Generative AI Question 03

Kissa Zahra[1 [i21−0572]]

FAST National University of Computer and Emerging Sciences, Islamabad H-11, Pakistan kissasium@gmail.com

Abstract. This paper presents the implementation and evaluation of Convolutional Neural Networks (CNNs) for image classification on the CIFAR-10 dataset. We investigate the impact of various hyperparameters including learning rate, batch size, number of filters, and network depth on model performance. Additionally, we compare models trained with and without data augmentation techniques. Experimental results demonstrate that data augmentation significantly improves model generalization, with the augmented model achieving 72.83% accuracy compared to 67.80% for the non-augmented model.

Keywords: CNN · Neural Network · Image Classification · CIFAR-10 · One-hot encoding ·

1 First Section

1.1 Introduction

Image classification remains a fundamental problem in computer vision with applications spanning across various domains including autonomous driving, medical imaging, and security systems.

In this paper, we implement and evaluate CNNs for classification on the CIFAR-10 dataset , which consists of 60,000 32×32 color images across 10 classes, providing a challenging yet manageable testbed for analyzing CNN performance. Our primary objectives are to:

- Implement an effective CNN architecture for CIFAR-10 classification
- Evaluate the impact of data augmentation on model performance
- Conduct a systematic ablation study to understand the effects of key hyperparameters
- Visualize feature maps to interpret the learned representations

1.2 Methodology

Dataset and Preprocessing The CIFAR-10 dataset consists of 60,000 32×32 color images divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. We split the dataset into training (80%) and testing (20%) sets. Prior to training, we performed the following preprocessing steps:

- Normalization: Pixel values were scaled to the range [0, 1] by dividing by 255
- One-hot encoding: Class labels were converted to one-hot encoded vectors

CNN Architecture The base CNN architecture implemented for this study consists of the following components:

- Convolutional layers with ReLU activation functions
- Max pooling layers for spatial downsampling
- Fully connected layers for classification
- Softmax output layer for multi-class probability distribution

Data Augmentation For the augmented model variant, we implemented the following data augmentation techniques:

- Random horizontal flipping
- Random rotation (±15 degrees)
- Random shifts in width and height (±10%)

These augmentations increase the effective size of the training dataset and improve the model's invariance to common transformations.

Training Details All models were trained using categorical cross-entropy loss and the following configuration:

- Optimizer
- Batch size
- Number of Epochs
- Learning rate

2 Second Section

2.1 Experimental Results

Performance Comparison Comparison the performance of models trained with and without data augmentation.

Table 1. Performance Metrics Comparison of CNN Models

Model	Accuracy	Precision	Recall	F1-Score
Without Augmentation	0.6780	0.6813	0.6780	0.6750
With Augmentation	0.7283	0.7322	0.7283	0.7223
Difference	+5.03%	+5.09%	+5.03%	+4.73%

As shown in Table 1, data augmentation resulted in significant performance improvements across all metrics, with an accuracy increase of 5.03% over the baseline model. This confirms that augmentation effectively enhances the model's generalization capability.

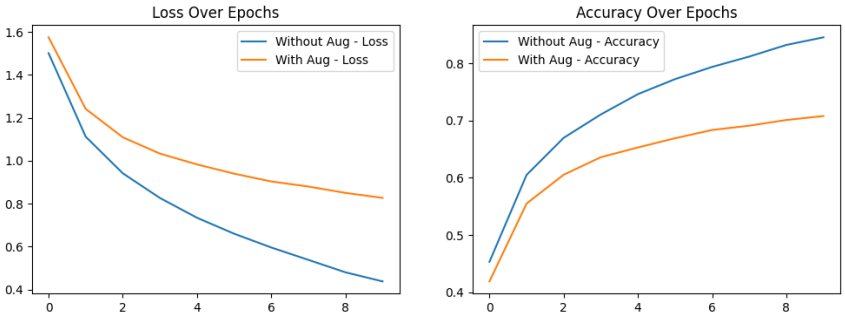

Fig. 1. Loss and Eccuracy over the epochs

Class-wise Performance:

Class	Without Aug	With Aug	Difference
airplane	75.10%	81.10%	+6.00%
automobile	77.50%	89.00%	+11.50%
bird	54.90%	58.10%	+3.20%
cat	48.60%	45.70%	-2.90%
deer	72.70%	62.20%	-10.50%
dog	42.50%	56.10%	+13.60%
frog	71.90%	91.00%	+19.10%
horse	73.80%	76.50%	+2.70%
ship	80.40%	82.50%	+2.10%
truck	80.60%	86.10%	+5.50%

Fig. 2. Class-wise Performance (Recall) for Both Models

2.2 Confusion Matrix Analysis

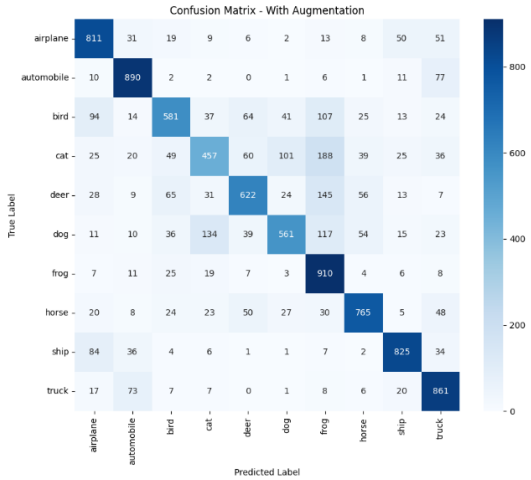

Fig. 3. Confusion Matrix - with augmentation

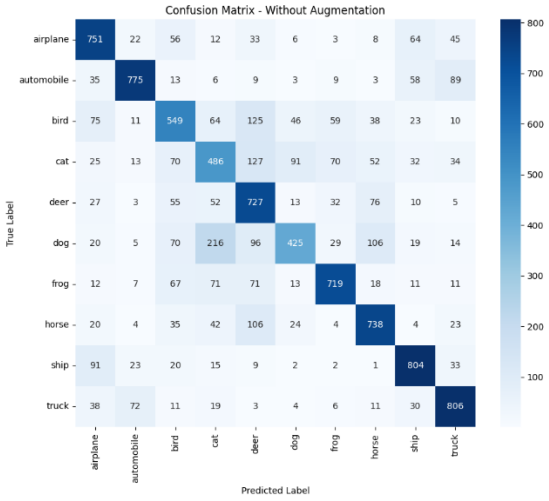

Fig. 4. Confusion Matrix - without augmentation

– Cats are frequently confused with dogs
– Deer are sometimes classified as horses

– Trucks are occasionally misidentified as automobiles

These patterns are expected and align with the visual similarities between these classes. The most distinctive classes (frog, ship, airplane) show the least confusion with other categories.

2.3 Impact of Learning Rate

We experimented with three different learning rates: 0.0001, 0.001, and 0.01. Table 2 presents the results.

Table 2. Impact of Learning Rate on Model Performance

Learning Rate	Accuracy	Precision	F1-Score
0.0001	0.6672	0.6922	0.6580
0.001	**0.7730**	**0.7769**	**0.7736**
0.01	0.6672	0.6922	0.6580
0.1	0.0968	0.0094	0.0171

A learning rate of 0.001 yielded the best performance across all metrics. Too low a learning rate (0.0001) resulted in slower convergence, while too high rates (0.01 and especially 0.1) led to unstable training and poor performance. With an extremely high learning rate of 0.1, the model failed to learn meaningful patterns, resulting in near-random performance.

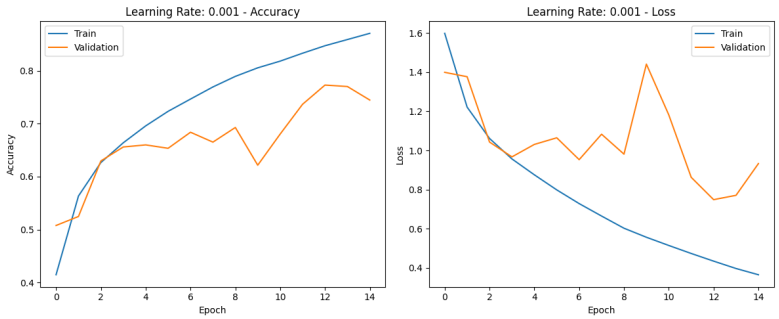

Fig. 5. Learning rate

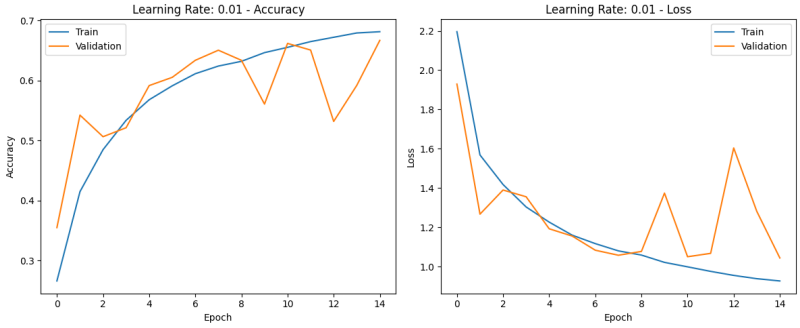

Fig. 6. Learning rate

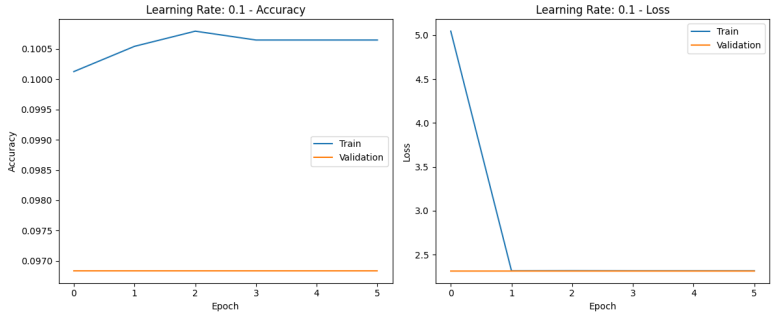

Fig. 7. Learning rate

2.4 Impact of Batch Size

We varied the batch size between 16, 32, and 64 to assess its impact on model performance. Table 3 presents the results.

Table 3. Impact of Batch Size on Model Performance

Batch Size	Accuracy	Precision	F1-Score
16	0.7462	0.7554	0.7487
32	**0.7720**	**0.7788**	**0.7711**
64	0.7355	0.7397	0.7322

A batch size of 32 provided the best balance between computational efficiency and model performance. Smaller batch sizes (16) resulted in noisier gradients and

slower convergence, while larger batch sizes (64) potentially prevented the model from escaping local minima.

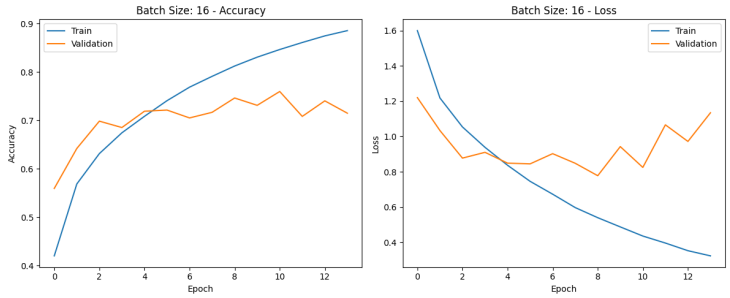

Fig. 8. Batch Size

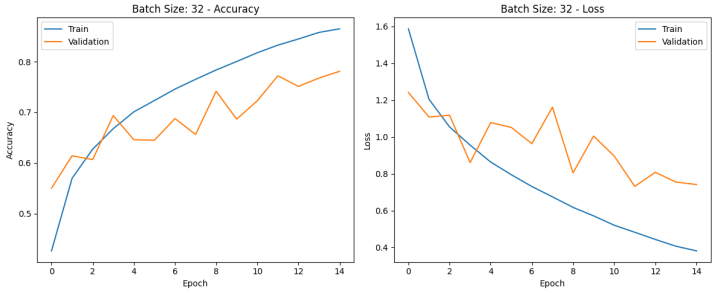

Fig. 9. Batch Size

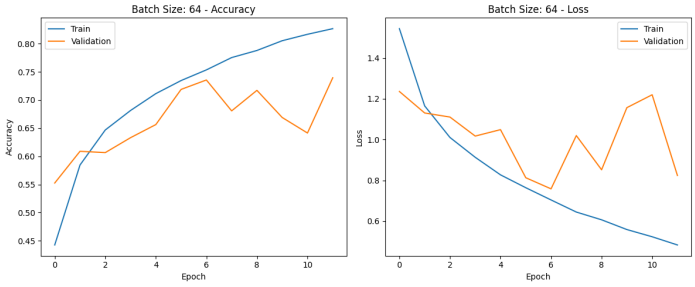

Fig. 10. Batch Size

2.5 Impact of Number of Filters

We experimented with different numbers of initial filters: 16, 32, and 64. Table 4 presents the results.

Table 4. Impact of Initial Filter Count on Model Performance

Initial Filters	Accuracy	Precision	F1-Score
16	0.7374	0.7412	0.7376
32	0.7535	0.7576	0.7510
64	**0.7448**	**0.7599**	**0.7449**

The impact of filter count was less dramatic than other hyperparameters, but models with 32 and 64 initial filters generally outperformed those with only 16 filters. The model with 32 filters achieved the highest accuracy, while the model with 64 filters had the best precision.

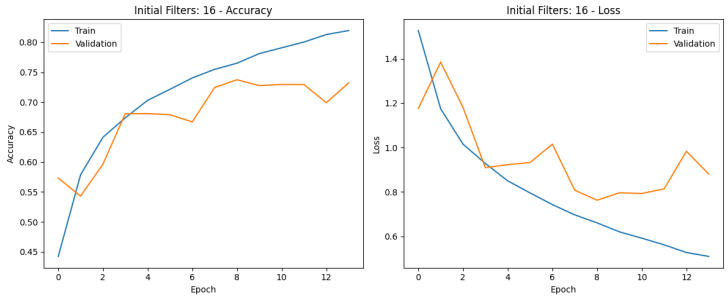

Fig. 11. Number of Filters

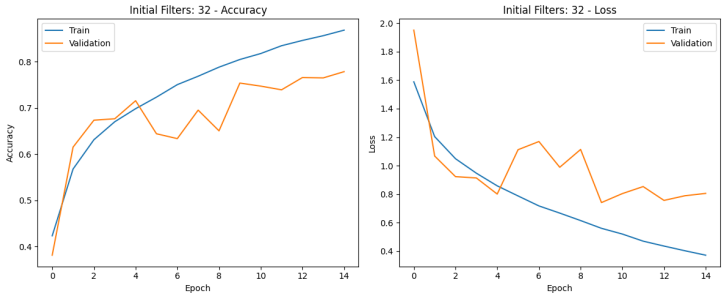

Fig. 12. Number of Filters

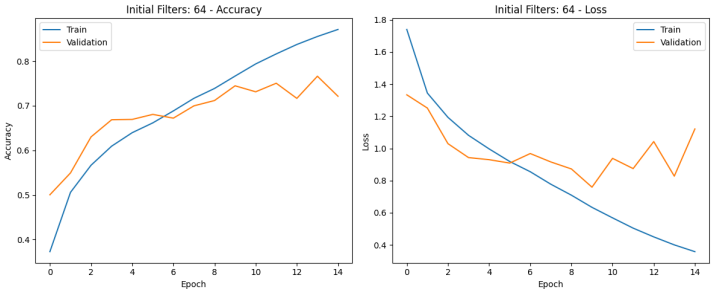

Fig. 13. Number of Filters

2.6 Impact of Network Depth

We varied the number of convolutional layers between 3 and 7 to understand the effect of network depth. Table 5 presents the results.

Table 5. Impact of Network Depth on Model Performance

Conv Layers	Accuracy	Precision	F1-Score
3	**0.7519**	**0.7675**	**0.7512**
5	0.7093	0.7379	0.7025

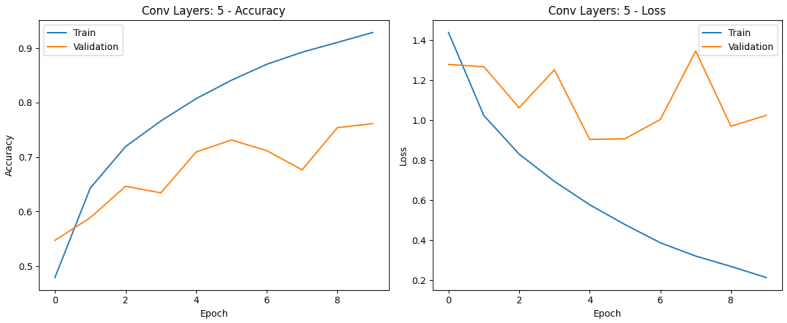

Fig. 14. Network Depth

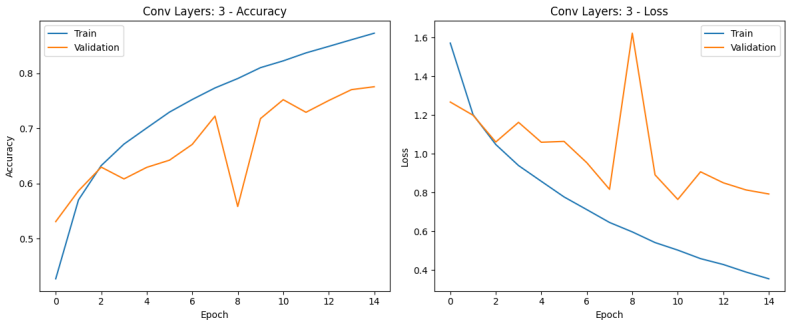

Fig. 15. Network Depth

A moderate network depth of 3 convolutional layers yielded the best performance. Increasing to 5 layers resulted in a drop in accuracy, suggesting that the additional complexity did not provide benefits for this dataset and may have led to overfitting.

3 Third Section

3.1 Visualization

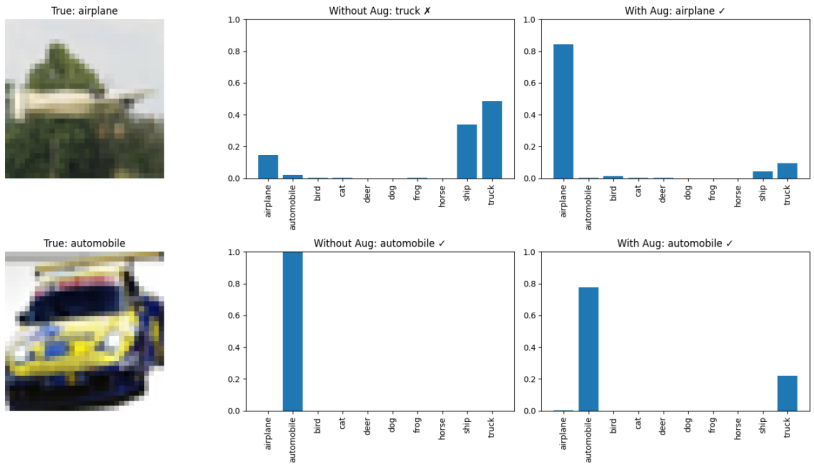

Fig. 16. Test Data

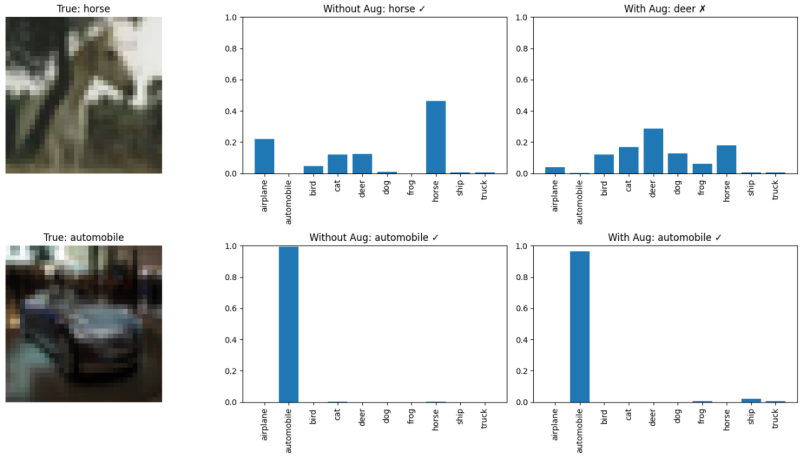

Fig. 17. Test Data

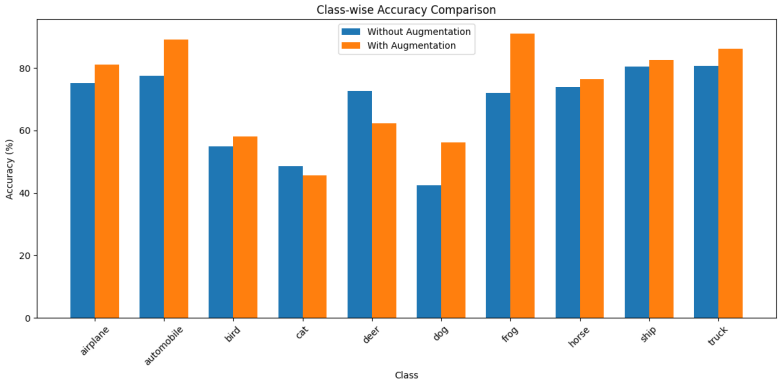

Fig. 18. Classwise Accuracy

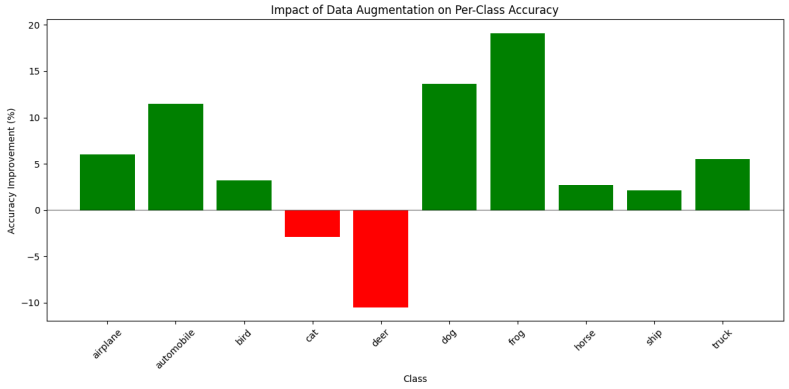

Fig. 19. Impact on per class

4 Fourth Section

4.1 Discussion

Our experiments provide several key insights into CNN design and optimization for image classification:

Data Augmentation Effect Data augmentation proved highly effective, improving accuracy by over 5%. The most substantial improvements were observed for classes with distinctive features and consistent orientations (frog, automobile, dog). Interestingly, augmentation reduced performance for cats and deer, which may be more sensitive to the specific transformations we applied. This underscores the importance of class-specific augmentation strategies.

Optimal Hyperparameters Best parameters configuration:

- Learning rate: 0.001
- Batch size: 32
- Network depth: 3 convolutional layers
- Initial filters: 32-64 (balance between performance and efficiency)

The learning rate had the most dramatic impact on performance, with a 0.001 learning rate achieving 77.30% accuracy, while a high learning rate of 0.1 resulted in model failure.

5 Fifth Section

5.1 Conclusion

In this paper, we implemented and evaluated CNN models for CIFAR-10 image classification. Our results demonstrate that data augmentation significantly

improves model performance, and that hyperparameter selection has a substantial impact on accuracy. The optimal configuration in our experiments included a learning rate of 0.001, batch size of 32, and a moderate network depth of 3 convolutional layers.