Image Classification Using Deep Learning on CIFAR-10 Dataset

Gajan Mohan Raj

1 Introduction

Image classification is a fundamental task in deep learning with the goal of assigning a label predefined set of categories to an input image. Convolutional Neural Networks (CNNs) have achieved remarkable success in image classification, underpinning various real-world applications such as self-driving cars, facial recognition systems, and medical image analysis.

The CIFAR-10 dataset is commonly used as a benchmark for image classification research. The dataset consists of 60,000 images, 32 x 32 pixels in size, in 10 distinct classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

This project focuses on building, training, and evaluating a CNN to classify images from the CIFAR-10 dataset. The methodology involved preparing the data, designing the CNN model architecture, training and testing using accuracy as the primary metric, and attempting to improve the CNN using techniques such as random data augmentation, hyperparameter tuning, and tweaking the model architecture.

2 Methodology

2.1 Environment and Libraries

Google Colab was utilized for all GPU computing. The V100 GPUs were utilized. Python (version 3.x) and other key libraries were utilized as well.

Pytorch was the deep learning framework utilized for building and training the neural networks. Torchvision was utilized for accessing the CIFAR-10 dataset and data transformations. Matplotlib was utilized to visualize the training progress and results through training and validation curves for loss and accuracy. Numpy was utilized for numerical computations. Scikit-learn was utilized to generate the confusion matrix and classification reports in the evaluation step of the project.

2.2 Data Loading and Preparation

As mentioned earlier, the CIFAR-10 dataset consisting of 60,000 images in 10 distinct classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) was used for training and evaluation of the CNN model. To preprocess the data, all images were transformed to tensors (for suitable input to the CNN model) and normalized by subtracting the channel-wise mean and dividing by the standard deviation, aiding in model convergence.

The dataset of 60,000 images was then split into a training set of 40,000 images, a validation set of 10,000 images, and a test set of 10,000 images. The training set was used to train the CNN architecture. The validation set was used to monitor the model's performance during training and tune hyperparameters. The held-out test set was used to get an unbiased evaluation of the trained CNN model.

2.3 CNN Architecture

The CNN design consisted of several convolutional and pooling layers, followed by fully connected layers.

The CNN architecture consists of three “blocks” with each consisting of two convolutional layers (3x3 kernel) to extract features from the input images, two batch normalization layers (one after each convolutional layer) to improve training stability and convergence, two RELU activation layers (one after each batch normalization layer) enabling the model to learn complex boundaries between classes, and a max pooling layer (2x2 kernel) to control overfitting and promote spatial variance. The number of feature maps or output channels doubles after each block (32 -> 64 -> 128) to continually extract features with increased, more complex depth.

The output of these three convolutional blocks is flattened into a one-dimensional vector with 256 * 4 * 4 elements (4096 elements). Three dense layers then process the flattened features to map them to one of the ten distinct classes. There are also two batch normalization layers between these three dense layers to enhance training stability. Dropout layers are also added in between these dense layers to prevent overfitting and improve performance. A table of this CNN architecture is displayed in Figure 1.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 32, 32]	896
BatchNorm2d-2	[-1, 32, 32, 32]	64
Conv2d-3	[-1, 64, 32, 32]	18,496
BatchNorm2d-4	[-1, 64, 32, 32]	128
MaxPool2d-5	[-1, 64, 16, 16]	0
Conv2d-6	[-1, 128, 16, 16]	73,856
BatchNorm2d-7	[-1, 128, 16, 16]	256
Conv2d-8	[-1, 128, 16, 16]	147,584
BatchNorm2d-9	[-1, 128, 16, 16]	256
MaxPool2d-10	[-1, 128, 8, 8]	0
Conv2d-11	[-1, 256, 8, 8]	295,168
BatchNorm2d-12	[-1, 256, 8, 8]	512
Conv2d-13	[-1, 256, 8, 8]	590,080
BatchNorm2d-14	[-1, 256, 8, 8]	512
MaxPool2d-15	[-1, 256, 4, 4]	0
Linear-16	[-1, 1024]	4,195,328
BatchNorm1d-17	[-1, 1024]	2,048
Dropout-18	[-1, 1024]	0
Linear-19	[-1, 512]	524,800
BatchNorm1d-20	[-1, 512]	1,024
Dropout-21	[-1, 512]	0
Linear-22	[-1, 10]	5,130

Figure 1: Model architecture

2.4 Training Process

The model was trained with the Adam optimizer, a learning rate of 0.001, and a batch size of 64. Cross-entropy loss was used to adjust the model weights during training. Training was done for 15 epochs, and the weights with which the model performed with the lowest validation loss was saved.

2.4.1 Evaluation

The model saved from the training process was then evaluated on the test set of 10,000 images. A confusion matrix and classification report were produced to evaluate the performance of the CNN model. Figure 2 outlines the basic layout of the training and testing process.

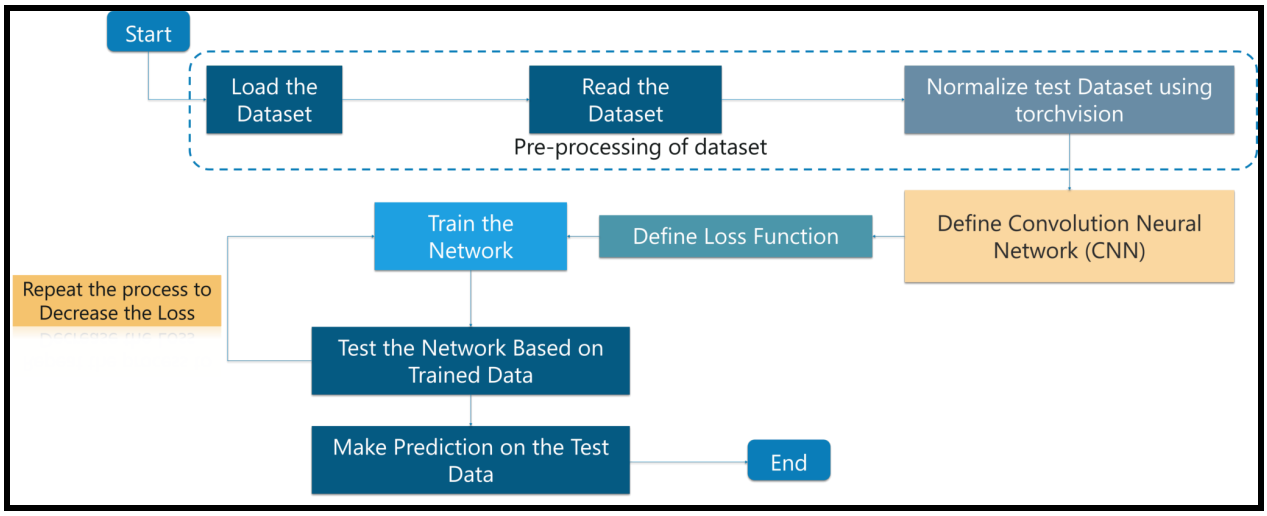


Figure 2: Basic Outline of Training and Testing

2.5 Model Improvement Strategies

Three model improvement strategies were then tested to try to prevent overfitting and increase accuracy. These three strategies include using (1) random data augmentations and (2) tuning hyperparameters with random search algorithm optimization, and (3) tuning hyperparameters with an Reduced Learning Rate on Plateau Scheduler . If any of these strategies caused improvements greater than 2% to the accuracy performance of the original CNN architecture, the strategy would then be used in the final implementation.

2.5.1 Model Improvement Strategies: Random Data Augmentations

In model improvement strategy 1, using random data augmentations, the normalization techniques were still used but the RandomHorizontalFlips and ColorJitter transformations were also used. The RandomHorizontalFlip data augmentation technique was used with a 50% probability to improve generalization by preventing the model from being overly reliant on left-right orientation of objects within the images. This helps the model learn features that are robust to orientation changes. The ColorJitter data augmentation technique was used to randomly modify the brightness, contrast, saturation, and hue of the images by a magnitude of 0.1. This makes the model more robust to natural variations in color, lighting, and object hues. It also reduces the model's sensitivity to specific color properties, making it focus on more generalizable features. The model was trained and tested with the same process as stated earlier.

2.5.2 Model Improvement Strategies: RSOA

In model improvement strategy 2, a random search optimization algorithm (RSOA) was implemented. Although grid search and manual search are the most widely used strategies for hyper-parameter optimization, the RSOA strategy was used as it only selects and tests a random combination of hyperparameters. This can help identify the best hyperparameters while also reducing testing time. Figure 3 outlines the RSOA strategy vs the ordinary grid search. As seen in Figure 3, RSOA is more effective than other searches since it can cover a wider range of possible parameters (gray circles) in the same number of searches (blue circles)

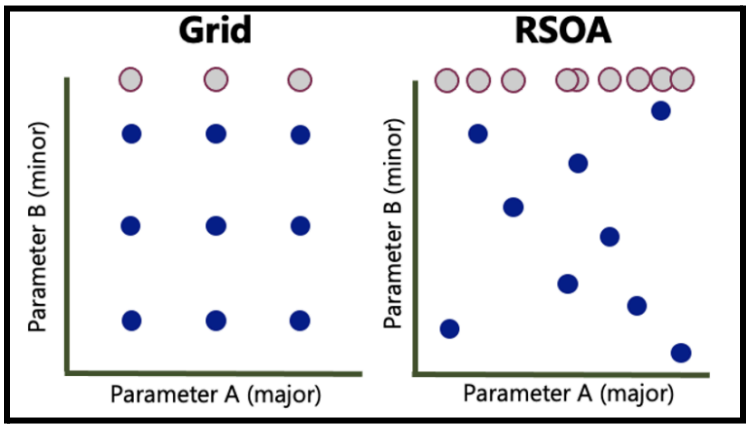


Figure 3: Grid Search vs RSOA

20 random combinations of various batch sizes and learning rates were tested. One of the three following batch sizes was randomly selected and used in each of the 20 trials: [32, 64, 128]. A random number between 0.001 and 0.01 was selected for each of these trials as well. The model was trained 20 times, each time with a random combination. Whether or not the random hyperparameter outperformed the original hyperparameters with a batch size of 64 and learning rate of 0.001, was evaluated.

2.5.3 Model Improvement Strategies: Reduced Learning Rate on Plateau Scheduler

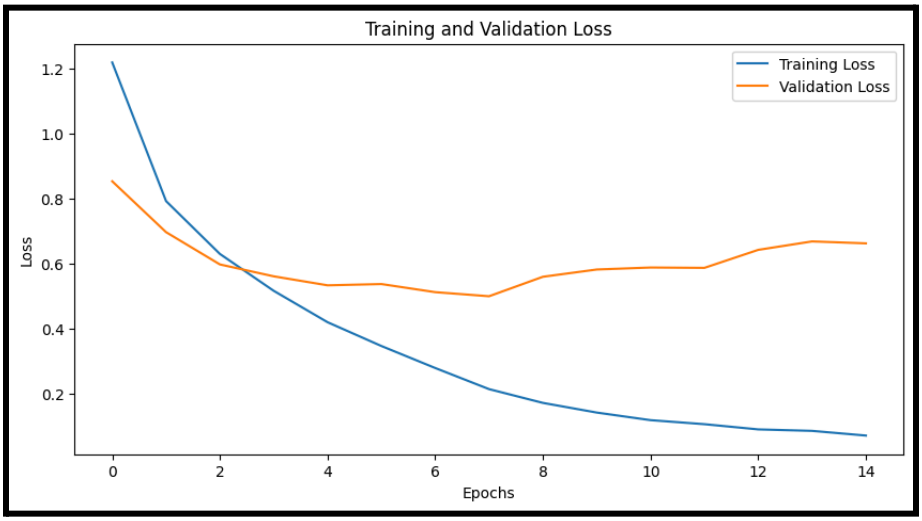
In model improvement strategy 3, a reduced learning rate on plateau scheduler was implemented. The learning rate started at 0.001 but decreased by ¼ every time that the validation loss increased throughout the 15 epochs.

3 Experiment and Results

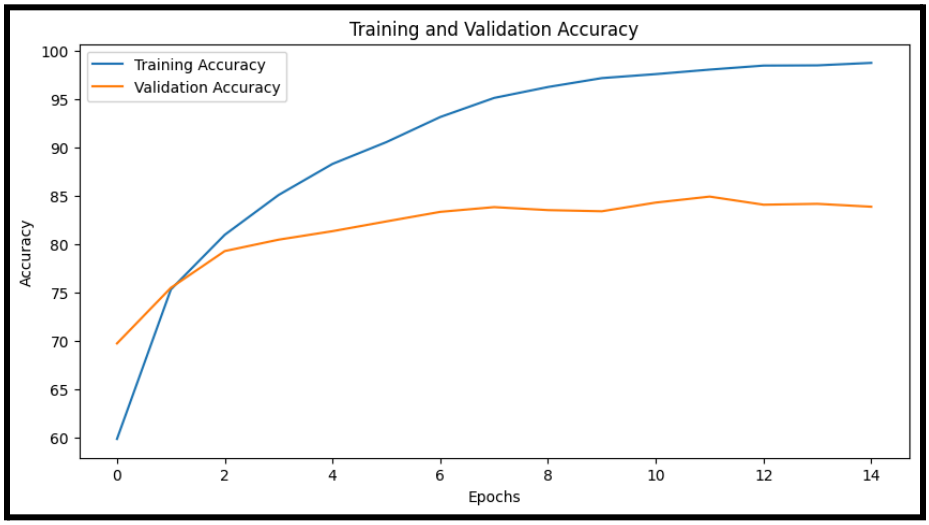
3.1 Results for Original CNN Architecture without Improvement Strategies

The original CNN model was trained on the training set then validated on the validation set for 15 epochs. Figure 2 shows the training and validation curves for the loss and accuracy of this model. The model achieved a final accuracy of 83.90% on the validation set with a

validation loss of 0.6627. As shown in Figure 2a, the validation loss increased after 7 epochs while the training loss continued to decrease, which indicates overfitting to the training data.



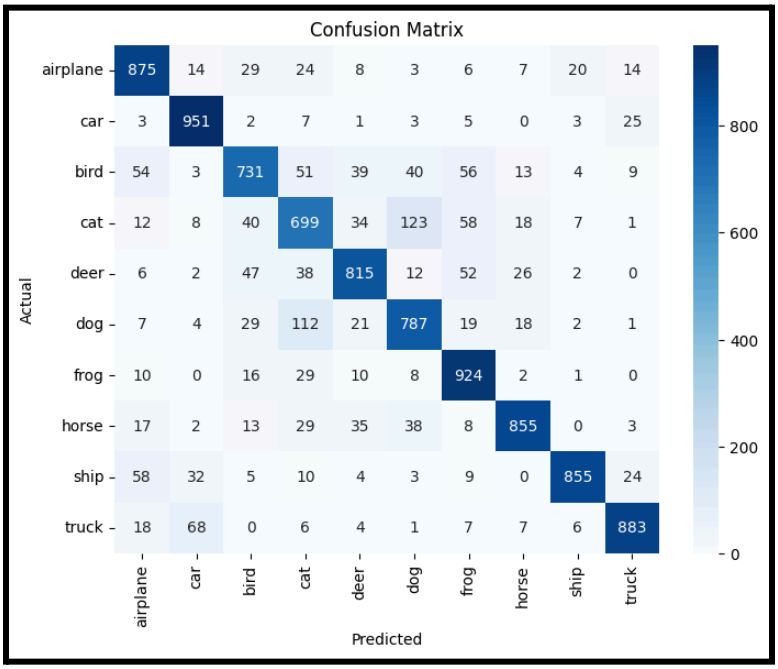
(a) Training and Validation Loss for the CNN model



(b) Training and Validation Accuracy for the CNN model

Figure 4: Graphs of training and validation loss and accuracy curves for the CNN Model

The CNN model was then tested on the testing set. Figure 3 shows the confusion matrix and classification report for this model. As can be seen in Figure 3b, the model achieved an accuracy of 84% on the unseen testset.



(a) Confusion Matrix of the CNN model

Classification Report				
	Precision	Recall	F1-Score	Support
Airplane	0.83	0.88	0.85	1000
Car	0.88	0.95	0.91	1000
Bird	0.80	0.73	0.76	1000
Cat	0.70	0.70	0.70	1000
Deer	0.84	0.81	0.83	1000
Dog	0.77	0.79	0.78	1000
Frog	0.81	0.92	0.86	1000
Horse	0.90	0.85	0.88	1000
Ship	0.95	0.85	0.90	1000
Truck	0.92	0.88	0.90	1000
Accuracy			0.84	10,000
Macro Avg	0.84	0.84	0.84	10,000
Weighted Avg	0.84	0.84	0.84	10,000

(b) Classification Report of the CNN model

Figure 5: Confusion Matrix for the CNN Model

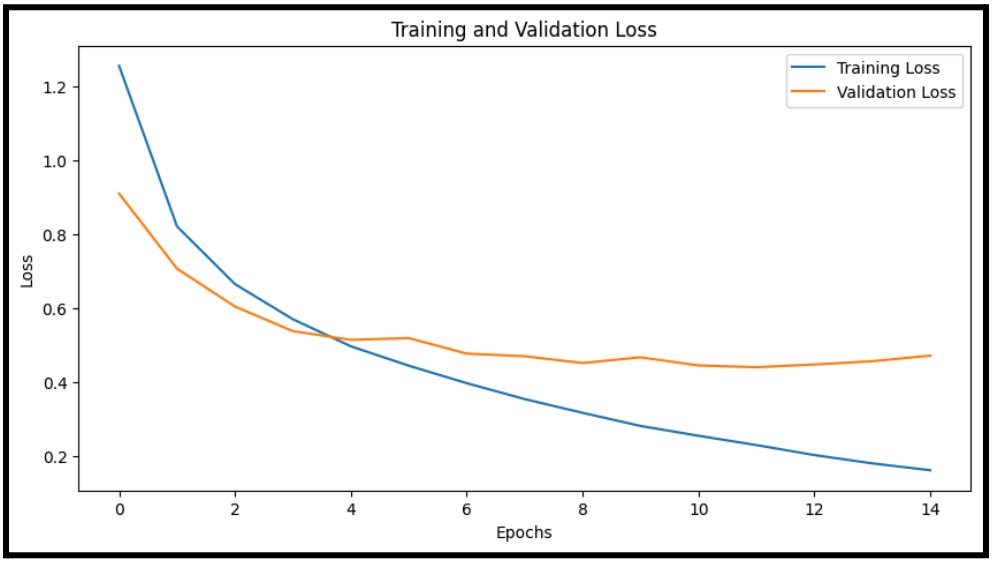
3.2 Results for CNN Architecture With Improvement Strategies

The three model improvement strategies were then tested and evaluated.

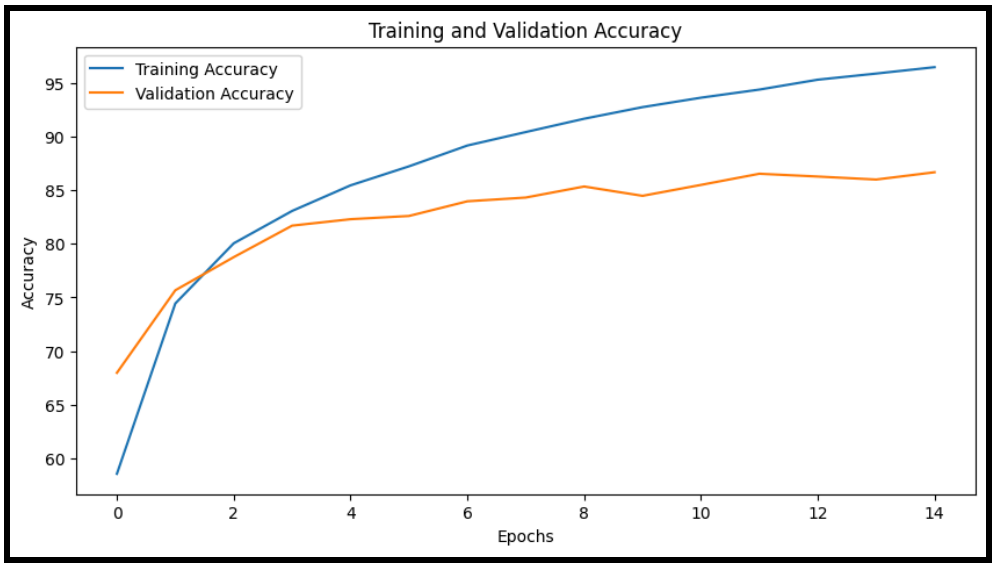
3.2.1 Results for CNN Architecture With Improvement Strategy 1: Random Data Augmentations

The random data augmentation strategy was tested first. Figure 6 outlines the training and validation curves of the CNN architecture with this strategy. The CNN model achieved an accuracy of 86.68% on the validation set and validation loss of 0.4718, improving the original accuracy by 2.78%.

Additionally, although there is a gap between the training and validation loss, the validation loss doesn't increase to the extent that it did without the random data augmentations. As a result, the random data augmentation strategy prevented overfitting.



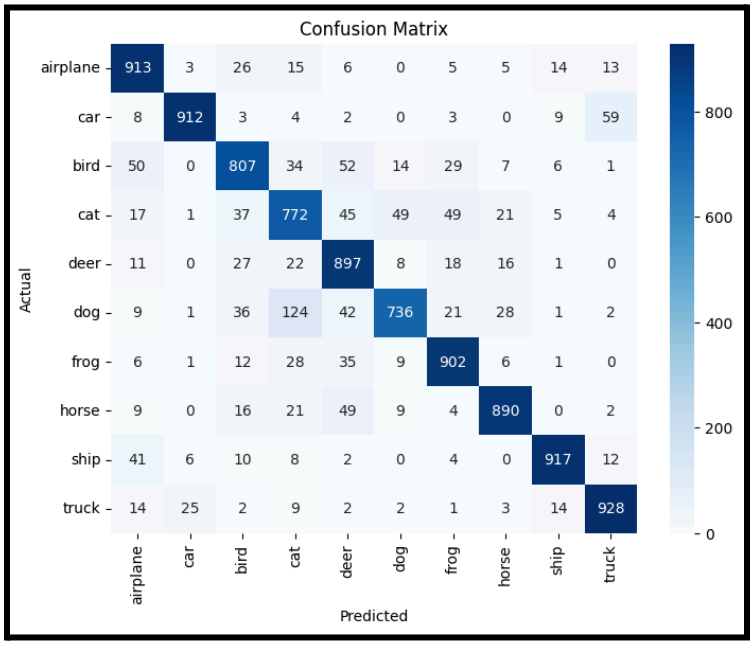
(a) Training and Validation Loss for the CNN model with Strategy 1



(b) Training and Validation Accuracy Curve for the CNN model with Strategy 1

Figure 6: Graphs of training and validation loss and accuracy curves for the CNN Model with Improvement Strategy 1

The CNN model with random data augmentations was then tested on the testing set. Figure 7 shows the confusion matrix and classification report for this model. As can be seen in Figure 7b, the model achieved an accuracy of 87% on the unseen testset, improving on the original accuracy by 3%.



(a) Confusion Matrix of the CNN model with Strategy 1

Classification Report				
	Precision	Recall	F1-Score	Support
Airplane	0.85	0.91	0.88	1000
Car	0.96	0.91	0.94	1000
Bird	0.83	0.81	0.82	1000
Cat	0.74	0.90	0.84	1000
Deer	0.79	0.90	0.84	1000
Dog	0.89	0.74	0.81	1000
Frog	0.87	0.90	0.89	1000
Horse	0.91	0.89	0.90	1000
Ship	0.95	0.92	0.93	1000
Truck	0.91	0.93	0.92	1000
Accuracy			0.87	10,000
Macro Avg	0.87	0.87	0.87	10,000
Weighted Avg	0.87	0.87	0.87	10,000

(b) Classification Report of the CNN model with Strategy 1

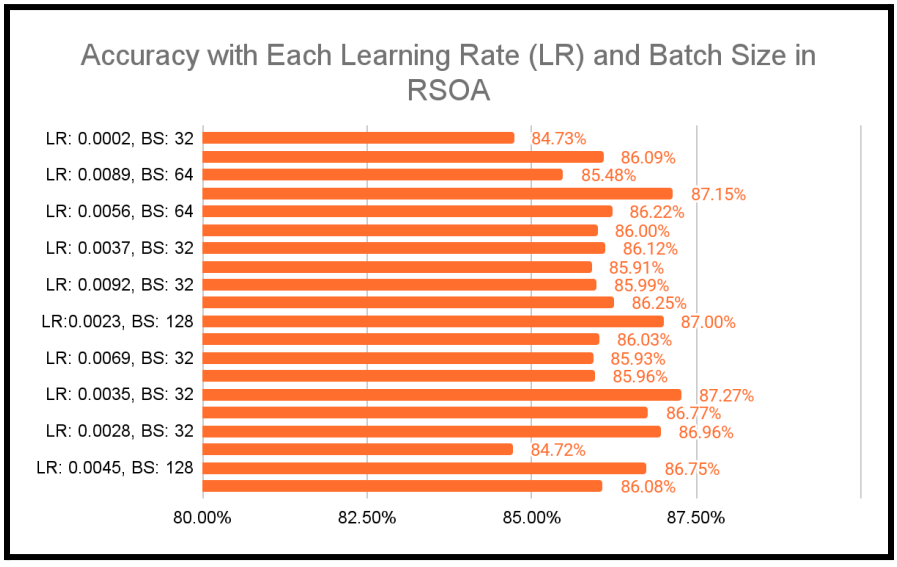
Figure 7: Confusion matrix and Classification Report for the CNN Model with Improvement Strategy 1

3.2.2 Results for CNN Architecture With Improvement Strategy 2: RSOA For Tuning Hyperparameters

As stated earlier, 20 trials with random combinations of various batch sizes and learning rates were conducted. One of the three following batch sizes was randomly selected and used in each of the 20 trials: [32, 64, 128]. A random number between 0.001 and 0.01 was selected for each of these trials as well. Figure 8 displays the results for these 20 trials with the RSOA. As can be seen in Figure 8a and Figure 8b, the learning rate of 0.0035 and batch size of 32 performed the best out of all 20 combinations with an accuracy of 87.27% on the validation set.

Learning Rate (LR) and Batch Size (BS)	Validation Accuracy
LR: 0.0002, BS: 32	84.73%
LR: 0.007, BS: 64	86.09%
LR: 0.0089, BS: 64	85.48%
LR: 0.0014, BS: 64	87.15%
LR: 0.0056, BS: 64	86.22%
LR: 0.0046, BS: 128	86.00%
LR: 0.0037, BS: 32	86.12%
LR: 0.0084, BS: 64	85.91%
LR: 0.0092, BS: 32	85.99%
LR: 0.0037, BS: 128	86.25%
LR:0.0023, BS: 128	87.00%
LR: 0.0051, BS: 32	86.03%
LR: 0.0069, BS: 32	85.93%
LR: 0.0043, BS: 64	85.96%
LR: 0.0035, BS: 32	87.27%
LR: 0.0018, BS: 128	86.77%
LR: 0.0028, BS: 32	86.96%
LR: 0.0099, BS: 64	84.72%
LR: 0.0045, BS: 128	86.75%
LR: 0.0056, BS: 64	86.08%

(a) Table for Random Learning Rates and Batch Sizes Selected and Their Respective Accuracies on the Validation Set

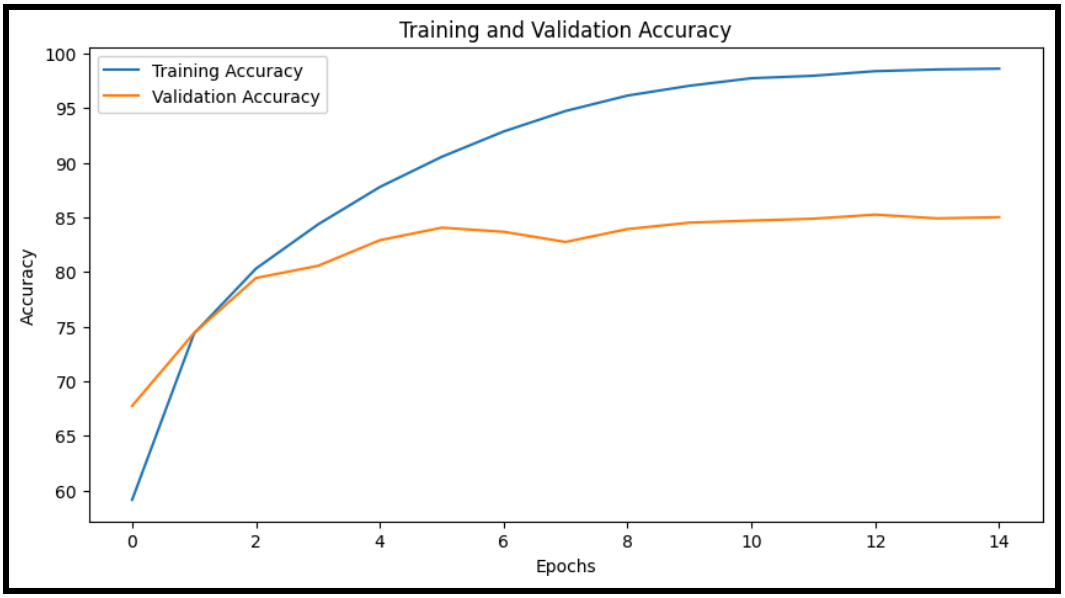


(b) Graph of Random Learning Rates and Batch Sizes Selected and Their Respective Accuracies on the Validation Set

Figure 8: Table and Graph of the RSOA Experiment

The model with the learning rate of 0.0035 and batch size of 32 was then retrained and evaluated on an unseen test set. The model did not perform as well with a validation accuracy of 85.02% although this is still 1.12% better than the model without any improvement strategies.

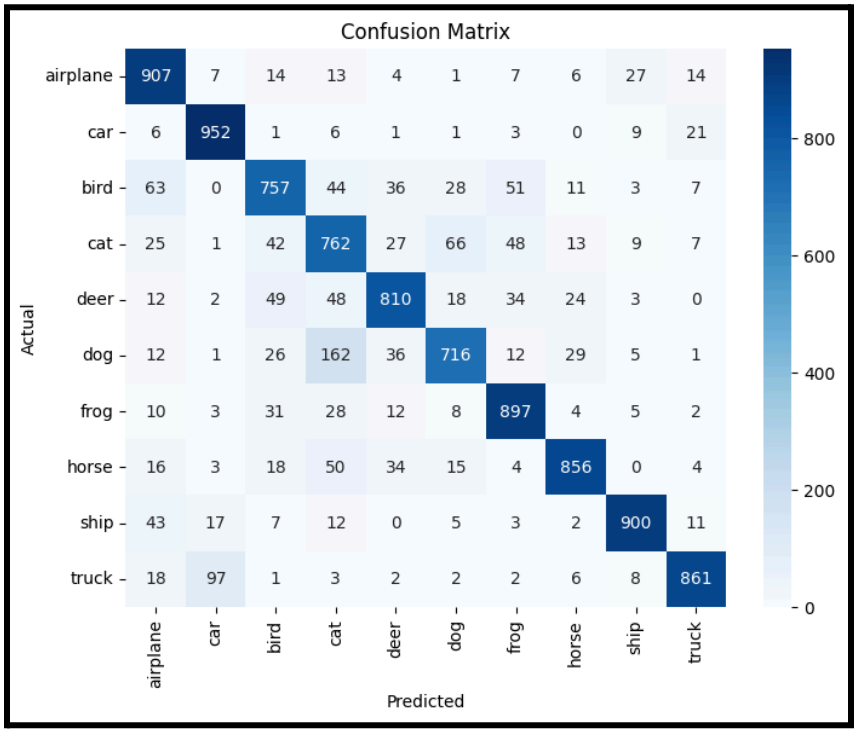
(c) Training and Validation Loss for the CNN model with Strategy 2



(d) Training and Validation Accuracy Curve for the CNN model with Strategy 2

Figure 9: Graphs of training and validation loss and accuracy curves for the CNN Model with Improvement Strategy 2

As can be seen in Figure 9, the model was then tested on the unseen testset but performed with an accuracy of 84%, not improving at all on the original model without this improvement strategy.



(c) Confusion Matrix of the CNN model with Strategy 2

Classification Report				
	Precision	Recall	F1-Score	Support
Airplane	0.82	0.91	0.86	1000
Car	0.88	0.95	0.91	1000
Bird	0.80	0.76	0.78	1000
Cat	0.68	0.76	0.72	1000
Deer	0.84	0.81	0.83	1000
Dog	0.83	0.72	0.77	1000
Frog	0.85	0.90	0.87	1000
Horse	0.90	0.86	0.88	1000
Ship	0.93	0.90	0.91	1000
Truck	0.93	0.86	0.89	1000
Accuracy			0.84	10,000
Macro Avg	0.84	0.84	0.84	10,000
Weighted Avg	0.84	0.84	0.84	10,000

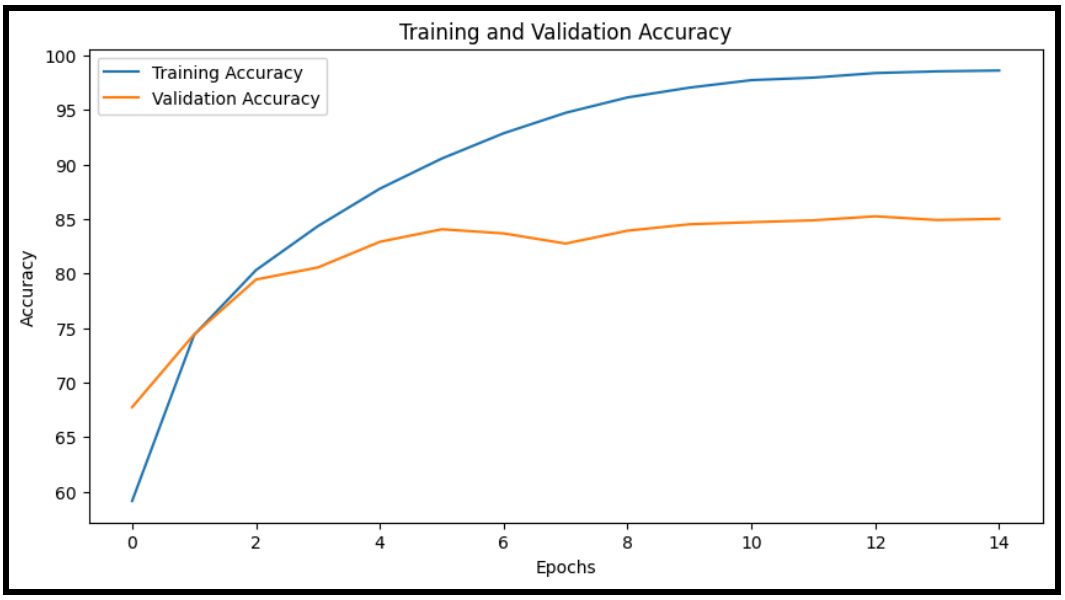
(d) Classification Report of the CNN model with Strategy 2

Figure 10: Confusion matrix and Classification Report for the CNN Model with Improvement Strategy 1

3.2.3 Results for CNN Architecture With Improvement Strategy 3: Reduced Learning Rate on Plateau Scheduler

The model started off with a learning rate of 0.001 and a batch size of 75, and trained for 15 epochs. The learning rate decreased by ¼ every time that the validation loss decreased. Figure 11 outlines the training and validation curves of the CNN architecture with this strategy. The CNN model achieved an accuracy of 86.66% on the validation set and validation loss of 0.5621, improving the original accuracy by 2.76%.

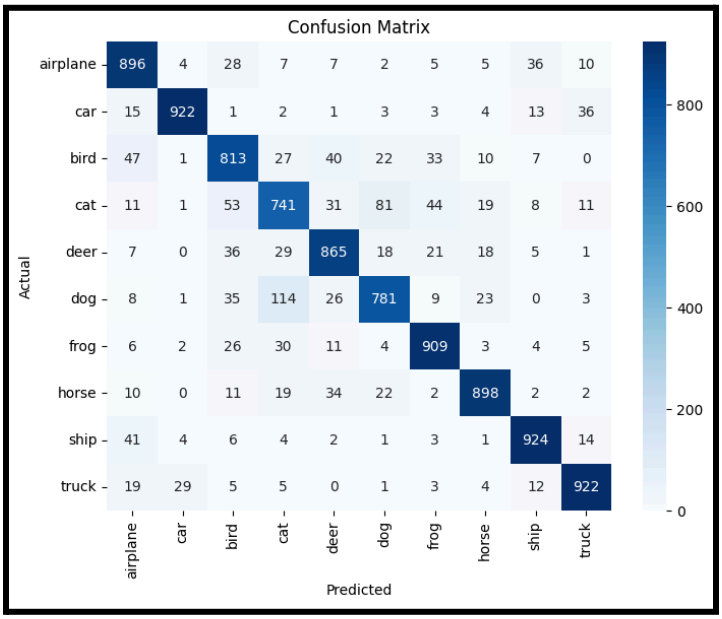
(a) Training and Validation Loss for the CNN model with Strategy 3



(b) Training and Validation Accuracy Curve for the CNN model with Strategy 3

Figure 11: Graphs of training and validation loss and accuracy curves for the CNN Model with Improvement Strategy 3

The CNN model with the Reduced LR on Plateau Scheduler was then tested on the testing set. Figure 12 shows the confusion matrix and classification report for this model. As can be seen in Figure 12b, the model achieved an accuracy of 87% on the unseen testset, improving on the original accuracy by 3%.



(a) Confusion Matrix of the CNN model with Strategy 3

Classification Report				
	Precision	Recall	F1-Score	Support
Airplane	0.85	0.90	0.87	1000
Car	0.96	0.92	0.94	1000
Bird	0.80	0.81	0.81	1000
Cat	0.76	0.74	0.75	1000
Deer	0.85	0.86	0.86	1000
Dog	0.84	0.78	0.81	1000
Frog	0.88	0.91	0.89	1000
Horse	0.91	0.90	0.90	1000
Ship	0.91	0.92	0.92	1000
Truck	0.92	0.92	0.92	1000
Accuracy			0.87	10,000
Macro Avg	0.87	0.87	0.87	10,000
Weighted Avg	0.87	0.87	0.897	10,000

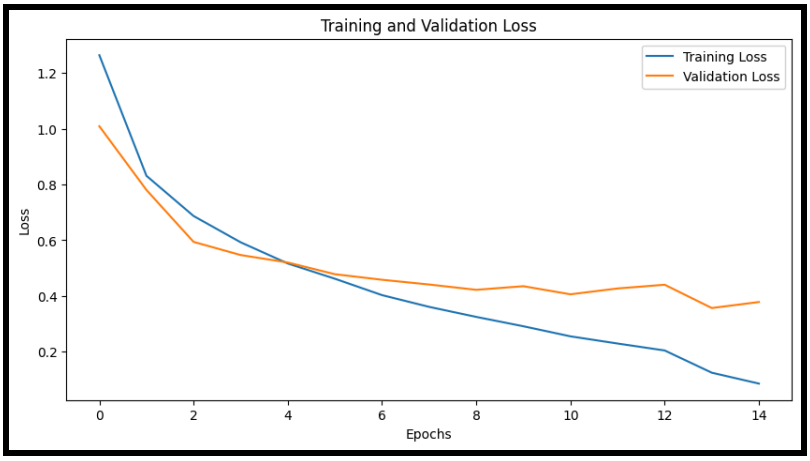
(b) Classification Report of the CNN model with Strategy 3

Figure 12: Confusion matrix and Classification Report for the CNN Model with Improvement Strategy 3

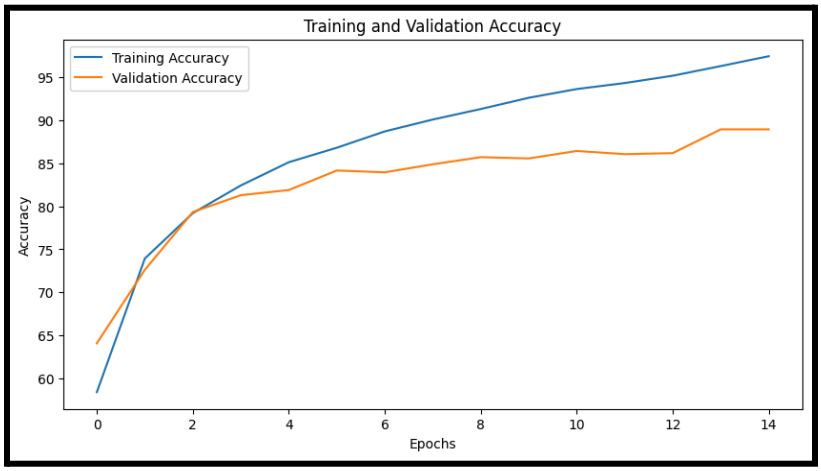
3.2.4 Results for CNN Architecture With Combining Strategy 1 and Strategy 3

Since Strategy 1, Random Data Augmentation and Strategy 3, Reduced Learning Rate on Plateau Scheduler, improved on the original CNN by the greatest margins, a CNN model using both of these strategies was tested.

Figure 13 outlines the training and validation curves for training and validation for the CNN model trained with both of these strategies. The CNN trained with this strategy achieved a validation accuracy of 88.94%, improving on the original CNN model by 5.04%. This strategy yielded the highest improvement. Additionally there is the smallest gap between the training and validation curves for the model trained with this two strategies, indicating that it prevented overfitting.



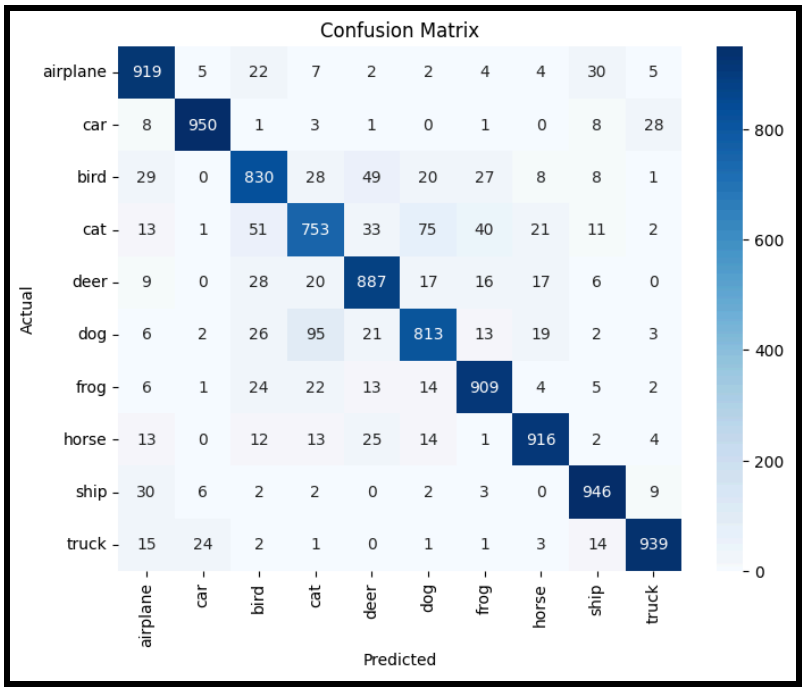
(e) Training and Validation Loss for the CNN model with Strategies 1 and 3



(f) Training and Validation Accuracy Curve for the CNN model with Strategies 1 and 3

Figure 13: Graphs of training and validation loss and accuracy curves for the CNN Model with Improvement Strategies and 3

The CNN model was then tested on the testing set. Figure 14 shows the confusion matrix and classification report for this model. As can be seen in Figure 14b, the model achieved an accuracy of 89% on the unseen testset, improving on the original accuracy by 5%. This is the highest improvement of any strategy used.



(a) Confusion Matrix of the CNN model with Strategy 1 and 3

Classification Report				
	Precision	Recall	F1-Score	Support
Airplane	0.88	0.92	0.90	1000
Car	0.96	0.95	0.96	1000
Bird	0.83	0.83	0.83	1000
Cat	0.80	0.75	0.77	1000
Deer	0.86	0.89	0.87	1000
Dog	0.85	0.81	0.83	1000
Frog	0.90	0.91	0.90	1000
Horse	0.92	0.92	0.92	1000
Ship	0.92	0.95	0.93	1000
Truck	0.95	0.94	0.94	1000
Accuracy			0.89	10,000
Macro Avg	0.89	0.89	0.89	10,000
Weighted Avg	0.89	0.89	0.89	10,000

(b) Classification Report of the CNN model with Strategy 1 and 3

Figure 14: Confusion matrix and Classification Report for the CNN Model with Improvement Strategies 1 and 3

The following table, Table 1, displays all of these CNN architectures and their testing accuracies to compare all of them.

CNN Model Training Strategy	Accuracy on Test set	Improvement to Original CNN
Original CNN Model	84%	-
CNN Model with Random Data Augmentations	87%	3%
CNN Model with RSOA	84%	0%
CNN Model with Reduced LR on Plateau Scheduler	87%	3%
CNN Model with Random Data Augmentations and Reduced LR on Plateau Scheduler	89%	5%

4 Conclusion

The CNN architecture trained with random data Augmentations and a reduced LR on plateau scheduler performed the best on the CIFAR-10 dataset with an accuracy of 89%.