On weight initialization in deep neural networks

Siddharth Krishna Kumar
siddharthkumar@upwork.com

May 4, 2017

Abstract

A proper initialization of the weights in a neural network is critical to its convergence. Current insights into weight initialization come primarily from linear activation functions. In this paper, I develop a theory for weight initializations with non-linear activations. First, I derive a general weight initialization strategy for any neural network using activation functions differentiable at 0. Next, I derive the weight initialization strategy for the Rectified Linear Unit (RELU), and provide theoretical insights into why the *Xavier initialization* is a poor choice with RELU activations. My analysis provides a clear demonstration of the role of non-linearities in determining the proper weight initializations.

1 Introduction

In recent years, there have been rapid advances in our understanding of deep neural networks. These advances have resulted in breakthroughs in several fields, ranging from image recognition ([13],[18],[19]) to speech recognition ([5],[11],[17]) to natural language processing ([2],[8], [15]). These successes have been achieved despite the notorious 臭名昭著 difficulty in training these deep models.

Part of the difficulty in training these models lies in determining the proper initialization strategy for the parameters in the model. It is well known [12] that arbitrary initializations can slow down or even completely stall the convergence process. The slowdown arises because arbitrary initializations can result in the deeper layers receiving inputs with small variances, which in turn slows down back propagation, and retards 妨碍 the overall convergence process. Weight initialization is an area of active research, and numerous methods ([12], [14], [16] to state a few) have been proposed to deal with the problem of the shrinking variance in the deeper layers.

In this paper, I revisit the oldest, and most widely used approach to the problem with the goal of resolving some of the unanswered theoretical questions which remain in the literature. The problem can be stated as follows: If the weights in a neural network are initialized using samples from a normal distribution, $\mathcal{N}(0, v^2)$, how should v^2 be chosen to ensure that the variance of the outputs from the different layers are approximately the same?

The first systematic analysis of this problem was conducted by Glorot and Bengio [3] who showed that for a linear activation function, the optimal value of $v^2 = 1/N$, where N is the number of nodes feeding into that layer. Although the paper makes several assumptions about the inputs to the model, it works extremely well in many cases and is widely used in the initialization of neural networks to date; this initialization scheme is commonly referred to as the *Xavier initialization*.

In an important follow up paper, He and colleagues [6] argue that the *Xavier initialization* does not work well with the RELU activation function, and instead propose an initialization of $v^2 = 2/N$ (commonly referred to as the *He initialization*). In support of their initialization, they provide an example of a 30 layer neural network which converges with the *He initialization*, but not under the *Xavier initialization*. To the best of my knowledge, the precise reason for the convergence of one method and the non-convergence of the other is not fully understood.

My main contributions in this paper are to (a) generalize the results of [3] to the case of non-linear activation functions and (b) to provide a continuum between the results of [3] and [6]. For the class of activation functions differentiable at 0, I provide a general initialization strategy. For the class of activation functions that are not differentiable at 0, I focus on the Rectified Linear Unit (RELU) and provide a rigorous 严谨的 proof of the *He initialization*. I also provide theoretical insights into why the 30 layer neural network converges with the *He initialization* but not with the *Xavier initialization*. As a small bonus 意外收获, I resolve an unanswered question posed in [3] regarding the distributions of activations under the hyperbolic tangent activation.

2 The setup

Consider a deep neural network with M layers. The relationship between the inputs to the m^{th} layer (x_m) and $m+1^{th}$ layer (x_{m+1}) are described by the recursions 递归

$$y_m(i) = \sum_{j=1}^{j=N} \mathbf{W}_m(i,j)x_m(j) = \sum_{j=1}^{j=N} p_{ij} \tag{1}$$

and

$$x_{m+1}(i) = \mathrm{g}(y_m(i)). \tag{2}$$

Here $p_j = \mathbf{W}_m(i,j)x_m(j)$, $\mathbf{W}_m$ is a matrix of weights for the m^{th} layer, g is the non-linear activation function, and N is the number of nodes in the hidden layers respectively. The weights $\mathbf{W}_m(i,j)$ are assumed to be independent identically distributed normal random variables with mean 0 and variance v^2. Consistent 一致性 with the assumptions in [3] and [6], I assume that the inputs to the first layer are independent and identically distributed random variables with mean 0 and variance 1. For convenience, I use r_m and u_m^2 to denote the mean and variance of $y_m(i)$ respectively.

Due to the symmetry 对称性 in the problem, all inputs to the m^{th} layer will have the same means and variances during the first forward pass (i.e., $E(x_m(i)) = \mu_m$ and $Var(x_m(i)) = s_m^2$ for all i); the covariances 协方差矩阵 between the inputs to the m^{th} layer need not be 0.

The goal is to find the value of v^2 which ensures that $s_1^2 \approx s_2^2 ... \approx s_M^2 = 1$ during the first forward pass. To accomplish this, I need to express the central moments of $x_{m+1}(i)$ in terms of the central moments 中心矩 of $x_m(i)$ for an arbitrary value of m. I begin by analyzing properties of the neural network that are independent of the activation function considered in the analysis.

Proposition 1. *During the first iteration, $W_m(i,j)$ is independent of $x_m(k)$ for all values of i, j and k*

Using the recursions in (1) and (2), $x_m(k)$ can be expressed as some non-linear function of the weights in the first $m-1$ layers, and the inputs to the first layer. Since the weights in the m^{th} layer are independent of the inputs to the first layer and the weights in all other layers, the weights in the m^{th} layer will also be independent of any non-linear function of these quantities. Therefore, $W_m(i,j)$ is independent of $x_m(k)$ for all values of i, j and k. Furthermore, since $W_m(i,j)$ is independent of $x_m(k)$ and $x_m(l)$, $W_m(i,j)$ will also be independent of $x_m(k)x_m(l)$□

Taking expectations in (1) and using proposition 1, along with the fact that $E(\mathbf{W}_m(i,j)) = 0$ yields

$$r_m = E(y_m(i)) = \sum_{j=1}^{j=N} E(\mathbf{W}_m(i,j))E(x_m(j)) = 0. \tag{3}$$

Therefore, $u_m^2 = Var(y_m(i)) = E\left(y_m(i)^2\right) - \left(E(y_m(i))\right)^2 = E\left(y_m(i)^2\right)$. Using (1),

$$\begin{aligned} u_m^2 &= E\left(\sum_{j=1}^{j=N} \mathbf{W}_m(i,j)x_m(j)\right)^2 \\ &= \sum_{j=1}^{j=N} E\left(\mathbf{W}_m(i,j)^2 x_m(j)^2\right) + 2\sum E\left(\mathbf{W}_m(i,j)x_m(j)\mathbf{W}_m(k,l)x_m(l)\right). \end{aligned} \tag{4}$$

From proposition 1, $\mathbf{W}_m(i,j)$ and $\mathbf{W}_m(k,l)$ will (a) be independent of each other, and (b) be independent of $x_m(j)x_m(l)$. Using these results, along with the fact that $E(\mathbf{W}_m(i,j)) = 0$ gives

$$E\left(\mathbf{W}_m(i,j)x_m(j)\mathbf{W}_m(k,l)x_m(l)\right) = E(\mathbf{W}_m(i,j))E(\mathbf{W}_m(k,l))E(x_m(j)x_m(l)) = 0. \tag{5}$$

Plugging (5) into (4) gives

$$\begin{aligned} u_m^2 &= \sum_{j=1}^{j=N} E\left(\mathbf{W}_m(i,j)^2 x_m(j)^2\right) \\ &= \sum_{j=1}^{j=N} E\left(\mathbf{W}_m(i,j)^2\right) E\left(x_m(j)^2\right) \\ &= Nv^2(s_m^2 + \mu_m^2) \end{aligned} \tag{6}$$

for all i. Interestingly, this result holds for any arbitrary covariance structure of the inputs to the m^{th} layer.

Equations (3) and (6) provide insights into the central moments of $y_m(i)$, but can we derive insights into the distribution of $y_m(i)$? To answer this question, I make the additional assumption that the number of nodes in the hidden layer (N) is 'large'; this assumption is reasonable given that most neural networks have several hundred nodes in the hidden layers. Under this assumption, we have the following result

Proposition 2. *$y_m(i)$ will be approximately normally distributed for all values of m and i.*

For the first iteration, note that $E(p_{ij}) = E\left(\mathbf{W}_m(i,j)x_m(j)\right) = E\left(\mathbf{W}_m(i,j)\right) E\left(x_m(j)\right) = 0$. Furthermore, for $j \neq k$,

$$\begin{aligned} Cov(p_{ij}, p_{ik}) &= E(p_{ij}p_{ik}) - E(p_{ij})E(p_{ik}) \\ &= E(p_{ij}p_{ik}) \\ &= E\left(\mathbf{W}_m(i,j)x_m(j)\mathbf{W}_m(i,k)x_m(k)\right) = 0, \end{aligned} \tag{7}$$

where the last equality follows from (5). This implies that p_{i1}, p_{i2} ... p_{iN} are independent and identically distributed random variables. Therefore by the Central Limit Theorem, we expect $y_m(i) = \sum_{j=1}^{j=N} p_{ij}$ to converge to a normal distribution when N is large [10]. Even when p_{i1}, p_{i2} ... p_{iN} are dependent and not identically distributed, the conditions required to ensure that $y_m(i) = \sum_{j=1}^{j=N} p_{ij}$ converges to a normal distribution are weak (for a list of all the conditions, see Theorem 2.8.2 in [10]). Thus, $y_m(i)$ is expected to to be approximately normally distributed during most iterations. □

The analysis thus far has focused on providing general insights into the distribution of $y_m(i)$ resulting from equation (1). In order to analyze the role of the non-linearity induced by (2), assumptions need to be made about the nature of $g(x)$. In particular, my analysis critically hinges on the differentiability of $g(x)$ at 0. Accordingly, I split the analysis into two cases. The first case deals with the general class of activation functions differentiable at 0. In the second case, instead of considering all possible non-differentiable functions, I focus on the Rectified Linear Unit (RELU) which is commonly used in the analysis of neural networks.

3 Activation functions differentiable at 0

When $g(x)$ is differentiable at 0, we can perform a Taylor expansion in (2) about $E(y_m(i)) = 0$. Assuming that the higher order terms can be ignored,

$$x_{m+1}(i) \approx g(0) + (y_m - 0)g'(0). \tag{8}$$

Taking the expectation in (8) gives

$$\mu_{m+1} = E(x_{m+1}(i)) \approx g(0). \tag{9}$$

This equation suggests that the expected value of the inputs to the $(m+1)^{th}$ layer has little dependence on the moments of the inputs to the m^{th} layer. Using this result recursively suggests that for all layers (barring the first),

$$\mu_j = g(0) \text{ for all } j \geq 1. \tag{10}$$

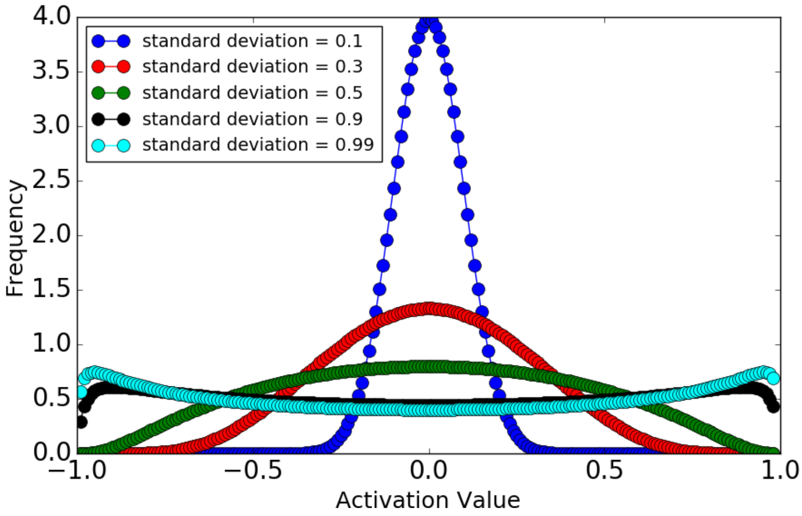

Figure 1: Plots of the pdf described in (16) for different values of the standard deviation (u_m) of $y_m(i)$.

Using (6) and (10), the variance of $x_{m+1}(i)$ can be computed from (8) as

$$\begin{aligned} s_{m+1}^2 = Var(x_{m+1}(i)) &\approx (g'(0))^2 Var(y_m(i)) \\ &= N(g'(0))^2 v^2 (s_m^2 + \mu_m^2) \\ &= N(g'(0))^2 v^2 (s_m^2 + g(0)^2). \end{aligned} \tag{11}$$

Using the condition $s_m^2 \approx s_{m-1}^2 ... = s_1^2 = 1$ along with (10) and (11) gives

$$v^2 = \frac{1}{N(g'(0))^2(1+g(0)^2)}, \tag{12}$$

Equation (12) provide a general weight initialization strategy for any arbitrary differentiable activation function. I use the results developed in this section to analyze the optimal value of v^2 for two commonly used differentiable activation functions - the hyperbolic tangent and the sigmoid.[1]

3.1 Hyperbolic tangent activation

For the hyperbolic tangent

$$g(x) = \tanh(x), \tag{13}$$

we have $g(0) = 0$ and $g'(0) = 1$. Plugging these results in (12) yields

$$v \approx \frac{1}{\sqrt{N}}, \tag{14}$$

which is precisely the *Xavier Initialization.*

Sequential saturation with the hyperbolic tangent

In their analysis of a neural network with the hyperbolic tangent activations, [3] find that the deeper layers in the neural network have a greater proportion of unsaturated nodes than the shallower layers. As is stated in their paper, ‘why this is happening remains to be understood’.

To explain their finding, I begin by noting that in [3], the authors initialize the weights using samples from a uniform distribution $\left(U[-1/\sqrt{N}, 1/\sqrt{N}]\right)$ having a variance of $1/3N$. Therefore, from (10) and (11), with $g(0) = 0$ and $g'(0) = 1$, we have $\mu_m = 0$ and

[1] In their calculations, [3] and [6] impose an additional set of constraints to ensure that the variance is maintained even during the backward pass. I believe that this is not required, since the requirement that the variance of the inputs at each layer be the same ensures that the gradient flows through in the backward pass.

$$s_{m+1}^2 = \frac{1}{3} s_m^2 < s_m^2 \tag{15}$$

respectively. From (6), this implies that $u_{m+1}^2 < u_m^2$ for all m (i.e., u_m^2 is a decreasing function of m). Furthermore from proposition 2, $y_m(i) \sim \mathcal{N}(0, u_m^2)$. Therefore, $x_{m+1}(i)$ will be the tanh transformation of a normal random variable. Using results from [4], $x_{m+1}(i) = \tanh(y_m(i))$ will have a probability density function (pdf) given by

$$f(y) = \frac{1}{1-y^2} \frac{1}{\sqrt{2\pi u_m^2}} \mathrm{e}^{\left(-\frac{t_y^2}{2u_m^2}\right)}, \tag{16}$$

where

$$t_y = \frac{1}{2} ln \left(\frac{1+y}{1-y}\right). \tag{17}$$

Plots of this pdf for different values of u_m (provided in Figure 1) produce trends similar to those observed by the simulation studies of [3] (figure 4 in their paper). A comparison of Figure 1 and figure 4 of [3] suggests that u_m^2 is a decreasing function of m, as is expected.

From the results in [4], we expect the activations to be (a) approximately normally distributed when u_m is close to 0 and (b) bimodally distributed with local maximas near -1 and +1 when u_m is close to 1. Accordingly, since u_m^2 is a decreasing function of m, we expect the activations from the shallower layers to be more saturated (i.e., more concentrated near -1 and +1), and the saturation in the activations to reduce as we go to the deeper layers in the network.

3.2 Sigmoid activation

For the sigmoid activation defined as

$$g(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{18}$$

we have $g(0) = 0.5$, $g'(0) = 1/4$. Plugging these values in (12) yields

$$v \approx \frac{3.6}{\sqrt{N}}, \tag{19}$$

To compare the initialization described in (19) with the *Xavier initialization*, I use a simple 10 layer neural network whose architecture is described in Figure 2. For my experiments, I use the CIFAR 10 dataset [9] comprising 60,000, 32 × 32 color images evenly split over 10 classes. The dataset comprises 50,000 training examples (which forms the training dataset in my analyses) and 10,000 test examples (which forms the validation dataset in my analyses).

First, I train the neural network with the *Xavier initialization* for 10 iterations and compute the top 5 accuracy on the validation dataset for each iteration. Next, I repeat the process using the initialization stated in (19). A comparison of the validation accuracies for the 2 cases is provided in Figure 3, which shows that the convergence appears to stall with the *Xavier initialization*, but proceeds rapidly with the initialization proposed in (19).[2]

4 Activation functions not differentiable at 0

When $g(x)$ is not differentiable at 0, the analysis seems more difficult than in the previous section. Instead of attempting to provide a general solution, I focus on the most important non-differentiable activation function used in the analysis of neural networks - the Rectified Linear Unit (RELU).

4.1 RELU activation

Since the RELU activation is not differentiable at 0, the results from (8 − 11) cannot be used to compute the optimal value of v^2. To proceed, I use proposition 2 and (3) which state that

[2] Python code (using the package Keras [1]) to replicate Figure 3 can be downloaded from `https://github.com/sidkk86/weight_initialization`

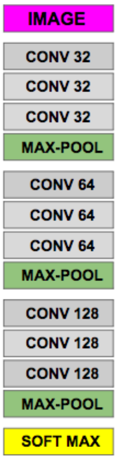

Figure 2: Architecture of deep nural network used in analysis of sigmoid; stride lengths in all layers are 2 × 2.

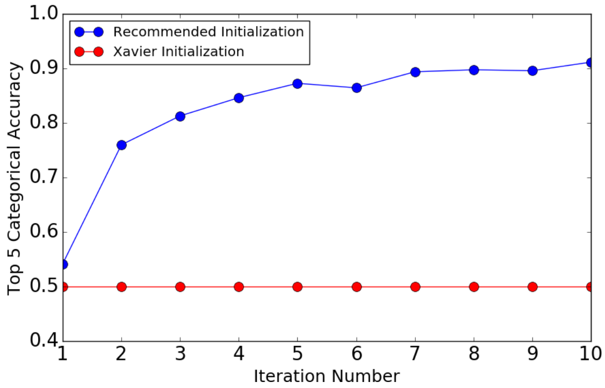

Figure 3: Convergence comparison of *Xavier initialization* with initialization recommended in (19). The *Xavier initialization* stalls while the initialization recommended in (19) converges proceeds rapidly towards convergence.

for the first iteration, $y_m(i) \sim \mathcal{N}(0, u_m^2)$. We are interested in in the mean and variance of $x_{m+1}(i) = \max(0, y_m(i))$. The mean will be given by

$$\mu_{m+1} = E(y_m(i)\mathbf{I}(y_m(i) > 0)) = \frac{1}{u_m\sqrt{2\pi}}\int_0^\infty x\mathrm{e}^{-\frac{x^2}{2u_m^2}}\,dx = \frac{u_m}{\sqrt{2\pi}}\int_0^\infty \mathrm{e}^{-t}dt = \frac{u_m}{\sqrt{2\pi}}. \tag{20}$$

Similarly,

$$E(x_{m+1}(i)^2) = \frac{1}{u_m\sqrt{2\pi}}\int_0^\infty x^2\mathrm{e}^{-\frac{x^2}{2u_m^2}}\,dx = \left(\frac{1}{2}\right)\frac{1}{u\sqrt{2\pi}}\int_{-\infty}^\infty x^2\mathrm{e}^{-\frac{x^2}{2u^2}}\,dx = \frac{u_m^2}{2}. \tag{21}$$

Using (20) and (21),

$$s_{m+1}^2 = E(x_{m+1}(i)^2) - \mu_{m+1}^2 = \frac{u_m^2}{2} - \frac{u_m^2}{2\pi} \approx 0.34u_m^2. \tag{22}$$

For the variance to be maintained at each iteration, we require $s_{m+1}^2 \approx 1$ which yields

$$u_m^2 \approx 3. \tag{23}$$

Plugging (23) in (20) yields $\mu_{m+1} \approx 0.7$. By the symmetry of the problem during the first iteration, we expect $\mu_{m+1} \approx \mu_m ... \approx \mu_2 \approx 0.7$. Using this result in (6) yields

$$3 = Nv^2(1 + 0.49) \tag{24}$$

or

$$v^2 \approx 2/N, \tag{25}$$

which is consistent with that obtained by [6].

To converge or not to converge, that is the question.

In [6] paper, the authors provide an example of a 22 layer neural network using RELU activations which converges with the *Xavier Initialization*, and a 30 layer neural network which does not converge with the same initializations and activation functions.

To understand why this happens, I compute the central moments of x_{m+1} in terms of the central moments of x_m when $v^2 = 1/N$. From (6) we have

$$u_m^2 = s_m^2 + \mu_m^2. \tag{26}$$

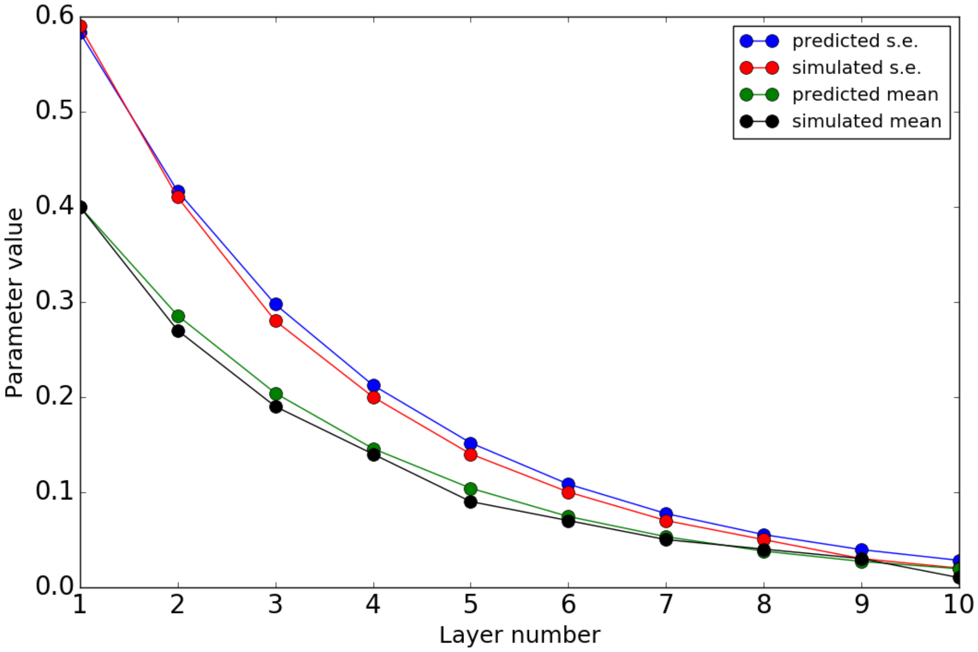

Figure 4: A comparison of the predicted means and standard errors obtained from (29 – 30) with the simulated values reported in slide 61 of [7].

Plugging results from (26) into (20) yields the recursion

$$\mu_{m+1}^2 = \frac{1}{2\pi}(s_m^2 + \mu_m^2) \approx 0.16(s_m^2 + \mu_m^2). \tag{27}$$

Similarly, plugging results from (26) and (27) in (22) yields

$$s_{m+1}^2 \approx 0.34(s_m^2 + \mu_m^2). \tag{28}$$

Simple manipulations of equations (26 – 28) gives

$$\mu_m^2 \approx 0.16 \times (0.51)^{m-1} \tag{29}$$

and

$$s_m^2 \approx 0.34 \times (0.51)^{m-1}. \tag{30}$$

for all $m \geq 1$. These approximations are remarkably accurate, as is demonstrated by comparisons with the simulation experiments described in Figure 4.

Equation (30) shows that the variance of the inputs to the deeper layers is exponentially smaller than the variance of the inputs to the shallower layer. Therefore, the deeper the neural network, the worse the performance of the *Xavier Initialization* will be. From (30), $s_{22}^2 \approx 1.62 \times 10^{-7}$ and $s_{30}^2 \approx 6.33 \times 10^{-10}$. Thus the variance in the input to the 30^{th} layer will be $(0.51)^8 = 3 \times 10^{-3}$ times smaller than the variance to the 22^{nd} layer, and explains the possible reason why the 30 layer neural network described in [6] converges, but the 22 layer neural network does not. [3]

5 Conclusion

In this paper, I have provided a general framework for weight initialization with non-linear activation functions. First, I provide a general formula for the ideal weight initialization for all activation functions differentiable at 0. I show how the weight initializations change for the hyperbolic tangent and sigmoid activation functions. Second, I focus only on the Rectified Linear Unit (RELU) from the class of functions that are non-differentiable at 0, and I provide a rigorous proof of the *He*

[3] It is surprising that the 22 layer neural network converges!

Initialization. Finally, I show why the *Xavier initialization* fails to work with the RELU activation function. Given the sharp increase in non-differentiable activation functions over the years, a more general version of my (largely incomplete) analysis of non-differentiable functions is warranted. My analysis repeatedly illustrates the drastic difference in dynamics which can result from introducing non-linearities in the system.

References

[1] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[2] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[4] Michael D Godfrey. The tanh transformation. *Information Systems Laboratory, Stanford University*, 2009.

[5] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[6] 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.

[7] Andrej Karpathy, Justin Johnson, and Fei Fei Li. Cs 231n: Convolutional neural networks for visual recognition, lecture 5, slide 61. http://cs231n.stanford.edu/slides/2016/winter1516_lecture5.pdf, 2016.

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[10] Erich Leo Lehmann. *Elements of large-sample theory*. Springer Science & Business Media, 2004.

[11] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[12] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[14] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[15] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[16] David Sussillo and LF Abbott. Random walk initialization for training very deep feedforward networks. *arXiv preprint arXiv:1412.6558*, 2014.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[18] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[19] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.