $E\{\mathbf{w}\}$ and bias $E\{b\}$ under the independence assumptions [12] as is done in linear adaptive systems. For this analysis, we require the quantities

$$E\{d^n u(d)\} = \int_0^\infty \frac{x^n}{\sqrt{2\pi a^2}} \exp\left(-\frac{(x-\mu_d)^2}{2a^2}\right) dx. \quad (15)$$

It can be shown that

$$E\{u(d)\} = \Phi\left(\frac{\mu_d}{|a|}\right) \quad (16)$$

$$= \frac{1}{2}\left(\text{erf}\left(\frac{\mu_d}{\sqrt{2}|a|}\right) + 1\right) \quad (17)$$

$$E\{du(d)\} = \frac{|a|}{\sqrt{2\pi}} \exp\left(-\frac{\mu_d^2}{2a^2}\right) + \mu_d E\{u(d)\} \quad (18)$$

$$E\{d^2 u(d)\} = \frac{|a|\mu_d}{\sqrt{2\pi}} \exp\left(-\frac{\mu_d^2}{2a^2}\right) + (\mu_d^2 + a^2) E\{u(d)\} \quad (19)$$

With the above results, we have

$$E\{\mathbf{w}_{new}\} = (\mathbf{I} - \eta\mathbf{R})\, E\{\mathbf{w}\} - \eta\mathbf{r}E\{b\} + \eta\mathbf{p} \quad (20)$$

$$E\{b_{new}\} = (1 - \eta E\{u(d)\})\, E\{b\} - \eta\mathbf{r}^T E\{\mathbf{w}\} + \eta E\{du(d)\}, \quad (21)$$

where we have defined

$$\mathbf{R} = E\{u(d)\mathbf{x}\mathbf{x}^T\} \quad (22)$$

$$\mathbf{r} = E\{u(d)\mathbf{x}\} \quad (23)$$

$$\mathbf{p} = E\{du(d)\mathbf{x}\}. \quad (24)$$

The entries for $\mathbf{R}$, $\mathbf{r}$, and $\mathbf{p}$ can be evaluated using the formulas for $E\{d^n u(d)\}$ shown previously, the details of which are omitted for brevity. The expressions obtained are shown at the top of the next page.

We can combine the updates for both $\mathbf{w}$ and b in the $(L+1)$-element vector

$$\overline{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}. \quad (28)$$

Let us define

$$\overline{\boldsymbol{\mu}} = [1 \;\; \mu_1 \;\; \cdots \;\; \mu_{L-1} \;\; h]^T \quad (29)$$

where

$$h = \mu_L + \frac{\text{sgn}(a)}{\sqrt{2\pi}E\{u(d)\}} \exp\left(-\frac{\mu_d^2}{2a^2}\right). \quad (30)$$

Then, the evolutionary equation for the means of the elements of $\overline{\mathbf{w}}$ is

$$E\{\overline{\mathbf{w}}_{new}\} = (\bar{\mathbf{I}} - \eta\mathbf{A})E\{\overline{\mathbf{w}}\} + \eta\mathbf{b}, \quad (31)$$

where

$$\mathbf{A} = E\{u(d)\}(\bar{\mathbf{I}} + \overline{\boldsymbol{\mu}}\overline{\boldsymbol{\mu}}^T - \bar{\boldsymbol{\delta}}_1\bar{\boldsymbol{\delta}}_1^T) - K\bar{\boldsymbol{\delta}}_{L+1}\bar{\boldsymbol{\delta}}_{L+1}^T \quad (32)$$

$$\mathbf{b} = \overline{\boldsymbol{\mu}}E\{u(d)\}(c + ah) + \bar{\boldsymbol{\delta}}_{L+1}a(E\{u(d)\} - K) \quad (33)$$

$$K = \frac{1}{\sqrt{2\pi}}\left(\mu_L \text{sgn}(a) + \frac{c}{|a|}\right)\exp\left(-\frac{\mu_d^2}{2a^2}\right) + \frac{1}{2\pi E\{u(d)\}}\exp\left(-\frac{\mu_d^2}{a^2}\right). \quad (34)$$

Examining the form of $\mathbf{A}$ above, we see that it has $(L-2)$ eigenvalues equal to $E\{u(d)\} = \Pr[d > 0] \approx \Pr[y > 0]$ for our assumed analytical model. Thus, convergence in almost all signal dimensions will be exponential with the factor $(1 - \eta\Pr[y > 0])$ near convergence. It is generally not possible to choose a large step size to obtain faster convergence, as other signal dimensions have larger eigenvalues which limit the maximum step size allowed. Thus, if $\Pr[y > 0]$ is small, the unit will have slow convergence in these dimensions.

We have performed Monte Carlo simulations to verify the equations derived. In these simulations, we have implemented an $L = 11$-parameter single ReLU adaptive unit updated using the analysis model in (7)–(8) with $\eta u(d)$ step size, Gaussian input and desired response signals with $\boldsymbol{\mu} = [2 \;\; 1.6 \;\; 1.2 \;\; \cdots \;\; -1.6 \;\; -2.0]^T$, $a = 0.5$, and c chosen to obtain $\Pr[d > 0]$ values given by one of $\{0.8, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05\}$. The initial weight errors are distributed at $\mathcal{N}(0, 0.01)$, and the average squared weight error norm is computed across 10000 algorithm iterations and 1000 simulation runs to compare with the theory. Figure 6 shows the convergence of the algorithm predicted by the theoretical relations and from the numerical simulations, where close agreement is seen. In addition, it is observed that the convergence speed is slowed as $\Pr[d > 0]$ is reduced, which verifies the analytical prediction.

$$[\mathbf{R}]_{ij} = \begin{cases} (\mu_i\mu_j + \delta_{ij})E\{u(d)\} & 1 \le i \le L-1 \text{ and } 1 \le j \le L-1 \\ \frac{1}{\sqrt{2\pi}}\mu_i \text{sgn}(a) \exp\left(-\frac{\mu_d^2}{2a^2}\right) + \mu_i\mu_L E\{u(d)\} & 1 \le i \le L-1 \text{ and } j = L \\ \frac{1}{\sqrt{2\pi}}\mu_j \text{sgn}(a) \exp\left(-\frac{\mu_d^2}{2a^2}\right) + \mu_L\mu_j E\{u(d)\} & i = L \text{ and } 1 \le j \le L-1 \\ \frac{1}{\sqrt{2\pi}}\left(\mu_L \text{sgn}(a) - \frac{c}{|a|}\right) \exp\left(-\frac{\mu_d^2}{2a^2}\right) + (\mu_L^2+1)E\{u(d)\} & i = j = L \end{cases} \quad (25)$$

$$[\mathbf{p}]_i = \begin{cases} a[\mathbf{R}]_{iL} + \mu_i c E\{u(d)\}, & 1 \le i \le L-1 \\ a\left(\frac{1}{\sqrt{2\pi}}\mu_L \text{sgn}(a) \exp\left(-\frac{\mu_d^2}{2a^2}\right) + (\mu_L^2 + 1 + \frac{c}{a}\mu_L)E\{u(d)\}\right) & i = L \end{cases} \quad (26)$$

$$[\mathbf{r}]_i = \begin{cases} \mu_i E\{u(d)\} & 1 \le i \le L-1 \\ \frac{1}{\sqrt{2\pi}} \text{sgn}(a) \exp\left(-\frac{\mu_d^2}{2a^2}\right) + \mu_L E\{u(d)\} & i = L \end{cases} \quad (27)$$

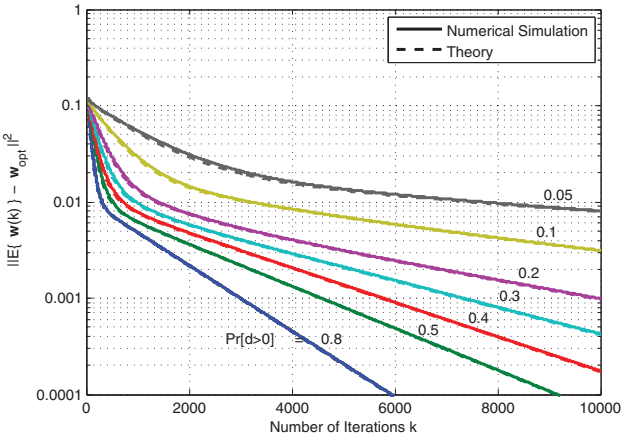

Fig. 6. Analysis model performance verification; see text for explanation.

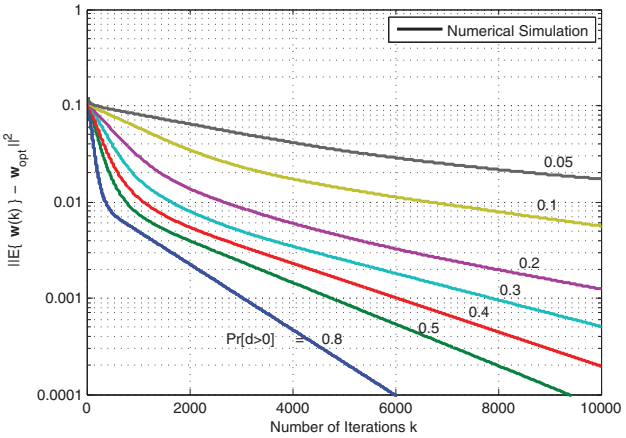

Fig. 7. Original model performance verification; see text for explanation.

To verify that the analytical model accurately depicts the behavior of the original single-unit ReLU update in (3)–(4) with $\eta u(y)$ step size, we repeat the above simulations for the original algorithm. Figure 7 shows the results, in which the observed behavior of the original algorithm is quite similar to that predicted by the analysis model near convergence, except that the original algorithm is initially slower for small $\Pr[d > 0]$ values.

4. CONCLUSIONS

In this paper, we study issues surrounding "dying ReLUs" in deep neural networks. Simulation studies applying well-known architectures to the CIFAR-10 benchmark show that output activations within the networks are sparse, particularly for layers with no skip connections. We also analyze a simple single-unit model for understanding the convergence behaviors of dying ReLU units. Additional work is ongoing.

5. REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, and J.L. McClelland, "A general framework for parallel distributed processing," in *Parallel Distributed Processing, Vol. 1: Foundations*, D.E. Rumelhart and J.L. McClelland, eds. (Cambridge, MA: Bradford Books, 1986), pp. 45-76.

[2] V. Nair and G.E. Hinton, "Rectified linear units improve restricted Boltzmann machines," *Proc. 27th Int. Conf. Machine Learning,* Hafia, Israel, pp. 807-814, June 2010.

[3] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Proc. 14th Int. Conf. Artifical Intell., Stat.,* Ft. Lauderdale, FL, pp. 315-323, 2011.

[4] M.D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q.V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G.E. Hinton, "On rectified linear units for speech processing," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Vancouver, BC, Canada, pp. 3517-3521, May 2013.

[5] A.L. Maas, A.Y. Hannun, and A.Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Proc. 39th Int. Conf. Machine Learning,* Atlanta, GA, 6 pp., 2013.

[6] N.J. Bershad, J.J. Shynk, and P.L. Feintuch, "Statistical analysis of the single-layer backpropagation algorithm: Part I- mean weight behavior," *IEEE Trans. Signal Processing,* vol. 41, no. 2, pp. 573-582, Feb. 1993.

[7] N.J. Bershad, J.J. Shynk, and P.L. Feintuch, "Statistical analysis of the single-layer backpropagation algorithm: Part II- MSE and classification performance," *IEEE Trans. Signal Processing,* vol. 41, no. 2, pp. 583-591, Feb. 1993.

[8] S.C. Douglas and T.H.-Y. Meng, "Stochastic gradient adaptation under general error criteria," *IEEE Trans. Signal Processing,* vol. 42, no. 6, pp. 1335-1351, June 1994.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. 2015 Int. Conf. on Learning Representations,* San Diego, CA, arXiv:1409.1556, 14 pp., May 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. 29th IEEE Conf. Computer Vision and Pattern Recog.,* pp. 770-778, June 2016.

[11] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., Apr. 8, 2009, 58 pp. Available at http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf; accessed Dec. 2018.

[12] J.E. Mazo, "On the independence theory of equalizer convergence," *Bell Syst. Tech. J.,* vol. 58, no. 5, pp. 963-993, May-June 1979.