Successfully and Efficiently Training Deep Multi-layer Perceptrons with Logistic Activation Function Simply Requires Initializing the Weights with an Appropriate Negative Mean

Ahmet Yilmaz[a,b,*], Riccardo Poli[a]

[a]*School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, UK*
[b]*Department of Computer Engineering, Karamanoglu Mehmetbey University, Karaman, Turkey*

Abstract

The vanishing gradient problem (i.e., gradients prematurely becoming extremely small during training, thereby effectively preventing a network from learning) is a long-standing obstacle to the training of deep neural networks using sigmoid activation functions when using the standard back-propagation algorithm. In this paper, we found that an important contributor to the problem is weight initialization.

We started by developing a simple theoretical model showing how the expected value of gradients is affected by the mean of the initial weights. We then developed a second theoretical model that allowed us to identify a sufficient condition for the vanishing gradient problem to occur. Using these theories we found that initial back-propagation gradients do not vanish if the mean of the initial weights is negative and inversely proportional to the number of neurons in a layer.

Numerous experiments with networks with 10 and 15 hidden layers corroborated the theoretical predictions: if we initialized weights as indicated by the theory, the standard back-propagation algorithm was both highly successful and efficient at training deep neural networks using sigmoid activation functions.

Keywords: Deep neural networks, vanishing gradient, weights initialization, logistic activation function, supervised learning.

1. Introduction

The Multi-layer Perceptron (MLP) is one of the most widely used types of artificial neural network [1]. The connection weights of an MLP are traditionally optimized by the standard back-propagation (SBP) algorithm which is a form of gradient descent [2, 3, 4].

Initially, only MLPs with one or a few hidden layers (now called "shallow" networks) were common. So, their properties were studied in depth. For instance, they were proven to be general function approximators [5, 6, 7, 8, 9, 10, 11]. Deeper MLPs, having more than a few hidden layers, were tested, but they suffered from the so called *vanishing gradient problem* [12] which was a strong obstacle to train such networks.

The vanishing gradient problem may be briefly described as a situation where gradients prematurely become extremely small during the training, thereby effectively freezing the corresponding weights and preventing the network from learning. This usually happens when a deep network uses the logistic or hyperbolic tangent (TanH) activation functions, and in recurrent neural networks where gradients are propagated through time as well as layers [12, 13, 14, 15, 16]. So, *deep* MLPs did not receive much attention by the neural network scientific community until Hinton [17, 18] found a method (based on a greedy learning algorithm and pre-training) that could successfully

*Corresponding author
Email addresses: `yilmazahmet@kmu.edu.tr` (Ahmet Yilmaz), `rpoli@essex.ac.uk` (Riccardo Poli)

train them. After this achievement, deep learning has become a very popular area of machine learning.

While there are a small number of techniques that can ameliorate the vanishing gradient problem (we will review them in Section 3), it is fair to say that *no method based on a precise identification of root causes of the problem and its solution has been proposed so far.*[1]

In this paper, we found that *an important contributor to the problem is weight initialization* and that, if properly initialized, deep MLPs using *sigmoid activation functions* can be effectively trained using the standard back-propagation algorithm without experiencing the vanishing gradient problem.

The paper is organized as follows. In Section 2, we provide minimalist descriptions of MLPs and the SBP both for completeness and for the purposes of defining the notation used in the rest of the paper. We look at previous work on the vanishing gradient problem and methods proposed to overcome it (including, among other techniques, also a small number of initialization strategies) in Section 3. In Section 4 we present two theoretical contributions and use them to derive a new initialization method for deep neural networks with sigmoid activation function. More specifically, we develop a simple theoretical model that illustrates how the expected value of the gradient is affected by the expected value of the initial weight distribution (Section 4.1) and we identify a sufficient condition for the vanishing gradient problem to occur in MLPs (Section 4.2). We then (Section 4.3) use these models to understand the interplay between number of neurons in a layer, mean and standard deviation of the weight distribution and gradient magnitudes. Using the lessons learnt from the theory, in Section 4.4 we derive a *new initialization method that should prevent vanishing gradients at least at the very beginning of the training process.* The method sets the mean initial weights to `max(-1, -8 / number_of_neurons_in_layer)`. In Section 5 we report results of experiments with six classification problems (Section 5.1) using deep neural networks with 10 and 15 hidden layers and 10 and 100 neurons in the hidden layers. More specifically, we first corroborate the predictions on initial gradients obtained from the theory (Section 5.2), finding that they were accurate. Then (Section 5.3), we train neural networks initialized with our method and three other initialization methods proposed in the literature, finding that, with our initialization, the networks do not suffer from the vanishing gradient problem at all, suggesting that *our initialization method is an effective solution to the vanishing gradient problem.* We discuss the lessons learnt from theoretical and empirical evidence in Section 5.4. Finally, in Section 6, we present some conclusions, highlight limitations of the work and describe possible avenues for future research.

2. Multilayer Perceptron

2.1. Normal Operation

An MLP consists of multiple layers of neurons connected by corresponding layers of weights.

Let a^l be a vector representing the activation of the neurons in layer l. If l is not the output layer, this can also be considered as the input vector for the neurons in layer $l+1$. Let w^l be a matrix representing the weights between layer l and layer $l+1$.[2] Let net^l be net input vector of layer l, which has elements

$$net_j^l = w_{0j}^{l-1} + \sum_i a_i^{l-1} w_{ij}^{l-1}, \tag{1}$$

where

$$a_j^l = \begin{cases} f(net_j^l) & \text{if } 1 < l \leq L, \\ x_j & \text{if } l = 1, \end{cases} \tag{2}$$

[1]Over time, several improvements to SBP have been developed including gradient-based and non-gradient-based ones [19, 20, 21, 22, 23, 24, 25]. Naturally, the vanishing gradient problem and, so, our results, are not relevant for methods that are not based on gradient descent. Also, here we only consider gradients under the SBP learning rule, leaving extensions of the work to gradient-based improvements of SBP to future research (as discssed in Section 6).

[2]Following standard convention, we include in this matrix also the bias vector w_0^l which includes the biases of the neurons in layer $l+1$.

x_j being the j-th element of input pattern x and f is the activation function. We will use the *logistic activation function* in the experiments in this study, for which

$$f(net) = \frac{1}{1 + e^{-net}}. \tag{3}$$

2.2. Learning by Error Back-propagation

The aim of back-propagation is to decrease the error of the network calculated at the output layer, by apportioning and back-propagating the "blame" to the weights using a gradient descent on the error function. The error is the difference between the actual activation values in the output layer and the desired values over the training set of the network. In vector form, the error produced when training pattern x^k is presented in input to the network is

$$e^k = a^L(x^k) - y^k, \tag{4}$$

where $a^L(x^k)$ is the activation vector of the last (L-th) layer in the presence of input pattern x^k and y^k is the corresponding desired output vector.

Various methods have been proposed in the literature to evaluate training performance. In this study we used the *average cross-entropy loss*, that is:

$$\mathcal{L} = -\frac{1}{m} \sum_{k=1}^{m} y^k \cdot \log a^L(x^k), \tag{5}$$

where m is the number of examples in the data set.

In the online version of the SBP learning rule, at each epoch, t, each weight is updated with a fraction of the derivative of the error with respect to the weight, i.e.,

$$\Delta w_{ij}^l(t) = \eta \delta_j^{l+1} a_i^l, \tag{6}$$

where η is the learning rate and

$$\delta_j^l = \begin{cases} e_j & \text{if } l = L, \\ f'(net_j^l) \sum_i w_{ji}^l \delta_i^{l+1} & \text{otherwise,} \end{cases} \tag{7}$$

where e_j in the error in output neuron j and f' is the derivative of the activation function f. For the logistic function considered here, this can be written as $f'(net_j^l) = a_j^l(1 - a_j^l)$. This equation is applicable on the assumption that the output layer is a soft-max layer.

3. Prior Work the Vanishing Gradient Problem and Methods to Tackle It

The vanishing gradient problem is a major obstacle to the training of deep networks, especially when using a sigmoidal activation function. In this section, we will review prior work on this problem as well as attempts to solve it.

3.1. Vanishing Gradient Problem

The vanishing gradient problem was first identified in [12], in particular in the context of recurrent neural networks. As we indicated above, it occurs when gradients prematurely become extremely small during the training, thereby preventing the network from learning. A correspondingly problematic situation was also described, the exploding gradients problems, where gradients grow exponentially bigger over time leading to instability that also prevent the network from learning.

A theoretical exploration of this problem was provided in [26] *for the case of recurrent neural networks*, which proved that a *sufficient* condition for vanishing gradients to occur in one such network is that the spectral radius ρ of the weight matrix be smaller than a constant $c = \frac{1}{\max_x f'(x)}$. By inverting the direction of they proof, it was found that a *necessary* condition for the occurrence of exploding gradients is that the spectral radius be larger than c. In the case of the sigmoid activation function considered in here, $c = 4$.

3.2. Methods to Tackle the Vanishing Gradient Problem

3.2.1. Pre-training and Normalization

Bengio *et al.* [27] proposed to use unsupervised greedy layer-wise pre-training. This method initially trains the first hidden layer of a network and then adds a new hidden layer. The outputs of the pre-trained layer are used as inputs for the added hidden layer. The process of adding and pre-training layers continues until all hidden layers in the architecture have been processed. At the end of pre-training, the algorithm adds an output layer and trains the whole network. This method was reported to improve not only the network performance but also its generalization. However, it may be computationally expensive depending on the depth of the network trained.

Batch normalization [28] is another technique to avoid the vanishing gradient problem. This basically normalizes the net input values in the hidden layers before feeding them into the activation function. This technique was tested on a three-hidden-layered fully-connected network with the logistic activation function, resulting in significantly accelerated learning.

3.2.2. Changing Activation Function

The Rectified Linear Unit (ReLU) [29] uses an activation function that was proposed to make deep neural networks trainable. This function directly transfers the net input to the output of the neuron when the net input is positive; otherwise, it sets the output to zero. Because the derivative of ReLU is 1 when the net input is positive, the error can be propagated to the weights in the earlier layers through the active neurons without getting smaller. Therefore, ReLU does not suffer from the vanishing gradient problem as much as sigmoid activation functions. However, it suffers from the so-called *dying neuron problem*: if there is a large gradient for a connection weight, when the weight is updated, the corresponding neuron may become sufficiently inhibited to output zero thereafter (or die). If a neuron dies, it can never be active again. Another drawback of ReLU is that non-zero mean activations lead to the *bias shift problem* [30, 31, 32]. With ReLU, the mean of the activations in the layers is bigger than zero. Therefore, layer by layer, the activations grow bigger and bigger. This causes oscillations in the learning process.

To obviate to such problems, other versions of ReLU, such as the Leaky-ReLU [33] and Parametric-ReLU [34], were introduced. The Leaky-ReLU and Parametric-ReLU produce non-zero activations in the negative interval by replacing the constant part of ReLU with a linear function with a very small slope. The slope of the line (α) is a predefined constant value (typically 0.01) in the Leaky-ReLU, while it is a parameter learnt by the network in the Parametric-ReLU. These functions do not suffer from the dying neuron problem. However, Clevert *et al.* [30] reported that these activation functions are not robust to noise.

The Exponential Linear Unit (ELU) [30] and Hyperbolic Linear Unit (HLU) [31] were proposed to overcome this sensitivity to noise and the bias shift problem. ELU and HLU use the same function as the other ReLU family functions in the positive interval. The difference is that they are non-linear functions in the negative interval.

A related approach was presented in [35] which slightly improves the vanishing gradient problem for the logistic activation function. More specifically, the authors found that adding a carefully chosen small positive constant to the derivative of the logistic function made it possible to train networks with up to 7 hidden layers.

3.2.3. Weight Initialization

Weight initialization is another approach that has been proposed to reduce the vanishing gradient problem in deep networks. Glorot and Bengio [36] analyzed the difficulty of training deep neural networks and proposed to initialize the connection weights for deep MLPs using the distribution $W^l \sim \mathcal{N}(0,\ \sigma^2)$, where $\sigma = \sqrt{2/(n^l + n^{l+1})}$ and n^l is the number of neuron in layer l. Learning in networks with the logistic, TanH and soft-sign [37] activation functions was analyzed and compared. It was found that this method shows poor performance in networks where the logistic activation function is used. Kumar [38] suggested that the distribution of initial weights should vary according to activation function used and proposed to use $\sigma = 3.6/\sqrt{n^l}$ to initialize networks with the logistic activation function.

Unfortunately, these methods did not identify and directly address the root causes for the vanishing gradient problem. So, they suffer from it in a very significant manner when networks are deep and neurons use the logistic activation function.

4. Theoretical Derivation of a new Initialization Strategy

In this section we first develop a theoretical model that shows how the expected value of the gradient in weight space is affected by the expected value of the initial weight distribution (Section 4.1). Then, we extend the theory developed in [26] for recurrent neural networks (Section 4.2) to the case of feed forward MLPs deriving a sufficient condition for vanishing gradients to occur in such networks. We show that both theories imply that initial gradients may not vanish if weights are initialized with a mean that is inversely proportional to the number of neurons in a layer (Section 4.3). Finally, we determine that the proportionality constant should be negative and propose a new initialization method based on this (Section 4.4).

4.1. Approximate Model of Expected Initial Gradients

In Section 2 we have reviewed the equations determining the dynamics of net^l_j, a^l_j, δ^{l+1}_j and Δw^l_{ij}. In this section, we want to perform an analysis of the magnitude of Δw^l_{ij} from these equations *at the very first iteration of the learning process*, i.e., when a training example has been propagated forward once and error has been back-propagated once, but the weights have not been updated yet, in a batch-type back-propagation learning algorithm. We will perform our analysis by introducing drastic approximations. However, results are reasonably accurate as will be confirmed both theoretically (later in the rest of Section 4) and empirically (in Section 5.2).

In relation to the initialization phase of a network, the weights can be seen as stochastic variables $w^{l-1}_{ij} \sim \mathcal{N}(\mu^l,\ \sigma^2)$.[3] Because of this, also the net input net^l_j is a stochastic variable, and so are the activations, a^l_j, and the delta's, δ^{l+1}_j.

The net input is a sum of products of stochastic variables. Let us consider its expectation. From Equation (1), we have:

$$\begin{aligned}
\mathrm{E}[net^l_j] &= E\Big[w^{l-1}_{0j} + \sum_i a^{l-1}_i w^{l-1}_{ij}\Big] \\
&= \mu^{l-1} + \sum_i E[a^{l-1}_i w^{l-1}_{ij}] \\
&= \mu^{l-1}\Big(1 + \sum_i E[a^{l-1}_i]\Big), \qquad (8)
\end{aligned}$$

where the last step is the result of a^{l-1}_i and w^{l-1}_{ij} being initially uncorrelated and, so, $E[a^{l-1}_i w^{l-1}_{ij}] = E[a^{l-1}_i]E[w^{l-1}_{ij}] = \mu^{l-1}E[a^{l-1}_i]$.

Note that this equation shows that $\mathrm{E}[net^l_j]$ does not depend on j, so we can use the term $\mathrm{E}[net^l]$ to indicate the expected net input of any neuron in layer l. Note also that, with the exception of the input layer, due to symmetries, initially $E[a^l_i]$ does not depend on i. So, we can use the term $E[a^l]$ to indicate the expected activation of any neuron in layer l. Thus, we can rewrite Equation (8) as:

$$\mathrm{E}[net^l] = \mu^{l-1}\Big(1 + \sum_i E[a^{l-1}]\Big) = \mu^{l-1}\Big(1 + n^{l-1}E[a^{l-1}]\Big). \qquad (9)$$

Let us now consider the expectation of a^l. By definition, we have that

$$E[a^l] = E\left[f(net^l)\right], \qquad (10)$$

[3]Normally $\mu^l = 0$ for all l, but here we do not make this assumption. Also, we do not assume that all μ^l are the same. However, for simplicity, we use the same σ for all l.

where we omitted the subscript j for the reasons mentioned above. This cannot be computed readily. However, as a first order approximation we have:

$$E[a^l] \quad \cong \quad f\left(E\left[net^l\right]\right). \tag{11}$$

Then, substituting Equation (9) into this equation, we obtain the following *approximate forward propagation* equation:

$$E[a^l] \quad \cong \quad f\left(\mu^{l-1}\left(1 + n^{l-1}E[a^{l-1}]\right)\right). \tag{12}$$

We should note that this equation gives us an approximate recursion which allows us to estimate $E[a^l]$ for the whole network. The initial condition for the recursion is $E[a^0] = E[x]$, $E[x]$ being the mean value of the input activation across all neurons and input patterns in the training set.

Turning now our attention on the expectation for the δ's, we have

$$\begin{aligned} E[\delta_j^{l+1}] &= E\Big[f'(net_j^{l+1})\sum_i w_{ji}^{l+1}\delta_i^{l+2}\Big] \\ &\cong E[f'(net_j^{l+1})]E\Big[\sum_i w_{ji}^{l+1}\delta_i^{l+2}\Big], \end{aligned} \tag{13}$$

where the last step is true on the assumption that $f'(net_j^{l+1})$ and $\sum_i w_{ji}^{l+1}\delta_i^{l+2}$ are weakly correlated. Since δ_i^{l+2} and w_{ij}^{l-1} are initially uncorrelated, the last equation transforms into:

$$\begin{aligned} E[\delta_j^{l+1}] &\cong E[f'(net_j^{l+1})]\sum_i E[w_{ji}^{l+1}]E[\delta_i^{l+2}] \\ &= \mu^{l+1}E[f'(net_j^{l+1})]\sum_i E[\delta_i^{l+2}] \qquad (14) \\ &\cong \mu^{l+1}f'(E[net_j^{l+1}])\sum_i E[\delta_i^{l+2}]. \qquad (15) \end{aligned}$$

For the same reasons that $E[net_j^l]$ does not depend on j, so does any function of net_j^l. Hence, in Equation (15), the subscript j can be omitted in $f'(E[net_j^{l+1}])$. Because of this also E$[\delta_j^{l+1}]$ does not depend on the subscript j. Also, for reasons of symmetry (except for the output neurons), $E[\delta_i^{l+2}]$ does not depend on i. So, the previous equation can be rewritten as

$$E[\delta^{l+1}] \cong f'(E[net^{l+1}])\mu^{l+1}n^{l+2}E[\delta^{l+2}]. \tag{16}$$

Substituting Equation (9) into it, we obtain the following *approximate backpropagation equation*:

$$E[\delta^{l+1}] \quad \cong \quad f'\left(\mu^l\left(1 + n^l E[a^l]\right)\right)\mu^{l+1}n^{l+2}E[\delta^{l+2}]. \tag{17}$$

We should note that, once the values of $E[a^l]$ are known via Equation (12), Equation (17) gives us a recursion that allows us to estimate $E[\delta^{l+1}]$ for the whole network. The initial conditions are $E[\delta^L] \cong E[a^L](1 - E[a^L])E[e]$ for the logistic activation function, where $E[e]$ is the mean error across all output neurons and teaching-input patterns in the training set.[4]

We should note that for specific activation functions, alternative formulations to Equation (17) can be derived. For the logistic activation function $f'(net_j^l) = a_j^l(1 - a_j^l)$. So, we can directly use the more precise Equation (14), which can be rewritten as

$$\begin{aligned} E[\delta^{l+1}] &\cong E[a^{l+1}(1 - a^{l+1})]\mu^{l+1}n^{l+2}E[\delta^{l+2}] \\ &\cong E[a^{l+1}](1 - E[a^{l+1}])\mu^{l+1}n^{l+2}E[\delta^{l+2}]. \end{aligned} \tag{18}$$

[4]In the initial conditions for the forward and backward recursions, we have ignored, without loss of generality, the fact that the expected input $E[x]$ and the expected error $E[e]$ may differ from neuron to neuron. This is because with all weights in a layer having the same mean in the first training epoch, any neuron-to-neuron differences in $E[x]$ and $E[e]$ are averaged out at the level of first hidden layer and penultimate hidden layer, respectively.

Similarly, for the hyperbolic tangent activation function $f'(net_j^l) = 1 - (a_j^l)^2$ and, so, we have

$$\begin{aligned} E[\delta^{l+1}] &\cong E[1-(a^{l+1})^2]\mu^{l+1}n^{l+2}E[\delta^{l+2}] \\ &\cong (1-E[a^{l+1}]^2)\mu^{l+1}n^{l+2}E[\delta^{l+2}]. \end{aligned} \tag{19}$$

With similar arguments we have that $\mathrm{E}[\Delta w_{ij}^l]$ does not depend on either i or j and, so, we term it $\mathrm{E}[\Delta w^l]$ hereafter. Then

$$E[\Delta w^l] = \eta E[a^l \delta^{l+1}] \cong \eta E[a^l]E[\delta^{l+1}], \tag{20}$$

on the assumption that a^l and δ^{l+1} are weakly correlated.

In conjunction with the results of the previous recursions for $E[\delta^{l+1}]$ and $E[a^l]$, Equation (20) allows one to estimate the *expected initial gradients* in weight-space for every layer in the network, as a function of the choice of the mean initial weights μ^l, the depth of the network L, the number of neurons in each layer n^l, the characteristics of the problem with its encoding (represented by $E[x]$ and $E[e]$) and, of course, the learning rate η.

4.2. A Sufficient Condition for Vanishing Gradients in Feed-forward Networks

To obtain a more rigorous theory and one that would also apply to phases of training other than the very beginning, in this section we adapt some of the theoretical results obtained in [26] for recurrent networks to the case of feed-forward neural networks. For simplicity we considered only the case of networks with equally sized hidden layers, i.e., where the weight matrices are square.

Equations (1) and (2) can be combined and re-written in vector notation as:

$$\mathbf{a}_l = f(\mathbf{W}_{l-1}\mathbf{a}_{l-1} + \mathbf{b}_{l-1}) \tag{21}$$

where subscripts represent layers and quantities in boldface represent vectors and matrices.

If f is a sigmoid, it is invertible (being monotonic). So, we can apply its inverse function, f^{-1}, to both sides of Equation (21) obtaining

$$f^{-1}(\mathbf{a}_l) = f^{-1}(f(\mathbf{W}_{l-1}\mathbf{a}_{l-1} + \mathbf{b}_{l-1})).$$

Simplifying both sides produces

$$\mathbf{net}_l = \mathbf{W}_{l-1}\mathbf{a}_{l-1} + \mathbf{b}_{l-1}.$$

Finally, this leads to

$$\mathbf{net}_l = \mathbf{W}_{l-1}f(\mathbf{net}_{l-1}) + \mathbf{b}_{l-1} \tag{22}$$

which is the parametrization used in [26, Equation (2)] for recurrent neural networks.

Adapting the steps in [26, Equations (3)–(5)], computing the gradient of the cost function, $\mathcal{E}$, to be minimised with respect to a generic weight θ_l from layer l, and applying the chain rule we obtain:

$$\frac{\partial \mathcal{E}}{\partial \theta_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{net}_L}\frac{\partial \mathbf{net}_L}{\partial \mathbf{net}_l}\frac{\partial \mathbf{net}_l}{\partial \theta_l}, \tag{23}$$

where

$$\frac{\partial \mathbf{net}_L}{\partial \mathbf{net}_l} = \frac{\partial \mathbf{net}_L}{\partial \mathbf{net}_{L-1}}\frac{\partial \mathbf{net}_{L-1}}{\partial \mathbf{net}_{L-2}}\cdots\frac{\partial \mathbf{net}_{l+1}}{\partial \mathbf{net}_l}, \tag{24}$$

L is the index of the output layer and $\frac{\partial \mathcal{E}}{\partial \mathbf{net}_L}$ is the error gradient at the output layer that we want to back propagate. Then, using Equation (22), we obtain

$$\frac{\partial \mathbf{net}_k}{\partial \mathbf{net}_{k-1}} = \mathbf{W}_{k-1}^T diag(f'(\mathbf{net}_{k-1}))$$

and

$$\frac{\partial \mathbf{net}_l}{\partial \theta_l} = f(\mathbf{net}_{l-1}).$$

Substituting these expressions in Equation (23), we obtain

$$\frac{\partial \mathcal{E}}{\partial \theta_l} \quad = \quad \frac{\partial \mathcal{E}}{\partial \mathbf{net}_L} \left(\prod_{k=l+1}^{L} \mathbf{W}_{k-1}^T \mathit{diag}(f'(\mathbf{net}_{k-1})) \right) f(\mathbf{net}_{l-1}). \tag{25}$$

This equation makes it clear that whether the gradients grow or shrink as l decreases (as we move from the output to the input layer) depends on whether each additional factor of the form $\mathbf{W}_{k-1}^T \mathit{diag}(f'(\mathbf{net}_{k-1}))$ produces an amplification or a contraction.

To explore this, let us take the (operator induced) ∞ *matrix norm*, $||A|| = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$, of the term in bracket in Equation (25). Because this norm is submultiplicative, we have that

$$\left|\left| \prod_{k=l+1}^{L} \mathbf{W}_{k-1}^T \mathit{diag}(f'(\mathbf{net}_{k-1})) \right|\right| \quad \leq \quad \prod_{k=l+1}^{L} \left|\left|\mathbf{W}_{k-1}^T\right|\right| \, ||\mathit{diag}(f'(\mathbf{net}_{k-1}))|| \, . \tag{26}$$

By the definition of ∞ norm and $\mathit{diag}(f'(\mathbf{net}_{k-1}))$, $||\mathit{diag}(f'(\mathbf{net}_{k-1}))||$ is the maximum element of $\mathit{diag}(f'(\mathbf{net}_{k-1}))$. Also, if we consider the logistic activation function, each such element is in the interval $[0, 1/4]$. From these two facts, it follows that

$$||\mathit{diag}(f'(\mathbf{net}_{k-1}))|| \leq \frac{1}{4}. \tag{27}$$

Combining this result with Equation (26), we obtain

$$\left|\left| \prod_{k=l+1}^{L} \mathbf{W}_{k-1}^T \mathit{diag}(f'(\mathbf{net}_{k-1})) \right|\right| \quad \leq \quad \left(\frac{1}{4}\right)^{L-l} \prod_{k=l+1}^{L} ||\mathbf{W}_{k-1}^T||. \tag{28}$$

This result implies that if for all k

$$||\mathbf{W}_{k-1}^T|| \leq 4\gamma, \tag{29}$$

where γ is a constant strictly smaller than 1, i.e., if the 1-norm[5] of all weight matrices in the network is strictly less than 4, then

$$\left|\left| \prod_{k=l+1}^{L} \mathbf{W}_{k-1}^T \mathit{diag}(f'(\mathbf{net}_{k-1})) \right|\right| \quad \leq \quad \gamma^{L-l}, \tag{30}$$

i.e., *gradients decrease exponentially* as they are propagated further and further back in the network.

We should note that Equation (29) is a *sufficient condition for gradients to vanish in a feed-forward neural network with logistic activation function.* This condition is related to the condition on the spectral radius $\rho < 4$ introduced in [26] and mentioned in Section 3.1, which applies to recurrent neural networks. This is because for any operator induced norm (including the norm ∞ used here), $\rho(A) \leq ||A||$. So, our condition is weaker than that of [26].

4.3. Vanishing Gradients Problem in the Early Stages of Training

In this section we use the theoretical results obtained in Sections 4.1 and 4.2 to study the potential presence of the vanishing gradients problem at the beginning of the learning process.

[5]The ∞ norm of the transpose of a matrix corresponds to the 1-norm of the matrix.

4.3.1. Evidence from the Expected Gradients Model

If we unwind the recursion in Equation (17) and substitute the result in Equation (20), we obtain

$$E[\Delta w^l] \cong \eta E[a^l] f'\left(\mu^l\left(1+n^l E[a^l]\right)\right)\mu^{l+1} n^{l+2} f'\left(\mu^{l+1}\left(1+n^{l+1}E[a^{l+1}]\right)\right)\mu^{l+2}n^{l+3}.... \quad (31)$$

Of course, at this stage, we know nothing about the cumulative effect on the end result of the approximation made in the derivations in Section 4.1. However, qualitatively this equation gives as the following useful indications that we will later verify both theoretically and empirically:

1. **$|\mu^l| < 1$ exponentially reduces errors, $|\mu^l| > 1$ exponentially amplifies them**: In Equation (31), if $|\mu^l| > 1$ for all l, then $|\mu^l \times \mu^{l+1} \times \cdots|$ grows exponentially. Conversely, if $|\mu^l| < 1$, $|\mu^l \times \mu^{l+1} \times \cdots|$ decreases exponentially. More precisely, if $\bar{\mu}^l$ is the geometric mean of $|\mu^l|$, $|\mu^{l+1}|$, etc., then $|\mu^l \times \mu^{l+1} \times \cdots| = (\bar{\mu}^l)^{L-l}$.

   Given this observation, if $\mu^l = 0$ for all l, we have that $\bar{\mu}^l = 0$. As a result, $E[\Delta w^l] \cong 0$ in every layer. So, *the traditional choice of $\mu^l = 0$ may be an initial cause of the vanishing gradients problem.* If this prediction is correct, we should expect better initial error backpropagation (and potentially an improvement on the vanishing gradient problem) if we initialize the weights using μ^l sufficiently different from zero.
2. **Logistic-function derivatives exponentially reduce back-propagated errors**: With the logistic activation function we have that $f' \leq 0.25$. So, in Equation (31), for any l, we have that $f'\left(\mu^l\left(1+n^lE[a^l]\right)\right) \leq 0.25$. The presence of $L-l$ terms of this form in the equation implies that, together, they scale down the gradient by *at least* a factor $(0.25)^{L-l}$, as errors are back propagated. More precisely, if $\bar{f'}^l$ is the geometric mean of $f'\left(\mu^l\left(1+n^lE[a^l]\right)\right)$, $f'\left(\mu^{l+1}\left(1+n^{l+1}E[a^{l+1}]\right)\right)$, etc., then gradients will be scaled down by a factor $(\bar{f'}^l)^{L-l}$.

   Both this observation and directly Equation (31) suggest that to have some hope of successfully back-propagating errors initially, the choice of values of μ^l for all l, must be such as to ensure that $E[a^l]$ (via Equation (12)), $f'\left(\mu^l\left(1+n^lE[a^l]\right)\right)$, $f'\left(\mu^{l+1}\left(1+n^{l+1}E[a^{l+1}]\right)\right)$, etc. are also sufficiently different from zero. In other words, neurons must not be in saturation.
3. **The wider a network, the better it back-propagates errors**: Equation (31) makes it clear that terms of the form n^l contribute to countering the above-mentioned error-reducing tendency, as, typically, $n^l > 1$, and, so, $n^l \times n^{l+1} \times \cdots$ grows exponentially. More precisely, if $\bar{n}^l$ is the geometric mean of $n^l \times n^{l+1} \times \cdots$, then the error amplification produces by the n terms grows like $(\bar{n}^l)^{L-l}$.

   Interestingly, all this implies that *wider networks initially back propagate errors better than narrower ones*, everything else being equal.

Given these observations, we can see that Equation (31) effectively predicts that errors will be back-propagated without attenuation if the three exponential relationship discussed above balance each other in such a way that

$$(\bar{\mu}^l \times \bar{f'}^l \times \bar{n}^l)^{L-l} \geq 1. \quad (32)$$

which, for a generic l, implies

$$\bar{\mu}^l \times \bar{f'}^l \times \bar{n}^l \geq 1. \quad (33)$$

Because the product of geometric means is the geometric mean of the products, a simple choice of μ^l and n^l that guarantees this balance is one where

$$|\mu^l|\, n^l\, f'(\mu^l(1+n^lE[a^l])) \geq 1. \quad (34)$$

Solving this equation for $|\mu^l|$, produces

$$|\mu^l| \geq \frac{1}{n^l\, f'(\mu^l(1+n^lE[a^l]))}. \quad (35)$$

Because $f' \leq 0.25$, one might think that, in principle, Equation (35) would be satisfied if $|\mu^l| \geq \frac{4}{n^l}$. However, this is not the case as $f' = 0.25$ only when the net input is zero. So,

$f'(\mu^l(1+n^l E[a^l])) = 0.25$ only if $\mu^l = 0$, which as we know makes all gradients zero in expectation. So, we need to set:

$$\mu^l = \frac{d}{n^l}. \tag{36}$$

with $|d|$ sufficiently bigger than 4 for every l. However, in general, there exist also an upper limit on $|d|$, as too large a value would lead to large $|\mu^l|$ which would saturate the neurons in either direction, leading to $f' \cong 0$ and $E[\Delta w^l] \cong 0$.

Within these limits for $|d|$, *the choice of μ^l suggested in Equation (36) might remove a potential initial cause for the vanishing gradient problem and, in fact, might even produce initial gradients that grow (rather than shrink) as we move from the output layer towards the input layer.*

4.3.2. Evidence from the Matrix Model

Because most initialization methods draw initial weights from a Normal distribution (e.g., see Section 5.3.2), focusing on the very first iteration of training, we can assume that the matrices $\mathbf{W}_k$ are random with i.i.d. entries drawn from a normal distribution $\mathcal{N}(\mu,\ \sigma^2)$. Let us focus on the ∞ norm of a generic one, which we will call $\mathbf{W}$. By definition

$$||\mathbf{W}|| = \max_{1\le i\le n} \sum_{j=1}^{n} |w_{ij}| \tag{37}$$

where $w_{ij} \sim \mathcal{N}(\mu,\ \sigma^2)$. The terms $|w_{ij}|$ are i.i.d. according to $|w_{ij}| \sim \mathcal{F}(\mu_f, \sigma_f)$, where $\mathcal{F}$ is a *folded normal distribution* with mean $\mu_f = \sigma\sqrt{\frac{2}{\pi}}e^{-\mu^2/2\sigma^2} + \mu(1-2\Phi(-\frac{\mu}{\sigma}))$ and standard deviation $\sigma_f = \sqrt{\mu^2+\sigma^2-\mu_f^2}$, where Φ is the normal cumulative distribution function. As a result, for large enough numbers of neurons, n, by the *central limit theorem*, the sum of the absolute values of the elements in each row in Equation (37), $S_i = \sum_{j=1}^{n} |w_{ij}|$, follows the distribution $S_i \sim \mathcal{N}(\mu_s,\ \sigma_s^2)$ where $\mu_s = n\mu_f$ and $\sigma_s = \sqrt{n}\,\sigma_f$. So, $s_i = \frac{S_i-\mu_s}{\sigma_s}$ is a standard normal variable. Extreme value theory (in particular, the Fisher-Tippett-Gnedenko theorem) indicates that

$$E\left[\max_{1\le i\le n} s_i\right] \approx (1-\gamma)\Phi^{-1}(1-1/n) + \gamma\Phi^{-1}(1-1/(e\,n)), \tag{38}$$

where $\Phi^{-1}(.)$ is the inverse of Φ, i.e., the Probit function, and $\gamma \approx 0.5772156649$ is the Euler-Mascheroni constant. Thus,

$$\begin{aligned}
&E\left[||\mathbf{W}||\right] \\
&= E\left[\max_{1\le i\le n} S_i\right] \\
&= \mu_s + \sigma_s E\left[\max_{1\le i\le n} s_i\right] \\
&\approx \mu_s + \sigma_s\left((1-\gamma)\Phi^{-1}(1-1/n) + \gamma\Phi^{-1}(1-1/(e\,n))\right) \\
&= n\mu_f + \sqrt{n}\,\sigma_f\left(\left((1-\gamma)\Phi^{-1}(1-1/n) + \gamma\Phi^{-1}(1-1/(e\,n))\right)\right) \\
&= n\left(\sigma\sqrt{\frac{2}{\pi}}e^{-\mu^2/2\sigma^2} + \mu\left(1-2\Phi\left(-\frac{\mu}{\sigma}\right)\right)\right) \\
&+ \sqrt{n}\sqrt{\mu^2+\sigma^2-\left(\sigma\sqrt{\frac{2}{\pi}}e^{-\mu^2/2\sigma^2} + \mu\left(1-2\Phi\left(-\frac{\mu}{\sigma}\right)\right)\right)^2} \\
&\quad \times\left(\left((1-\gamma)\Phi^{-1}(1-1/n) + \gamma\Phi^{-1}(1-1/(e\,n))\right)\right).
\end{aligned} \tag{39}$$

The accuracy of this formula has been verified for many values of n, σ and μ, by creating 100 random matrices for each setting and computing the mean and standard error of the mean their norms. Some representative results will be reported in Section 5.4 (Table 3). However, in all cases the estimates from Equation (39) are correct within experimental errors.

As an approximation, we can interpret our sufficient condition for gradients to vanish in Equation (29) as $E[||\mathbf{W}||] < 4$. Substituting Equation (39) in it yields:

$$\begin{aligned} 4 \quad > \quad & n\left(\sigma\sqrt{\frac{2}{\pi}}e^{-\mu^2/2\sigma^2} + \mu\left(1-2\Phi\left(-\frac{\mu}{\sigma}\right)\right)\right) \\ + \quad & \sqrt{n}\sqrt{\mu^2+\sigma^2-\left(\sigma\sqrt{\frac{2}{\pi}}e^{-\mu^2/2\sigma^2} + \mu\left(1-2\Phi\left(-\frac{\mu}{\sigma}\right)\right)\right)^2} \\ & \times\left((1-\gamma)\Phi^{-1}(1-1/n)+\gamma\Phi^{-1}(1-1/(e\,n))\right). \end{aligned} \tag{40}$$

While this is a complex formula, it is easy to study theoretically in special cases and it is also easy to implement numerically. Therefore, *it makes it possible to study under which combinations of μ, n and σ the vanishing gradient problem is guaranteed to be present at the beginning of the training process.*

Let us consider Equation (40) in two cases: (1) the traditional initialization with $\mu = 0$ and any σ, and (2) the case $|\mu| \gg \sigma$ which practically guarantee that the entries in $\mathbf{W}$ are either all positive or all negative:

1. In the $\mu = 0$ *case*, Equation (40) simplifies to

$$4 \quad > \quad \sigma\left[n\sqrt{\frac{2}{\pi}} + \sigma\sqrt{n(1-2/\pi)}\left((1-\gamma)\Phi^{-1}(1-1/n)+\gamma\Phi^{-1}(1-1/(e\,n))\right)\right]. \tag{41}$$

   If the typical value of $\sigma = 0.1$ is used to initialize a network, this is satisfied for any network with $n < 40$ neurons in the hidden layers. *In the typical conditions $\mu = 0$, $\sigma = 0.1$ and $n < 40$, gradients will be* ***guaranteed*** *to be vanishingly small at the first epoch of training.*

2. If we now consider the $|\mu| \gg \sigma$ *case*, we have that terms of the form $e^{-\mu^2/2\sigma^2}$ are approximately 0, while $\Phi\left(-\frac{\mu}{\sigma}\right) \approx 0$ if $\mu > 0$ and $\Phi\left(-\frac{\mu}{\sigma}\right) \approx 1$ otherwise. So, $1 - 2\Phi\left(-\frac{\mu}{\sigma}\right) = |\mu|$ and Equation (40) becomes:

$$4 \quad > \quad n|\mu| + \sqrt{n}\,\sigma\left((1-\gamma)\Phi^{-1}(1-1/n)+\gamma\Phi^{-1}(1-1/(e\,n))\right). \tag{42}$$

   In this case, we see how $|\mu|$ and σ each contribute linearly to $E[||\mathbf{W}||]$, with $|\mu|$ having a much larger coefficient than σ. In any case, it is clear that if one set $|\mu| > 4/n$, the first term of the equation is, on its own, bigger than 4 and, so, the weight matrix would *not* satisfy the sufficient condition for the vanishing gradients problem to occur.

   In practice, when $|\mu| \gg \sigma$, for a typical value of $\sigma = 0.1$, the sufficient condition for initially vanishing gradients is only satisfied with particular choices of n and $|\mu|$, such as $n \leq 11$ and $|\mu| = 0.3$, $n \leq 8$ and $|\mu| = 0.4$, $n \leq 7$ and $|\mu| = 0.5$, etc. That is, *when $|\mu| \gg \sigma$ the sufficient condition for the vanishing gradients problem can be met only for very narrow networks*, suggesting that in general the use of $|\mu| \neq 0$ may be beneficial in relation to ensuring gradients do not vanish.

We should note a special instance of the $|\mu| \gg \sigma$ case is when $|\mu| \neq 0$ (but finite) and $\sigma \to 0$. In this case, from Equations (40) and (42), *the sufficient condition for the vanishing gradient problem to initially occur simplifies to $n \times |\mu| < 4$ for the logistic activation function.* To avoid this condition then one should set $\mu = d/n$ where $|d| \geq 4$ (at least), which is exactly the same result we obtained independently with our approximate model of expected gradients at the end of Section 4.3.1. This suggests that the approximations we introduced to develop the theory in Sections 4.1) did not render its conclusions useless.

Table 1: Information on the data sets used in this study

Problem	Inst.	NF	Classes	IM	$E[x]$ after normalization
Mux	128	6	2	0.0000	0.500000
Iris	150	4	3	0.0000	0.614489
Wine	178	13	3	0.0125	0.562137
Authorship	841	70	4	0.0833	0.252697
BCW	699	9	2	0.0963	0.289938
DNA	3186	180	3	0.0776	0.252671

4.4. New Initialization Method

While the analyses in Sections 4.3.1 and 4.3.2 identified the need to set $\mu \neq 0$ and inversely proportional to the number of neurons in the hidden layers when σ is small, they did not specify whether the proportionality constant, d, should be positive or negative. Fortunately, this is easily determined by reasoning on the issue or with different forms of numerical simulations (including the results in Section 5.2).

For instance, we can iterate the approximate forward-propagation equation (Equation (12), starting from typical values of $E[x]$ such as the ones reported in Table 1, when $\mu^l = d/n^l$ for a variety of positive and negative values of d such that $|d| \geq 4$. Doing so, one finds that, when $d \geq 4$ (positive), the sequence $E[a^l]$ rapidly converges to a value very close to 1 (saturation), resulting in all the f' terms in Equation (31) being near 0, leading gradients to vanish. On the contrary, when $d \leq -4$ (negative), with the same initial conditions, the sequence $E[a^l]$ is oscillating (with period 2) and converges to an asymptote, $E[a^\infty]$ (that does not depend on $E[x]$), that neither completely saturates nor completely inhibits the neurons. For instance, $E[a^\infty] = 0.218544$ for $d = -8$ and $E[a^\infty] = 0.133642$ for $d = -4$.

Because of this, for networks with logistic activation function, to avoid initially vanishing gradients, it seems reasonable to set μ^l according to Equation (36) with $d \leq -4$ so that $\mu^l < 0$, for all hidden layers.

Finally, we should note that in networks with narrow layers (or with very few input neurons) Equation (36) results in very large (in absolute value) values of μ^l. We think it is reasonable avoid such extremes and set a lower limit, μ_{min}, for μ^l.

Based on the analysis and discussion in this and the previous sections, we propose the following *New Initialization Method (NIM)* for MLPs using the logistic activation function:

$$w^l \sim \mathcal{N}\left(\max\left(\mu_{min}, \frac{d}{n^l}\right),\ \sigma^2\right), \tag{43}$$

where $\sigma = 0.1$ as standard, $\mu_{min} = -1$ somewhat arbitrarily and $d = -8$ as this value combats the decline in the magnitude of gradients better than the threshold value, $d = -4$.

5. Experimental Results

5.1. Benchmark Problems

As indicated in Section 1, we used six classification problems in our experiments: the iris, wine, analcat data authorship (authorship), DNA, breast cancer (BCW) and 4-bit multiplexer classification problems from the Penn Machine Learning Benchmarks [39, 40]. Information of these data sets — number of instances (Inst.), number of features (NF), number of class (Classes), imbalance metric (IM) and mean input $E[x]$ (after $[0, 1]$ normalization) — is provided in Table 1. IM (from [40]) shows the level of class imbalance of the instances in each data set.

5.2. Empirical Estimation of Initial Gradients in Deep Networks

In order to verify our predictions that negative values of μ could be beneficial to initially counter the vanishing gradient problem, we did the following experiment: (1) we initialized deep networks of different widths (number of neurons in each hidden layer) with distributions of weights having a range of different means, μ, both positive and negative (the same μ for all layers); (2) the weight distributions were Gaussian with standard deviation 0.1 in all cases; (3) we performed one step of forward propagation of the training set and one of error back-propagation using Equations (1)–(7), recording the mean $|\Delta w|$ in all layers. We repeated this process for the different problems in Table 1, testing 30 different initial random seeds for each problem and averaging the corresponding results.

As an illustrative example (very similar results were obtained for all other problems and depths), Figure 1 shows the mean $|\Delta w|$ for the input layer and several other layers in the network, for the authorship problem in networks with 15 hidden layers and a varying number of neurons, $n \in \{30, 40, 50, 60, 70, 80, 90, 100\}$, in the hidden layers. Figure 2 represents the same quantities at a finer resolution ($n = 30, 31, \cdots, 99, 100$) using heat maps.

As one can see in Figures 1.1–1.4 and 2.1–2.4, for most layers, the magnitude of the initial weight changes, $|\Delta w|$, is non-zero for negative values of μ. For such layers, we see how the optimal μ is negative and is inversely proportional to the number of neurons in a layer, as we postulated in Section 4 (for reference in Figure 2 we plotted the line $\mu = -\frac{8}{n}$). However, as we move closer to the output layer, a phase transition occurs. First, we see the emergence of a second optimum near $\mu = 0$ (or for slightly positive μ), as illustrated in Figures 1.5 and 2.5. This, then, becomes the only optimum as we move further towards the output layer, as illustrated in Figures 1.6 and 2.6.

From the figures we also see that the initial gradients in the input layer and several hidden layers near it are very sensitive to the specific value of μ (that is, the values of $|\Delta w|$ decay quite rapidly as we move away from the optimal μ). On the contrary, a much broader set of μ values (both positive and negative) provide appreciable gradients in hidden layers near the output layer.

This confirms the prediction of our theoretical analysis that $\mu = 0$ may lead to initially vanishing gradients. The data show that while the standard practice of setting $\mu = 0$ gives good initial gradients in the output layer and hidden layers near it, it results in near-zero gradients in the input layer and many hidden layers. On the contrary, selecting a negative value of μ that gives a good initial gradient in the input layer (such as $\mu = -0.1$ for the authorship problem) produces non-zero initial gradients everywhere in the network.

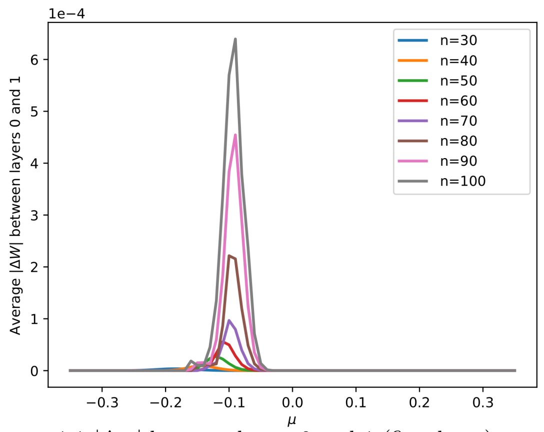
5.3. Training Deep Networks after our New Initialization

In both the analysis in Section 4 and its empirical corroboration in Section 5.2, we focused on initial gradients. The question we now need to address is to what degree initializing weights with a optimal negative μ solves the vanishing gradient problem when we continue training the network with the SBP algorithm for many interactions. In this section, we address this question.

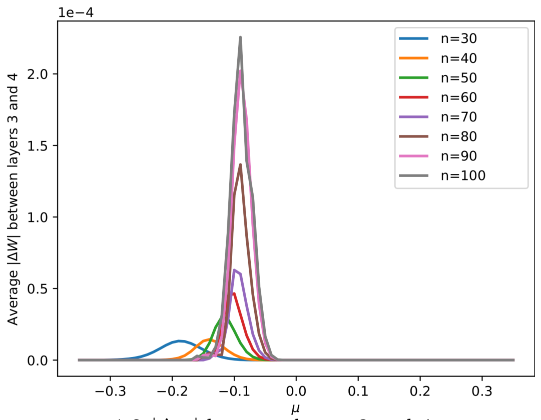
5.3.1. Network Structures and Meta-parameters

In these experiments, we also used the problems in Table 1 to verify the benefits of our initialization method over multiple training epochs.

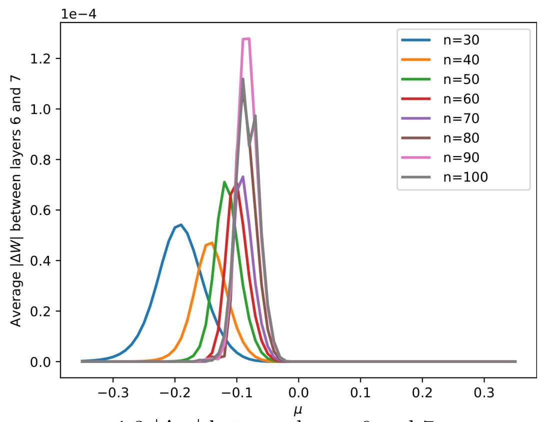
The network structures (number of neurons in the layers and number of layers) are shown in Table 2. These were chosen so as to represent both relatively compact and relatively big deep networks. The input layer of each network consisted of as many neurons as the features in the problem. The output layer of each network had as many neurons as the number of classes in that problem. All other layers had an equal number of neurons, n. All networks were trained using a learning rate of $\eta = 0.25$. For each problem we used two network structures to test whether the initialization method is sensitive to network size: one with 10 hidden layers of 10 neurons each (having of the order of 1,000 weights and 100 neurons) and one with 15 hidden layers of 100 neurons each (with approximately 150,000 weights and 1,000 neurons).



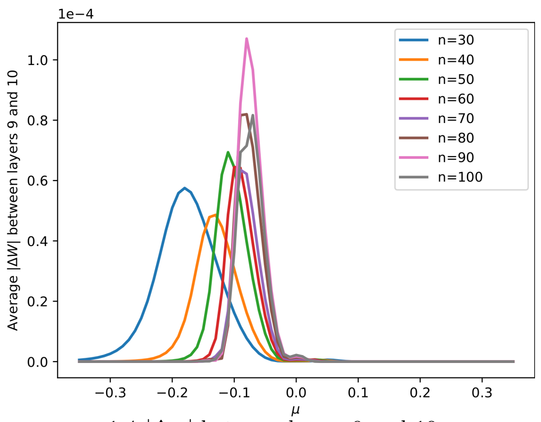
1.1 $|\Delta w|$ between layers 0 and 1 (first layer).



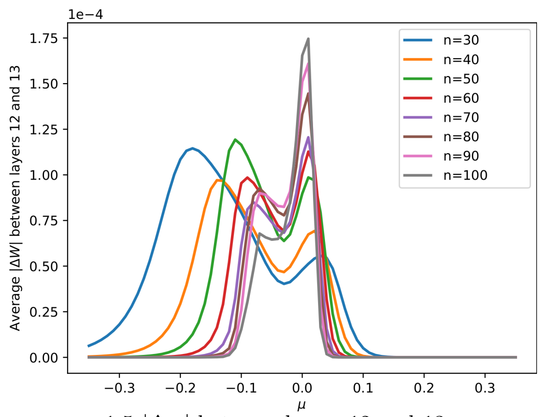
1.2 $|\Delta w|$ between layers 3 and 4.



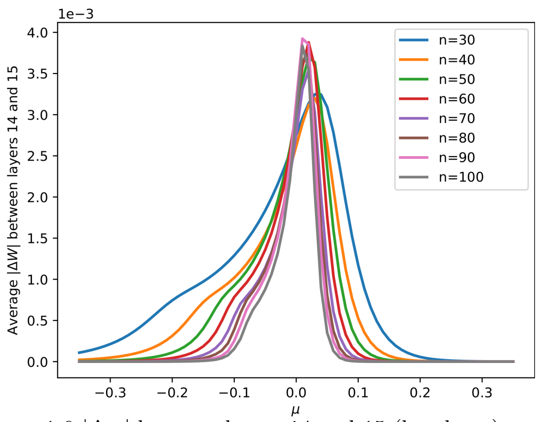
1.3 $|\Delta w|$ between layers 6 and 7.



1.4 $|\Delta w|$ between layers 9 and 10.

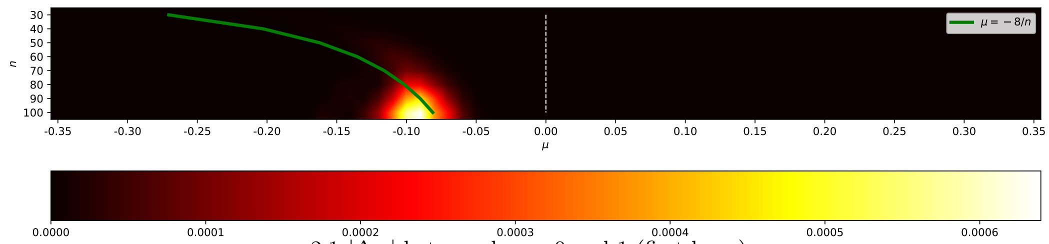


1.5 $|\Delta w|$ between layers 12 and 13.

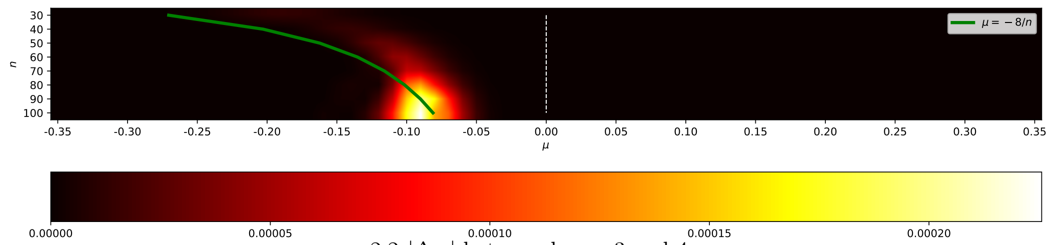


1.6 $|\Delta w|$ between layers 14 and 15 (last layer).

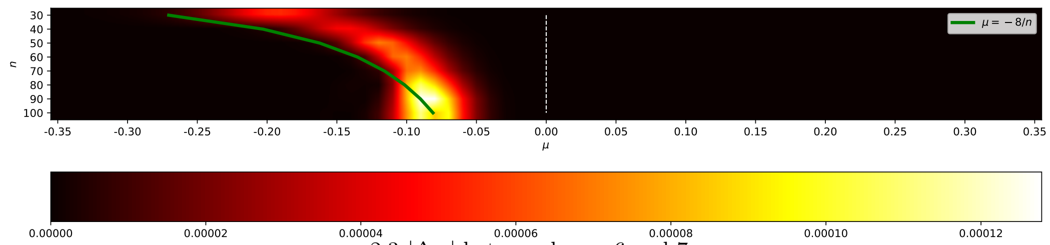
Figure 1: Mean $|\Delta w|$ for the authorship problem for a network with 15 hidden layers and different numbers of neurons, n, in several representative layers.



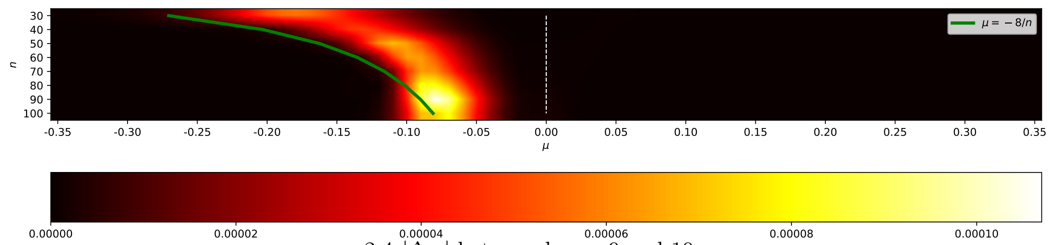
2.1 $|\Delta w|$ between layers 0 and 1 (first layer).



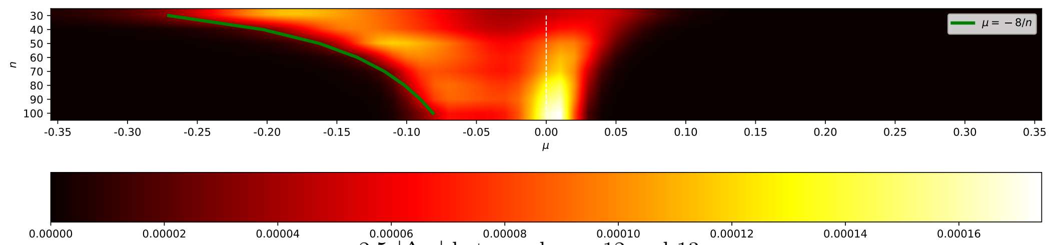
2.2 $|\Delta w|$ between layers 3 and 4.



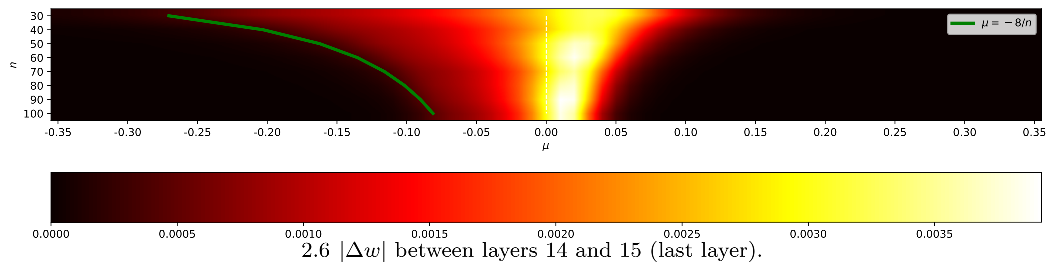
2.3 $|\Delta w|$ between layers 6 and 7.



2.4 $|\Delta w|$ between layers 9 and 10.



2.5 $|\Delta w|$ between layers 12 and 13.



2.6 $|\Delta w|$ between layers 14 and 15 (last layer).

Figure 2: Heat maps representing the mean $|\Delta w|$ for the authorship problem for a network with 15 hidden layers and different numbers of neurons, n, in several representative layers.

Table 2: Network structures adopted for different test problems used for comparing different forms of initialization

Problem	Number of hidden layers	Number of neurons in each hidden layer
Iris	10	10
Wine	10	10
Authorship	10	10
DNA	10	10
Mux	10	10
BCW	10	10
Iris	15	100
Wine	15	100
Authorship	15	100
DNA	15	100
Mux	15	100
BCW	15	100

5.3.2. Initialization Methods Compared

Except with NIM, when a network is initialized with random weights, the values are usually assigned using a zero-mean normal distribution with a small standard deviation such as 0.01, 0.05, 0.1 and 0.2. To determine if NIM was competitive with other methods, we pitted it against a Standard Initialization Method (*SIM*), Glorot and Benjo's method [36] and Kumar's method [38] (both reviewed in Section 3). So, we compared the following distributions of initial weights:

$$\begin{aligned} \textit{SIM}: &\quad W^l \sim \mathcal{N}(0,\ 0.01), \\ \textit{Glorot}: &\quad W^l \sim \mathcal{N}(0, 2/(n^l + n^{l+1})), \\ \textit{Kumar}: &\quad W^l \sim \mathcal{N}(0,\ 12.96/n^l), \\ \textit{NIM}: &\quad W^l \sim \mathcal{N}(\max(-1, -8/n^l),\ 0.01), \end{aligned} \tag{44}$$

where n^l is the number of neurons (including bias) in layer l.

5.3.3. Results

Figures 3 and 4 show plots of the average cross-entropy loss and test- and training-set accuracy when training networks initialized with the four initialization methods mentioned above for the problems in Table 1. Results are medians of 30 independent runs.[6]

As shown in the figures, in SIM- and Glorot-initialized networks both the median loss and the median accuracy get stuck to sub-optimal values. This is because, with such initializations, SBP could not train the network to solve *any* of the test problems in *any* of the independent runs. On the contrary, with NIM and Kumar initialization, the standard back-propagation can train the networks successfully in nearly all problems and in nearly all independent runs.[7] A key difference between NIM and Kumar is that with NIM both the loss and the accuracy start improving straight away, while with Kumar it typically takes several thousand epochs before loss and/or accuracy improvements start being appreciable.

Overall, the results corroborate our theoretical analyses and the hypothesis that focusing on initial gradients is an effective approach to understanding and dealing with the vanishing gradient problem in deep neural networks with logistic activation function. They demonstrate that *when starting from good initial conditions* (i.e., from a state where gradients are not vanishingly small everywhere in a network) *even a normal gradient descent by SBP can be extremely efficient and effective at training deep networks.*

[6]Please note, we do not report a *testing* accuracy for the Mux problem as this is identical to the training accuracy given that this problem requires learning a full truth table.

[7]The only exception is Mux for networks with 15 hidden layers where, with both NIM and Kumar initialization, there was good progress but the 10,000 training epochs used here were insufficient to complete the training.

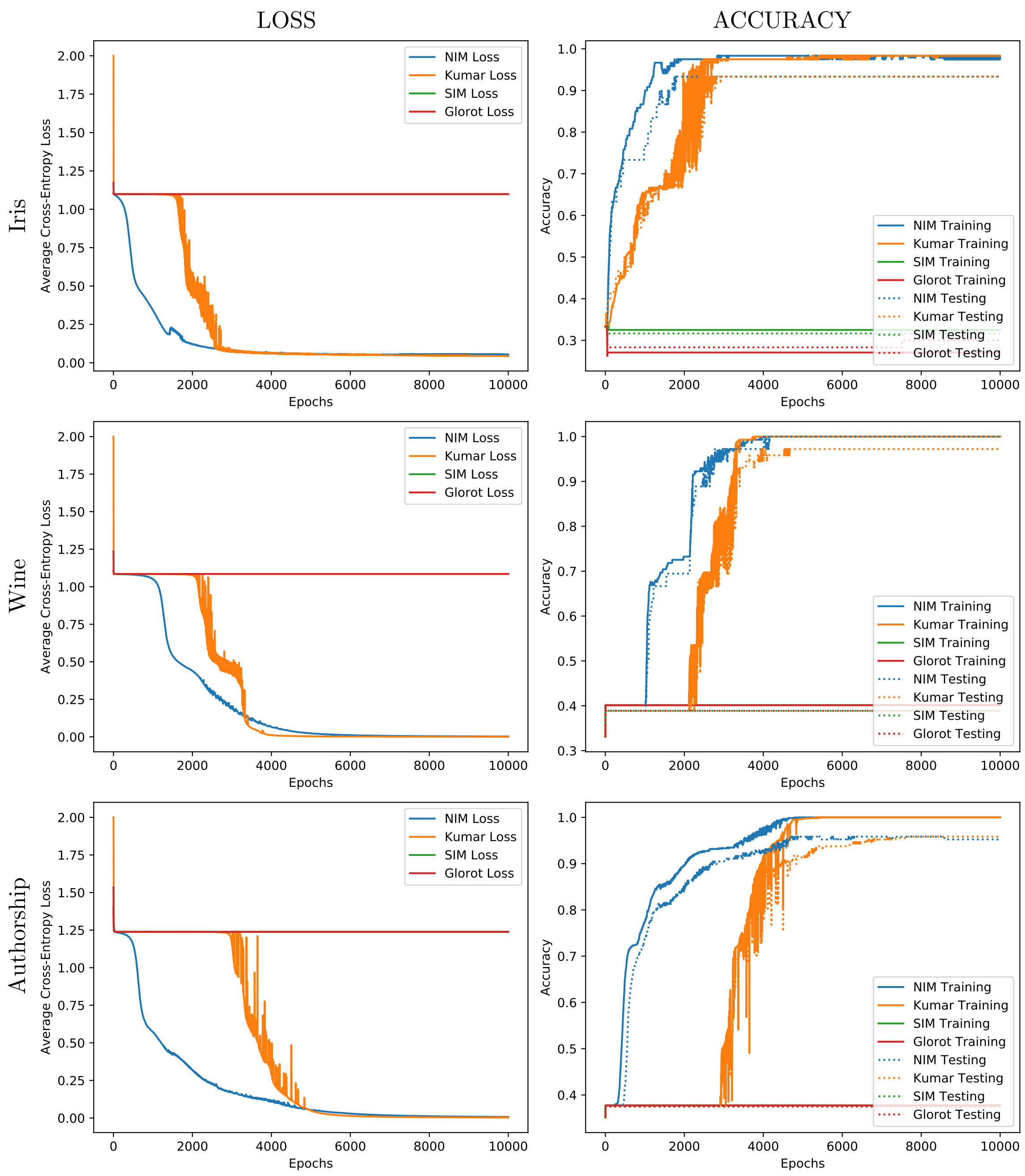


Figure 3: Results for our six test problems when the *SIM*, *Glorot*, *Kumar* and *NIM* methods are used to initialize networks with 10 hidden layers and 10 neurons in each hidden layer.

5.4. Are Sufficient Conditions Sufficient?

In the previous section we have seen that NIM presents remarkable advantages in relation to effectively and efficiently training deep networks over other initialization methods. However, we also found that Kumar's method was successful, albeit requiring a longer training process. On the contrary, SIM and Glorot's methods were completely unsuccessful. In this section we want to try and understand the reasons for these results.

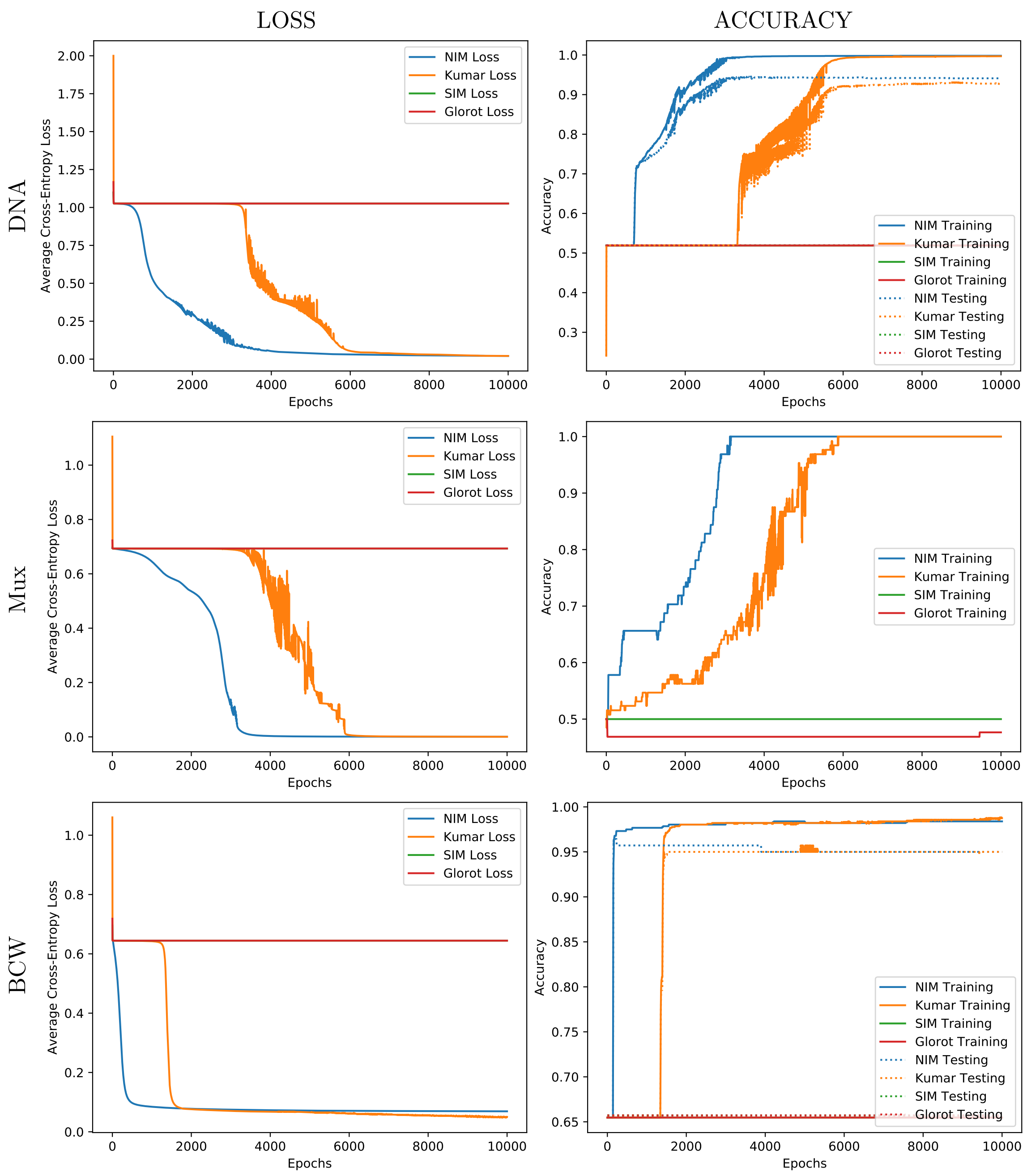


Figure 3: (Cont.) Results for our six test problems when the *SIM*, *Glorot*, *Kumar* and *NIM* methods are used to initialize networks with 10 hidden layers and 10 neurons in each hidden layer.

5.4.1. Initial Weight Matrix Norms

We start by looking at the norms of the initial weight matrices created by different initialization algorithms (see Table 3), to see if the theoretical conditions on $||\mathbf{W}||$ for the vanishing gradient problem to occur are actually sufficient to explain the success or otherwise in training networks when using different initialization algorithms. What we mean by this is the following: while $||\mathbf{W}|| < 4$ is only a sufficient condition, one might postulate that beyond a certain threshold or phase-transition value for the initial $||\mathbf{W}||$, SBP would start from an initial state where gradients are not vanishingly small, which then would lead to successfully training the network. Conversely

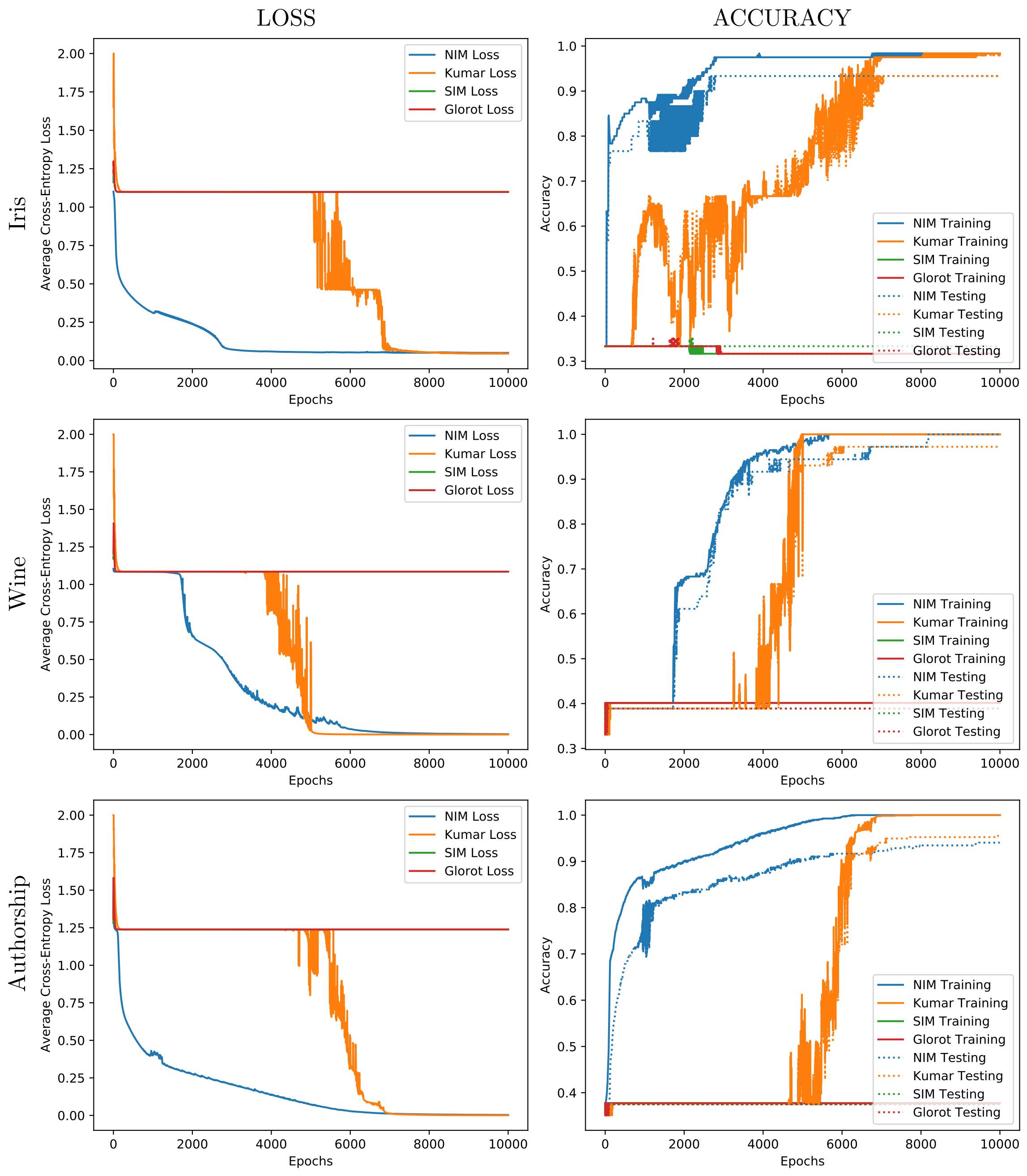


Figure 4: As in Figure 3 but for networks of depth 15 with 100 neurons in each hidden layer.

one might expect that below such a phase-transition value, SBP might struggle.

To test this hypothesis, in Table 3 we report both theoretical and numerical estimates for $E\left[||\mathbf{W}||\right]$ for different algorithms and layer widths. The values of the parameters in each row were determined using Equations (44).

Focusing on the case $n = 10$, we see that both SIM and Glorot produce $E\left[||\mathbf{W}||\right] < 4$. So, for such methods and layer width, initial gradients are guaranteed to be vanishing in corresponding

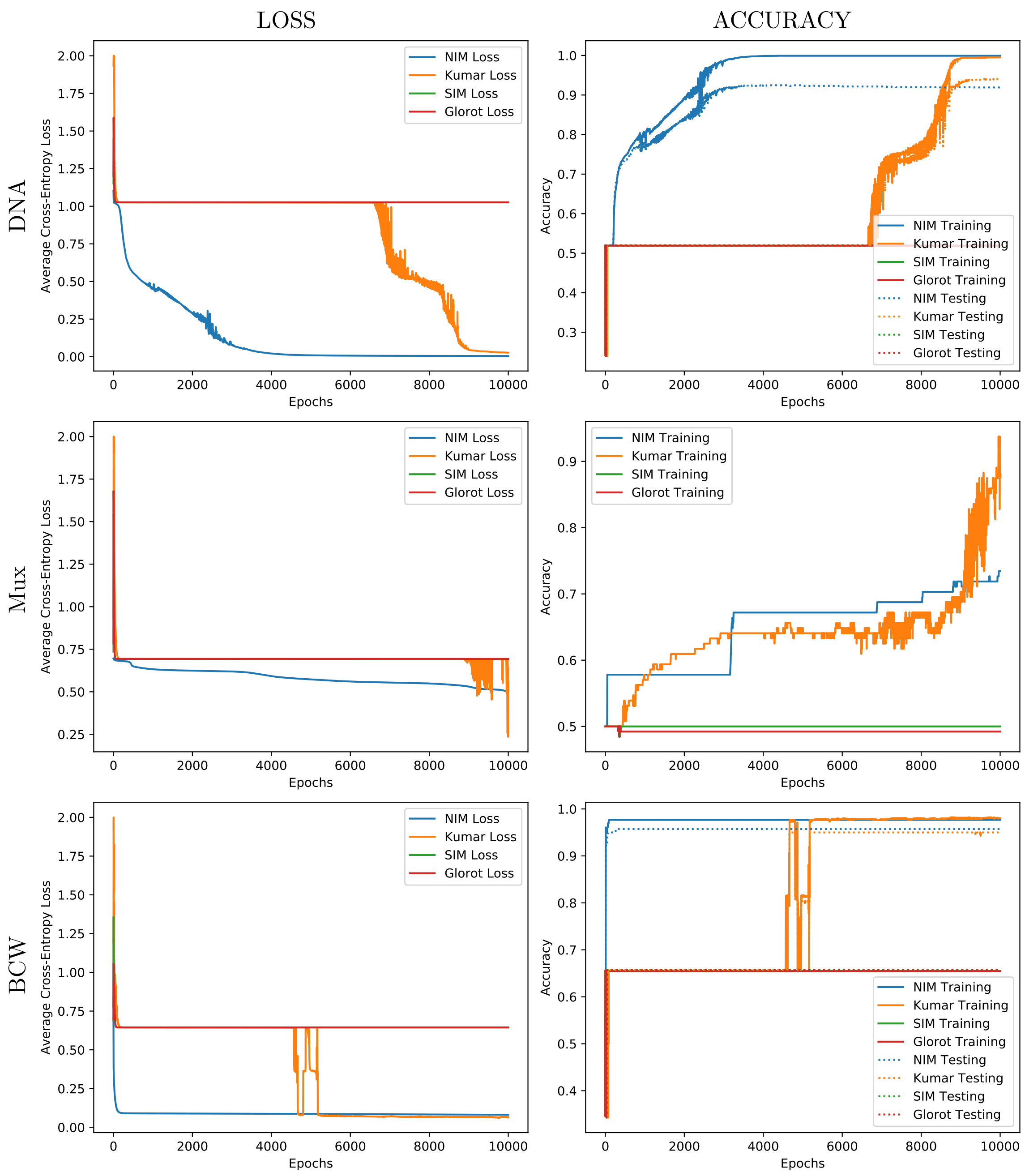


Figure 4: (Cont.) As in Figure 3 but for networks of depth 15 with 100 neurons in each hidden layer.

networks, which may be the reason why SBP cannot train them.[8] The success of NIM and Kumar, and the correspondingly higher $E\left[||\mathbf{W}||\right]$ (8.5 and 12.5, respectively), suggest that the postulated phase-transition value for $||\mathbf{W}||$ might be between 4 and 8.5.

If we now look at the case $n = 100$, we see some similarities and differences. The similarities are that both SIM and Glorot have the smallest $E\left[||\mathbf{W}||\right]$ of the four algorithms, and SBP fails to

[8]Here we focus on $E\left[||\mathbf{W}||\right]$ because standard deviations, $\texttt{StdDev}\left[||\mathbf{W}||\right]$, are only a fraction (approximately 11% for $n = 10$ and 3% for $n = 100$) of $E\left[||\mathbf{W}||\right]$. One could replace $E\left[||\mathbf{W}||\right]$ with $E\left[||\mathbf{W}||\right] + 3 \times \texttt{StdDev}\left[||\mathbf{W}||\right]$, should more precision on the argument be required.

Table 3: Theoretical and numerical estimates of $E[||\mathbf{W}||]$ for the different initialization methods considered in this paper. Theoretical estimates are from Equation (39). Numerical estimates are based on averages of the norm of 100 random weight matrices per configuration. SEM stands for Standard Error of the Mean. Corresponding standard deviations are $10 \times$ SEM.

Initialization method	μ	σ	n	*Eq.* (39)	*Mean*$\pm$*SEM*
SIM	0.00	0.10	10	1.10	1.10 $\pm$ 0.012
Glorot	0.00	0.32	10	3.47	3.50 $\pm$ 0.039
NIM	-0.80	0.10	10	8.50	8.49 $\pm$ 0.017
Kumar	0.00	1.14	10	12.50	12.69 $\pm$ 0.142
SIM	0.00	0.10	100	9.50	9.52 $\pm$ 0.026
Glorot	0.00	0.10	100	9.50	9.54 $\pm$ 0.025
NIM	-0.08	0.10	100	12.29	12.36 $\pm$ 0.033
Kumar	0.00	0.36	100	34.22	34.44 $\pm$ 0.104

train corresponding deep networks. Like for the case $n = 10$, we also see that Kumar produces the largest initial $E[||\mathbf{W}||]$ (34.22 vs. 12.29 for NIM). In this case too, SBP is successful after NIM and Kumar's initialization. However, in this case the postulated phase-transition value for $||\mathbf{W}||$ would need to be bigger than for $n = 10$: somewhere in between 9.5 and 12.29.

This suggests that while $||\mathbf{W}||$ needs to be above a certain value to make it possible for SBP to train a network, other factors may be at play, in that this threshold value appears to depend on n.

5.4.2. Magnitudes of Gradients

To further analyse the reasons for the success of NIM and Kumar and the failure of SIM and Glorot, we recorded the average of the magnitudes of the gradients measured in each layer over time. As a representative example, Figure 5 reports the results for the Iris problem and a network with 10 hidden layers for the four initialization methods.[9] For networks initialized with *SIM* and *Glorot*, we used a log-log scale because all the dynamics is concentrated in the first 70 or so training epochs.

Focusing on the first epoch of training, with all three initialization methods where $\mu = 0$ (SIM, Glorot and Kumar), in Figure 5 initially we see reasonably big gradients near the output layers which become smaller and smaller as we move towards the input layer, as we also observed in the simulations described in Section 5.2. So, initially all three methods suffer from the vanishing gradients problem, although the bigger the σ, and hence the bigger the $||\mathbf{W}||$, the bigger the initial gradients. Indeed, for $n = 10$, Kumar's method has a σ approximately 10 times bigger than that of Glorot's method and over 100 times bigger than that of SIM and correspondingly the initial gradients observed in the three methods are $|\Delta w| \in [10^{-4}, 0.04]$, $|\Delta w| \in [10^{-8}, 10^{-2}]$ and $|\Delta w| \in [10^{-13}, 10^{-3}]$, respectively.

In terms of temporal dynamics, as shown in Figure 5, with all three methods, within the first few tens of training epochs, all gradients become very small, thereby apparently freezing the networks, into a very-low-gradient state from which it is very hard to recover, at least judging from the fact that the loss function (Figure 3(top left)) stops decreasing.

However, in the case of Kumar, the network is not entirely frozen as after a few thousand epochs (two thousand, in the case of the Iris problem) the loss function suddenly starts decreasing eventually leading to successfully learning. In the case of the Iris problem, during the period of constant loss, the training-set accuracy (Figure 3(top right)) kept growing from the initial 33.3% to over 70% indicating that under the bonnet significant changes were talking place. However, for most other problems no changes in accuracy can be observed in that period. In all cases, the sudden drop in loss function appears to be associated with a state where *hidden layers near the input layer have fairly large gradients and such gradients are bigger than those in layers closer*

[9]The gradient magnitudes of the very last layer are not plotted as the networks use a soft-max output layer.

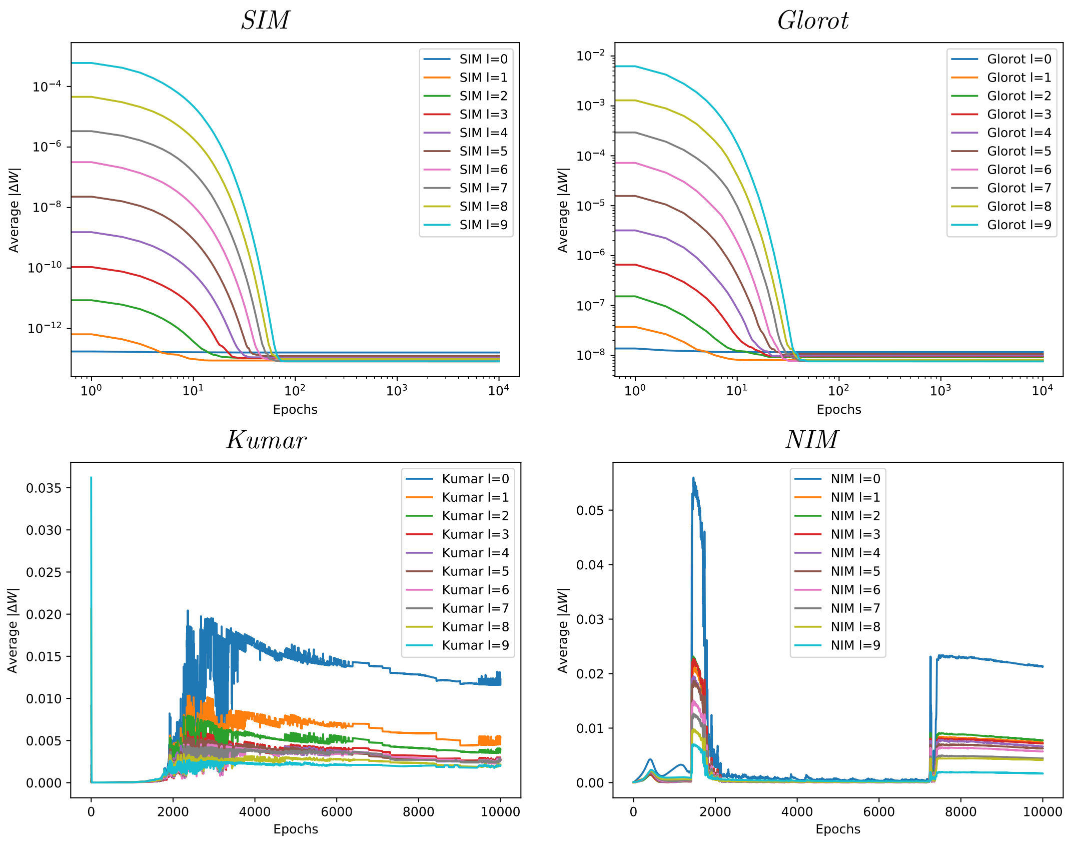


Figure 5: Mean $|\Delta w|$ in each layer when for a network with 10 hidden layers for four different initialization methods in the Iris problem.

to the output layer. As shown in Figure 3(top left) this "high input-gradients state" is what is required for the loss function to drop.

In the case of NIM, the situation is quite different. Firstly, in the first iterations, the network is already in a high input-gradients state, the gradients in the input layer being the biggest (see dark blue line in Figure 5(bottom right)), thus corroborating the prediction we made at the end of Section 4.3.1. Also, from such an initial state, unlike with SIM, Glorot and Kumar, the gradients grow progressively bigger, rather than smaller, for 400 epochs or so. Following this phase, the dynamics is not monotonic and there are bursts where gradients become even bigger. However, in all cases, the biggest gradients are at the level of the input layer. This seems to be the reason why, for NIM, both the loss function and the accuracy grow fairly smoothly and rapidly. For instance, for Iris, NIM allows SBP to obtain an accuracy of 80% within tens of epochs whereas with Kumar SBP does not get there for several thousand of epochs, and with SIM and Glorot SBP never gets there.

5.4.3. Two Theories are Better than One

It is clear from the observations in the previous two sections, that looking at $||\mathbf{W}||$, as we did when developing the theory in Section 4.2, can only partially capture the reasons for the vanishing gradients problem. So, while it can explain the relative magnitude of initial gradients in SIM, Glorot and Kumar, on its own, it does not explain why NIM produces so much better gradient descents than other initializations.

Similarly, the theory developed in Section 4.1 predicts that all gradients should be zero if $\mu = 0$ and can be non-zero only with an appropriate choice of $\mu \neq 0$. While this does explain why initial gradients are big and larger in the input layers than anywhere else for NIM, it does not capture the fact that while small, gradients are non-zero in Kumar, Glorot and SIM.

The two theories are complementary, but neither on their own nor together they appear to be able to explain why with Kumar initialization there is significant dynamics in classification accuracy, but not in the loss function, and why after thousands of generations eventually large gradients emerge and finally the training is successful.

6. Conclusions

There has been relatively little research on the root cause and the solution of the vanishing gradient problem in deep networks using the *sigmoid activation functions*, very few papers on this topic having being published in the last 20 years. Perhaps this is natural, since interest rapidly shifted towards other activation functions such as ReLU, PReLU and LReLU, that did not suffer from this problem, leaving the issue unsolved.

In this paper, we wanted to understand to what extent the vanishing gradient problem depends on the choice of the initial weight distribution in networks using a sigmoid activation function. Our hypothesis was that if one could start the training of a deep neural network from an initial position where gradients were non-vanishing, this might allow the standard back-propagation algorithm to succeed in training the network.

We started (Section 4) by developing two theoretical models that shows how the expected value of the gradient is affected by the expected value and the standard deviation of the initial weight distributions as well as the number of neurons in a layer. Using the lessons learnt from the theory, we then derived a new initialization method (NIM) that was designed to prevent vanishing gradients at least at the very beginning of the training process.

In Section 5, we empirically verified the predictions on initial gradients obtained from the theory for the logistic activation function finding that they are essentially correct. We also tested to what extend starting from such a good initial position in weight-space, the vanishing gradient problem could be reduced in networks with the logistic activation function. To do so we used a set of standard benchmark classification problems (four multi-class and two binary), including real-world, large-scale problems, also comparing NIM to three other initialization algorithms proposed in the literature. We found that when a network is initialized with NIM, the back-propagation algorithm was both successful and efficient at training deep networks with 10 and 15 hidden layers (and 10 and 100 neurons per hidden layer, respectively). In other words, *NIM appears to solve the vanishing gradient problem.* Another initialization algorithm, Kumar, was also able to solve all test problems but it required between one thousand and five thousand more epochs, depending on network size and problem complexity.

Also, at the end of Section 5, we analysed the reasons for the relative performance differences offered by different initialization algorithms, finding that our theoretical models offer explanatory insights, but they still do not capture the full picture, in particular in relation to the delayed training success associated with Kumar's method. This is a limitation of this work.

The study has also other limitations. For instance, while we developed our theoretical models in such a way that they could accommodate other sigmoid activation functions, all empirical validation of ideas and study of the learning process were conducted only for the logistic activation function. In the future, we plan to use other sigmoid activation functions (such as TanH) to see whether our methods can model and optimize initial gradients leading to successfully training deep neural networks based on such activation functions. Also, we only tested our initialization method with a limited number of problem. So, in the future, we will also want to see whether our methods work on a wider range of problems.

In the future we also want to consider more sophisticated forms of training, including the use of a momentum term [3], Rprop [41], RMSprop [23] and Adam [24]. Furthermore, it is clear from the theory, that it would be possible to refine our initialization rule so as to consider the expected input, $E[x]$. Also, in this work, we optimized the initial (epoch 0) gradients. However, if one used numerical optimization to identify optimal values of μ and σ in each layer, one could perform not just one pass of forward and backward propagation but a few, thereby being able to get a better idea of the initial *dynamics* of the gradients, likely yielding better choices of parameters. We will explore this in the future.

Most modern deep neural networks draw their power from their use of the ReLU activation function (or related functions). Naturally, there are many symmetries at the beginning of the training process even in such networks. So, there is the theoretical possibility that, with a suitable initialization strategy, the negative effects of symmetries could be reduced, thereby speeding up the learning process. Naturally, in this case the solution will not be in the form of an initial weight distribution with a negative mean, as this would cause most ReLU neurons to die straight away even before the learning process starts. In future research we will extend our theoretical approaches to ReLU-type activation functions to see whether anything can be learnt from this in relation to initializing modern deep neural networks.

Finally, the vanishing gradient problem is particularly insidious in applications involving recurrent neural networks. We believe most of the work presented here would be applicable to such networks with the proviso that l (the layer number) must be interpreted to represent time and that all W's, μ's, and n's do not depend on l. As a result, we believe our initialization method could speed up learning also in recurrent neural networks based on the logistic activation function. We hope to be able to explore in more depth and test this in future research.

References

[1] J. Anderson, E. Rosenfeld (Eds.), Neurocomputing: Foundation of Research, MIT Press, Cambridge, MA, 1988.

[2] P. J. Werbos, Backpropagation through time: what it does and how to do it, Proceedings of the IEEE 78 (10) (1990) 1550–1560.

[3] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[4] D. E. Rumelhart, J. L. McClelland, P. R. Group, et al., Parallel distributed processing, Vol. 1, MIT press Cambridge, MA, 1987.

[5] A. J. Surkan, J. C. Singleton, Neural networks for bond rating improved by multiple hidden layers, in: 1990 IJCNN International Joint Conference on Neural Networks, IEEE, 1990, pp. 157–162.

[6] G.-B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Transactions on Neural Networks 14 (2) (2003) 274–281.

[7] S. Tamura, M. Tateishi, Capabilities of a four-layered feedforward neural network: four layers versus three, IEEE Transactions on Neural Networks 8 (2) (1997) 251–255.

[8] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (4) (1989) 303–314.

[9] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural networks 4 (2) (1991) 251–257.

[10] M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a non-polynomial activation function can approximate any function, Neural networks 6 (6) (1993) 861–867.

[11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (5) (1989) 359–366.

[12] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157–166. `doi:10.1109/72.279181`.

[13] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.

[14] S. Hochreiter, Recurrent neural net learning and vanishing gradient, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (2) (1998) 107–116.

[15] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, in: S. C. Kremer, J. F. Kolen (Eds.), A Field Guide to Dynamical Recurrent Neural Networks, IEEE Press, 2001.

[16] S. Squartini, A. Hussain, F. Piazza, Attempting to reduce the vanishing gradient effect through a novel recurrent multiscale architecture, in: Proceedings of the International Joint Conference on Neural Networks, 2003., Vol. 4, IEEE, 2003, pp. 2819–2824.

[17] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2006) 1527–1554.

[18] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science 313 (5786) (2006) 504–507.

[19] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, science (2004).

[20] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1-3) (2006) 489–501.

[21] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (Jul) (2011) 2121–2159.

[22] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701 (2012).

[23] T. Tieleman, G. Hinton, Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning (2012).

[24] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[25] T. Dozat, Incorporating Nesterov momentum into Adam (2016).

[26] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, PMLR, 2013, pp. 1310–1318.

[27] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.

[28] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167 (2015).

[29] V. Nair, G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[30] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (ELUS), arXiv preprint arXiv:1511.07289 (2015).

[31] J. Li, H. Xu, J. Deng, X. Sun, Hyperbolic linear units for deep convolutional neural networks, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 353–359.

[32] X. Wang, Y. Qin, Y. Wang, S. Xiang, H. Chen, Reltanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis, Neurocomputing (2019).

[33] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. ICML, Vol. 30, 2013, p. 3.

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

[35] M. Roodschild, J. G. Sardiñas, A. Will, A new approach for the vanishing gradient problem on sigmoid activation, Progress in Artificial Intelligence 9 (4) (2020) 351–360.

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

[37] J. Bergstra, G. Desjardins, P. Lamblin, Y. Bengio, Quadratic polynomials learn better image features, Technical report, 1337 (2009).

[38] S. K. Kumar, On weight initialization in deep neural networks, arXiv preprint arXiv:1704.08863 (2017).

[39] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, J. H. Moore, PMLB: a large benchmark suite for machine learning evaluation and comparison, BioData Mining 10 (1) (2017) 36.

[40] M. J. H., EpistasisLab/pmlb: PMLB: a large, curated repository of benchmark datasets for evaluating supervised machine learning algorithms (2017).
URL `https://github.com/EpistasisLab/pmbl`

[41] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks, Vol. 1, 1993, pp. 586–591.