Investigation of weight initialization using Fibonacci Sequence on the performance of neural networks*

1st Dr. Dipanwita Sinha Mukherjee  
 Enterprise Analytics & Data Science  
 Artificial Intelligence - Center of Excellence  
 Wells Fargo International Solutions Private Limited  
 Bangalore, India  
 Dipanwita.Sinhamukherjee@wellsfargo.com

2nd Naveen Gururaja Yeri  
 Head of Enterprise Analytics & Data Science  
 Artificial Intelligence - Center of Excellence  
 Wells Fargo International Solutions Private Limited  
 Bangalore, India  
 naveen.yeri@wellsfargo.com

Abstract—Initializing weights are important for fast convergence and performance improvement of Artificial Neural Network models. This study proposes a heuristic method to initialize weights for Neural Network with Fibonacci sequence. Experiments have been carried out with different network structures and datasets and results have been compared with other initialization techniques such as Zero, Random, Xavier and He. It has been observed that for small sized datasets, Fibonacci weight initialization technique reports 94% of test accuracy which is better than Random (85%) and close to Xavier (93%) and He (96%) initialization methods. Also, for medium sized dataset, we have noted that performance of Fibonacci weight initialization method is comparable with the same for Random, Xavier and He initialization techniques.

Index Terms—Weight Initialization Technique, Artificial Neural Networks, Fibonacci Sequence

I. INTRODUCTION

Famous Italian mathematician Leonardo da Pisa who was also known as Fibonacci, while calculating the population growth of rabbits, introduced Fibonacci Sequence as follows:

\[
F_n = \begin{cases} 
0 & \text{if } n = 0 \\
1 & \text{if } n = 1 \\
F_{n-1} + F_{n-2} & \text{if } n > 1 
\end{cases}
\]  \quad (1)

The limit of the ratio between two consecutive numbers \(F_{n+1} \text{ and } F_n\) of Fibonacci sequence as \(n\) approaches to \(\infty\), is defined as \(\phi\) \(\left(\lim_{n \to \infty} \frac{F_{n+1}}{F_n} = \phi\right)\) \cite{1}. This irrational number \(\phi\) is called as Golden Ratio or Divine Proportion \((\phi = 1.618033...\)) . One of the interesting property of \(\phi\) is that, \(\phi = \frac{1}{\phi} + 1\) which forms the quadratic equation \(\phi^2 - \phi - 1 = 0\).

Presence of the golden ratio constant has been observed in DNA structure \cite{2}, human brain waves during thinking \cite{3}, human facial proportions and facial attractiveness assessment \cite{4}, \cite{5} scenarios. Therefore, this ratio has been used in performance evaluation of various face detection algorithms \cite{6}. Also Convolution Neural Net (CNN) has been used to learn the attributes of facial attractiveness with the help of \(\phi\) \cite{4}.

Previous studies have explored the contribution of Golden ratio in different algorithmic improvement scenarios in the areas of Optimization, Artificial Intelligence (AI) and Machine Learning (ML). In optimization, \(\phi\) has been used in Golden Section Search (GSS) \cite{7} method. This can be obtained while breaking down the whole interval into three parts with the help of \(\phi\) in a iterative way. Though GSS has been primarily developed to find the extremum (minimum or maximum) for a univariate and unimodal function over a closed interval, it could also be applied in multi-dimensional cases with certain limitations.

Similarly, efficient ML algorithms for data classification tasks have been developed using spiral golden angle which is parametrized by radius \((r)\), angle \((\theta)\), \(\phi\) and Golden Angle \((\psi)\) \cite{8}, \cite{9}. Luwes \cite{10} has investigated the performance of neural network with
weight matrix that has been initialized according to the Fibonacci sequence and has compared the results with randomly initialized weight matrix. It was observed that initialization of the weight matrix according to Fibonacci series lead to a steeper learning curve. This paper mainly focuses on the application of Fibonacci sequence for weight initialization in optimization algorithms of Artificial Neural Network (ANN) to improve the performance. Later on, the result has been compared with various initialization techniques on different datasets. The idea of initialization with Fibonacci sequence has been inspired from Golden ratio which can be observed in nature particularly for symmetrical, balanced and reproducible systems. This paper has been organized in the following structure. Section I introduces Fibonacci sequence and Golden ratio and associated applications, Section II introduces standard initialization techniques including Fibonacci sequence for optimization algorithms in ANN, Section III discusses experimental set-up and finally, results have been discussed with conclusions in section IV and V respectively.

II. NEURAL NETWORKS AND VARIOUS WEIGHT INITIALIZATION METHODS

A. Artificial Neural Network (ANN)

As the name suggests, ANN is primarily inspired from biological nervous systems. Architecture of a generalized ANN model could primarily be divided into three layers namely input, hidden and output layer [11]. In general, $a^{m+1} = f^{m+1}(W^{m+1}a^m + b^{m+1})$ for $m = 0, 1, \ldots, (M - 1)$; $a^0 = p$; $a = a^M$; where $p$ is Input Variables and $a$ is Network Outputs. Number of layers in the network structure is $M$. $f, b$ and $W$ are activation function, bias and weight respectively. Information from the input layer is propagated in feed forward direction from one layer to the next. Transfer function in hidden layers can introduce non-linearity in the model. Thus, ANN is capable of approximating high non-linear complex phenomena without prior knowledge of the relationship. Back-Propagation [11] and different variations of the same are widely used algorithms to train the ANN model. The error in prediction is used in back-propagation learning algorithm that is fed in backward direction to optimize the weights and biases [11]. This process of error minimization is repeated across many training cycles till the network performance reaches a predefined acceptable accuracy. Hence, a multi-layer neural network learn to map the relationship between a set of input and corresponding output space and predict accordingly.

In any NN structure (Convolution Neural Network-CNN, Recurrent Neural Network-RNN, Deep Neural Network-DNN etc.), activation function has significant importance for capturing non-linear relationship. Various common activation functions are Logistic Sigmoid ($f(x) = \frac{1}{1+e^{-x}}$), Hyperbolic Tan ($f(x) = \frac{e^x-e^{-x}}{e^x+e^{-x}}$), Rectified Linear Unit-ReLU ($f(x) = \max(0, x)$) etc. Aside from activation functions, recent studies have found that the weight initialization also plays a crucial role for optimizing the performance. This problem is bi-directional, both small and big weights may lead to loss of information during forward propagation while creating vanishing and exploding effects of gradients respectively during backward propagation. This vanishing or exploding gradient problem was first approached by Hochreiter [12]. Since then, various studies have been attempted in this direction. Later on, Xavier [13] proposed an initialization technique that performs satisfactorily in DNN structure except for ReLU activation function. This limitation has been solved by He [14]. Both this initialization (Xavier and He) performs well for balanced datasets but fails for imbalanced datasets observed in practical scenario. This study focuses on initialization concept that has been investigated by Luwes [10] previously. Here, we have initialized the weights of neural network with Fibonacci sequence. Experimental investigation has been done on different datasets while NN model was initialized with Fibonacci sequence. Results of this experiment have been compared with popular initialization methods for the same NN structure and datasets. Brief discussions about different initialization techniques like Random, Zero, Xavier, He and the proposed Fibonacci Sequence (FS) have been included below.

B. Different Initialization techniques

- Zero Initialization: In zero initialization, weights and biases are set to be zero initially for all neurons. As a result, all neurons in each layer learn the same thing and fails to break symmetry and therefore, the network will behave like a linear classifier.

- Random Initialization: In random initialization, weights are initialized randomly from a standard Gaussian distribution ($\mu = 0, \sigma = 1$). This helps
the neurons to break symmetry and improves performance significantly. At the same time, random initialization with very high or low value will lead to exploding or vanishing problem respectively particularly in DNN structure.

- Xavier Initialization: Xavier Glorot and Yoshua Bengio [13] has proposed a initialization method called as ‘normalized initialization or Xavier initialization’. In this initialization method, weights in the layer \( l \) are initialized randomly from a Gaussian distribution with \( \mu = 0 \) and \( \sigma = \frac{1}{\sqrt{(n_l-1)}} \) where \( n_l \) is the number of neurons in the previous \((l-1)^{th}\) layer and biases are initialized at zero. This initialization reduces exploding or vanishing gradient problem by making the same variance for outputs of activation functions at each layer and back-propagated associated gradients throughout the network layers. Xavier initialization performs well with hyperbolic tan (tanh) or sigmoid activation functions, but it fails to converge for ReLU. The reason behind this is the assumption that activation functions are linear, which is not applicable for ReLU and Parametric ReLU (PReLU).

- He Initialization: In this method [14], weights for layer \( l \) are initialized randomly from a Gaussian distribution with \( \mu = 0 \) and \( \sigma = \frac{2}{\sqrt{(n_l-1)}} \) where \( n_l \) are the number of neurons in the previous \((l-1)^{th}\) layer and biases are initialized at zero. This initialization method takes care of the non-linearity of the activation functions, hence could address the limitation of Xavier initialization technique.

- Fibonacci Initialization: In this work we have proposed Fibonacci Sequence (FS) initialization where weights are initialized randomly from the Fibonacci sequence. Random samples from the log scaled and normalized Fibonacci series (starting from 1) have been considered as initial weights based on the size of the neural network.

III. EXPERIMENTAL SETUP

A. Datasets

We have created two concentric circular datasets with different values for radius using scikit-learn python package[1][2]. 300 points are generated for training and 100 points are kept for testing purpose. This is known as two concentric circles problem where each circle represents a class and hence this becomes a binary classification problem. The aim of the classifier is to separate two different classes designated by different colours (Fig. 1).

Fig. 1: Circle Plot

Also, MNIST[2] dataset of handwritten digits have been used for the purpose of training and validating the model. This dataset has 10 different digits as label from 0 to 9 and consists of 60,000 train and 10,000 test images. Each element is a grayscale image with resolution of 28x28. The aim is to develop ANN model to efficiently classify the handwritten images.

B. Model Architecture

For circle datasets, neural network model has been constructed with 2 hidden layers having 10 and 5 nodes respectively. ReLU activation function has been used in each hidden layer and for the final output layer, sigmoid activation function has been considered. Model is trained with learning rate of 0.01 and for 20,000 iterations. Model performance has been observed in terms of both train and test accuracy. Similarly, for MNIST dataset again, neural network model has been structured with 2 hidden layers having 100 and 50 nodes respectively. Softmax activation function \( ((\sigma(x_i) = \frac{\exp(x_i)}{\sum j \exp(x_j)}) \) has been used at the output layer where at each hidden layer, we have employed sigmoid activation function. In order to ensure robustness of the classifier, random noise from uniform distribution has been added purposefully to both train and test datasets. Model has been trained using “adam” optimize[3] for 10 epochs and the

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html; accessed on 27 March, 2021
[2] http://yann.lecun.com/exdb/mnist/; accessed on 27 March, 2021
[3] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam; accessed on 28 March, 2021
performance has been computed in terms of loss and accuracy for both training and test set of images.

C. Development of initial weights using Fibonacci Sequence

A custom initializer class has been defined using "initializers" function of the "TensorFlow" framework that has been used for implementing the proposed method and conducting associated experiments in this study. This custom initializer generates Fibonacci sequence numbers up to the provided number. Next, log scaling and normalization have been applied to the generated Fibonacci sequence numbers. Finally, according to the structure of the neural net, random samples from this log-scaled and normalized Fibonacci sequence numbers are passed as the argument for initial weights to the "layers.Dense" function.

IV. Results and Discussion

A. Investigation: Circle Dataset

In this problem, we have attempted to build an efficient binary classifier using ANN model to differentiate generated datasets that fall into 2 concentric circles. The model has been trained for 20,000 iterations and associated cost function has been plotted. Also, model accuracy has been computed and reported with different weight initializations for both train and test datasets.

1) Model with Zero Initialization: Here, the parameters for the ANN model with $l$ number of layers (weights : $(w^1, w^2, ..., w^{l-1}, w^l)$ and bias : $(b^1, b^2, ..., b^{l-1}, b^l)$) are initialized to zero. The performance of the model is poor as the cost function (Fig. 2a) is not decreasing over iteration and model is predicting '0' for every example (Fig. 2b). This could occur as every neuron is learning the same information when initializing all the weights to zero or some constant value and it fails to break symmetry. Therefore, it could be concluded that, weights need to be initialized randomly to break symmetry even though zero initialization of biases. In this case, after 19000 iterations, loss is 0.6932 and accuracy reaches 50% for test dataset.

2) Model with Random Initialization: Here, weights are initialized randomly from a standard Gaussian distribution ($\sim N(0, 1)$). It could be observed that Cost function decreases with number of iterations (Fig. 3a), hence the symmetry has been broken successfully as each neuron start learning from different starting point. Initializing weights with small random values does not perform well. Therefore, to initialize weights with high random values, initial weights are multiplied by 10 where biases are assigned to zero. With large random-valued weights, the last layer outputs (sigmoid activation function) results tend towards 0 or 1, indicating different classes. It has been noted in the previous literature that initializing with large random numbers slows down the optimization process. Various studies have also revealed that improper initialization might lead to vanishing or exploding gradients particularly for Deep Net structure [15], [16] and slows down the optimization algorithm of cost function. In this case, we have observed that after 19,000 iterations, loss is 0.3716 and accuracy is 85% for test dataset.
3) **Model with Xavier Initialization:** Xavier initialization uses the scaling factor of $\sqrt{\frac{1}{n_{l-1}}}$ with random initialization from standard Gaussian distribution for the weights $W^l$ of the network where $n_{l-1}$ is the number of nodes in the previous layer. In this case, after 19,000 iterations, loss and accuracy has reached 0.06114 and 93%. Thus, significant improvement could be noted here compared to previous two initialization methods. The same could be observed clearly from Fig. 4b where blue and red dots separated nicely.

4) **Model with He Initialization:** In this initialization method, the weights $W^l$ for the $l^{th}$ layer of the neural net, gets initialized with random numbers generated from standard Gaussian distribution with scaling factor of $\sqrt{\frac{2}{n_{l-1}}}$ where $n_{l-1}$ is the number of nodes in the previous layer ($l - 1$). In this case, after 19,000 iterations, loss is observed to be 0.0437 while accuracy is reported as 96% for test dataset. Therefore, as observed in Fig. 5b, ‘He Initialization’ successfully separates 2 classes and achieves significant improvement compared to Zero and Random initialization techniques.
5) Model with Fibonacci Initialization: In this case, after 19,000 iterations, loss and accuracy have been reported as 0.0966 and 94% respectively (Fig. 6a) for test dataset. This newly proposed model with Fibonacci initialization clearly separates the blue and the red dots (Fig. 6b) with high level of accuracy.

B. Investigation: MNIST Dataset

To evaluate the performance of ANN model, Confusion matrix and Classification Reports have been generated for this multi-class classification scenario. The plotted confusion matrix is a 10x10 matrix where rows represent the real values and columns represent the predicted values. Later, data from the confusion matrix has been used to compute different metrics which are used to quantify the performance of the model. Here a classification report has been generated with Accuracy, Precision, Recall, F1 Score for every class. Each of these reported scores are explained briefly below:

- Accuracy: Accuracy is viewed as the overall prediction performance of a model. It is calculated as the ratio of true prediction to overall prediction. This metric is useful for balanced class distribution scenario; 
  \[ \text{Accuracy} = \frac{TP + FN}{TP + FN + FP + TN} \]
  Note: TP and TN represent True Positive and True Negative values. Also, False Positive and False Negative values are denoted by FP & FN respectively.
- Precision: Precision is viewed as classifier’s exactness. It is calculated as the ratio of true positives (TP) to the sum of true and false positives (TP+FP) for each class; 
  \[ \text{Precision} = \frac{TP}{TP + FP} \]
- Recall: Recall is viewed as the classifier’s completeness or the ability to detect all positive instances correctly. It is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (TP+FN) for each class; 
  \[ \text{Recall} = \frac{TP}{TP + FN} \]
- F1-score: F1 score is used to compare classifier performance. It is a harmonic mean of precision and recall. It’s value ranges between 0 to 1.0 and a high F1-score means high values for both Precision and Recall; 
  \[ F1 - score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]

Classification Report has also been generated with other information like Support (observation count for each class), Macro Average (arithmetic mean of values for considered metrics) and Weighted Average (weighted mean of metrics considering number of observation points for each class). To measure the model performance, ‘Cohen’s Kappa’ (κ) [17] has also been reported.

1) Model with Zero Initialization: From the confusion matrix plot (Fig. 7), it is clear that except diagonal elements, some other elements in matrix are also quite high which signifies that this model predicts wrongly for some classes. These observations are also supported from classification report (Table II) where the overall accuracy is reported as 65.12% and the reported kappa score is 0.5679. It could be easily observed that loss is not decreasing significantly with
TABLE II: Classification Report with Zero Initialization

| Handwritten Digits | Precision   | Recall   | F1-score  | Support |
|--------------------|-------------|----------|-----------|---------|
| Digit 0            | 0.8478      | 0.9477   | 0.8984    | 975     |
| Digit 1            | 0.8748      | 0.9450   | 0.8967    | 913     |
| Digit 2            | 0.6641      | 0.6512   | 0.6576    | 1029    |
| Digit 3            | 0.4087      | 0.7356   | 0.5255    | 1012    |
| Digit 4            | 0.6545      | 0.7896   | 0.7157    | 988     |
| Digit 5            | 0.2731      | 0.0727   | 0.1148    | 886     |
| Digit 6            | 0.6863      | 0.7495   | 0.7163    | 962     |
| Digit 7            | 0.6962      | 0.8395   | 0.7611    | 1025    |
| Digit 8            | 0.4871      | 0.2528   | 0.3329    | 978     |
| Digit 9            | 0.6931      | 0.4796   | 0.5669    | 1010    |
| Accuracy           | 0.6512      | 0.6512   | 0.6512    | 10000   |
| Weighted Avg.      | 0.6285      | 0.6414   | 0.6171    | 10000   |
| Macro Avg.         | 0.6354      | 0.6512   | 0.6261    | 10000   |

2) Model with Random Initialization: The confusion matrix plot corresponding to Random Initialization technique (Fig. 9) clearly indicates significant improvement in performance compared to Zero Initialization method as the diagonal elements are the highest numbers in the matrix. The accuracy is reported here as 90.94% (Table III). Reported loss also has decreased significantly as the number of epochs progresses (Fig. 10). Kappa score is noted as 0.9017.

Fig. 7: Confusion Matrix: Zero Initialization

Fig. 8: Loss and Accuracy: Zero Initialization

Fig. 9: Confusion Matrix: Random Initialization

TABLE III: Classification Report with Random Initialization

| Handwritten Digits | Precision   | Recall   | F1-score  | Support |
|--------------------|-------------|----------|-----------|---------|
| Digit 0            | 0.9339      | 0.9569   | 0.9453    | 975     |
| Digit 1            | 0.9561      | 0.9789   | 0.9673    | 1135    |
| Digit 2            | 0.9214      | 0.9067   | 0.9138    | 1029    |
| Digit 3            | 0.8644      | 0.9071   | 0.8852    | 1012    |
| Digit 4            | 0.9014      | 0.916    | 0.9086    | 886     |
| Digit 5            | 0.8933      | 0.888    | 0.8908    | 886     |
| Digit 6            | 0.8969      | 0.9584   | 0.9266    | 962     |
| Digit 7            | 0.8969      | 0.9249   | 0.9107    | 1025    |
| Digit 8            | 0.9556      | 0.7715   | 0.8534    | 978     |
| Digit 9            | 0.8786      | 0.8743   | 0.8764    | 1010    |
| Accuracy           | 0.9094      | 0.9094   | 0.9094    | 9094    |
| Macro Avg.         | 0.9098      | 0.9082   | 0.9078    | 10000   |
| Weighted Avg.      | 0.9104      | 0.9094   | 0.9087    | 10000   |
3) **Model with Xavier Initialization**: Reported confusion matrix (Fig. 11) and overall accuracy of 90.94% (Table IV) for Xavier Initialization are very similar to that of Random Initialization method. However, the macro and weighted average values (Table IV) have been noted to be higher than the corresponding values reported for Random Initialization techniques. Fig. 12 clearly shows that the loss function progressively decreases with epochs. Also, for this initialization method, high value of 0.9053 has been observed for $\kappa$.

4) **Model with He Initialization**: Though the calculated accuracy is slightly higher (91.34%) for He initialization method (Table V) compared to the same for previously reported techniques (Xavier and Random), there has not been any significance difference observed for Confusion matrix (Fig. 13) and $\kappa$ score which has been reported as 0.9044. Similar to previous methods of Random & Xavier initialization, the loss function for He initialization technique also consistently decreases with number of epochs (Fig. 14).
5) Model with Fibonacci Initialization: The confusion matrix for Fibonacci initialization method (Fig. 15), reveals high level of similarities with the same for He, Xavier & Random methods. Also, comparable values for accuracy (90.21% ; Table VI) and \( \kappa \) (0.8881) have been reported for this newly proposed method. As observed previously for He, Xavier & Random techniques, loss function for Fibonacci initialization method also decreases with progression of epochs (Fig. 16).
V. Conclusion

In this study, a new weight initialization method has been proposed and investigated with two datasets. The results have been compared with standard initialization techniques (Zero, Random, Xavier and He). We have observed that for small (circle dataset) sized datasets, Fibonacci initialization technique performs better than Random and close to Xavier and He initialization methods. Also, for medium sized dataset (MNIST), performance of Fibonacci weight initialization method is comparable with the same for Random, Xavier and He initialization techniques.

Observation, in reverse also leads to the conclusion that Fibonacci weight initialization method tends to perform on par with established techniques, particularly for symmetrical datasets. In continuity to our quest, we will be extending our research to arrive at analytical soundness of results drawn from symmetrical and balanced datasets.

As in this study, we have primarily employed shallow neural net architecture, further investigation needs to be conducted on Deep Net structure to understand the performance of Fibonacci initialization method for DNN and to verify whether it slows down the learning by saturating the output activation function or not.

References

[1] T. Omotehinwa and S. Ramon, “Fibonacci numbers and golden ratio in mathematics and science,” International Journal of Computer and Information Technology, vol. 2, no. 4, pp. 630–638, 2013.

[2] J.-C. Perez, “Chaos, dna and neuro-computers: a golden link,” Speculations in Science and Technology, vol. 14, no. 4, pp. 336–346, 1991.

[3] H. Weiss and V. Weiss, “The golden mean as clock cycle of brain waves,” Chaos, Solitons & Fractals, vol. 18, no. 4, pp. 643–652, 2003.

[4] X. Liang, S. Tong, T. Kumada, and S. Iwaki, “Golden ratio: The attributes of facial attractiveness learned by cnn,” in 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019, pp. 2124–2128.

[5] M. Hassaballah, K. Murakami, and S. Ido, “Face detection evaluation: a new approach based on the golden ratio \( \phi \)," Signal, Image and Video Processing, vol. 7, no. 2, pp. 307–316, 2013.

[6] L. Cuimei, Q. Zhiliang, J. Nan, and W. Jianhua, “Human face detection algorithm via haar cascade classifier combined with three additional classifiers,” in 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI). IEEE, 2017, pp. 483–487.

[7] Y.-C. Chang, “N-dimension golden section search: Its variants and limitations,” in 2009 2nd International Conference on Biomedical Engineering and Informatics. IEEE, 2009, pp. 1–6.

[8] M. W. Guerfala, A. Sifaoui, and A. Abdelkrim, “Data classification using logarithmic spiral method based on rbf classifiers,” in 2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT). IEEE, 2016, pp. 416–421.

[9] ———, “Logarithmic spiral-based construction of rbf classifiers,” INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, vol. 8, no. 2, pp. 263–271, 2017.

[10] N. Luwes, “Fibonacci numbers and the golden rule applied in neural networks,” Interim: Interdisciplinary Journal, vol. 9, no. 1, pp. 33–43, 2010.

[11] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, Neural network design. Pws Pub. Boston, 1996, vol. 20.

[12] S. Hochreiter, “Untersuchungen zu dynamischen neuronalen netzen,” Diploma, Technische Universität München, vol. 91, no. 1, 1991.

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

[14] K. He, X. Zhang, S. Ren, and 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.

[15] S. Koturwar and S. Merchant, “Weight initialization of deep neural networks (dnn) using data statistics. arxiv 2017,” arXiv preprint arXiv:1710.10570.

[16] M. Taki, “Deep residual networks and weight initialization,” arXiv preprint arXiv:1709.02956, 2017.

[17] T. O. Kválueth, “Note on cohen’s kappa,” Psychological reports, vol. 65, no. 1, pp. 223–226, 1989.