Weight Initialization Based ReLU Activation Function to Improve the Performance of a Convolutional Neural Network model.

Abstract: Convolutional Neural Networks (CNNs) have made a great impact on attaining state-of-the-art results in image task classification. Weight initialization is one of the fundamental steps in formulating a CNN model. It determines the failure or success of the CNN model. In this paper, we conduct a research based on the mathematical background of different weight initialization strategies to determine the one with better performance. To have smooth training, we expect the activation of each layer of the CNN model follow the standard normal distribution with mean 0 and standard deviation 1. It prevents gradients from vanishing and leads to more smooth training. However, it was obtained that even with the appropriate weight initialization technique, a regular Rectified Linear Unit (ReLU) activation function increases the activation mean value. In this paper, we address this issue by proposing weight initialization based (WIB) ReLU activation function. The proposed method resulted in more smooth training. Moreover, the experiments showed that WIB-ReLU outperforms ReLU, Leaky ReLU, parametric ReLU and exponential linear unit (ELU) activation functions and results in up to 20% decrease in loss value and 5% increase in accuracy score on both Fashion-MNIST and CIFAR-10 databases.

Keywords: Deep Learning, Convolutional Neural Networks, Weight Initialization, Activation Function, ReLU.

1. Introduction

CNNs are the most widely used models in Deep Neural Networks (DNNs) [1]. They have demonstrated excellent performance in several fields, such as Image Recognition [2], Semantic Segmentation [3], Speech Recognition [4], Object Detection [5] and Natural Language Processing [6]. Since CNN models require great amount of data and computational skills, their success was attributed to dramatic growth in the amount of data as well as an increase in the computational speed [7], [8], [9]. Based on these factors, researchers and practitioners were able to train deeper networks ([10], [11], [12]), which resulted in a higher increase in the number of the CNN models.

However, as the networks get larger, it becomes much more complicated to conduct smooth training. This results in the activation values of each layer being exponentially small or large. These phenomena are called activations gradient vanishing or exploding respectively ([13], [14], [15]). Therefore, in designing the model architecture, it is advisable to take proper precaution since the deeper model does not always mean better performance.

CNN comprises forward and backward passes. Both of them greatly depend on matrix multiplication. In the forward pass, each hidden layer executes multiplication between a specific part of the input and kernel weight matrix. It creates feature maps, which act as inputs for the next layer. Regarding back-propagation [16], the same matrix multiplication operations are executed in the opposite direction. A typical sort of CNN has a large number of layers, thus there are numerous matrix multiplication operations in it. The only available parameter that requires modification is the weight matrix.

Suppose we initialize the weight values from the same standard normal distribution as the inputs, then the model will produce exploding activation values. However, if the initial weight values are small, then we obtain a vanishing activation problem. It will also decrease gradient, which in turn will hinder weight updates and consequently lead to poor learning process [17].

To understand the best strategy for weight initialization that prevents the activations from either exploding or vanishing, we conduct theoretical and practical research. The main contributions of this work are as follows:

- We provide mathematical background to identify the most optimal weight initialization strategy and show its practical implementation in graphs.
- We propose the WIB-ReLU activation function, which performs better for normally distributed values.
- We demonstrate the positive effect of the proposed method on the training process of a CNN model in two different databases.

In the research, we obtain that the most optimal method is to make the activations have standard normal values with mean 0 and standard deviation 1. However, we obtain that the most widely used activation function, which is ReLU, increases the mean value with certain units. In this paper, we address this issue by proposing a method that keeps the activations mean and standard deviation close to 0 and 1 respectively. From the experiments, we obtain that the proposed Weight Initialization Based ReLU (WIB-ReLU) function performs better than regular ReLU in several metrics.

The paper is organized as follows. In Section 2, we present related research works. Section 3 contains the description of the proposed methodology. Section 4 contains the information concerning the experiments and their results. Finally, Section 5 summarizes the work with a conclusion. Directions for further study are suggested as well.

2. Related work

An early method of layer-wise pre-training that is still widely used in a multilayer perceptron was proposed by Bergstra et al. [18], however, it is not suitable for training CNNs to obtain state-of-the-art outcomes.

The first most prevalent weight initialization method was attributed to the great achievement of [10], who used standard normal distribution with mean 0 and a standard deviation of 0.01, as weight values. However, this strategy mainly considers the shallow neural network architectures, therefore the later research works [11] proved inefficiency of [10] weight initialization for deeper models. Also, He et al. [19] emphasized the issue as a result of small or large activation and gradient values that lead to vanishing and exploding of the activation values respectively.

Glorot et al. [20] studied the weight initialization strategy by considering the number of input and output channels. It still performs better in a number of applications, despite the idea is incorrect. He et al. [19] modified the aforementioned method for ReLU activation function and demonstrated its success in very deep network architectures. Although, the proposed method performed well on addressing vanishing gradient problem by maintaining standard deviation values close to 1, it failed to provide 0 mean for the activation values.

Layer-Sequential Unit-Variance (LSUV) [21] is another initialization method that benefits from orthonormality and employs unit variance for the output of each layer. Sussillo et al. proposed a method [22] that stores constant the log of the norms of back-propagated errors called Random Walk Initialization. However, this strategy failed to compete with other works [21]. Moreover, Srivastava et al. presented a weight initialization technique to manage information and gradient movement through the model by training a thousand layer multilayer perceptron model on MNIST [23]. Ahmad et al. [24] presented a scalable and efficient system architecture that selected features by using Artificial Bee Colony that outperformed the existing approaches. Ullaha et al. [25] proposed a technique that detects and classifies optic disc from fundus images by using a trained classification procedure in cloud.

Krahenbuhl et al. [26] proposed data-dependent weight initialization to make all layers train at an equal rate. Romero et al. [27] developed and modified the distillation approach [28], based on replicating teacher network predictions. They permit the training of a student using the outputs as well as representation impacted by the teacher layer and performs better than the previous method with 10 times fewer parameters. Chen et al. [29] developed an efficient gradient normalization algorithm that automatically balances training in deep multitask models by dynamically tuning gradient magnitudes. Lin et al. [30] derived simple natural-gradient updates by using minimal conditional exponential-family representation and obtained a faster convergence compared to the existing gradient-based methods. Shankar et al. [31] presented CROSSGRAD method that used multi-domain training data to learn a classifier that generalizes to new domains. The notable point of this technique was that it did not require an adaptation phase in both supervised and unsupervised learning.

Wu et al. [32] proposed a model to mitigate high variance of gradient estimates in policy gradient methods. The authors derived a bias-free action-dependent baseline for variance reduction. Another work [33] demonstrated exceptional performance of orthonormal matrix initialization for linear models when compared to Gaussian noise. Also, they showed that the method generalizes for the models with non-linearities. Xie et al. [34] designed and analyzed a backward error modulation based on the quasi-isometry assumption between two consecutive parametric layers, which showed distinct improvements in deep neural network models.

3. The proposed methodology

Fig. 1. shows the graphical representation of our method. In the data preparation step, we standardize the dataset by making its values have mean 0 and standard deviation 1. Then, we input the data into our plain model made up of convolution layers followed by the proposed WIB-ReLU activation function. After the input passes three aforementioned combinations of layers with different numbers of kernel sizes, its output goes through the fully connected layer with SoftMax activation function, which produces certain number of values that are equal to the number of classes in the dataset. This technique mitigates gradient vanishing problem in back-propagation, which results in more smooth training, loss reduction and accuracy improvement, when compared to regular ReLU activation function.

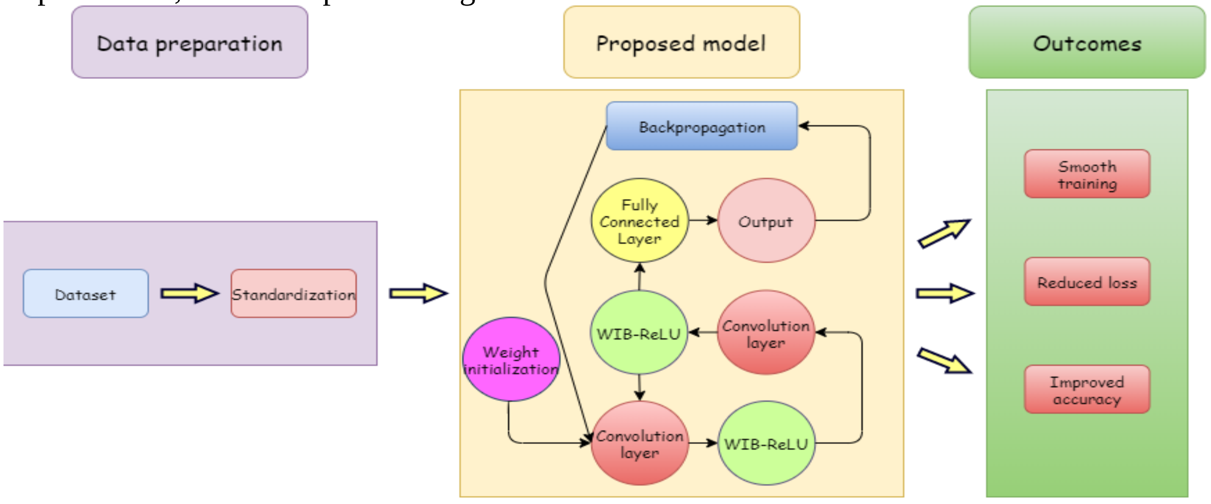


Fig. 1 The proposed method

4. Experiments

Motivation. Correctly initialized weights greatly influence the success of the model that has accurate weight initialization not only results in a competitive model with high accuracy but also increases inference time and decreases computational costs.

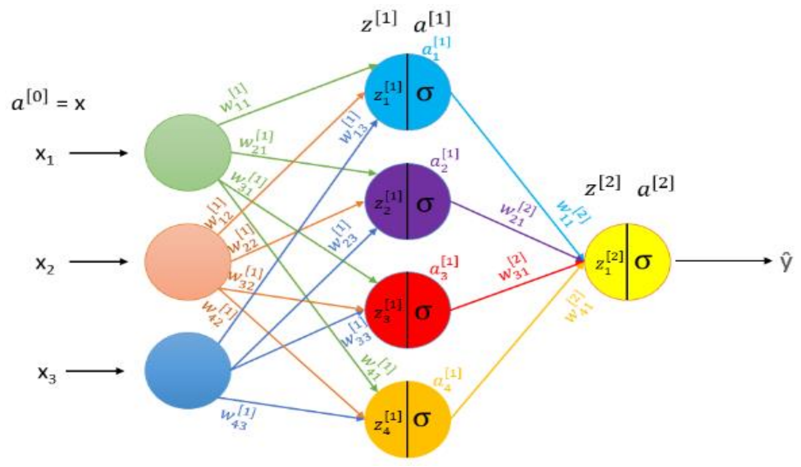


Fig. 2 Simple neural network architecture

Fig. 2 depicts a shallow neural network architecture that we used for the experiments with weight initialization. It is a two-layer neural network with three input units (x_1, x_2, x_3), one hidden layer ($a^{[1]}$) with four units and an output layer with one unit ($a^{[2]}$). The first activation of the first hidden layer, $z_1^{[1]}$ is calculated using Equation (1).

$$z_1^{[1]} = \sum_{k=1}^{n} w_{1k}^{[1]} x_k \tag{1}$$

In the experiments, we standardized inputs to be normally distributed so as to see the change in variance is changed after the first computation.

$$\sigma^2(z_1^{[1]}) = \sum_{k=1}^{n} \sigma^2(w_{1k}^{[1]})\sigma^2(x_k) \tag{2}$$

Using Equation (1), we can write variance of $z_1^{[1]}$ as:

Since both inputs and weights follow a standard normal distribution, variance from every single input and weight equals to 1, and the product of their summation is equal to the product

$$\sigma^2(z_1^{[1]}) = (n\sigma^2(x))(\sigma^2(w^{[1]})) \tag{3}$$

of variances multiplied by the number of variances. Therefore, we can re-express Equation (2) as follows:

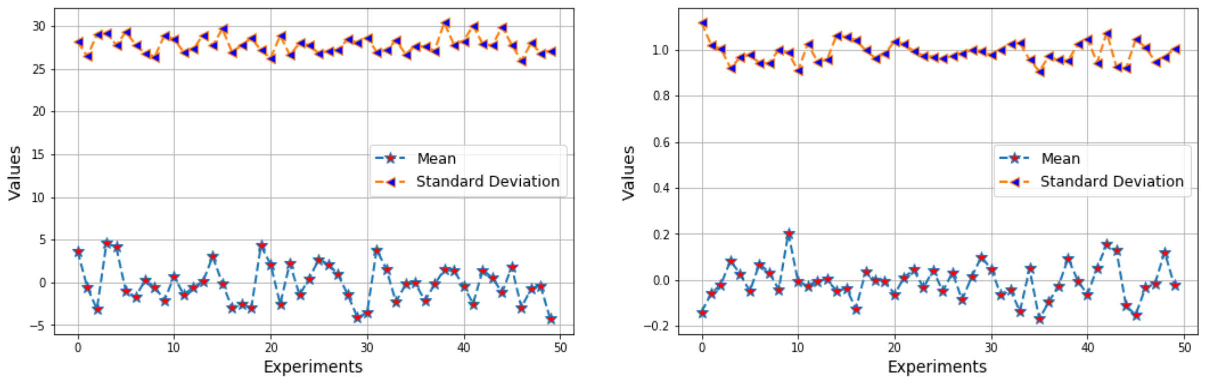
Continuing in this manner for l layers, we obtain the general formula for the variance of the *lth* layer output as follows:

$$\sigma^2\left(z^{[l]}\right) = [n\sigma^2(w)]^{[l]}(\sigma^2(x)) \tag{4}$$

From Equation (4), we ensure that variance of the weights is very close to 1 or exactly 1. Theoretically, we may obtain the variance of 1 for the weights by taking randomly distributed variable *y* and scaling it by square root value of the number of layers *n:*

$$n\sigma^2(w) = n\sigma^2(\frac{y}{\sqrt{n}}) \tag{5}$$

From Equation (5), we obtained that if a value is drawn from a normal distribution and divided it into the number of neurons in the layer, the variance of the weights will be equal to 1.



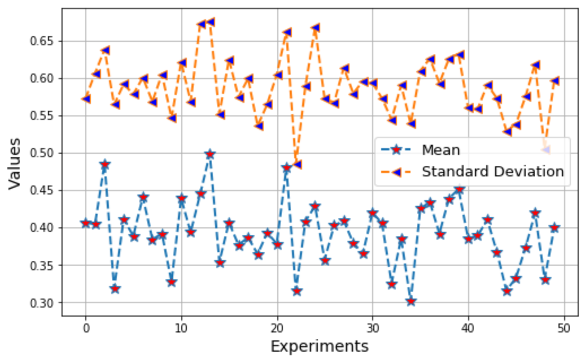
(a) (b)

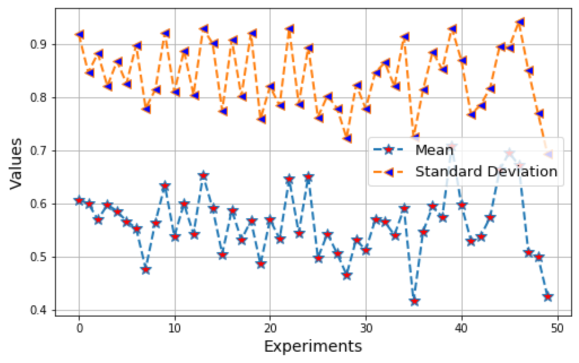
Fig. 3 Mean and standard deviation values of the output from linear matrix multiplication of weights and normally distributed random inputs: (a) standard weight initialization; (b) weight initialization based on Equation (5)

Fig. 3. shows the benefit of weight initialization. From Fig. 3(a), we can observe that even though mean fluctuates around 0, standard weight initialization blows up the standard deviation of the outputs. In 50 experiments, the values range from 25 to 30. However, when weight initialization from Equation (5) is used, we obtain desired outputs for both mean and standard deviation, close to 0 and 1 respectively.

However, we note that the aforementioned experiments were conducted by ignoring activation functions. Although the activation functions are an inextricable part of CNN, we added non-linear function (ReLU) and noticed that the mean and standard deviation of the output were not stable. This is because the ReLU activation function eliminates all negative values replacing them with 0, thus nearly half of the values are set to 0. Consequently, to maintain a standard deviation of 1, we divide the number of neurons into 2 [19]. After a little modification of Equation (5), we get a robust weight initialization method for the output of the activation function.

$$n\sigma^2(w) = n\sigma^2(\frac{2}{\sqrt{n}}) \tag{6}$$





(a) (b)

Fig. 4 Mean and standard deviation values of the output from ReLU activation function: (a) weight initialization based on (5); (b) Kaiming weight initialization

Fig. 4(a) shows the decrease in standard deviation values after applying the ReLU activation function and (b) illustrates that Kaiming weight initialization assists in recovering standard deviation values close to 1. However, we observe that mean values are considerably higher than 0. It should be noted that ReLU activation function addresses a vanishing gradient problem, but it could not keep mean of the activations close to 0 even though the input value was zero centered. Also, it significantly increases the mean value of the activations, which causes a great issue for the network, since non zero-centered activations affect fast convergence. According to the experiments, we compute the average value of the mean increase using Equation (7):

$$\mu_{exp} = \frac{1}{n}\sum_{k=1}^{n}\mu_k \tag{7}$$

Equation (7) shows that the mean value of the activations in different experiments increases a certain amount of value, μ_{act}. Therefore, to deal with the issue of zero-centered activation values, we subtract the mean value of the iteration from the output from ReLU activation function.

$$a^{[l]} = max(0, z^{[l]}) - \mu_{exp} \tag{8}$$

In Fig. 5, we observe that this modification performs better, and the activations are now zero-centered.

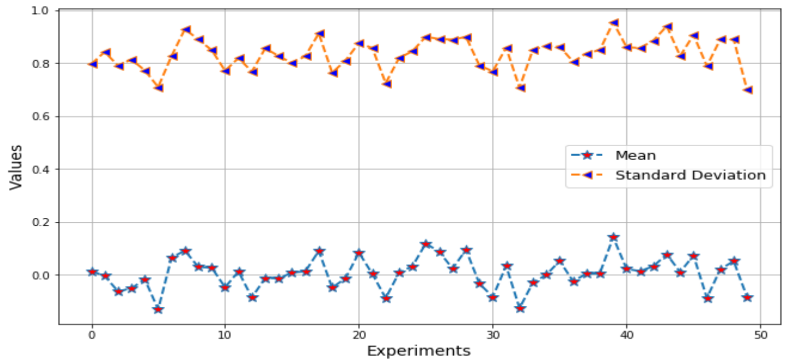


Fig. 5 Mean and standard deviation values of the output from WIB-ReLU activation function

Datasets. We use Fashion-MNIST database [35] and CIFAR-10 dataset [36] for our experiments. The former is quite novel and the classification task is fairly complex when compared to the MNIST database of handwritten digits [23], while the latter contains color larger size and color images. Table 1 provides a general description of both datasets.

TABLE 1
General information about Fashion-MNIST database and CIFAR-10 dataset.

Name	**Number of training examples**	**Number of testing examples**	**Number of classes**	**Image size**
Fashion MNIST database	60,000	10,000	10	28×28
CIFAR-10 dataset	50,000	10,000	10	32×32

Network architecture. To implement our proposed method, we chose a plain network to conduct fast experiments. The network architecture made up of convolutional layers with a regular ReLU activation function is depicted in Fig. 6.

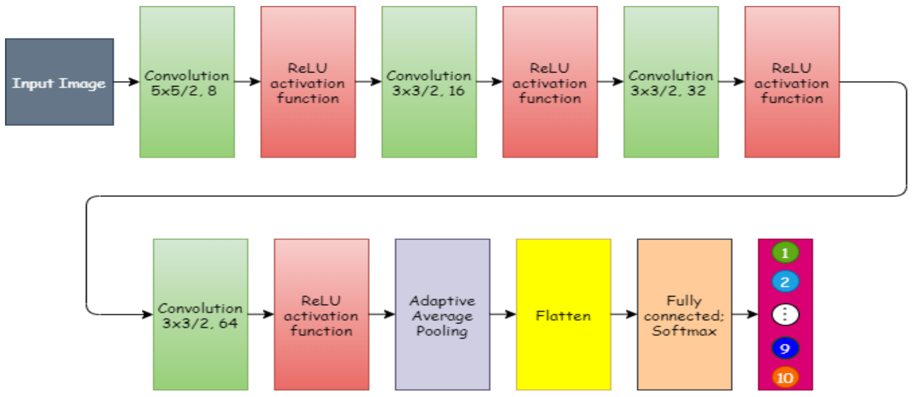


Fig. 6 Plain CNN network for the experiments

Fig. 6. describes a simple CNN model used for the experiments. The input image is transformed into a 28×28 matrix for the Fashion-MNIST database and 32×32×3 for the CIFAR-10 dataset, respectively. We observe that 8 5×5 kernels convolve the first layer with a stride of 2 and padding of 2 followed by ReLU activation function. 2~4th layers have the same parameters, such as stride of 2 and padding of 1. Also, we used 16, 32, 64 3×3 kernels for the second, the third and the fourth convolution layers respectively. The outcome of the 4th layer passed through the Adaptive Average Pooling layer followed by the Flattening layer. Finally, the result goes through 64×10 dense layer outputting 10 values for each 10 classes.

Training parameters. We formulated a deep CNN model using 1.4.0 version of the PyTorch library and conducted experiments using NVIDIA GeForce RTX 2060 SUPER GPU. For all experiments, we initialized weights using Kaiming weight initialization [19] and used Adam optimizer [37] with a momentum of 0.9 and a learning rate of 0.4. Also, loss function was sparse categorical crossentropy and evaluation metrics was accuracy. We also trained the classifier for 20 epochs with a batch size of 256.

Experiment results. After training plain CNN network on the datasets, we studied each layer's mean and standard deviation.

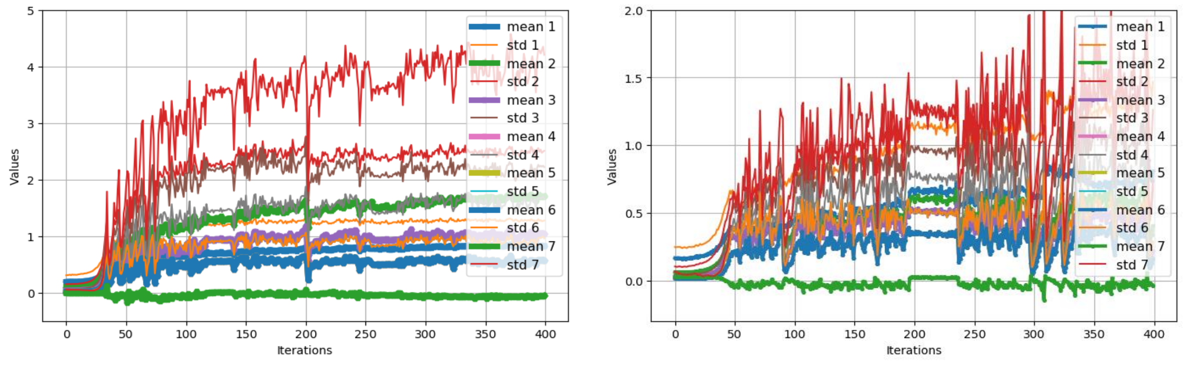


Fig. 7 Mean and standard deviation of the model's each layer activations
(a) Fashion-MNIST database; (b) CIFAR-10 dataset

Fig. 7 illustrates that the training process is not perfect. Regarding (a), the mean values of the activations that are expected to be close to 0 are nearly 1 in all layers except the last layer 7. Also, standard deviations are meant to have values close to 1, but they totally blow up excluding layer 6. The activations of the other layer's standard deviation vary from 2 to 4, which is several times larger than our desired value of 1. Similarly, we observe the same tendency in the case of CIFAR-10 dataset. However, mean and standard deviation values are significantly smaller when compared to the Fashion-MNIST database. In general, both mean and standard deviation values get exponentially large due to a collapse until iteration 150 and only after that time they become stable. Since the standard deviation of the activations are almost equal to 0 at the beginning of the learning phase, it is highly likely that most of the gradient is almost equal to 0 until iteration 200. We can check the percentage of activations with a standard deviation of 0 in Fig. 8.

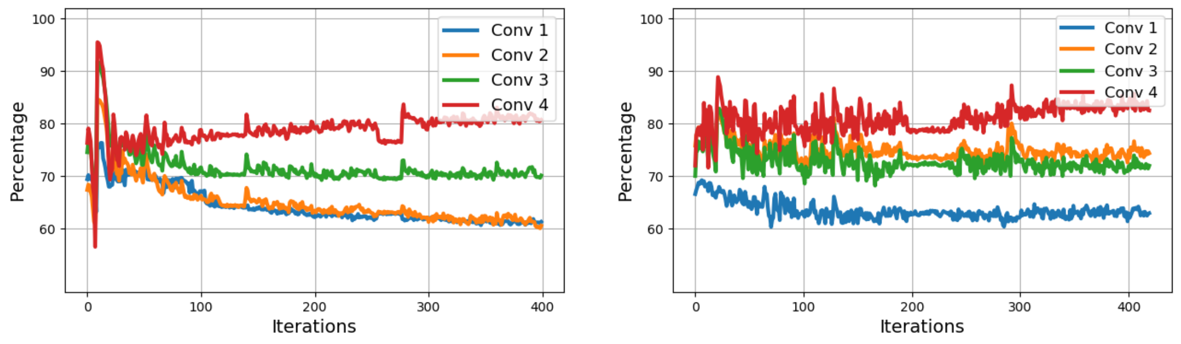
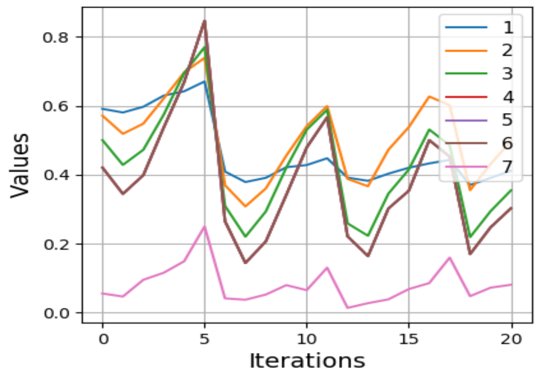


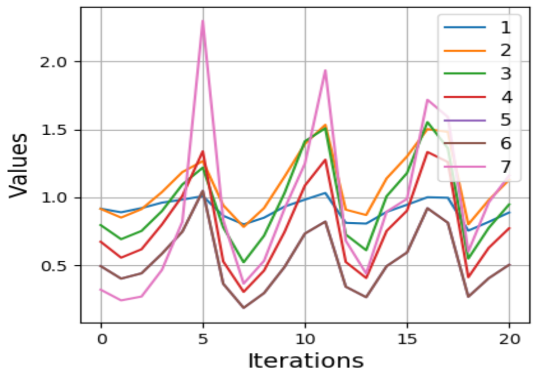
Fig. 8 Percentage of activations with standard deviation of nearly zero in the convolution layers
(a) Fashion-MNIST database; (b) CIFAR-10 dataset

Fig. 8 shows that more than half of the activations are nearly 0 in the first convolution layer. The situation in the subsequent convolution layers is even worse. The amount of 0 activations range from 70 to 90% in the second, third and fourth convolution layers.

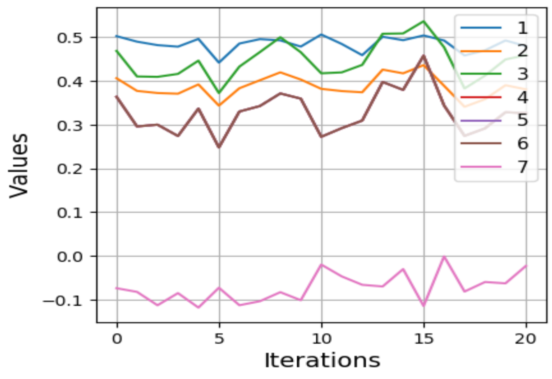
Since our vanilla CNN model was created with the default parameters provided by PyTorch, there is an issue caused by poor weight initialization. Considering the weight initialization technique used as default, we observe that they initialize weight using Kaiming initialization not for normal random numbers but for uniform random numbers. Since uniformly distributed values have ranged from −1 to 1, their standard deviation is 0.57735, which is approximately the root square of 3. However, in PyTorch Conv2d default initialization value (also called as gain) set as root square of 5, which resulted in unsmooth training. We tried to re-implement weight initialization using gain of square root of 3 and the results can be seen in Fig. 9.



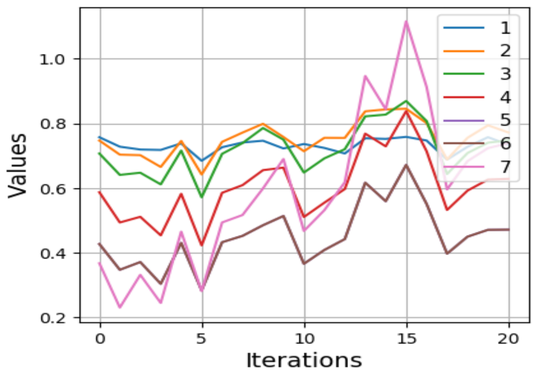
(a)



(b)



(c)

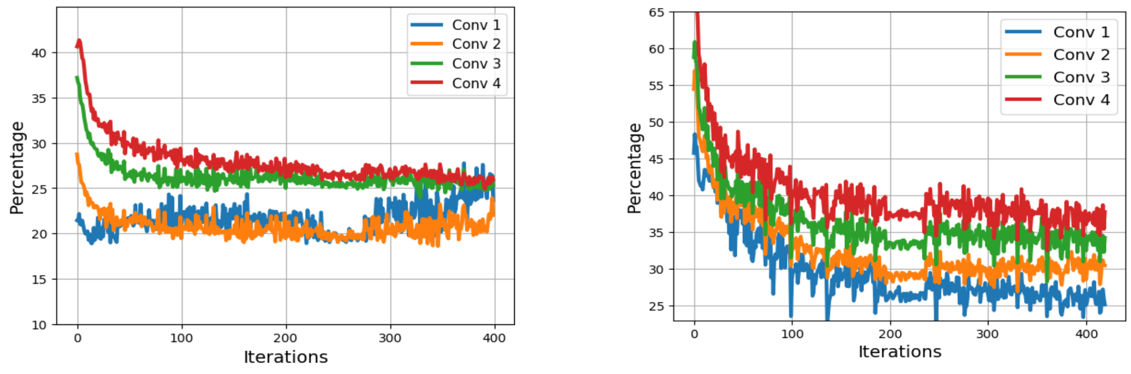


(d)

Fig. 9 Mean (a), (c) and standard deviation (b), (d) of the model's layer activations after modified weight initialization on Fashion-MNIST database and CIFAR-10 dataset, respectively

To demonstrate the situation with mean and standard deviation values in details, we show the first 20 iterations in Fig. 9. It shows that modified weight initialization improves the standard deviation values of the first activations of every layer of the model. With the standard initialization almost all layers had a standard deviation of close to 0. However, after correct initialization the values were very close to 1. But the model still experiences a sharp increase and drastic decrease in both mean and standard deviation of activation values. Also, the mean of the activations starts nearly at 0.5, not at 0 as desired. We address these issues by using the proposed WIB-ReLU.

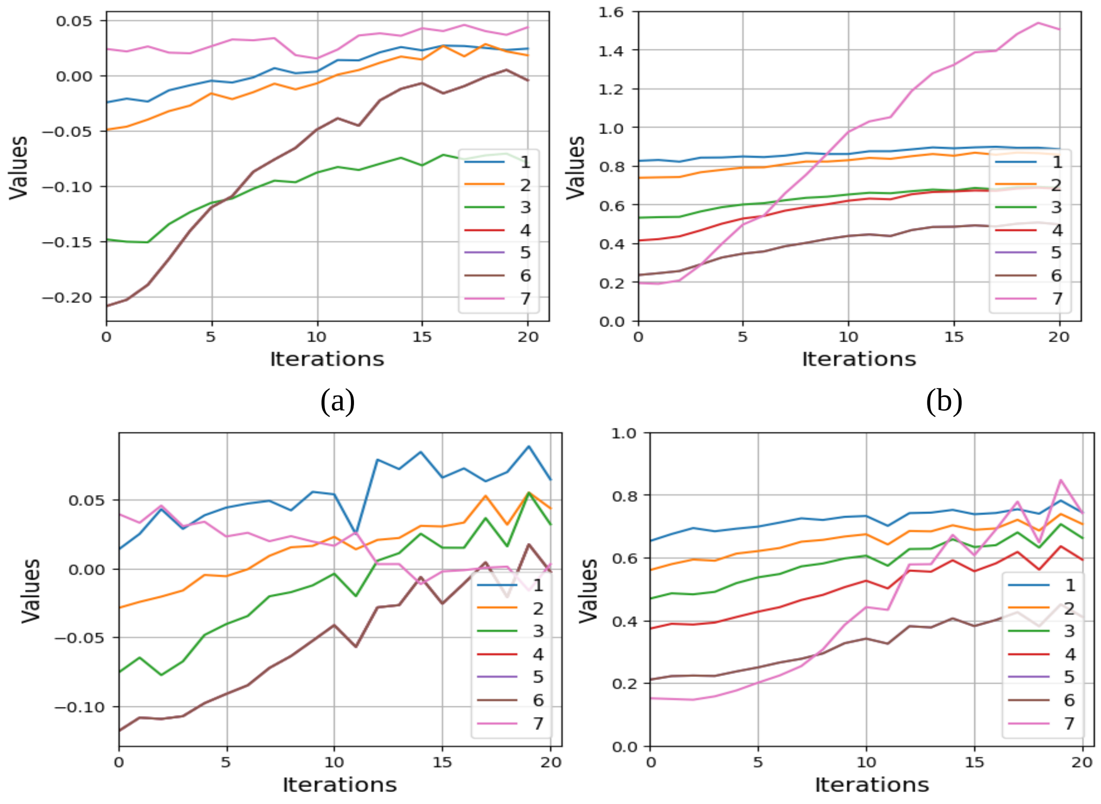
Applying the proposed activation function improves the aforementioned problems. Fig. 10 shows the percentage of activations with a standard deviation of nearly 0.



(a) (b)

Fig. 10 Percentage of activations with standard deviation of nearly 0 in the convolution layers
after implementing the proposed method
(a) Fashion-MNIST database; (b) CIFAR-10 dataset

When compared to Fig. 8, Fig. 10 shows high decline in the number 0 activations. From Fig. 10, we observe that the highest percentage values equal to approximately 40 and 65%, which is contrary to 90% using regular ReLU activation function. Also, we observe that most of the 0 activations range from 20% to 45% when compared to 55%~90% in Fig. 8. We conclude that WIB-ReLU activation function resulted in almost three times decrease in the number of activations with standard deviation of 0.

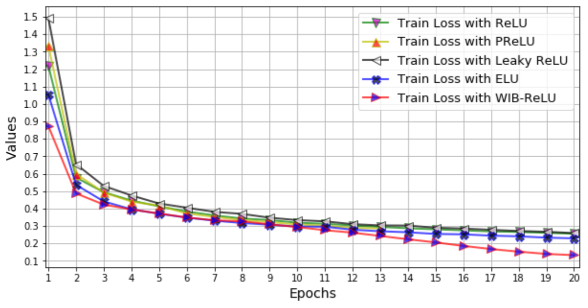


(a) (b)

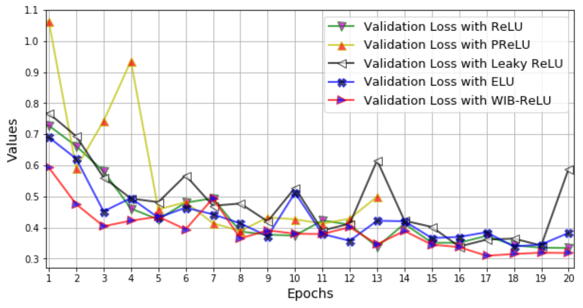
(c) (d)

Fig. 11 Mean (a), (c) and standard deviation (b), (d) of the model's layer activations after implementation
of WIB-ReLU activation function on Fashion-MNIST database and CIFAR-10 dataset, respectively

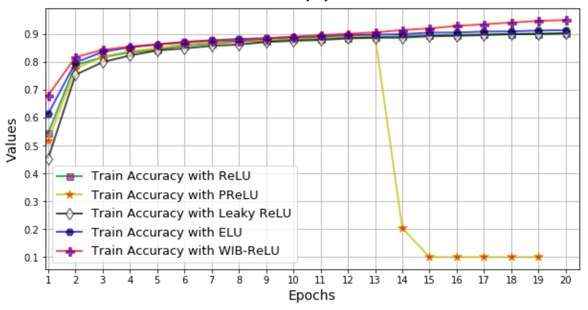
Fig. 11 shows the application of WIB-ReLU solving the problems of high mean of the activations and drastic fluctuations in both mean and standard deviation. In contrast to the results of Fig. 9, mean of all the layers is somewhere around 0 as expected. Furthermore, after implementation of WIB-ReLU, sharp increase due to a fall does not occur anymore. Moreover, the range of standard deviation values is much less when compared to default ReLU.



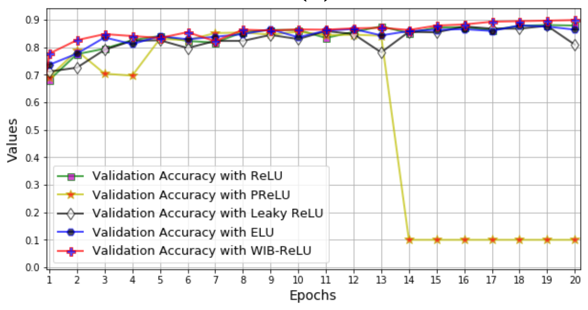
(a)



(b)



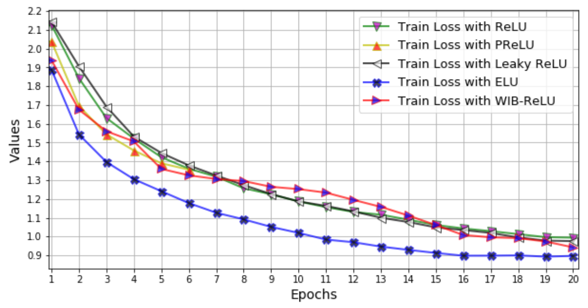
(c)



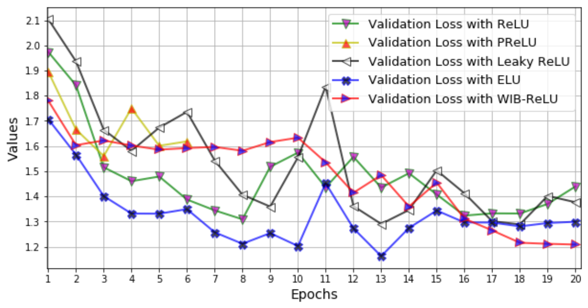
(d)

Fig. 12 Experiment results with the Fashion-MNIST database:
(a) train loss; (b) validation loss; (c) train accuracy; (d) validation accuracy

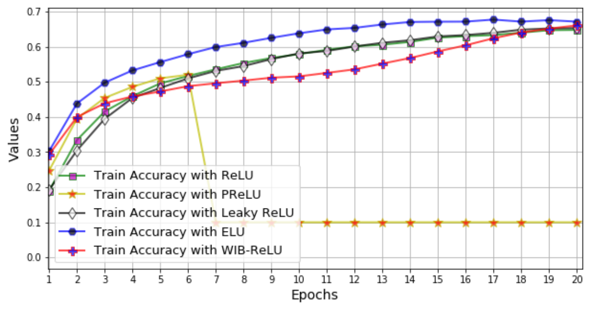
Fig. 12 shows results of the experiments, where we compared train, validation loss as well as train, validation accuracy using the proposed WIB-ReLU and regular ReLU, PReLU, Leaky ReLU, ELU activation functions. The proposed activation function performed better than the other activation functions in all 4 aspects of the comparison. In the training process, the loss value decreased to 0.133, while in the validation stage it declined to 0.318. Regarding the accuracy scores, the model with the proposed activation function attained 0.951 and 0.899 in the training and validation phases respectively.



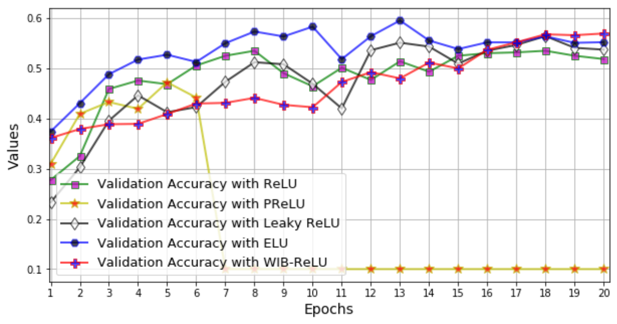
(a)



(b)



(c)



(d)

Fig. 13 Experiment results with the CIFAR-10 dataset:
(a) train loss; (b) validation loss; (c) train accuracy; (d) validation accuracy

As it can be seen from Fig. 13. ELU, the proposed activation function outperformed the other considered activation functions, apart from ELU, which attained slightly lower training loss and negligibly higher training accuracy in comparison with the WIB-ReLU. However, the proposed method performed better in the validation phase by attaining the lowest validation loss value and highest validation accuracy score among the considered activation functions. As validation phase indicators are considerably more important than those of a training stage, we can conclude that WIB-ReLU performed better than the other activation functions in CIFAR-10 dataset too.

5. Conclusion and future work

In this paper, we proposed and analyzed mathematical background of weight initialization in CNNs. We illustrated the results of different strategies and their consequences to the performance of the model. Also, we obtained and proved that default weight initialization in PyTorch Conv2d method performs poorly since it cannot produce variance of 1 for the activations in convolution layers. Thus, it needs to be corrected by fixing the gain hyperparameter. We proposed a WIB-ReLU activation function, which corrects the issue of regular ReLU activation function that was unable to provide 0 mean for the activations. Collecting all components into one model, we obtained approximately 20% decrease in loss value and nearly 5% increase in accuracy score when compared to the models with regular ReLU, PReLU, Leaky ReLU, ELU activation functions on the Fashion-MNIST database and the CIFAR-10 dataset.

Regarding future research, we plan to investigate research on using WIB-ReLU activation function in Recurrent Neural Network (RNN)s. By utilizing the proposed method, we expect to achieve better results not only in image classification, but also in sequence prediction problems, such as image captioning and Speech Recognition.

References

[1] S. Wu, S. Zhong, and Y. Liu, “Deep residual learning for image steganalysis,” *Multimed. Tools Appl.*, 2017, doi: 10.1007/s11042-017-4440-4.

[2] M. Sornam, K. Muthusubash, and V. Vanitha, “A Survey on Image Classification and Activity Recognition using Deep Convolutional Neural Network Architecture,” *2017 9th Int. Conf. Adv. Comput. ICoAC 2017*, pp. 121–126, 2018, doi: 10.1109/ICoAC.2017.8441512.

[3] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, “Image Segmentation Using Deep Learning: A Survey,” pp. 1–23, 2020, [Online]. Available: http://arxiv.org/abs/2001.05566.

[4] A. Kumar, S. Verma, and H. Mangla, “A Survey of Deep Learning Techniques in Speech Recognition,” *Proc. - IEEE 2018 Int. Conf. Adv. Comput. Commun. Control Networking, ICACCCN 2018*, pp. 179–185, 2018, doi: 10.1109/ICACCCN.2018.8748399.

[5] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *IEEE Transactions on Neural Networks and Learning Systems*. 2019, doi: 10.1109/TNNLS.2018.2876865.

[6] D. W. Otter, J. R. Medina, and J. K. Kalita, “A Survey of the Usages of Deep Learning in Natural Language Processing,” vol. XX, no. X, pp. 1–22, 2018, [Online]. Available:

http://arxiv.org/abs/1807.10854.
[7] S. Zerdoumi *et al.*, “Image pattern recognition in big data: taxonomy and open challenges: survey,” *Multimed. Tools Appl.*, 2018, doi: 10.1007/s11042-017-5045-7.
[8] A. Rehman, A. Paul, A. Ahmad, and G. Jeon, “A novel class based searching algorithm in small world internet of drone network,” *Comput. Commun.*, 2020, doi: 10.1016/j.comcom.2020.03.040.
[9] M. M. Rathore, A. Paul, A. Ahmad, M. Anisetti, and G. Jeon, “Hadoop-Based Intelligent Care System (HICS),” *ACM Trans. Internet Technol.*, 2017, doi: 10.1145/3108936.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “2012 AlexNet,” *Adv. Neural Inf. Process. Syst.*, 2012, doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.