Evolutionary Intelligence (2024) 17:2081–2089
https://doi.org/10.1007/s12065-022-00795-y

SHORT NOTE

An analysis of weight initialization methods in connection with different activation functions for feedforward neural networks

Kit Wong[1] · **Rolf Dornberger**[2] · **Thomas Hanne**[3]

Received: 31 January 2022 / Revised: 1 October 2022 / Accepted: 6 November 2022 / Published online: 24 November 2022

Abstract
The selection of weight initialization in an artificial neural network is one of the key aspects and affects the learning speed, convergence rate and correctness of classification by an artificial neural network. In this paper, we investigate the effects of weight initialization in an artificial neural network. Nguyen-Widrow weight initialization, random initialization, and Xavier initialization method are paired with five different activation functions. This paper deals with a feedforward neural network, consisting of an input layer, a hidden layer, and an output layer. The paired combination of weight initialization methods with activation functions are examined and tested and compared based on their best achieved loss rate in training. This work aims to better understand how weight initialization methods in neural networks, in combination with activation functions, affect the learning speed in comparison after a fixed number of training epochs.

Keywords Feedforward neural network · Weight initialization · Activation function

1 Introduction

Artificial neural networks (ANNs) are composed of three types of layers, an input layer, hidden layers, and an output layer. ANNs consist of multiple nodes called neurons. They are arranged in layers, with the output of one layer serving as the input for the next layer. As inputs are transmitted between neurons, weights are applied to the inputs and the sum of all weighted inputs is passed into an activation function along with a bias. Activation functions produce the output of the neurons and thus ultimately determine the output of the entire neural network. Activation functions are either linear or non-linear functions. Weights are the parameters that are adjusted by learning. They are optimized during the training process. All parameters are adjusted toward the desired output for a given input. The weights connected to each input define how much influence the particular inputs have on the processing in the neuron and thus on its output. Sometimes, biases are additionally introduced and adjusted during the learning process. They serve as additional parameter values attached to neurons.

The training of ANNs highly depends on how the weights are initialized prior to the training [15]. The initialization affects the learning speed, the convergence rate, and the probability of correct classification of ANNs [6, 11, 12, 13]. Although different methods of weight initialization have been suggested over time, there is a lack of systematic research regarding the suitability of the weight initialization methods in connection with the choice of activation functions for different kinds of machine learning problems.

In this paper, Nguyen-Widrow weight initialization, random initialization, and Xavier initialization methods for optimizing the weight initialization are explored on a simple feedforward neural network (FNN). Thus, the contribution of our paper is the investigation of these three different weight initialization approaches each of them combined with five different activation functions, such as hyperbolic tangent

✉ Thomas Hanne
thomas.hanne@fhnw.ch

Kit Wong
kit.wong@bluewin.ch

[1] School of Business, University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland

[2] Institute for Information Systems School of Business, University of Applied Sciences and Arts Northwestern Switzerland, Basel, Switzerland

[3] Institute for Information Systems, School of Business, University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland

sigmoid activation function, log-sigmoid activation function, rectified linear unit, Gaussian error linear unit, and linear activation function. The resulting 15 configurations of FNNs are trained on six different datasets, each with 100 training runs to evaluate the results based on the average loss score. The results are assessed in terms of how these weight initialization methods are affected by the activation functions and the dataset variables, resulting in the performance of the learning process. Our study allows to determine favorable combinations of weight initialization methods in combination with different types of activation functions.

2 The problem of weight initialization

2.1 Problem statement

The problem model is to find the best weights to best reduce the loss score between the desired and the computed output given the normal distribution or the uniform probability distribution. Small coefficients close to zero enhance the convergence rates [12]. The goal is to find the suitable upper and lower bounds of the distribution function. Weights with big initial magnitude may lead to saturated activation functions at the start of the training. Weights that are too small slow down the training.

The weights after the training process contain information that the network has learned after being exposed to training data. In supervised learning, for example, neural network models can be adapted using the stochastic gradient descent, which incrementally changes the network weights to minimize the loss function. This function calculates the number of bits required to transmit an average event between two different probability distributions. An appropriate strategy for initializing the weights may help to achieve a sufficiently good local minimum in a short time.

2.2 Existing methods for weight initialization

Random weight initialization is an often-used approach to provide symmetrical random values around zero [3, 8, 13]. Bottou suggested an interval-based weight initialization method. It was defined as $[-\frac{a}{\sqrt{d_{in}}}, \frac{a}{\sqrt{d_{in}}}]$, where a refers to the maximum curvature value of the activation function [8, 15]. This parameter a with value 1.5 is identified for logistic sigmoidal activation functions [8]. d_{in} refers to the number of input neurons linked to the concerned neuron.

Drago and Ridella presented the statistically controlled activation weight initialization (SCAWI) [8, 15]. They identified the maximum magnitude of the weight before the neurons move in the saturation state. The weights and biases between layers are initialized uniformly in the range of $[-W, W]$. The weights and biases between the input and hidden layer W_{ih} are defined in (1),

$$W_{ih} = 1.3 * \left(1 + E\left[x^2\right]\right)^{\frac{1}{2}} \tag{1}$$

where *E[x]* refers to the expected value of the input term x. The weights and biases between the hidden and output layer W_{ho} are defined as given in (2),

$$W_{ho} = 1.3 * \left(1 + 0.3 * N_h\right)^{\frac{1}{2}} \tag{2}$$

Nguyen and Widrow suggested accelerating the training process by initializing the weights of a node within a uniform random interval [3, 8, 12, 15]. The active region of each neuron in a layer is almost evenly distributed over the layer's entire input space. The weights between the input and hidden layer W_{ih} are defined as given in (3),

$$W_{ih} = 0.7 * (N_h)^{\frac{1}{N_i}} \tag{3}$$

where N_h refers to the number of neurons in the hidden layer of the network and N_i to the number of neurons of the input layer.

Nielsen and Glorot suggested the Xavier initialization 12. This approach makes use of the Gaussian distribution function for the random values. It determines the scale of weight initialization based on the number of input and output neurons 14. The goal is to initialize the weights so that the variance of the activations is the same in each layer.

Masood et al. 8 and Sodhi and Chandra 13 (also cf. Dolezel et al. 3 developed a weight initialization scheme for non-linear activation functions using sigmoidal feedforward artificial neural network (SFFANN). SFFANN is an ANNs with an input layer, a hidden layer, an output layer and sigmoidal activation functions in the hidden layer [13]. The authors used a logistic sigmoidal function as the activation function. The magnitude of weight matrix between the input and the hidden layer is set as given in (4),

$$W_{ih} = \sqrt{3 * \left(\lambda^2 - \theta^2\right)} \tag{4}$$

where λ is 3.6369, which is the active 95% region of the derivative from the activation function used in the paper. The bias Θ is a uniformly distributed random value from the interval $[-\delta, \delta]$. The weights between the hidden and output layer are set with uniform distribution in the range $[-\mathrm{W_{ho}}, \mathrm{W_{ho}}]$ where $\mathrm{W_{ho}}$ is a specified in (5),

$$W_{ho} = \sqrt{3 * \left(\lambda^2 - \theta^2\right)} \tag{5}$$

where $\delta = 0.9 * \lambda$. Here, λ is generally 1 and Θ is within $[0, \delta]$.

Sadhi and Chandra 13 developed an Interval Based Weight Initialization method (IWI) for SFFANN using a hyperbolic tangent activation function. The weights leading into the *i-th* hidden node in the IWI method are distributed in the interval of $[(2i-1)/(H-1), (2i+1)/(H-1)]$, where i is the number of input and H is the number of hidden nodes. The weights are defined as shown in (6),

$$W_{ih} = 2i/(H-1) \tag{6}$$

The weights from the hidden nodes to the output nodes W_{ho} are set as given in (7)

$$W_{ho} = -C + 2iC/(H-1) \tag{7}$$

where C is the connection weight between the *i-th* hidden nodes and $C = H^{\frac{-1}{2}}$.

In this paper, Nguyen and Widrow, random weight initialization and Xavier weight initialization formulars are used for the experiment.

3 Method

This section explains the data acquisition process. The raw data is divided into a training set, a validation set, and a test set to test the weight optimization method for a neural network. The raw data are divided randomly to avoid sampling bias.

3.1 System overview

In this study, random (uniform between -1 and $+1$), Nguyen-Widrow, and Xavier weight initialization methods are used to initialize the weights of a FNN to assess the training performance. To focus on the effects of the initialization methods, the set up FNN consists of an input layer, only one hidden layer and an output layer. The number of inputs for the FNN depends on the dataset. The number of hidden neurons is 10. The data is classified into 2–4 outputs, depending on the dataset.

The dataset is divided into three sets, 75% were randomly selected for the training set, 15% for the validation set and the remaining 10% were for the test set.

The described 15 configurations of the FNN are trained with the datasets. For each dataset and in every training run the training took place over 6 epochs, because learning was not yet complete on all datasets in the first six epochs. After the six epochs, the learning curve on some datasets already flattened and the process showed signs of convergence. In addition, we were only interested in the learning performance directly after the weight initialization in the experiments. The best performance of all 15 configurations after 6 epochs is compared. Each of the 15 configurations is a pair of activation function and weight initialization strategy. Each neuron in all configurations has the same fixed bias, which is initialized with random numbers.

For training the FNN, the same weight initialization and bias are used in the hidden layer for each configuration as in the input layer. Each configuration runs 100 times on one dataset. This number of repetitions enables a statistical analysis of significance. The outcome is an average value of the loss score over the 100 training runs. The result aims to observe how five different activation functions respond to the three different weight initialization methods.

3.2 Data sets

Six different datasets are used in this study to train the models: Wine dataset, Thyroid Disease dataset, Simpleclass dataset, Iris dataset, Glass dataset and Cancer dataset. All six datasets are example datasets provided by the University of California Machine Learning Repository [4] and included in tools such as MATLAB 9. Table 1 presents an overview of the dataset characteristics.

- The Wine dataset is based on the results of a chemical analysis of wine grown on three different cultivations in the same region in Italy.
- The Thyroid Disease dataset consists of three classes, 21 attributes and 7200 instances. All data in the dataset are labelled to one of the three classes: hyperthyroidism, hypothyroidism, and normal subject.

Table 1 Data sets

Dataset	No. of instances	No. of attributes	Type of attributes	No. of classes
1. Wine dataset	178	13	Real values	3
2. Thyroid Disease dataset	7200	21	15 categorical, 6 numerical	3
3. Simpeclass dataset	1000	2	Real values	4
4. Iris dataset	150	4	Real values	3
5. Glass dataset	214	10	Real values	2
6. Cancer dataset	699	10	Integer numbers	2

- The Simpleclass dataset is a classification dataset with two input data and four separate output categories.
- The Iris dataset classifies iris flowers into three species.
- The Glass dataset was motivated by criminological investigation.
- The Cancer dataset is related to the breast cancer cases from a clinic.

For the considered datasets we do not consider any preprocessing such as oversampling or undersampling as we are not concerned with specific quality metrics for the datasets but primarily with the learning speed depending on different weight initialization methods.

3.3 Feedforward neural network architecture

The FNN models are first tested on the Wine dataset and the average loss score is evaluated for all the 15 configurations. Afterwards, the same set-up is tested on the Thyroid dataset and it is assessed whether the same average loss score is achieved with the same configuration.

In this work, an FNN setup with one hidden layer was chosen as the architecture for the network. The FNN for the Wine dataset is depicted in Fig. 1. It consists of an input layer of 13 input variables (= number of input nodes), each with one input bias unit and one input weight unit. The hidden layer consists of ten neurons and related weights and biases. The output layer contains three output neurons, which corresponds to the number of classes to be predicted.

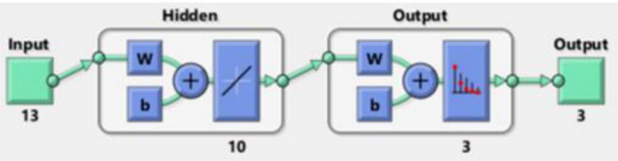

Fig. 1 Set up of the FNN wine dataset

The FNNs for the other datasets are constructed in a similar manner. In each case the number of input variables corresponds to the number of attributes of the respective dataset (see Table 1), each with one input bias unit and an input weight unit on the input layer. In each case, the hidden layer is constructed of hidden layer bias, weights and ten hidden neurons. The output layer includes a number of output neurons which correspond to the number of classes to be predicted for each dataset (see Table 1).

15 configurations of the FNNs (different combinations of weight initialization methods and activation functions) are evaluated per dataset. The bias is a randomly generated number and remains the same for all configurations. The 15 configurations are shown in Table 2.

3.4 FNN activation functions

In an FNN, the activation function determines the maximal magnitude of the signal that can come from a neuron [1, 10]. It defines how the weighted sum of the input is transformed into an output. In this study, five different activation functions are used. The five activation functions used in the model are stated below:

3.4.1 Hyperbolic tangent sigmoid activation function

The hyperbolic tangent sigmoid activation function is an S-shaped anti-symmetric function [13]as defined by (8). Its output is in the range of $[-1, +1]$.

Table 2 Configuration methods

Configuration	Weight optimization method	Activation function
1. NY TanSigmoid	Nguyen-widrow initialization	Hyperbolic tangent sigmoid
2. NYLogSigmoid	Nguyen-Widrow initialization	Log-sigmoid
3. NY RELU	Nguyen-Widrow initialization	Rectified linear unit
4. NY GELU	Nguyen-Widrow initialization	Gaussian error linear unit
5. NY Linear	Nguyen-Widrow initialization	Linear
6. Rand TanSigmoid	Random initialization [−1,1]	Hyperbolic tangent sigmoid
7. Rand LogSigmoid	Random initialization [−1,1]	Log-sigmoid
8. Rand RELU	Random initialization [−1,1]	Rectified linear unit
9. Rand GELU	Random initialization [−1,1]	Gaussian error linear unit
10. Rand linear	Random initialization [−1,1]	Linear
11. Xavier TanSigmoid	Xavier initialization	Hyperbolic tangent sigmoid
12. Xavier LogSigmoid	Xavier initialization	Log-sigmoid
13. Xavier RELU	Xavier initialization	Rectified linear unit
14. Xavier GELU	Xavier initialization	Gaussian error linear unit
15. Xavier linear	Xavier initialization	Linear

$$\psi_1(x) = \tanh(x) \tag{8}$$

3.4.2 Log-sigmoid activation function

The log-sigmoid activation function is an S-shaped function with an output range of $[0, +1]$ according to Eq. (9) (also cf. Larose and Larose 7.

$$\psi_2(x) = \log\left(\frac{1}{1+\exp(-x)}\right) \tag{9}$$

3.4.3 Rectified linear unit

The Rectified Linear Unit (RELU) zeros-out all negative values [2], i.e., RELU returns zero if it receives a negative input. According to Eq. (10), it passes the input value directly to its output value if it is positive, otherwise the output will be zero.

$$\psi_3(x) = \max(0, x) \tag{10}$$

3.4.4 Gaussian error linear unit

The Gaussian Error Linear Unit (GELU) is a nonlinear activation function based on the standard Gaussian cumulative distribution function according to Eq. 11 [5].

$$\psi_4(x) = xP(X \leq x) \tag{11}$$

3.4.5 Linear activation function

In the linear activation function, the function is a linear passing of the input value to the output value. The output of the function is not limited to a specific range.

3.5 Loss function

The cross-entropy loss function is used to evaluate the performance of the training. During the training, these 15 different configurations are compared based on their loss score. The loss is calculated on training, validation, and test dataset. For the final comparison of the performance of the configurations the average loss score on the evaluation set is used. The loss score calculates the error for the current state of the model. The cross-entropy can be calculated as shown in (8),

$$CE = -\sum_{i}^{C} t_i log(s_i) \tag{12}$$

where CE is the cross entropy, t is the target distribution and s is the approximation of the target distribution. The higher the loss score, the worse the model and vice versa.

4 Results and discussion

In order to evaluate the effectiveness of training of a FNN with different weight initialization methods and different activation functions, the average loss score from each model is calculated. The average loss score after six epochs is represented by the blue dot. The error bar represents the standard deviation of the training result. The training is aborted after six epochs because by limiting the number of epochs, a conclusion about the convergence speed of the training can also be drawn. Exemplarily, Fig. 2 shows the training progress over six epochs on the wine dataset in one of the 100 test runs.

4.1 Wine dataset results

The results of the FNN on the Wine dataset are shown in Fig. 3. The Wine dataset consists of continuous variables. From the result, Nguyen-Widrow weight initialization with linear activation function has the best performance with 0.03595 as the lowest loss score. On average, Nguyen-Widrow and Xavier weight initialization perform better on the dataset than the random weight initialization.

4.2 Thyroid disease dataset results

The results of the Thyroid Disease dataset are shown in Fig. 4. The Thyroid Disease dataset consists of categorical numerical variables, which are a mix of quantitative and qualitative variables. The results show that random weight initialization with log-sigmoid activation function has the best performance with 0.11001 as the lowest loss score. On average, the random weight initialization method performs better on this mixed dataset than the Nguyen-Widrow or the Xavier weight initialization method.

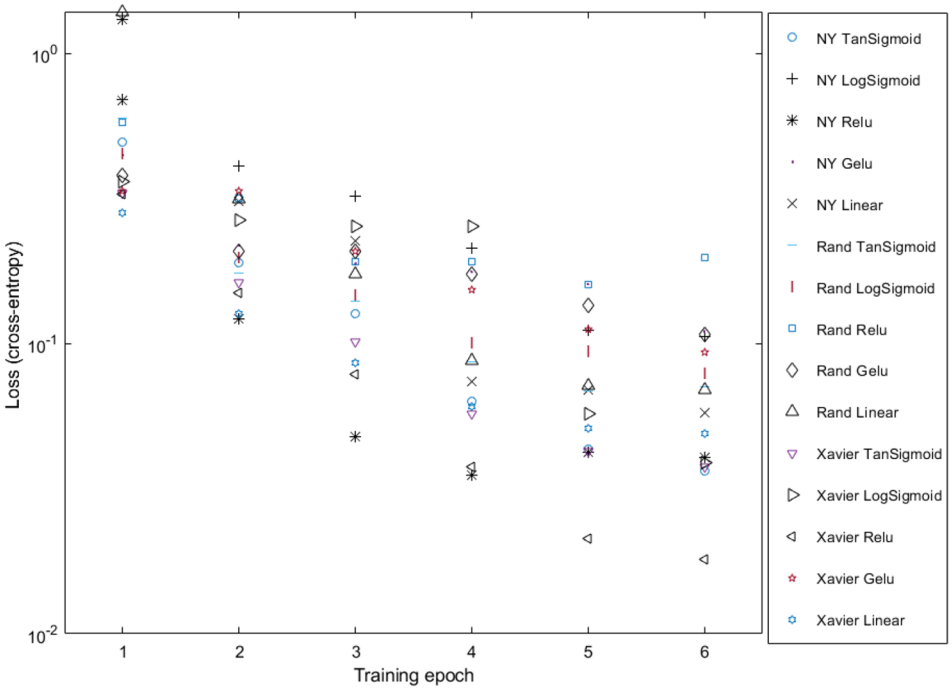

Fig. 2 Training progress example on the wine dataset

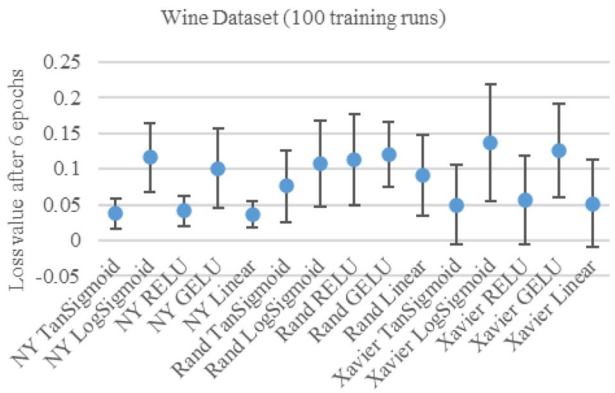

Fig. 3 Wine datasets results of the 15 configurations. The abbreviated method names on the *x* axis correspond to the methods shown in Table 1

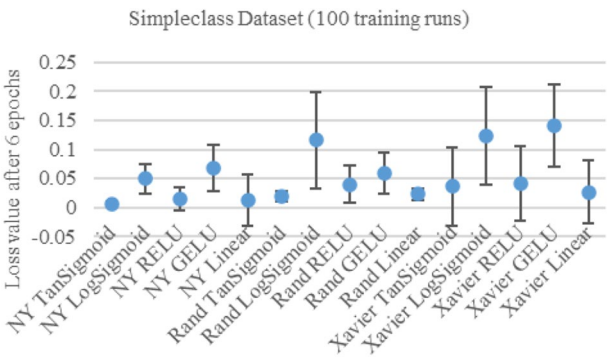

Fig. 5 Simpleclass dataset results of 15 configurations

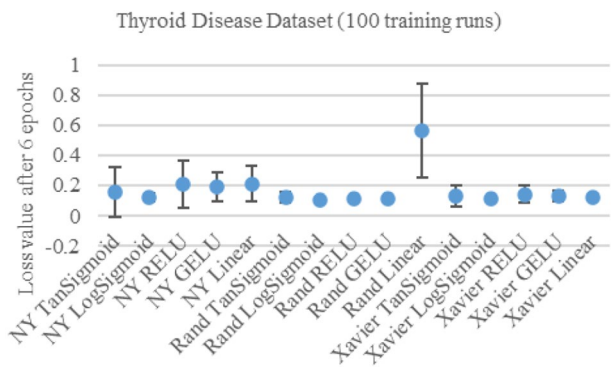

Fig. 4 Thyroid disease dataset results of 15 configurations

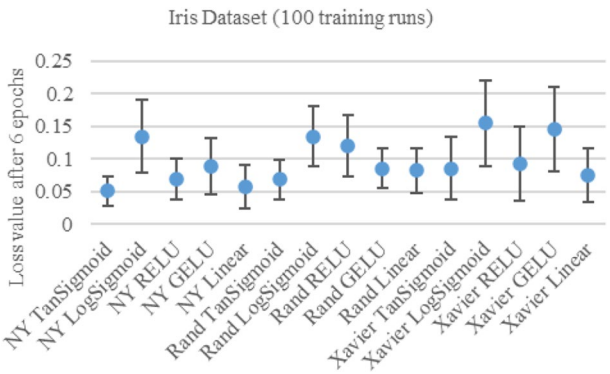

Fig. 6 Iris dataset results of 15 configurations

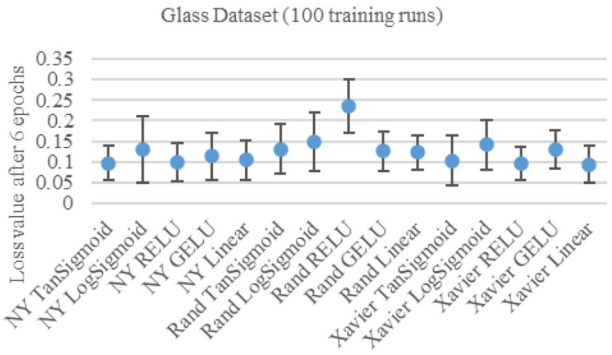

Fig. 7 Glass dataset results of 15 configurations

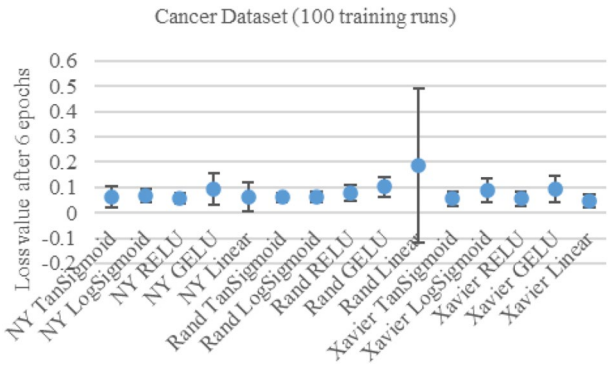

Fig. 8 Cancer dataset results of 15 configurations

4.3 Simpleclass dataset results

The results of the Simpleclass dataset are shown in Fig. 5. The results of the Simpleclass dataset show that the Nguyen-Widrow weight initialization with a hyperbolic tangent sigmoid activation function has the best performance with 0.00680 as the lowest loss score. On average, the Nguyen-Widrow and the Xavier weight initialization are better than the random weight initialization on this dataset.

4.4 Iris dataset results

The results of the training on the Iris dataset are shown in Fig. 6. The Iris dataset consists of real number variables. The results reveal that the Nguyen-Widrow weight initialization with the hyperbolic tangent sigmoid activation function has the best performance with 0.05076 as the lowest loss score. On average, the Nguyen-Widrow and the Xavier weight initialization perform better on the dataset than the random weight initialization.

4.5 Glass dataset results

The results of the Glass dataset are shown in Fig. 7. The Glass dataset consists of real number variables. The results show that the Xavier weight initialization with a linear activation function has the best performance with 0.09401 as the lowest loss score. On average, the Nguyen-Widrow and the Xavier weight initialization perform better on the dataset than the random weight initialization.

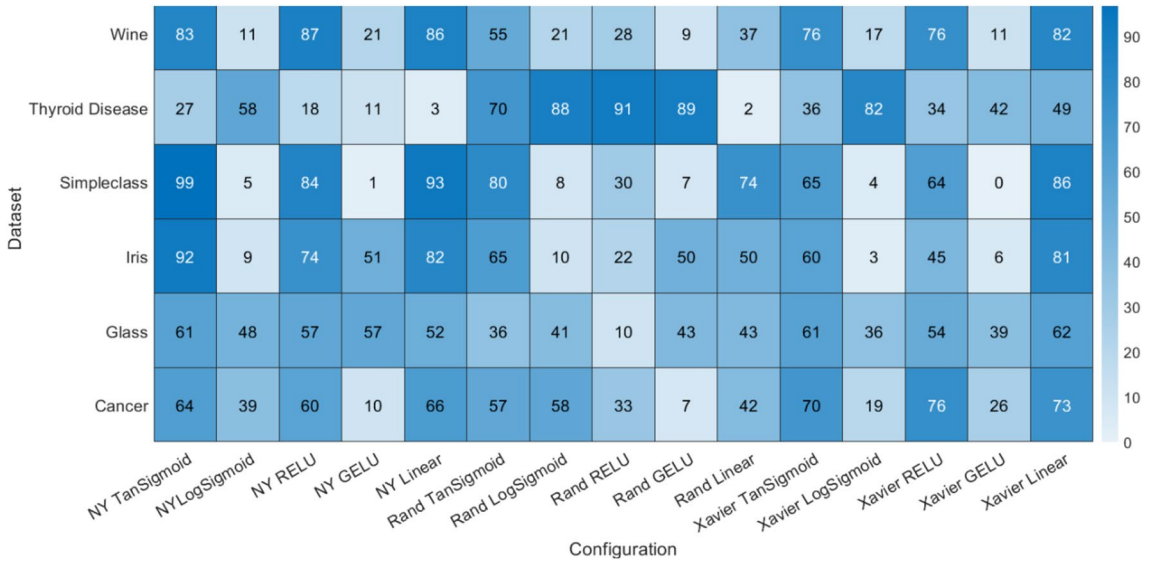

Fig. 9 Average performance of each configuration on all datasets

4.6 Cancer dataset results

The results of the Cancer dataset are shown in Fig. 8. The Cancer dataset consists of integer variables. The results show that the Xavier weight initialization with a linear activation function has the best performance with 0.04909 as the lowest loss score. On average, the Nguyen-Widrow and the Xavier weight initialization perform better on the dataset than the random weight initialization.

4.7 Overall results

Figure 9 shows an overall result of the average performance of each configuration on each dataset. It is the percentage of training runs where the configuration was better than the median of all configurations on this dataset in a training round.

The results shown in Fig. 9 are tested for statistical relevance. We can reject the hypothesis that we produce random results by doing a test with the cumulated binomial distribution function ($p = 0.5$). The 95% confidence interval with $p = 0.5$ and 100 tests is between 40 and 59. If less than 40 tests out of 100 tests give a result lower than the median, the hypothesis $p = 0.5$ is rejected, because this initialization scheme performs worse. If more than 59 tests out of 100 give a result that is better than the median, the hypothesis $p = 0.5$ can be rejected because this initialization performs better.

The result indicates that the Xavier weight initialization method with a linear activation function performs the best on average among all configurations. It has a more stable performance no matter which variables are in the dataset.

Random weight initialization performs better than Nguyen-Widrow and Xavier weight initialization only for qualitative variables.

5 Conclusions

In this research, three initialization methods are tested and analyzed, applied to a FNN and combined with five different activation functions. The 15 configurations are trained on six different standard datasets. The results indicate that it is worthwhile to use a more sophisticated initialization method than just uniform distributed random values in the range from -1 to 1 as weight initialization method. When testing these 15 configurations on 6 standard datasets, the Nguyen-Widrow weight initialization with hyperbolic tangent sigmoid activation function and the Xavier initialization with linear activation function performed best among all 15 configurations. From these two methods, the Xavier weight initialization with linear activation function is superior for these 15 configurations, six epochs and 100 trainings over each dataset in the FNN configuration of one input layer, one hidden and one output layer with the mentioned number of neurons in each layer. This implies that the majority of datasets include linearly separable classes that can be classified by a linear classifier. Therefore, the results could be different for datasets with stronger nonlinear separability, which should be investigated in future research. In addition, the performance could be different for a multilayer FNN, but this needs to be the subject of further research, as do further activation functions and initialization methods. In addition, it would be useful to consider similar studies on the effect of weight initialization methods with larger or more complex datasets in future research.

Funding Open access funding provided by FHNW University of Applied Sciences and Arts Northwestern Switzerland.

References

1. Apicella A, Donnarumma F, Isgrò F, Prevete R (2021) A survey on modern trainable activation functions. Neural Netw 138:14–32. https://doi.org/10.1016/j.neunet.2021.01.026
2. Chollet F, Allarire JJ (2017) Deep learning with R. Manning Publications Co., Shelter Island
3. Dolezel P, Skrabanek P, Gago L (2016) Weight initialization possibilities for feedforward neural network with linear saturated activation functions. IFAC-PapersOnLine 49(25):49–54. https://doi.org/10.1016/j.ifacol.2016.12.009
4. Dua D, Graff C (2019) UCI machine learning repository [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science, Irvine, CA (2019)
5. Hendrycks D, Gimpel K (2016) Gaussian error linear units (GELUs). http://arxiv.org/abs/1606.08415
6. Kumar SK (2017) On weight initialization in deep neural networks. ArXiv 1: 1–9
7. Larose DT, Larose CD (2015) Data mining and predictive analytics. John Wiley & Sons Inc., Hoboken
8. Masood S, Doja MN, Chandra P (2020) Architectural parameter-independent network initialization scheme for sigmoidal feedforward ANNs. Arab J Sci Eng 45(4):2901–2913. https://doi.org/10.1007/s13369-019-04200-2
9. MATLAB: version 9.10.0 (R2021a) (2021) The Math works Inc, Natick
10. Mittal A, Singh AP, Chandra P (2021) Weight and bias initialization routines for sigmoidal feedforward network. Appl Intell 51(4):2651–2671. https://doi.org/10.1007/s10489-020-01960-

11. Nienhold D, Schwab K, Hanne T, Dornberger R (2015) Effects of weight initialization in a feedforward neural network for classification using a modified genetic algorithm. In: Proceedings - 2015 3rd international symposium on computational and business intelligence, ISCBI 2015, 6–12. https://doi.org/10.1109/ISCBI.2015.9
12. Ramos EZ, Nakakuni M, Yfantis E (2017) Quantitative measures to evaluate neural network weight initialization strategies. In: 2017 IEEE 7th annual computing and communication workshop and conference, CCWC 2017, 3. https://doi.org/10.1109/CCWC.2017.7868389
13. Sodhi SS, Chandra P (2014) Interval based weight initialization method for sigmoidal feedforward artificial neural networks. AASRI Proced 6:19–25. https://doi.org/10.1016/j.aasri.2014.05.004
14. Sun W, Su F, Wang L (2018) Improving deep neural networks with multi-layer maxout networks and a novel initialization method. Neurocomputing 278:34–40. https://doi.org/10.1016/j.neucom.2017.05.103
15. Yam JYF, Chow TWS (2000) A weight initialization method for improving training speed in feedforward neural network. Neurocomputing 30(1–4):219–232. https://doi.org/10.1016/S0925-2312(99)00127-7