Reducing Run-to-Run Variability in Neural Networks: A Comparative Study of Weight Optimization Methods

First Author[1 [0000−1111−2222−3333]], Second Author[2,3 [1111−2222−3333−4444]], and Third Author[3 [2222−−3333−4444−5555]]

No Institute Given

Abstract. Reproducibility is a crucial aspect of neural network training, but it remains challenging due to the stochastic nature of the optimization during the training process. This paper examines the effects of two weight optimization techniques: weight selection optimization and traditional weight update optimization, on the run-to-run variability of neural network performance. Our extensive experiments across various convolutional layer sizes reveal that weight selection optimization reduces variability by approximately 25% to 40%, depending on the model, compared to traditional training methods. However, this reduction in variability does not lead to improved model accuracy, which remains comparable to conventional training. These findings provide valuable insights into enhancing the stability and reproducibility of neural network models without compromising accuracy.

Keywords: Reproducibility · Determinism · Convolutional Neural Networks.

1 Introduction

Similar to any other scientific research, reproducibility is also a cornerstone in deep learning. It means obtaining consistent results when experiments are repeated under the same conditions, such as using identical input data, computational steps, and training methods [1]. However, consistent outcomes in deep learning are difficult to achieve due to the stochastic nature of model optimization, which makes training models unstable. According to [7], model instability refers to how sensitive a neural network's performance is to tiny changes in a single weight during training. Even a small change as little as 10^{-10} can cause the model to converge to very different outcomes. These small perturbations during training can lead to significant run-to-run variability [3]. This variability poses challenges, especially in safety-critical fields like healthcare, autonomous systems, and finance, where reliable performance is crucial [20]. It undermines the reliability of results and makes it hard to distinguish genuine improvements in model performance from random noise.

Traditional methods to reduce variability, such as training multiple copies of a model, are computationally expensive and do not fully address the root causes

of instability [8]. While substantial research [2,4,13] has focused on modifying algorithmic design choices to improve reproducibility, there is growing recognition of the need to explore optimization techniques that influence model weight structures. These techniques are crucial as they directly affect model stability and address the core of instability issues. However, the impact of weight optimization methods on reducing training variability has not been thoroughly examined. This paper addresses this gap by investigating two weight optimization techniques: weight selection optimization and traditional weight update optimization. Unlike traditional methods that continuously adjust weights during training, weight selection optimization employs a strategic selection process where a subset of potential weights is chosen based on a predefined criterion [19]. This approach is inspired by the slot machine analogy, where the parameter k represents the number of weights randomly assigned to each node of the network and evaluated at each update, allowing the model to select the optimal weights from this subset rather than simply adjusting existing weights.

Through experiments on neural networks with varying convolutional layer sizes, we show that weight selection optimization reduces run-to-run variability. However, this reduction does not necessarily improve model accuracy, which remains similar to traditional approaches. Additionally, determining the best value of k for a particular architecture is challenging. These findings highlight the importance of strategies that enhance stability in neural network training without compromising performance, emphasizing the need for approaches that maintain both consistency and accuracy.

2 Related Work

Reproducibility in deep learning is a significant challenge due to the stochastic nature of training, which causes instability in model performance. Key sources of nondeterminism, such as random initialization, data shuffling, and stochastic regularization, contribute to unpredictable outcomes [9]. Studies by [10,6,2] highlighted the impact of variations in random seeds and asynchronous training, emphasizing the need for standardized protocols to minimize variability. Research by [13] showed that changes in architecture and hyperparameters affect model stability but do not fully resolve the underlying instability issues. [18] demonstrated that minor adjustments, like altering nonlinearity, can significantly influence model performance, underscoring the complexities of achieving reproducible results. Environmental factors such as software versions and tooling also play a role; [9] and [11] highlighted the importance of controlling hardware and software environments to reduce training variability. [4] and [5] noted that randomness in training processes, including data shuffling and adversarial training, adds layers of variability that further complicate reproducibility. [8] emphasized the challenges posed by randomness in model comparisons, advocating for robust research practices, such as conducting multiple training runs, although this approach is often computationally expensive. [7] critically examined model instability, showing that even minor changes in weight initialization can cause models

to converge to different outcomes. They highlighted the need for optimization techniques that specifically target weight selection and instability, which lays the groundwork for exploring weight selection optimization. Building on these insights, our study investigates the effects of weight optimization techniques on reducing training variability and enhancing reproducibility in neural network training.

3 Weight Selection Optimization Approach

Weight selection optimization was introduced in [19] as a novel method for training neural networks using fixed, randomly assigned weights instead of the traditional approach of continuously updating weights through backpropagation. In this method, each node in the network is provided with a selection of potential weights, referred to as "slots." These weights are randomly assigned and remain fixed, allowing the network to choose the best-performing weight without altering the weights themselves during training.

3.1 The Parameter k

Each node is associated with a parameter k, which represents the number of random weight options available at that node. During training, backpropagation is not used to adjust the weights but to evaluate each weight option based on its performance. A scoring mechanism assigns a score S_k to each of the k weights, reflecting how well each weight contributes to the network's overall objective. These scores are dynamically updated throughout the training process, measuring each weight's effectiveness under various conditions.

3.2 Selection of a Weight Through Scores S_k

Once the scores are evaluated, the network selects the highest-scoring weight option for each node. This selection process allows the network to choose the best weight from the available set rather than modifying the weights continuously as in conventional training. The network structure is thus optimized by selecting weights that best fit the training data based on their performance scores S_k , without directly altering the weights through gradient updates. The key idea is that while the weights remain static, the scoring system actively guides the network in selecting the most suitable weights.

3.3 Importance of Weight Selection

This approach minimizes the impact of stochastic factors, such as random initialization and continuous weight adjustments, which often lead to variability and instability in traditional training methods. By fixing the weight options and utilizing a score-based selection system, the focus shifts from constant adjustments to intelligently selecting optimal weights based on performance. This approach leads to more stable and reliable neural network training outcomes.

4 Experiment

The experiments were designed to evaluate the effectiveness of the weight selection optimization approach, focusing on its impact on model variability and performance stability. Following the weight selection framework, the experiments were conducted on convolutional neural networks (CNNs) with varying architectures and configurations as in [19]. This section outlines the experimental setup, including dataset, initialization methods, and training procedures.

4.1 Dataset

Experiments were mainly conducted using the CIFAR-10 dataset [12], a widely used benchmark in image classification tasks. The dataset consists of 60,000 32x32 color images divided into 10 classes, with 50,000 images for training and 10,000 for testing. Standard data augmentation techniques such as random cropping and horizontal flipping were applied during training to enhance model generalization [21].

4.2 Weight Initialization

The Slot Machine approach utilizes fixed random weights, which are initially selected from a predefined distribution. For this study, the weights were sampled from the Glorot Uniform distribution [14], a commonly used method that initializes weights to maintain the variance of activations through the layers. This initialization method helps in stabilizing the training process by ensuring that the weights are neither too large nor too small, preventing vanishing or exploding gradients [15].

4.3 Parameter k and Training Process

Each node in the network is associated with a parameter k, representing the number of random weight options available at each node. We experimented with $k = 2$, 4, 8 and 16 for each of the CNN architectures: Conv2, Conv4, and Conv6, which represent networks with two, four, and six convolutional layers, respectively. During training, backpropagation was used not to update the weights but to evaluate the performance of each weight option. A scoring mechanism dynamically assigned scores S_k to each of the k weights based on how well they contributed to the network's objective, such as minimizing the loss. The weight with the highest score S_k at each node was selected, allowing the network to optimize its performance by choosing the most suitable weights.

4.4 Experimental Setup

Hyperparameters: Each model was trained with a learning rate of 0.01, batch size of 128, and momentum of 0.9. Cross-entropy loss was used as the objective function, and training was performed for 100 epochs for each configuration of k. No additional regularization techniques, such as dropout, were applied to maintain the focus on evaluating the weight selection method.

Software: Experiments were conducted using Python 3.8 with PyTorch 1.9.0 as the deep learning framework [16]. All experiments were run on CUDA 11.2, leveraging GPU acceleration for efficient training.

Hardware: All experiments were performed on the HPC cluster ARA using the SLURM workload manager. This system consists of multicore nodes for high computational performance and offers a variety of GPU systems. For our experiments, we used 2 NVIDIA Tesla V100 GPUs, supported by a 24-core Intel CPU and 128 GB of RAM. This setup provided high computational power, enabling efficient execution of multiple training runs and accurate measurement of variability.

Metrics for Variability and Performance To evaluate the effect of the weight selection optimization, we measured two key metrics: the average accuracy and the standard deviation of accuracy across five independent training runs for each architecture and k configuration. The average accuracy indicated overall performance, while the standard deviation captured the variability between runs, highlighting how stable the model's performance was under different conditions.

5 Results

The results of the experiments were analyzed to evaluate the impact of weight selection optimization compared to the traditional training method, which is represented as "learned weights" in the results. The analysis focused on different convolutional neural network architectures, specifically Conv2, Conv4, and Conv6. The experiments aimed to assess the test accuracy and the variability in model performance, quantified by the standard deviation across multiple training runs. By comparing weight selection optimization with the conventional learned weights approach, we sought to determine how effectively each method maintains accuracy while minimizing variability.

5.1 Test Accuracy Across Architectures

The test accuracy results for the different architectures are summarized in Figure 1. Each bar represents the test accuracy for a given architecture (LeNet, Conv2, Conv4, and Conv6) with different values of k (2, 4, 8, and 16), as well as the traditional learned weights approach.

The results indicate that as the complexity of the architecture increases, test accuracy generally remains stable across different values of k. For LeNet, the accuracy remains consistently high, close to 100%, indicating that simpler architectures derive minimal benefit from weight selection optimization. For deeper architectures such as Conv2, Conv4, and Conv6, the accuracy achieved with weight selection optimization is comparable to that of the traditional learned weights approach, demonstrating that this method does not significantly compromise performance. However, learned weights still achieve the highest accuracy

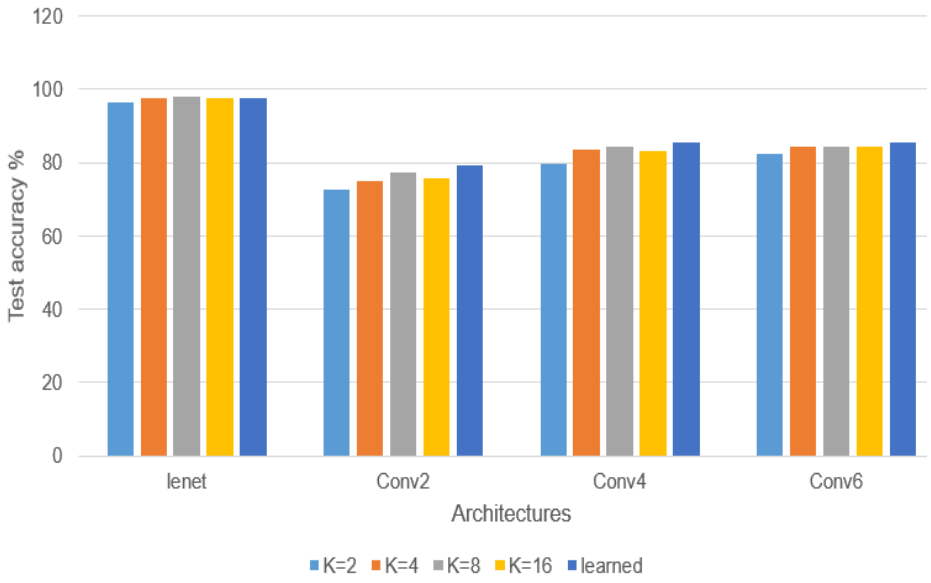

Fig. 1. Shows the average test accuracy of different CNN architectures (LeNet, Conv2, Conv4, Conv6) with varying k values (2, 4, 8, 16) and learned weights.

across all architectures. Among the weight selection configurations, $k = 8$ appears to be the optimal choice, offering the highest accuracy across different architectures within the weight selection approach.

5.2 Variability in Model Performance

Figure 2 shows the standard deviation of accuracy for the Conv2, Conv4, and Conv6 architectures with varying values of k. This metric highlights the variability in performance across multiple training runs, with lower standard deviation indicating more stable and reproducible results.

The results demonstrate that increasing the parameter k generally reduces the standard deviation, leading to more consistent performance across training runs. For Conv2, the standard deviation is highest at $k = 4$, indicating significant variability, while higher k values (e.g., $k = 16$) significantly stabilize the training outcomes, with the accuracy standard deviation decreasing from 0.40 to 0.25, representing a 37.5% reduction. A similar trend is observed in Conv4, where increasing k results in a reduction in standard deviation from 0.35 to 0.21, constituting a 40% reduction.

For the deeper Conv6 architecture, variability also decreases as k increases, though the effect is less pronounced compared to Conv2 and Conv4. The accuracy standard deviation reduces from 0.39 to 0.28, amounting to a 28.2% reduction. This suggests that deeper networks inherently possess greater stability, and weight selection optimization further enhances this attribute. However, the

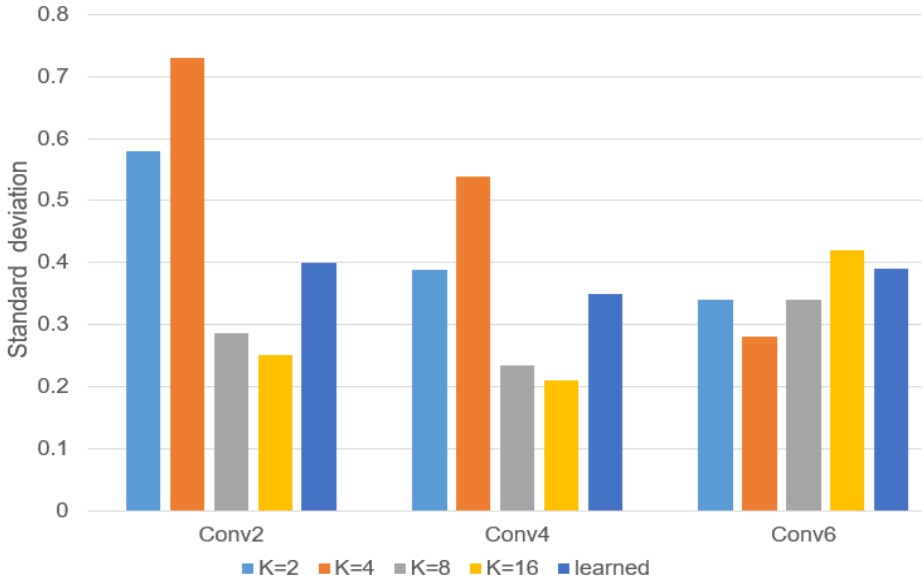

Fig. 2. shows the standard deviation of accuracy across different CNN architectures (Conv2, Conv4, Conv6) with varying k values (2, 4, 8, 16) and learned weights, indicating the variability in performance across multiple training runs.

optimal k value to minimize variability varies across architectures, underscoring the need for architecture-specific tuning. Notably, the traditional learned weights approach generally exhibits higher variability than the highest k values, emphasizing that weight selection optimization can outperform conventional training methods in terms of stability.

6 Conclusion

The results confirm the effectiveness of the weight selection optimization approach in reducing run-to-run variability while maintaining accuracy. By selecting weights based on performance scores rather than continuously adjusting them, this method offers a robust alternative to traditional backpropagation, especially in deeper architectures. However, there is a need to test this approach on larger and more complex models, as increasing the k parameter value could lead to higher memory requirements, highlighting a potential limitation of the approach. Future work will focus on applying this method to larger architectures and exploring techniques that enable the use of higher k values without imposing significant memory constraints. These findings demonstrate the potential of weight selection optimization as a practical strategy for enhancing reproducibility in neural network training, particularly in applications where stability and consistent performance are crucial.

References

1. Albertoni, R., Colantonio, S., Skrzypczyński, P., Stefanowski, J.: Reproducibility of machine learning: Terminology, recommendations and open issues. arXiv preprint arXiv:2302.12691 (2023)
2. Bouthillier, X., Laurent, C., Vincent, P.: Unreproducible research is reproducible. In: International Conference on Machine Learning, pp. 725–734. PMLR (2019)
3. Gundersen, O. E., Coakley, K., Kirkpatrick, C., Gil, Y.: Sources of irreproducibility in machine learning: A review. arXiv preprint arXiv:2204.07610 (2022)
4. Ji, Y., Kaestner, D., Wirth, O., Wressnegger, C.: Randomness is the root of all evil: more reliable evaluation of deep active learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 3943–3952 (2023)
5. Langroudi, H. F., Merkel, C., Syed, H., Kudithipudi, D.: Exploiting randomness in deep learning algorithms. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)
6. Ahmed, H., Lofstead, J.: Managing randomness to enable reproducible machine learning. In: Proceedings of the 5th International Workshop on Practical Reproducible Evaluation of Computer Systems, pp. 15–20 (2022)
7. Summers, C., Dinneen, M. J.: Nondeterminism and instability in neural network optimization. In: International Conference on Machine Learning, pp. 9913–9922. PMLR (2021)
8. Gundersen, O. E., Shamsaliei, S., Kjærnli, H. S., Langseth, H.: On reporting robust and trustworthy conclusions from model comparison studies involving neural networks and randomness. In: Proceedings of the 2023 ACM Conference on Reproducibility and Replicability, pp. 37–61 (2023)
9. Pham, H. V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., Yu, Y., Nagappan, N.: Problems and opportunities in training deep learning software systems: An analysis of variance. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, pp. 771–783 (2020)
10. Picard, D.: Torch.manual_seed (3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision. arXiv preprint arXiv:2109.08203 (2021)
11. Zhuang, D., Zhang, X., Song, S., Hooker, S.: Randomness in neural network training: Characterizing the impact of tooling. Proceedings of Machine Learning and Systems **4**, 316–336 (2022)
12. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical Report, University of Toronto (2009)
13. Madhyastha, P., Jain, R.: On model stability as a function of random seed. arXiv preprint arXiv:1909.10447 (2019)
14. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256. JMLR Workshop and Conference Proceedings (2010)
15. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02), 107–116 (1998)
16. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems **32** (2019)

17. Alahmari, S. S., Goldgof, D. B., Mouton, P. R., Hall, L. O.: Challenges for the repeatability of deep learning models. IEEE Access **8**, 211860–211868 (2020)
18. Snapp, R. R., Shamir, G. I.: Synthesizing irreproducibility in deep networks. arXiv preprint arXiv:2102.10696 (2021)
19. Aladago, M. M., Torresani, L.: Slot machines: Discovering winning combinations of random weights in neural networks. In: International Conference on Machine Learning, pp. 163–174. PMLR (2021)
20. Renard, F., Guedria, S., De Palma, N., Vuillerme, N.: Variability and reproducibility in deep learning for medical image segmentation. Scientific Reports **10**(1), 13724 (2020)
21. Shorten, C., Khoshgoftaar, T. M.: A survey on image data augmentation for deep learning. Journal of Big Data **6**(1), 1–48 (2019)