Improving AlexNet: Evaluating the Impact of Data Augmentation, Initialization, and Skip Connections

Abde - Abitalib Merchant, Jeongsik Park, Jake Fleischer, Nicklas Holmberg*
Department of Computer Science, University of Southern California
Los Angeles, CA 90089
{abdeabit, jeongsik, jmfleisc, nholmber}@usc.edu

Abstract

This study investigates the effects of data augmentation techniques, initialization methods, and skip connections on the performance of the AlexNet architecture for CIFAR-10 classification. The experimental framework incorporated three initialization strategies (Random, Xavier, and He), six distinct data augmentation techniques, and the inclusion of skip connections. The results demonstrate that data augmentation, particularly random flipping, substantially improved accuracy, achieving a maximum of 83.18%. In contrast, skip connections offered negligible benefits in the context of this relatively shallow architecture. While efficient initialization methods enhanced training stability, their impact on final performance was limited. These findings highlight the critical role of dataset-specific optimizations and carefully designed architectural modifications in achieving optimal model performance. The code can be found in: https://github.com/NicklasHolmberg/csci-567-project.git

1 Introduction

In the ever-expanding landscape of machine learning, image classification remains a cornerstone task, driving advancements in diverse applications such as autonomous vehicles [3], medical imaging[9], and facial recognition [4]. Among the many benchmarks used to evaluate models, CIFAR-10 holds a pivotal role due to its accessibility and suitability for testing a wide range of algorithms and architectures [7]. Its simplicity and balanced dataset of ten distinct classes allow researchers to explore the efficacy of novel techniques without the computational overhead of larger datasets.

Over the years, trends in CIFAR-10 research have mirrored the broader evolution of deep learning. Initial efforts focused on designing standalone architectures, such as AlexNet, which set the foundation for convolutional neural networks (CNNs) [8]. Subsequent research has emphasized enhancements through architectural innovations and training strategies. These developments include the integration of data augmentation [10], skip connections [6], and advanced initialization methods [2, 5], all of which aim to improve model generalization, convergence, and performance.

Data augmentation, a widely adopted technique, manipulates input images to artificially expand the training dataset, thereby reducing overfitting and enhancing model robustness . Common augmentation methods include random rotations, horizontal flipping, and scaling, each contributing unique invariance properties to the model [10]. Despite its proven benefits, the combined effects of different augmentation strategies warrant further exploration, especially in a controlled experimental setup.

Architectures such as ResNet have significantly advanced deep learning by addressing the vanishing gradient problem and enabling the training of deeper neural networks through the use of skip connections [6]. By enabling direct information flow across layers, these connections promote

*Authors are listed alphabetically.

efficient gradient propagation and enhance feature reuse. Integrating skip connections into AlexNet, a traditionally shallow model, presents an intriguing opportunity to evaluate their impact on model performance in a controlled environment.

Although often overlooked, initialization methods are critical in shaping the convergence behavior and overall performance of neural networks. Techniques such as Xavier [2] and He [5] initialization are designed to address issues of exploding or vanishing gradients, yet their relative effectiveness remains an open question when applied to specific architectures and datasets like CIFAR-10.

Motivated by these trends, we systematically investigate how different architectural and training modifications to AlexNet can improve performance on CIFAR-10. Importantly, each modification—data augmentation, skip connections, and initialization methods—was studied in isolation to provide clear insights into their individual contributions. Specifically, we explore three key research questions: (1) How do individual data augmentation methods affect performance? (2) Does the inclusion of skip connections enhance model performance? (3) What is the relative impact of different initialization methods? To address these questions, we designed a series of experiments:

- **Baseline Setup**: AlexNet using random initialization without any data augmentation.
- **Experiment 1 - Data Augmentation**: We applied six different augmentations—random rotation, flipping, color jitter, sharpness adjustment, random erasing, and a combination of all—individually to evaluate their specific impact on performance.
- **Experiment 2 - Skip Connections**: We introduced skip connections between layers and compared the modified model with the baseline.
- **Experiment 3 - Initialization**: We compared three initialization methods—Random, Xavier, and He—recording metrics such as accuracy and loss to assess their influence on convergence and stability.

In Section 2, we provide an overview of the methodologies we used, covering data augmentation, the AlexNet architecture, skip connection modifications, and initialization methods. Sections 3 and 4 detail the experimental setup, including hyperparameters, computational resources, evaluation strategies, and experimental procedures. Section 5 analyzes the results and discusses their implications, with a focus on architectural and training modifications. Finally, Section 6 summarizes the key outcomes and offers suggestions for future research directions.

2 Methods

Here, we describe the three methods applied in this study to systematically evaluate their impact on model performance. First, data augmentation, which improves generalization by diversifying the training set. Then, model architecture modifications, specifically the incorporation of skip connections to enhance gradient flow. Lastly, initialization methods, which aim to stabilize and accelerate training by setting appropriate starting weights. These approaches were implemented independently to isolate their contributions and provide actionable insights into optimizing AlexNet for the CIFAR-10 dataset.

2.1 Data Augmentation

We implemented a series of data augmentation techniques to enhance the diversity of the training dataset and improve model generalization. Each technique was applied independently to evaluate its specific impact. The augmentation methods and their configurations are as follows:

- Resize: All images were resized to 227x227 pixels to match the input requirements of AlexNet.
- **Random Rotation**: Images were randomly rotated within a range of ±20 degrees to introduce rotational invariance.
- **Random Flip**: Images were horizontally flipped with a probability of 0.1 to simulate mirrored perspectives.
- **Color Jitter**: Adjustments were made to brightness, contrast, and saturation within a factor of 0.1 to simulate varying lighting conditions.

- **Random Sharpness**: Sharpness was adjusted with a factor of 2 and applied with a probability of 0.1 to introduce texture variations.
- **Random Erasing**: Portions of the images were randomly erased with a probability of 0.75 and a scale range of 0.02 to 0.1 to mimic occlusions.
- **Combination (All Techniques)**: All the above augmentations - random rotation, random flip, color jitter, random sharpness and random erasing - were sequentially applied to evaluate their cumulative effect on model robustness and generalization.

Note that "Combination" does not mean we tried all possible combinations of data augmentation techniques, but rather a predefined sequence of selected methods applied together. Consequently, the size of the training set doubled after augmentation. Example of data augmentations applied to the frog class can be found in Figure 1., which illustrates how these techniques alter visual patterns while retaining the core characteristics of the class.

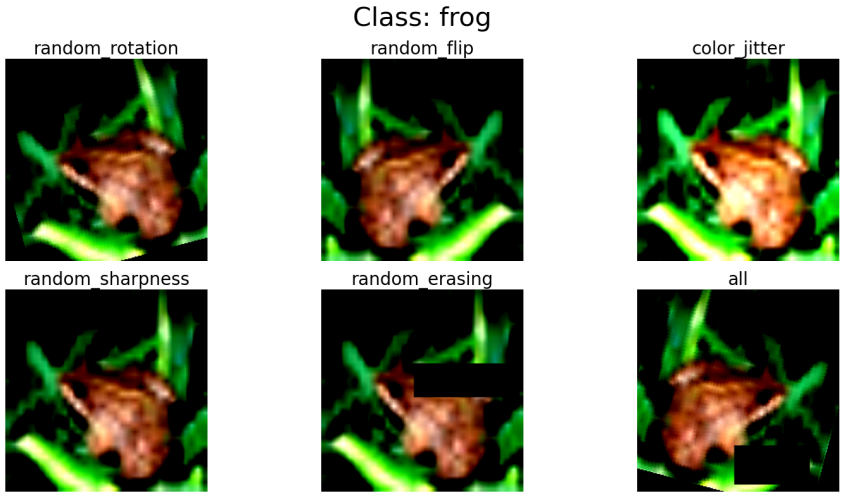


Figure 1: The augmentation example of frog.

2.2 Model Architecture

The AlexNet architecture was employed as the baseline model for our experiments due to its historical significance and simplicity, making it an excellent candidate for controlled studies on CIFAR-10. AlexNet consists of five convolutional layers, each followed by ReLU activations, interspersed with max-pooling operations to reduce spatial dimensions and increase feature abstraction [8]. These are followed by three fully connected layers with dropout applied to mitigate overfitting. All input images were resized to 227x227 pixels to match the network's input size requirements.

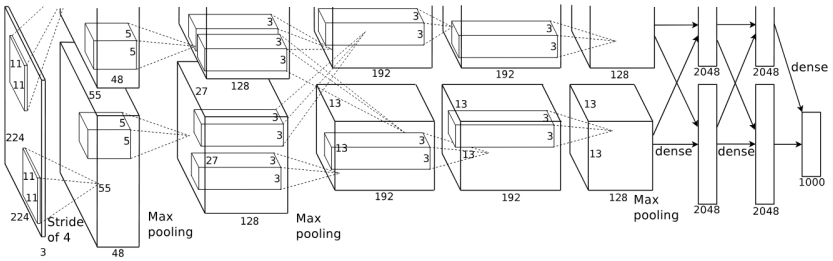


Figure 2: An illustration of the AlexNet architecture, as presented in the original AlexNet paper [2].

2.2.1 Skip Connections

Skip connections allow the output of a layer to bypass intermediate layers and directly contribute to subsequent layers, facilitating better gradient flow and feature reuse. This technique is particularly beneficial in deeper architectures, as demonstrated by ResNet [6].

For AlexNet, we introduced two skip connections. The first skip connection connects the input of the network to the output of the first convolutional block. A 1x1 convolutional layer is applied to the input to match the number of channels, followed by bilinear interpolation to align spatial dimensions. The transformed input is then added to the output of the first block. The second skip connection was implemented by directly adding the output of the second convolutional block to the output of the third convolutional block. Similarly, a 1x1 convolution was used to match the dimensions before addition.

2.3 Initialization Method

2.3.1 Importance of Initialization

In neural networks, initialization or weight initialization refers to the process of assigning initial values to the weights of the network before training. Initialization is an essential step because it can significantly affect the success and efficiency of the learning process [2, 5]. Incorrectly initialized weights can lead to problems such as:

- **Vanishing Gradients** - When gradients used to update the weights of the network become very small or "vanish" as they move to the earlier layers. This is primarily caused by the chain rule of calculus and can lead to slow convergence, getting stuck in a local minima, and poor learning performance.
- **Exploding Gradients** - Conversely, when the gradients of the network's loss with respect to the parameters get too large, it can lead to numerical instability and can obstruct the network from converging.

The goal of weight initialization is to set the weights in such a way that allows the network to converge to a good solution (global optimum) faster and more reliably.

2.3.2 Three Types of Initialization Methods

There are three types of initializations used in this experiment which will be discussed in detail below, along with the advantages and disadvantages of each.

1. **Random Initialization** - In this approach, weights are initialized to small random values, typically from either a normal or uniform distribution. The key objective is to have asymmetry between neurons, so that they do not learn the same features during training.

   $$W \sim \text{Uniform}(-\epsilon, \epsilon) \quad \text{or} \quad W \sim \mathcal{N}(0, \sigma)$$

   It is often used when the choice of initialization method is not known. It is a general-purpose method that is simple to implement and understand. However, one major flaw is that it can lead to vanishing or exploding gradients, particularly for deep networks.

2. **Xavier Initialization** - Xavier Initialization [2], proposed by Xavier Glorot, was designed to prevent the vanishing and exploding gradient problems, especially in deeper networks. The key idea is to maintain the variance of activations consistent throughout the layers.

   Uniform Distribution:

   $$W \sim \text{Uniform}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right)$$

   Where:

   - n_{in} is the number of input units
   - n_{out} is the number of output units

Normal Distribution:

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{\text{in}} + n_{\text{out}}}}\right)$$

Its performance lacks when the activations are ReLU-Based, as it leads to too large variances for the activations.

3. **He Initialization** - He Initialization [5], proposed by Kaiming He, is an activation function specifically designed for ReLU activation functions. ReLU activations have a property where a significant portion of the outputs are zero (due to negative inputs being mapped to zero). This leads to issues with the variance of the activations if not initialized properly. He initialization accounts for this by scaling the weights based on the number of input units.

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{\text{in}}}}\right)$$

Where:

- n_{in} is the number of input units

He initialization is best used for networks with ReLU or its variants, and is particularly effective in deep networks with a large number of layers where the risk of vanishing gradients is high. However, it is not suitable for activation functions like sigmoid or tanh.

The convergence rate for random initialization is largely unknown [1], however, the convergence rate for initialization methods largely depend on the activation function used in the network. Xavier converges faster when using activation functions of sigmoid or tanh. Similarly He initialization converges fastest when used with a ReLU activation function. Therefore, selecting the appropriate initialization method depends on the network architecture, depth of the network, and activation functions used.

3 Implementation Details

3.1 Data Preparation

The CIFAR-10 dataset is a widely used benchmark in the field of image classification, consisting of 60,000 color images divided into ten classes: plane, car, bird, cat, deer, dog, frog, horse, ship, and truck. Each image has a resolution of 32x32 pixels and contains only a single object. The simplicity and diversity of CIFAR-10 make it an ideal choice for evaluating the performance of various machine learning models, especially in scenarios where computational resources are limited. Furthermore, CIFAR-10's balanced class distribution ensures that models trained on this dataset do not exhibit biases toward specific classes.

	Total #	Plane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Train	50,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
Test	10,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Table 1: Class distribution of Train and Test datasets for CIFAR-10.

CIFAR-10 contains a total of 50,000 training images and 10,000 test images, with each class represented equally across both subsets. The breakdown of the dataset is shown in Table 1. During our experiments, the CIFAR-10 dataset was split into training, validation, and test subsets. Specifically, the training data was further divided into a training set and a development set into 9:1 ratio to monitor overfitting and fine-tune hyperparameters. Additionally, data augmentation techniques were applied exclusively to the training set to improve generalization, while the validation and test sets remained unaltered to provide consistent evaluation metrics.

3.2 Model Development and Environment Setup

To create our model, we built it from scratch using Python, implementing both data augmentation and initialization methods. We utilized the following external libraries: Numpy, Torch, Sklearn, Matplotlib, etc. These libraries, along with their version specifications, are listed in the `environments.yml` file provided in our shared GitHub repository. The execution of our Python code was automated through the use of shell scripts, specifically `src/run.sh`. Version control for the project was handled using `Git`, ensuring reproducibility and collaborative efficiency.

4 Experiments

This section provides comprehensive details about the hyperparameters, experimental setup, and computational resources.

4.1 Hyperparameters

We followed the hyperparameters from the original AlexNet paper, with modifications to accommodate the CIFAR-10 dataset. The learning rate was set to 0.005, weight decay to 0.005, momentum to 0.9, and batch size to 128. Training was performed for 15 epochs. We selected the best epoch based on the highest validation accuracy, which was monitored using a 9:1 training-to-validation split of the CIFAR-10 training set.

To ensure reproducibility, experiments were conducted using three fixed random seeds: 1, 42, and 99. Each combination of initialization methods, data augmentation techniques, and skip connections was evaluated separately to isolate their effects on model performance. The baseline model consisted of AlexNet without skip connections, trained on the CIFAR-10 dataset using the default resizing augmentation and random initialization.

4.2 Three Variants

We systematically varied the following components during our experiments, as described in Section 2. The overall experimental setups are summarized in Table 2, and the corresponding results are detailed therein.

Data Augmentations We initially applied image resizing followed by six data augmentation configurations, including resizing, random rotation, horizontal flipping, color jitter, sharpness adjustment, random erasing, and a combination of all these techniques applied sequentially.

Skip Connections We evaluated the effect of skip connections by testing AlexNet both with and without the incorporation of skip connections.

Initialization Methods The model weights were initialized using three different methods. Random initialization drew weights from a normal distribution with mean 0 and standard deviation 0.02. Xavier initialization aimed to maintain consistent variance of activations across layers, while He initialization was optimized for ReLU activations by scaling weights to prevent vanishing gradients.

4.3 Computational Resources

For computational resources, we utilized a third-party GPU service called RunPod, spending approximately $10. The computations were performed on an L4 model with 24GB VRAM, 12 VCPUs, and 50GB RAM. To parallelize the process, we executed four separate shell scripts, each running for about 10 hours. Without parallelization, the total computation time would have been 40 hours.

5 Results and Discussions

Architecture	Accuracy	Precision	Recall	F1	Top-2	Bottom-2
Baseline	79.63 ± 1.64	0.803	0.796	0.794	ship, car	cat, dog
+ Random Rotation	82.95 ± 0.34	**0.838**	0.829	0.829	car, ship	cat, bird
+ Random Flip	$\mathbf{83.18 \pm 1.03}$	0.835	**0.832**	**0.831**	car, ship	cat, dog
+ Color Jitter	82.34 ± 0.6	0.829	0.823	0.822	car, ship	cat, bird
+ Random Sharpness	82.17 ± 1.18	0.829	0.822	0.821	car, ship	cat, dog
+ Random Erasing	82.82 ± 0.27	0.835	0.828	0.828	car, ship	cat, dog
+ All Augmentations	82.53 ± 1.23	0.834	0.825	0.826	car, ship	cat, deer
+ Skip Connections	$\mathbf{79.71 \pm 1.22}$	**0.808**	**0.797**	**0.796**	car, truck	cat, bird
+ Xavier Initialization	$\mathbf{82.11 \pm 0.45}$	**0.823**	**0.821**	**0.82**	car, ship	cat, dog
+ He Initialization	81.33 ± 1.12	0.818	0.813	0.811	car, ship	cat, dog

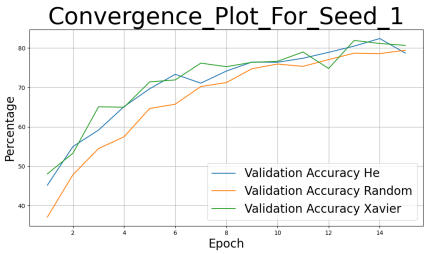
Table 2: **Performance Metrics for different architectures.** Each row represents a configuration and its corresponding evaluation metrics. For each method (Data Augmentation, Skip connections, and initialization), the overall best results are **bolded**. Accuracy, Precision, Recall, and, F1-score are reported in percentages (%).

In Table 2, we analyze the impact of three different methods on accuracy, precision, recall, and F1-score compared to the baseline model. For this task, the evaluation metric used is accuracy, providing a straightforward assessment of model performance. Precision, recall, and F1-score are included in the table as supplementary metrics to provide additional insights into the model's behavior. The first row presents the results of the baseline model, the next six rows illustrate the effects of different data augmentation techniques, the following row shows the impact of skip connections, and the final two rows highlight the results of two different initialization methods. Additionally, we identify the two classes with the highest accuracy and include them in the top-2 column, while the two classes with the lowest accuracy are listed in the bottom-2 column. Further details, including the confusion matrix, can be found in Appendix A.

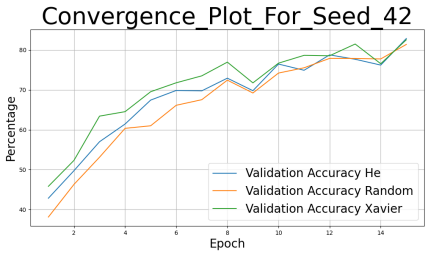
As observed in Table 2, the baseline model demonstrates an overall accuracy of 79.63%, serving as a foundation for comparison with other configurations and enhancements. It is interesting to note that the "car" category consistently performed the best across almost all setups, highlighting its distinct and well-learned features. Similarly, the "cat" and "dog" categories consistently performed the worst across most setups, indicating potential inter-class confusion due to their similar features. This suggests that class-specific augmentations, improved feature extraction, or dataset refinements may be needed to enhance the model's ability to differentiate these categories.

Data Augmentation When analyzing our results at a high level, data augmentation proves to be an effective strategy for improving performance, as all data augmentation methods outperform the baseline. Notably, methods such as random flip achieve the highest accuracy of 83.18 ± 1.03% overall. However, applying all types of data augmentations simultaneously does not outperform the individual augmentations, indicating that certain data augmentation combinations may introduce conflicting transformations or noise that offset their individual benefits.

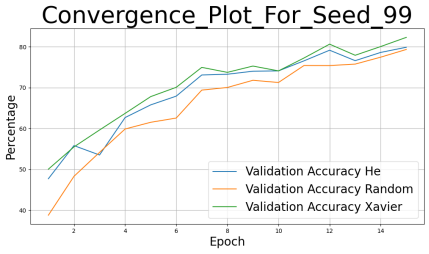
Skip Connections The idea of incorporating skip connections was inspired by deeper architectures like ResNet, which use his technique to mitigate vanishing gradients and improve back propagation. While AlexNet is not nearly as deep as ResNet, adding skip connections could theoretically enhance optimization, improve robustness, and even reduce the risk of overfitting by introducing alternative pathways for information flow. However, our results do not support these theoretical benefits in the context of Alexet. Adding skip connections slightly reduced the model's performance, with an accuracy of 79.71 ± 1.22%, falling below the baseline. This suggests that skip connections may not provide meaningful improvements in relatively shallow architectures like AlexNet, as its depth does not pose the same optimization challenges seen in much deeper networks. Instead, this outcome underscores the inherent limitations of AlexNet's architecture and highlights the diminishing returns of introducing techniques designed for modern, deeper networks. This result emphasizes the need to adapt such methods thoughtfully to the constraints and characteristics of the given model.



(a) Seed 1



(b) Seed 42



(c) Seed 99

Figure 3: Effect of Initialization Methods on Convergence Speed. The plots illustrate the convergence behavior for different random seeds (1, 42, and 99) using the same initialization method.

Initialization Method The choice of weight initialization clearly influences the early stages of model training. Xavier and He initialization methods enable faster convergence compared to random initialization, as demonstrated by the results in Figure 3. The plots show the convergence behavior for different random seeds (1, 42, and 99) across the three mentioned initialization methods (random, Xavier, and He). These methods help mitigate issues such as unstable gradients during backpropagation, particularly when using the ReLU activation function, which can otherwise result in dead neurons if the weights are not initialized appropriately. However, by the end of 15 epochs, all models achieve similar accuracy, suggesting that the impact of initialization methods is most significant during early training (≤ 10 epochs in this case).

6 Conclusion

In this study, we explored various enhancements and configurations to improve the performance of the AlexNet architecture. The baseline model demonstrated an accuracy of 79.63%, serving as a benchmark for evaluating the effectiveness of different methods. Among these, data augmentation emerged as the most impactful strategy, with random flip achieving the highest accuracy of 83.18 ± 1.03%. However, the results also revealed that combining all augmentations simultaneously did not outperform individual methods, highlighting the importance of selecting augmentations that complement the dataset and task.

In summary, the results highlight the low-hanging fruit of data augmentation as a straightforward and effective strategy for enhancing model performance. Techniques like random flip and random rotation demonstrate clear benefits with minimal computational overhead. Coupling these methods with careful architecture refinements, such as better feature extraction for under-performing classes, could further optimize results. Conversely, more complex enhancements like skip connections may not be directly applicable to shallow architectures like AlexNet, suggesting that deeper models are better suited to leverage these strategies. In a similar vein, Xavier and He initialization proved efficient for earlier convergence but did not result in a more accurate model after training was complete. This suggests that while initialization is important for stabilizing early training, its influence diminishes as the model progresses through training cycles.

Future work could explore more refined augmentation techniques, class-specific preprocessing, and architectural changes tailored to the characteristics of AlexNet. These findings reinforce the value of balancing simplicity and innovation when adapting classical architectures for modern tasks.

References

[1] Yuxin Chen, Yuejie Chi, Jianqing Fan, and Cong Ma. Gradient descent with random initialization: fast global convergence for nonconvex phase retrieval. *Mathematical Programming*, 176: 5–37, February 2019.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[3] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of field robotics*, 37(3):362–386, 2020.

[4] Guodong Guo and Na Zhang. A survey on deep learning based face recognition. *Computer vision and image understanding*, 189:102805, 2019.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

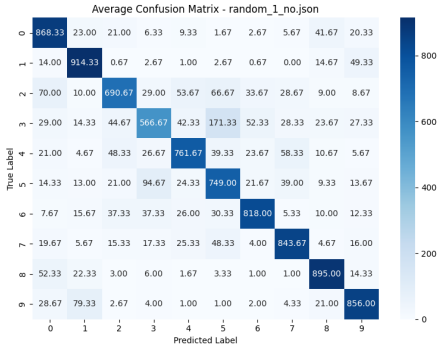
[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

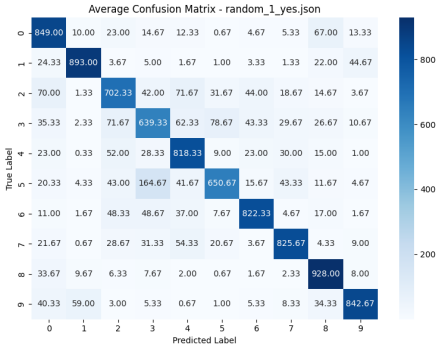
[9] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42: 60–88, 2017.

[10] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems*, 34:29935–29948, 2021.

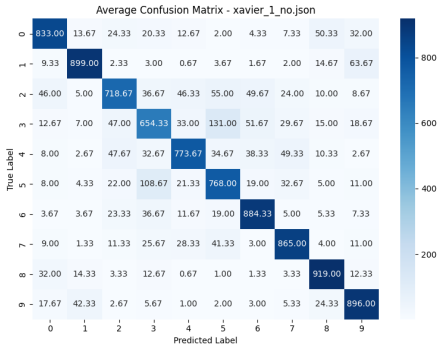
A Appendix: Confusion Matrix on Our Experiments



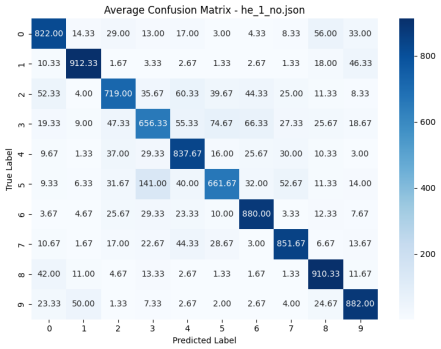
Random_1_No



Random_1_Yes

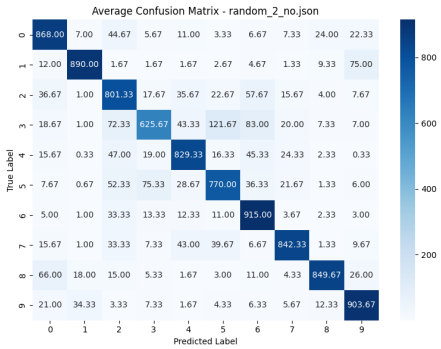


Xavier_1_No

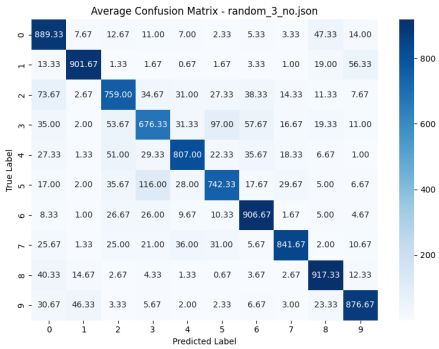


He_1_No

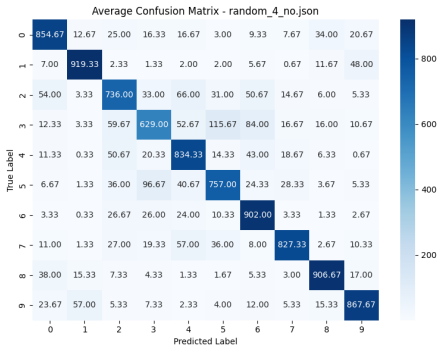
Figure 4: Grid of Confusion Matrices: Each cell represents a different confusion matrix. The subcaptions are labeled in the format [Initialization][AugmentNumber][SkipConnections], where we have 3 types of initialization, 6 types of augmentations encoded as numbers, and a boolean indicating the presence of skip connections. / [AugmentNumber] 2: random_rotation, 3: random_flip, 4: color_jitter, 5: random_sharpness, 6: random_erasing, 7: all augmentations / Labels 1 to 10 correspond to plane, car, bird, cat, deer, dog, frog, horse, ship, and truck, respectively.



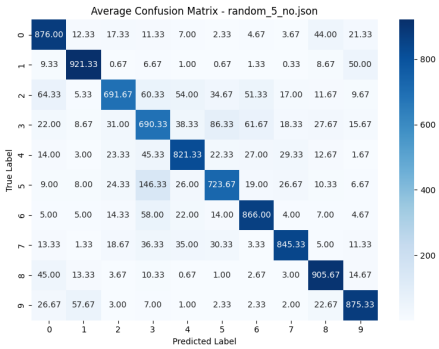
Random_2_No



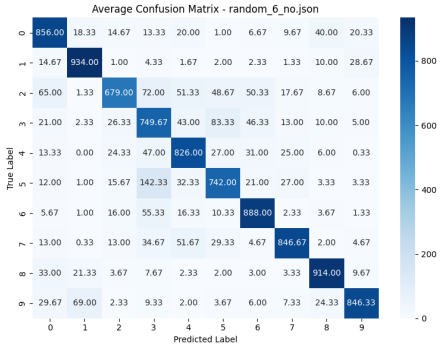
Random_3_No



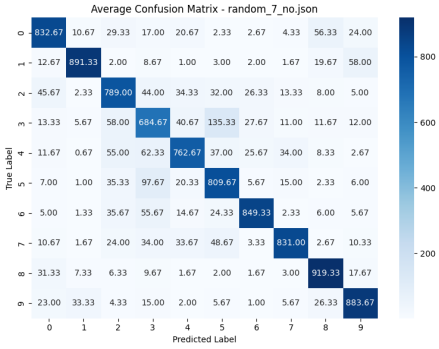
Random_4_No



Random_5_No



Random_6_No



Random_7_No

Figure 5: Grid of Confusion Matrices: Each cell represents a different confusion matrix. The subcaptions are labeled in the format [Initialization][AugmentNumber][SkipConnections], where we have 3 types of initialization, 6 types of augmentations encoded as numbers, and a boolean indicating the presence of skip connections. / [AugmentNumber] 2: random_rotation, 3: random_flip, 4: color_jitter, 5: random_sharpness, 6: random_erasing, 7: all augmentations / Labels 1 to 10 correspond to plane, car, bird, cat, deer, dog, frog, horse, ship, and truck, respectively.