International Journal of Smart Computing and Artificial Intelligence
International Institute of Applied Informatics
2024, Vol. 8, No. 2, IJSCAI846

Rotation Weight Update: Another Way Different from Dropout to Introduce Lazy Neurons in Image Recognition Tasks

Tetsuya Hori [*] , Yuki Sekiya [†] , Yoichi Takenaka [*]

Abstract

Numerous approaches have been developed to enhance the capabilities of deep learning in image recognition. We propose a method called ”Rotational-update,” which cyclically updates the weight of neurons. This approach segments the neurons in a fully connected layer into groups of equal size, each containing $\sqrt{N}$ neurons, with N representing the total neuron count in the layer. It selectively updates the weights of one group at a time per mini-batch. This selective updating mechanism aims to curb excessive learning, potentially reducing overfitting and enhancing validation accuracy. A notable aspect of this method is its compatibility for concurrent use with other techniques, including batch-normalization and dropout.

We used the CIFAR10 dataset for image recognition tasks to validate the method's efficacy, employing three neural network architectures: VGG-16, ResNet-110, and ResNet-152. Our findings indicate that integrating our proposed method with batch normalization outperforms the accuracy of the combination of dropout and batch normalization. Specifically, the proposed Rotational-update method achieved an accuracy improvement of up to 5 percentage points in VGG-16 and one percentage point in ResNet-110 compared to traditional methods. Thus, we deduce that substituting dropout with our proposed method enhances image recognition task performance and reduces overfitting.

Keywords: Deep Learning, Drop out, Image Recognition, Rotational-update

1 Introduction

Since Hinton et al.'s victory in the ILSVRC 2012 competition [12], deep learning has emerged as a prominent machine learning technique. This method involves using complex, multi-layered artificial neural networks designed to replicate the function of the human cerebral cortex. Deep learning finds diverse applications in several fields, including image recognition [12], natural language processing [21], audio processing [2], and computational biology [3], demonstrating its wide-ranging impact and utility.

[*] Kansai University, Osaka, Japan
[†] Amazon WEB service Japan, Tokyo, Japan

Object detection, a subset of image recognition, focuses on identifying and locating objects within an image. It is crucial for automating tasks like self-driving cars. Various methods, including R-CNN, have been developed for real-time object detection [6]. Later, Fast R-CNN [5] and Faster R-CNN [18] improved R-CNN's speed. Redmon then introduced YOLO (You Only Look Once), which further accelerated the process, enabling quicker and more efficient detection of multiple objects by employing grid cells [15][16][17][1].

Image Generation, a facet of image recognition, involves creating new images not found in the training dataset. Generative Adversarial Networks (GAN), introduced by Goodfellow [7], and its extension, Deep Convolutional Generative Adversarial Networks (DCGAN) [14], are vital algorithms for this task. This technology has been widely applied in various fields, including super-resolution by Ledig et al. [13] and artistic style transformation by Zhu et al.[24]. These advancements and applications have made deep learning an indispensable tool in the field of image recognition.

In the field of deep learning, foundational research is crucial. The key to building an effective neural network is carefully adjusting and optimizing hyperparameters, including structural and training variables [9][23]. Structural variables typically involve the number of hidden layers and the choice of activation functions. In contrast, essential training variables encompass the learning rate, the total number of training epochs, batch size, and momentum settings.

Batch Normalization, developed by Loffe and Szegedy, normalizes layer inputs by adjusting their center and scale [10]. This technique enhances the network's resilience to initial weight values, accelerating training and boosting performance and stability. Dropout, introduced by Srivastava et al., randomly omits neurons in each mini-batch during training, effectively preventing overfitting and enhancing validation accuracy [20]. Momentum, on the other hand, utilizes gradients from previous steps to stabilize neuron output fluctuations.

We propose a " Rotational-update " method, which cyclically updates neuron weights. The method divided neurons in a fully connected layer into $\sqrt{N}$ equally sized groups, with N being the total neuron count in the layer. For each mini-batch, only one of these groups undergoes a weight update. This selective updating mechanism aims to curb excessive learning, potentially reducing overfitting and enhancing validation accuracy.

While Rotational-update is akin to dropout, their differences are notable. In a neural network's training, encompassing forward propagation, backpropagation, and weight updating, Rotational-update functions solely during weight updating without influencing the other two phases. All neurons partake in both forward and backpropagation. Conversely, dropout excludes neurons from all three phases. Another distinction lies in neuron selection; Rotational-update assigns neuron groups before training, while dropout selects neurons randomly.

A parallel strategy also exists, restricting the number of neurons engaging in weight adjustment within the Hopfield neural network. This network operates on a single-layer framework, with all neurons interlinked. This approach, referred to as "N-parallel mode," has shown enhancements in resolving combinatorial optimization problems [22][4]. However, its application in convolutional networks and the realm of deep learning remains unexplored.

The remainder of this paper is organized as follows. Section 2 overviews deep neural networks, including their structure and learning process. Section 3 introduces the proposed Rotational-update method, detailing its mechanism and comparison with dropout. Section 4 presents the experiments to evaluate the Rotational-update method, including conditions, results, and discussion. Section 5 concludes the paper by summarizing the findings and discussing future research directions.

2 Deep Neural Network

Typically, a Deep Neural Network (DNN) is composed of several layers: convolutional, pooling, and fully connected layers. Each layer consists of a group of neurons functioning in unison at a particular neural network's structure level.

2.1 Layers

A fully connected layer in a deep neural network is a critical component where all neurons from a previous layer are connected to every neuron in the next layer. This structure enables the neural network to capture complex patterns and relationships in the data, as each neuron in the fully connected layer can receive signals from all neurons in the preceding layer. DNNs often use the fully connected layers towards the end of the network, especially in tasks like image classification, where they interpret and combine features extracted by earlier layers to make final predictions or classifications. The fully connected layer's ability to integrate learned features across the entire network is a crucial factor in the overall performance of deep learning models.

Let W be the weights and b for biases. With X as the input from the preceding layer, the layer generates output Y, depicted in equation (1). In the equation, $f()$ is a non-linear transformation such as sigmoid, tanh, and ReLU. The final fully connected layer applies a non-linear transformation tailored to its specific function, such as using the softmax function in classification tasks.

$$Y = f(X \cdot W) \tag{1}$$

A convolutional layer in a deep neural network is primarily used for processing data with a grid-like topology, such as images. This layer consists of a set of learnable filters (or kernels) that slide (convolve) over the input data to produce a transformed output. As each filter moves across the input, it performs element-wise multiplication with the part of the input it covers and sums up the results, producing a feature map. This process helps the network detect spatial hierarchies in the data by learning from local spatial features, making convolutional layers especially effective for tasks like image and video recognition and other areas involving spatial data.

A pooling layer in a deep neural network reduces the spatial dimensions (height and width) of the input image or feature map. It works by partitioning the input image into non-overlapping rectangles and summarizing the features within each rectangle. The most common forms are max pooling, which takes the maximum value from each rectangle, and average pooling, which calculates the average value. This layer decreases the computational load, memory usage, and the number of parameters, helping to reduce overfitting by providing an abstracted representation.

2.2 Neural Network Architectures

A neural network architecture is the structural framework of a neural network, defining the arrangement and interconnections of layers and neurons within the network. It includes the type, sequence, and configuration of layers and the methods of processing data through these layers.

AlexNet, VGG, and ResNet are distinguished neural network architectures in image recognition.

AlexNet, pioneering in its structure, comprises five convolutional layers and three fully connected layers and is recognized for introducing key elements like ReLU activation and dropout [12].

VGG, identified for its uniform design, utilizes repetitive blocks of convolutional layers, culminating in 16 or 19 layers in its most known forms [19]. They performed better than AlexNet [19].

ResNet, famous for its deep structure facilitated by residual connections, effectively addresses the vanishing gradient issue, allowing for the training of networks with an unprecedented depth of up to 152 layers [8]. These architectures have significantly advanced the capabilities of deep learning models, particularly in image processing tasks.

2.3 Learning Process

The learning process for Deep Neural Networks (DNNs) involves a three-step procedure: forward propagation, backpropagation, and the updating of weights. Detailed explanations of these critical stages in the learning process are provided in subsequent sections.

2.3.1 Forward propagation

Forward propagation in a neural network refers to the process where input data traverses from the initial layer through the network in a forward direction. It begins at the first layer, where each subsequent hidden layer receives and processes the data based on its specific function, and then transfers the processed output to the next layer. This propagation continues sequentially through the network until the final layer is reached, resulting in the final output of the neural network.

2.3.2 Backpropagation

Backpropagation assesses the deviation between the neural network's output and the actual data label by determining the loss function's gradient. This algorithm processes one layer at a time, moving backward from the end layer. It uses dynamic programming to streamline the process, reducing the need for recalculating intermediate variables in the chain rule during its reverse iteration through the network.

2.3.3 Weights updating

Neuron weights are updated using the computed gradients as per the equation (2):

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} \tag{2}$$

,where W is weights, $\frac{\partial L}{\partial W}$ is gradients, and η is learning rate, respectively.

Additionally, when momentum is incorporated, the weight update becomes equation (3).

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} + \alpha \Delta W \tag{3}$$

where ΔW is previous updated value and α is momentum parameter.

2.4 Techniques at Learning

2.4.1 Minibatch

Batch learning utilizes the entire training dataset simultaneously, whereas minibatch learning updates neuron weights using smaller portions of the training set. The training data are randomly divided into several subsets, known as minibatches. This approach enables the model to learn incrementally from these subsets, assisting in escaping from local optima during training.

2.4.2 Dropout

In DNN model training, overfitting is a common issue where models perform poorly on new data despite fitting well to the training set. Dropout, introduced to mitigate overfitting, randomly omits neurons within a layer during training [20].

This technique involves a probability parameter p. For a layer with m neurons receiving an n-sized input and producing a non-linear output $Y_{n\times m}$, Dropout generates a 'mask' matrix $mask_{n\times m}$ with Bernoulli random variables. The layer then forwards $\tilde{Y}$, an element-wise product, as detailed in (5).

$$mask_{n\times m} \sim \text{Bernoulli}(p) \tag{4}$$

$$\tilde{Y} = Y \circ mask \tag{5}$$

During the inference phase with the model, the Dropout technique does not drop neurons. Instead, to incorporate all neurons, the output from each neuron is scaled by multiplying with the probability p. This scaling ensures that the average output during inference aligns with what was learned during the training phase.

2.4.3 Batch Normalization

Training Deep Neural Networks (DNNs) often face the challenge of internal covariate shift, where distributional changes in the activation functions occur due to weight adjustments, impacting training efficiency, effectiveness, and DNN stability. This issue intensifies with an increase in layers. Batch normalization, which normalizes the inputs by adjusting their mean and variance, thereby maintaining a more stable distribution across different layers in the network, was proposed to mitigate this [10].

In Batch Normalization, each layer initially sets parameters $\gamma = 1$ and $\beta = 0$. During forward propagation, the layer input transforms Batch Normalization as defined in (6), using u_{out} as the input value.

$$u_{out} = \gamma \hat{u} + \beta \tag{6}$$

, where $\hat{u}$ is the normalized value by re-centering and re-scaling.

During backpropagation and weight update phases, the method calculates the gradients for γ and β and then updates γ and β.

3 Rotational-update

We propose a method called *Rotational-update* that limits the number of neurons updated during the learning process. In a fully connected layer with N neurons, this method initially divides these neurons into $\sqrt{N}$ groups, forming a set $G = g_1, \dots g_{\sqrt{N}}$. Each group contains $\sqrt{N}$ neurons, ensuring each neuron is exclusively one group member.

Throughout the learning phase, this method updates neuron weights for only one group, g_i, from the set $G = g_1, \dots g_{\sqrt{N}}$ per minibatch. The selection of groups occurs sequentially, cycling through from g_1 to $g_{\sqrt{N}}$, and upon reaching $g_{\sqrt{N}}$, it restarts with g_1.

Coping with momentum

The momentum technique in neural networks is proven to enhance training speed and accuracy. This paper outlines how to integrate momentum effectively with Rotational-updates. The approach is straightforward: apply each method independently. While momentum is applied across all neurons, Rotational-update targets the weight gradient component. When Rotational-update selects a neuron group g_i, the following equations define the weight update.

neuron in g_i

$$W \leftarrow W - \eta \frac{\partial L}{\partial W} + \alpha \Delta W \tag{7}$$

neuron not in g_i

$$W \leftarrow W + \alpha \Delta W \tag{8}$$

Rotational-update in multiple layers

Initially explained for a single layer, Rotational-update is also applicable to multiple layers in a neural network. It can be implemented independently across various layers. The technique creates distinct neuron groups, G, for each layer. Since the number of neurons varies across layers, the count of these groups differs accordingly. Consequently, the method independently selects a group from each layer' s set of groups, G, to facilitate Rotational-update across different layers in the network.

Differences with Dropout

Rotational update and dropout limit the number of neurons that update their weights, but significant differences exist between them. The main distinction lies in the application phase of neuron selection and weight update within the learning process. This differentiation is critical for understanding how each method impacts the neural network's learning and performance.

The fundamental distinction between Rotational-update and Dropout lies in the specific stage of the learning process where neuron numbers are limited. Rotational-update only restricts neurons during the weight update phase, meaning all neurons participate in forward propagation and backpropagation. In contrast, neurons excluded via Dropout are not involved at any stage of the learning process.

Additional distinctions between dropout and Rotational-update include their approaches to neuron selection. Dropout randomly removes neurons, while Rotational-update predefines neuron groups before learning and uses these groups consistently throughout the process.

Although Rotational-update and dropout are similar, they can be combined in a layer as follows.

Before training, Rotational-update organizes neurons into groups (G), and dropout sets its drop probability. During forward and backpropagation, only Dropout functions. In the weight updating phase, neurons get updated only if they are in a group (g_i in G) and not being excluded by dropout, where Rotational-update selects g_i.

4 Experiments

We conducted three experimental approaches to assess the efficacy of Rotational-update. The first approach involved integrating this method into an existing image classification neural network and monitoring its test accuracy over 100 epochs. Next, we compared the performance of Rotational-update against other leading methods regarding test accuracy. In the final phase of our study, we examined the performance of our method when synergistically combined with state-of-the-art techniques. For the benefit of the academic community and to foster further research, the source codes pertaining to these experiments have been made publicly accessible on the GitHub platform.`https://github.com/ryhoh/Rotational_update/tree/bn_dropout`. The proposed method is also provided as a library on pip (pip install rotational_update).

4.1 Rotational-update Only

4.1.1 Conditions

We integrated the Rotational-update method into the VGG-16 convolutional network architecture [19] to classify the CIFAR10 image set [11]. VGG-16 is a convolutional network architecture having 13 convolutional layers, followed by three fully connected layers. CIFAR10 is a commonly used dataset in machine learning comprising 60,000 images of 32×32 size across ten classes, where 50,000 are for training (5,000 images/class) and 10,000 for the test.

Adjustments were made to VGG-16 and the CIFAR10 dataset. We resized the images in CIFAR10 to 224×224 using bicubic interpolation and normalization, as well as modifying the output layer of VGG-16 to represent the ten classes in CIFAR10.

The Rotational-update was explicitly applied to the fully connected layers of VGG-16, excluding the final classification output layer. The Rotational-update divided 4096 neurons in each fully connected layer into $\sqrt{4,096} = 64$ groups, each containing 64 neurons. We used the following hyperparameters to train the neural network: a learning rate of 0.001, a momentum rate of 0.9, and a minibatch size 32. We trained them for 100 epochs.

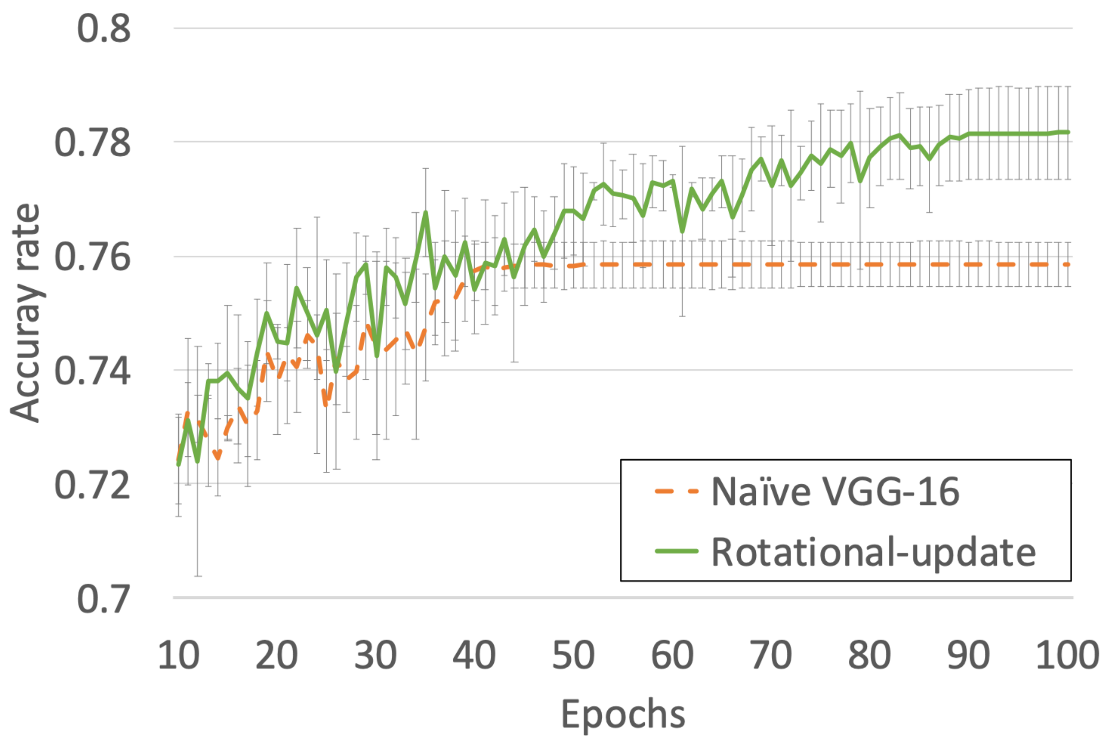

Figure 1: Transitions of testing accuracy rate on CIFAR10. Dashed line denotes previous method, and solid line denotes rotational-update. The whiskers indicate $\pm$ SD.

4.1.2 Results

Figure 1 illustrates the test accuracy progression for the naïve VGG-16 and VGG-16 with Rotational-update. The dashed and solid lines represent the accuracy of the the naïve and VGG-16 with Rotational-update, respectively. The lines' whiskers depict the $\pm$ SD sampling standard deviation over four runs.

Initially, the naïve VGG-16's accuracy increases rapidly, but by the 10th epoch, it levels with the Rotational-update model. Post this point, while the naïve VGG-16's accuracy shows marginal improvement, the Rotational-update model demonstrates a steady, albeit slight, increase. By the 100th epoch, the Rotational-update model outperforms its counterpart.

Table 1: Testing Accuracy of six models at the 50th epoch.

	Accuracy
1) Naïve VGG-16	0.757
2) Rotational-update	0.772
3) Dropout	0.778
4) BN (Fully connected)	0.822
5) BN (Convolution)	0.866
6) BN (Fully connected and Convolution)	0.819

4.2 Comparison with Other Methods

We conducted a comparison of Rotational-update with dropout and batch normalization techniques. Like Rotational-update, dropout restricts the number of neurons updated, as we mentioned before. Batch normalization is a crucial method for enhancing the efficacy and stability of convolutional network architectures.

4.2.1 Conditions

We established a set of six distinct neural network models for comparative analysis to evaluate the image recognition accuracy as outlined in section 4.1.1. These models are described as follows.

1. Naïve VGG-16
2. VGG-16 with Rotational-update

The two models are the same used in section 4.1.1.

3. VGG-16 with dropout at the two fully connected layers except the last.
4. VGG-16 with batch normalization in the fully connected layers except the last
5. VGG-16 with batch normalization in all the convolutional layers
6. VGG-16 with batch normalization in the fully connected layers except the last and all the convolutional layers

In this part of the research, while the hyperparameters remained consistent with those detailed in section 4.1.1, the duration of the training was set at 50 epochs. The model accuracies were compared at the 50th epoch, marking the point where further accuracy improvements had ceased.

4.2.2 Results

Table 1 presents the test accuracies of six models at their 50th epoch. The abbreviation ”BN” within the table stands for Batch Normalization.

The table reveals that VGG-16 equipped with batch normalization in its convolutional layers attained the highest level of accuracy. The Rotational-update model outperformed the naïve VGG-16, yet it did not surpass the accuracy of the remaining four models.

4.3 Combinational Use

It is possible to implement Rotational-updates, dropout, and batch normalization concurrently in a neural network architecture, prompting an evaluation of their combined efficacy. The neural network architecture incorporating batch normalization in its convolutional layers demonstrated superior performance as indicated in table 1 of section 4.2. Therefore, we verified the performance improvement by introducing Rotational-updates, dropouts, or both into the neural network architecture with batch normalization.

4.3.1 conditions

We used four types of neural network models as follows.

1. Naïve neural network architecture with batch normalization in all the convolutional layers (BN model)

2. BN model with Rotational-update

3. BN model with dropout

4. BN model with rotational-update and dropout

We implemented Rotational-update and dropout in the two fully connected layers, excluding the final one.

We employed Resnet-110 [8] and Resnet-152 [8] as naïve neural network architectures in addition to VGG-16. ResNet-110 and ResNet-152 are variations of the ResNet (Residual Network) architecture used in deep learning. These architectures are known for their depth, with ResNet-110 having around 110 layers and ResNet-152 having approximately 152. The critical feature of ResNet architecture is using ”residual connections” or ”skip connections,” which help mitigate the vanishing gradient problem in deep networks. These connections facilitate direct gradient passage across the network, enabling it to skip over one or more layers.

The hyperparameters remained as outlined in section 4.1.1, with a modification to the number of epochs, set at 50, aligning with section 4.2.1. We compared model accuracies at the 50th epoch, identified as the point at which accuracy enhancements ceased.

4.3.2 Results

Table 2 displays the testing accuracy of four types of combinational use on VGG16, ResNet-110, and ResNet-152. The table lists the average accuracy and the sample standard deviation, calculated from ten repetitions of the image classification task for each architecture.

For each neural network, we observed the best combination of methods. For VGG16 and ResNet-110, the neural network with both Rotational-update and dropout resulted in the highest accuracy. In the case of ResNet-152, the best performance was achieved with Rotational-update and without dropout. The second-best performance was obtained with dropout but without Rotational-update, indicating that using Rotational-update and dropout exclusively was more beneficial in this case.

Table 2: Testing Accuracy of the combinational models

	without Rotational Update		with Rotational Update	
Naïve NN	without Dropout	with Dropout	without Dropout	with Dropout
VGG16	0.7633 ± 0.0064	0.8197 ± 0.0037	0.7897 ± 0.0052	0.8680 ± 0.0057
ResNet-110	0.7534 ± 0.0100	0.7620 ± 0.0071	0.7576 ± 0.0103	0.7706 ± 0.0060
ResNet-152	0.7961 ± 0.0089	0.8019 ± 0.0082	0.8026 ± 0.0106	0.7984 ± 0.0085

4.4 Discussion

The three experiments have shed light on various characteristics of the Rotational-update method.

As detailed in Section 4.1.1, this method impacts image recognition tasks significantly when applied independently. It enhances accuracy in distinguishing images but at the expense of slower learning rates. The decrease in learning speed aligns with our expectations because Rotational-Update decreases the number of neurons updated in each mini-batch. The increase in final accuracy is linked to preventing overfitting, a principle also underlying the effectiveness of Dropout [20].

In Section 4.2.1, a comparison was made between Rotational-update, dropout, and batch normalization. Batch normalization emerged as the most accurate, followed by dropout, while Rotational-update method showed the most minor performance. However, we can use all three methods simultaneously.

In Section 4.3.1, an experiment was conducted to test the combined effectiveness of these methods. The result shows that using Rotational-update and batch normalization together yielded the highest performance. Conversely, pairing Rotational-update with dropout sometimes led to decreased accuracy, despite both methods improving accuracy when used alone or with batch normalization.

The results from the experiments suggest that Rotational-update could substitute dropout to enhance the performance of neural networks. Consequently, this method is anticipated to become a fundamental technology in deep neural networks.

5 Conclusion

In our research, we concentrated on the weight updating process in deep learning training, introducing the concept of Rotational-updating. This technique involves rotating the weight updates among neuron groups. It divides Neurons in a fully-connected layer into $\sqrt{N}$ groups of equal size, where N is the total number of neurons. Only the weights of neurons in one group are updated during each mini-batch. A notable aspect of this method is its compatibility with other techniques like batch normalization and dropout.

Rotational-update, similar to dropout, has distinct differences, which were clarified. Training a neural network involves a three-stage process: the initial phase of forwarding propagation, followed by backpropagation and concluding with updating neuron weights. Rotational-update operates exclusively during the neuron weight update without impacting the first two stages. Thus, all neurons participate in both forward and backpropagation. In contrast, Dropout excludes neurons from all three stages. Additionally, neuron selection differs: Rotational-update pre-determines neuron groups before training, whereas Dropout randomly selects neurons.

We employed the three neural networks VGG16, ResNet-110, and ResNet-152 for CIFAR-10 image classification to assess the Rotational-update's effectiveness. This experiment revealed that integrating Rotational-update with Batch Normalization led to the highest performance. However, combining Rotational-update with Dropout resulted in a reduction of accuracy in ResNet-152. Based on these findings, we deduced that Rotational-update could be a more effective alternative to dropout in improving neural network performance.

In this study, we applied the Rotational-update method to relatively small-scale neural networks. In future work, we plan to verify its effectiveness on larger-scale neural networks to assess its potential for broader applications. Additionally, in Rotational-update, we divided N neurons in the fully connected layer into $\sqrt{N}$ groups, which is analogous to the dropout rate in the dropout technique. In dropout, tuning the dropout rate is essential for performance improvement, and similar adjustments may be necessary for Rotational-update. Therefore, we intend to investigate how changing the number of groups affects performance, examining whether further optimization can enhance the efficacy of our method.

Acknowledgments

This research was financially supported by the Kansai University Fund for Research 2000, and Domestic and Overseas Research Fund 2021.

References

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[2] W. Dai, C. Dai, S. Qu, J. Li, and S. Das. Very deep convolutional neural networks for raw waveforms. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 421–425. IEEE, 2017.

[3] N. Erfanian, A. A. Heydari, A. M. Feriz, P. Iañez, A. Derakhshani, M. Ghasemigol, M. Farahpour, S. M. Razavi, S. Nasseri, H. Safarpour, et al. Deep learning applications in single-cell genomics and transcriptomics data analysis. *Biomedicine & Pharmacotherapy*, 165:115077, 2023.

[4] N. Funabiki, Y. Takenaka, and S. Nishikawa. A maximum neural network approach for n-queens problems. *Biological Cybernetics*, 76(4):251–255, 1997.

[5] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[9] I. Ilievski, T. Akhtar, J. Feng, and C. Shoemaker. Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31-1, 2017.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

[11] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[13] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[14] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[16] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[17] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[18] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[21] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[22] S. N. Yoichi Takenaka, Nobuo Funabiki. Maximum neural network algorithms for n-queen problems. *Journal of Information Processing (Information Processing Society of Japan)*, 37(10):1781–1788, 1996.

[23] T. Yu and H. Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.