Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2020-0165

Vol. **28**, No. 5, pp. 1671-1706
November 2020

Dying ReLU and Initialization: Theory and Numerical Examples

Lu Lu[1,†], Yeonjong Shin[2,*,†], Yanhui Su[3] and George Em Karniadakis[2]

[1] *Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.*
[2] *Division of Applied Mathematics, Brown University, Providence, RI 02912, USA.*
[3] *College of Mathematics and Computer Science, Fuzhou University, Fuzhou, Fujian 350116, China.*

Received 26 August 2020; Accepted 14 October 2020

Abstract. The dying ReLU refers to the problem when ReLU neurons become inactive and only output 0 for any input. There are many empirical and heuristic explanations of why ReLU neurons die. However, little is known about its theoretical analysis. In this paper, we rigorously prove that a deep ReLU network will eventually die in probability as the depth goes to infinite. Several methods have been proposed to alleviate the dying ReLU. Perhaps, one of the simplest treatments is to modify the initialization procedure. One common way of initializing weights and biases uses symmetric probability distributions, which suffers from the dying ReLU. We thus propose a new initialization procedure, namely, a randomized asymmetric initialization. We show that the new initialization can effectively prevent the dying ReLU. All parameters required for the new initialization are theoretically designed. Numerical examples are provided to demonstrate the effectiveness of the new initialization procedure.

AMS subject classifications: 60J05, 62M45, 68U99

Key words: Neural network, Dying ReLU, Vanishing/Exploding gradient, Randomized asymmetric initialization.

1 Introduction

The rectified linear unit (ReLU), $\max\{x,0\}$, is one of the most successful and widely-used activation functions in deep learning [30, 35, 40]. The success of ReLU is based on its superior training performance [16, 47] over other activation functions such as the logistic

[†]L. Lu and Y. Shin contributed equally to this work.
[*]Corresponding author. *Email addresses:* `lu_lu@mit.edu` (L. Lu), `yeonjong_shin@brown.edu` (Y. Shin), `suyh@fzu.edu.cn` (Y. Su), `george_karniadakis@brown.edu` (G. E. Karniadakis)

sigmoid and the hyperbolic tangent [15, 31]. The ReLU has been used in various applications including image classification [29, 49], natural language processes [33], speech recognition [23], and game intelligence [44], to name a few.

The use of gradient-based optimization is inevitable in training deep neural networks. It has been widely known that the deeper a neural network is, the harder it is to train [9, 46]. A fundamental difficulty in training deep neural networks is the vanishing and exploding gradient problem [6, 17, 39]. The dying ReLU is a kind of vanishing gradient, which refers to a problem when ReLU neurons become inactive and only output 0 for any input. It has been known as one of the obstacles in training deep feed-forward ReLU neural networks [1, 50]. To overcome this problem, a number of methods have been proposed. Broadly speaking, these can be categorized into three general approaches. One approach modifies the network architectures. This includes but not limited to the changes in the number of layers, the number of neurons, network connections, and activation functions. In particular, many activation functions have been proposed to replace the ReLU [7, 20, 28, 33]. However, the performance of other activation functions varies on different tasks and data sets [40] and it typically requires a parameter to be turned. Thus, the ReLU remains one of the popular activation functions due to its simplicity and reliability. Another approach introduces additional training steps. This includes several normalization techniques [4, 24, 42, 51, 53] and dropout [45]. One of the most successful normalization techniques is the batch normalization [24]. It is a technique that inserts layers into the deep neural network that transform the output for the batch to be zero mean unit variance. However, batch normalization increases by 30% the computational overhead to each iteration [34]. The third approach modifies only weights and biases initialization procedure without changing any network architectures or introducing additional training steps [15, 20, 31, 34, 43]. The third approach is the topic of our work presented in this paper.

The intriguing ability of gradient-based optimization is perhaps one of the major contributors to the success of deep learning. Training deep neural networks using gradient-based optimization fall into the nonconvex nonsmooth optimization. Since a gradient-based method is either a first- or a second-order method, and once converged, the optimizer is either a local minimum or a saddle point. The authors of [12] proved that the existence of local minima poses a serious problem in training neural networks. Many researchers have been putting immense efforts to mathematically understand the gradient method and its ability to solve nonconvex nonsmooth problems. Under various assumptions, especially on the landscape, many results claim that the gradient method can find a global minimum, can escape saddle points, and can avoid spurious local minima [2, 8–10, 13, 14, 25, 32, 52, 55, 57]. However, these assumptions do not always hold and are provably false for deep neural networks [3,26,41]. This further limits our understanding on what contributes to the success of the deep neural networks. Often, theoretical conditions are impossible to be met in practice.

Where to start the optimization process plays a critical role in training and has a significant effect on the trained result [36]. This paper focuses on a particular kind of bad