Image Classification Using Deep Learning on CIFAR-10 Dataset

Gajan Mohan Raj

TABLE OF CONTENTS

01

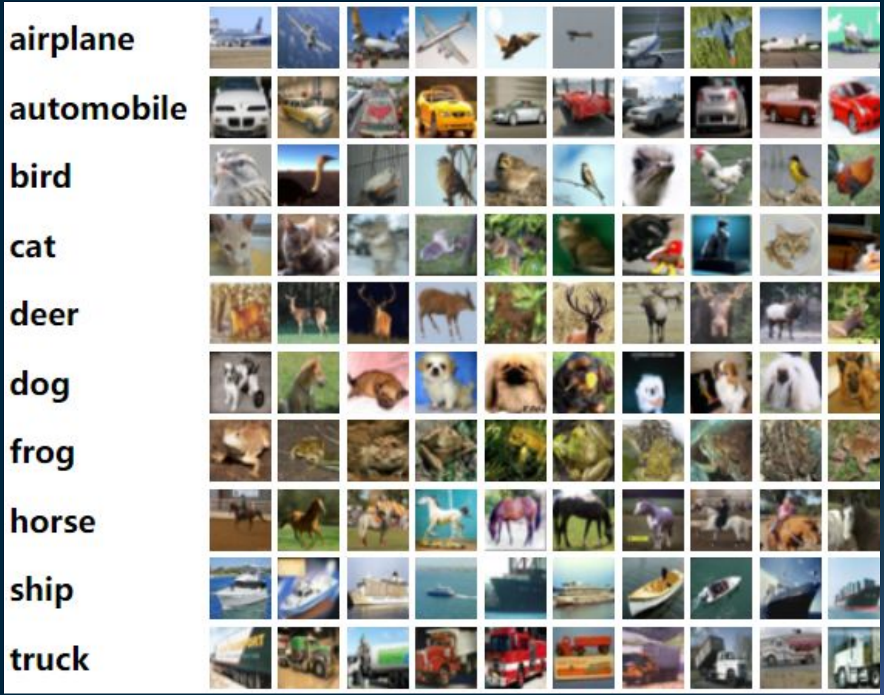
Background and Requirements

Background and Requirements

In this project, I was tasked with building and training a Convolutional Neural Network (CNN) to classify images from the CIFAR-10 dataset. CIFAR-10 is a well-known dataset used in the machine learning community, consisting of 60,000 32x32 color images in 10 different classes.

Requirements:

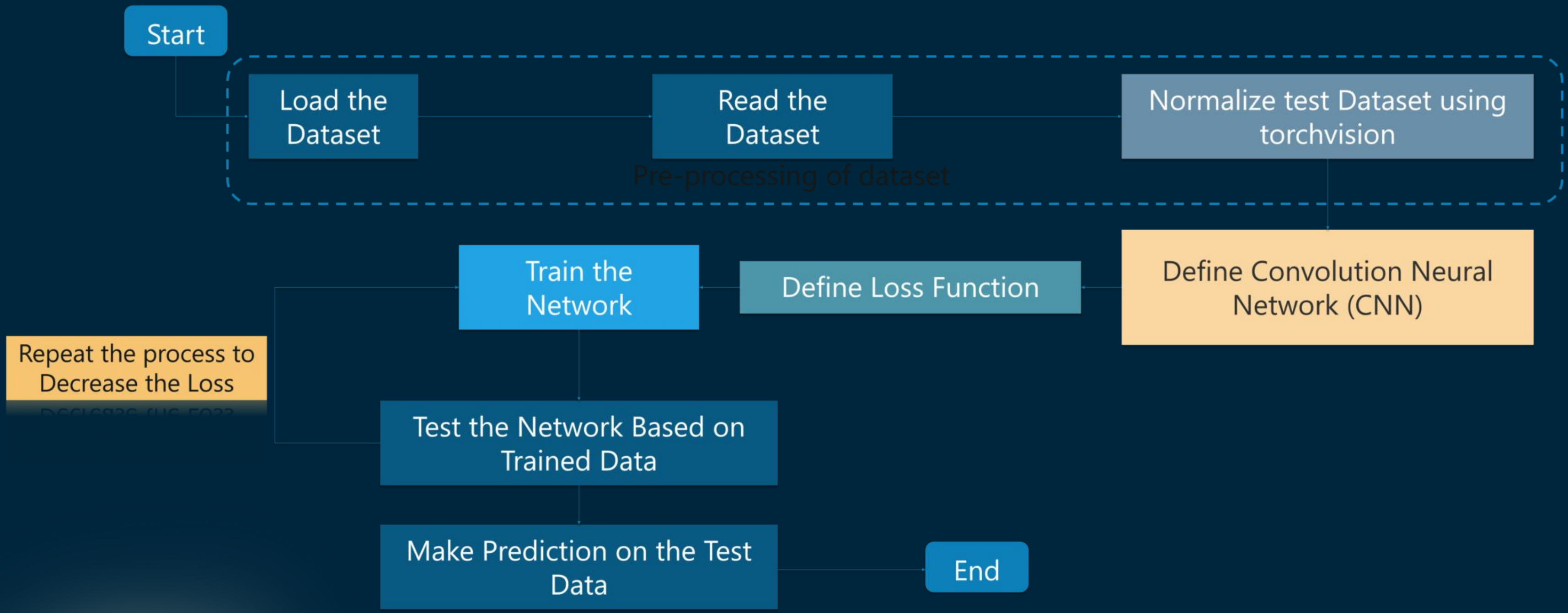
- Programming Language: Python
- Main Library: PyTorch Additional Libraries:
- Torchvision: For handling image loading and transformations.
- Matplotlib: For visualizing images and training results.



02

Project Approach

Project Approach: General Layout

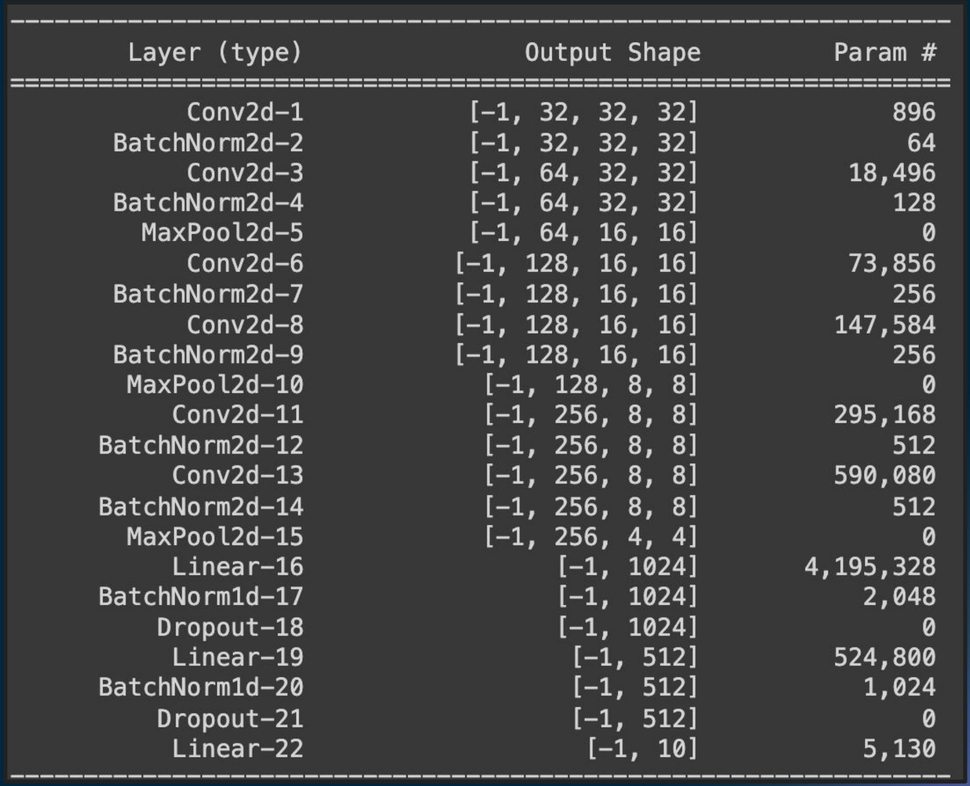

Start
Load the Dataset
Read the Dataset
Normalize test Dataset using torchvision
Pre-processing of dataset
Define Convolution Neural Network (CNN)
Define Loss Function
Train the Network
Repeat the process to Decrease the Loss
Test the Network Based on Trained Data
Make Prediction on the Test Data
End

Project Approach: Building the CNN Architecture

Convolutional neural networks (CNNs) are commonly used for image classification.

In this project, I constructed a CNN architecture with a series of convolutional blocks followed by dense (fully-connected) layers:

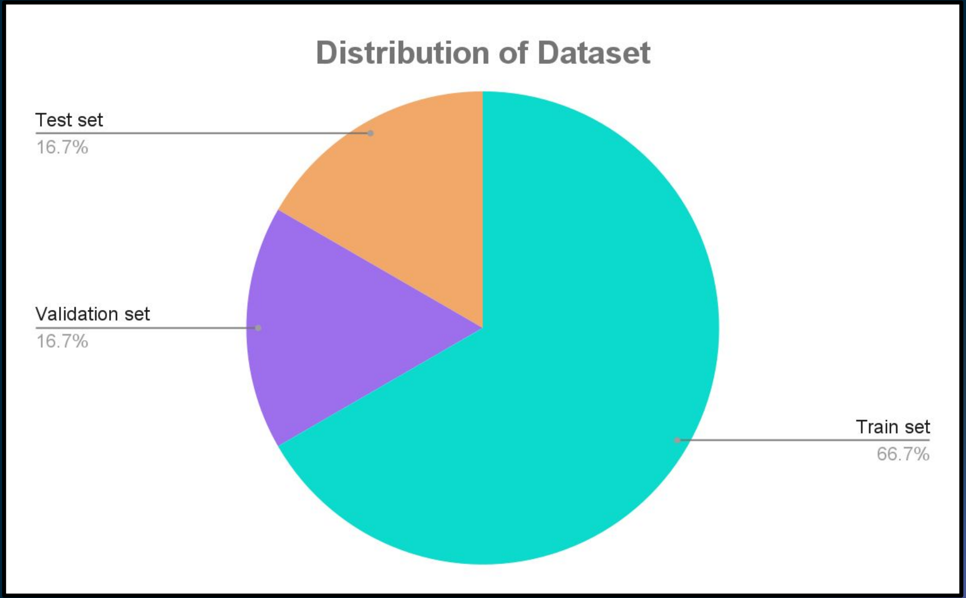
- Several convolutional blocks, where each block typically includes a convolutional layer followed by batch normalization and ReLU activation
- Max pooling layers placed periodically between convolutional blocks to reduce spatial dimension of feature maps and control overfitting
- Dense layers include batch normalization and dropout layers for regularization



```
----------------------------------------------------------------
        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
----------------------------------------------------------------
```

Project Approach: Training Process

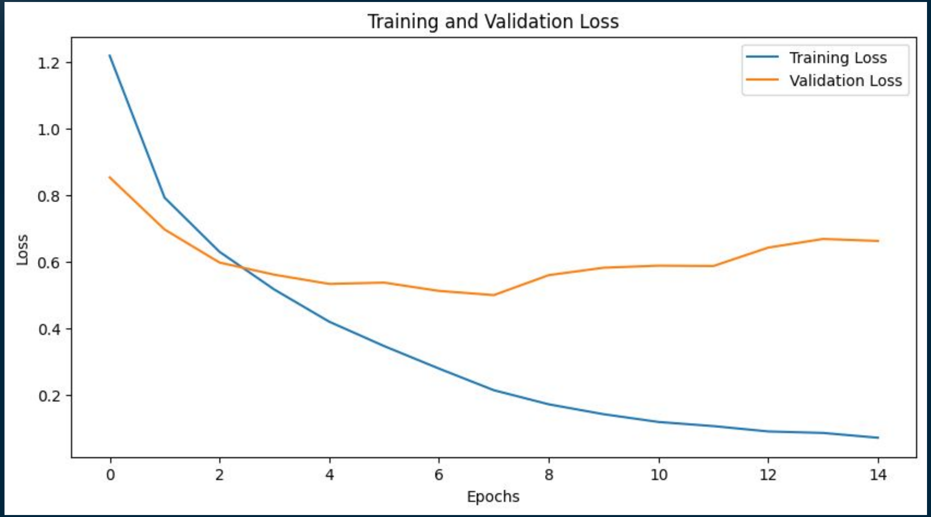
- **Dataset preparation**
  - There are 60,000 images of the CIFAR-10 dataset
  - All images were first normalized to improve performance and stability of the CNN
  - Then split into three distinct datasets using random_split
    - Train set
      - 40,000 images
    - Validation set
      - 10,000 images
    - Test set
      - 10,000 images
- **Hyperparameters**
  - *Loss function*: Cross entropy loss
  - *Optimizer*: Adam optimizer
  - *Epochs: 15*
  - *Learning rate: 0.001*
  - *Batch size: 64*

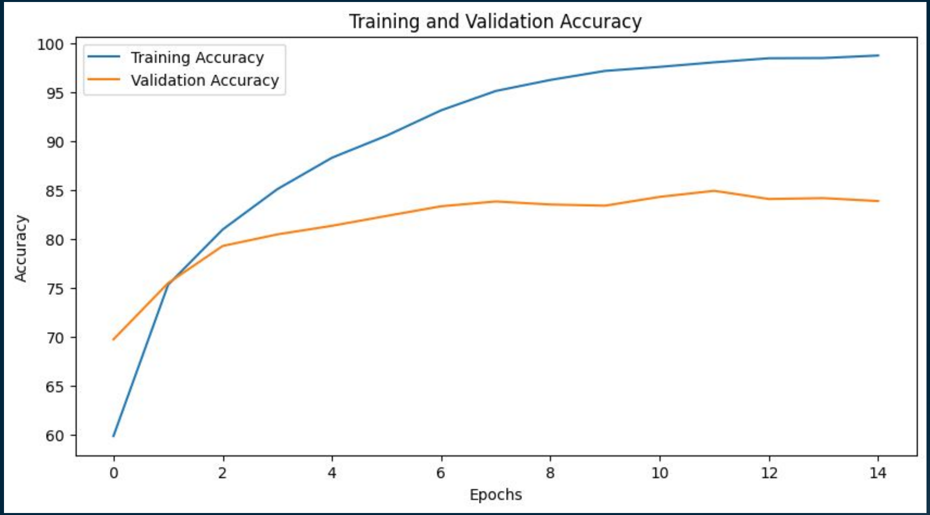


03

Findings

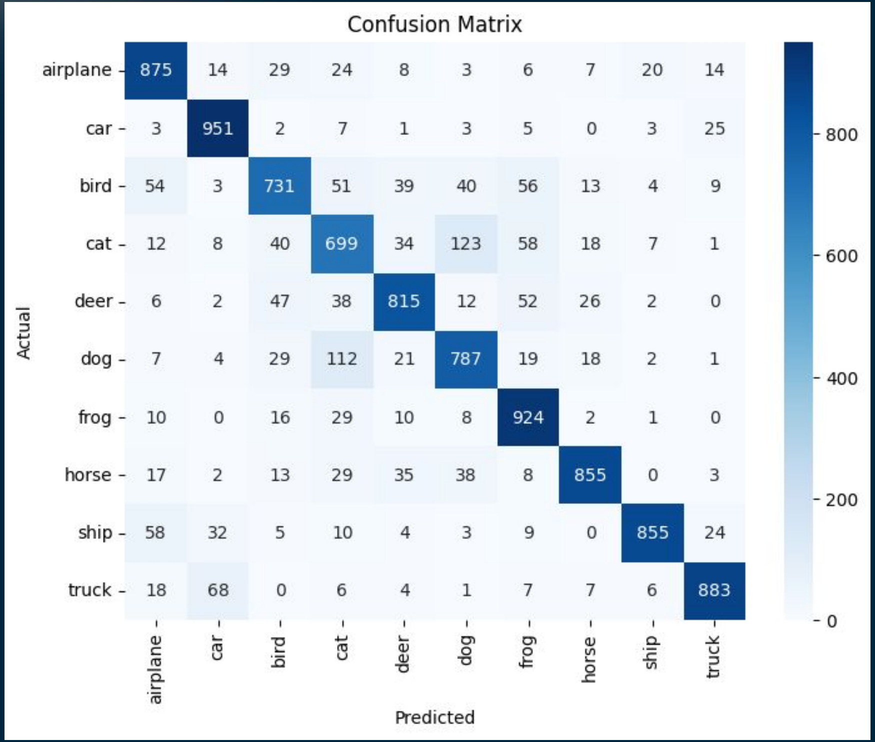
Findings: Training and Validation Curves for Original CNN





- The CNN architecture performed very well with an accuracy of **83.90%** on the validation set
- The validation loss tended to increase after the 7th epoch while the training loss continued to decrease, indicating overfitting
    - Early stopping and extra regularization techniques could be used to prevent overfitting

Findings: Performance on Test Set



Actual \ Predicted	airplane	car	bird	cat	deer	dog	frog	horse	ship	truck
airplane	875	14	29	24	8	3	6	7	20	14
car	3	951	2	7	1	3	5	0	3	25
bird	54	3	731	51	39	40	56	13	4	9
cat	12	8	40	699	34	123	58	18	7	1
deer	6	2	47	38	815	12	52	26	2	0
dog	7	4	29	112	21	787	19	18	2	1
frog	10	0	16	29	10	8	924	2	1	0
horse	17	2	13	29	35	38	8	855	0	3
ship	58	32	5	10	4	3	9	0	855	24
truck	18	68	0	6	4	1	7	7	6	883

- The CNN architecture performed very well on the test set with an impressive accuracy of **84%**!

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

04

Model Improvement Strategies

Model Improvement Strategies

3 different strategies were attempted to try to improve the CNN model's accuracy.

1

Random Data Augmentations

Random data augmentations can be performed to artificially increase the size and diversity of the training dataset through random flips, rotations, etc.

2

Use RSOA for Hyperparameter Tuning

Hyperparameter tuning with Random Search can be performed to find the most optimal hyperparameters for the training of CNN architecture.

3

Use Reduced Learning Rate on Plateau Scheduler

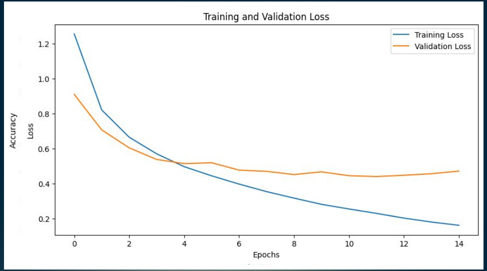
We can reduce the learning rate by ¼ every time the validation loss increases to improve model convergence

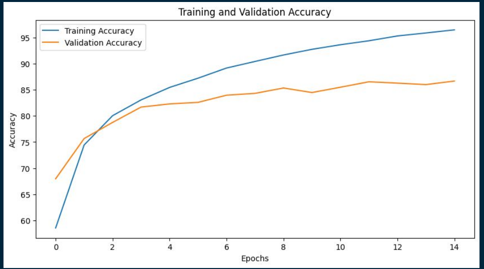
Strategies that improved on the original CNN model were implemented into the final implementation of the model.

Model Improvement: Random Data Augmentation

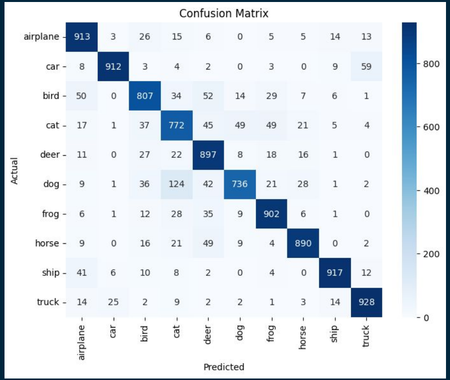
- Random horizontal flips with a 50% probability of occurring were added
- The brightness, contrast, saturation, and hue of image were randomly modified by a small degree
  - Forces the model to learn features that are less reliant on exact color values

Findings:

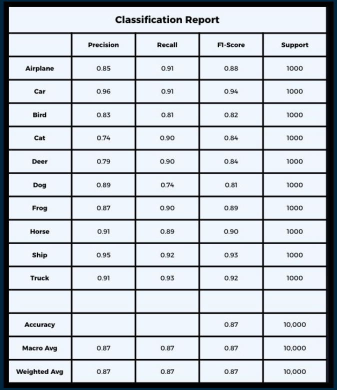




Findings Cont.:



	airplane	car	bird	cat	deer	dog	frog	horse	ship	truck
airplane	913	3	26	15	6	0	5	5	14	13
car	8	912	3	4	2	0	3	0	9	59
bird	50	0	807	34	52	14	29	7	6	1
cat	17	1	37	772	45	49	49	21	5	4
deer	11	0	27	22	897	8	18	16	1	0
dog	9	1	36	124	42	736	21	28	1	2
frog	6	1	12	28	35	9	902	6	1	0
horse	9	0	16	21	49	9	4	890	0	2
ship	41	6	10	8	2	0	4	0	917	12
truck	14	25	2	9	2	2	1	3	14	928



Classification Report

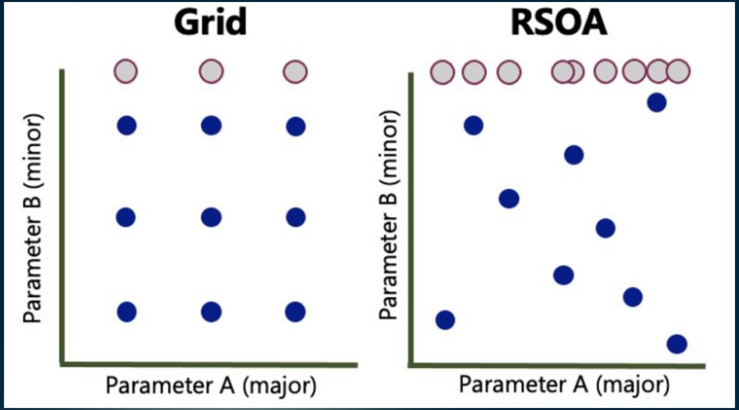
	Precision	Recall	F1-Score	Support
Airplane	0.85	0.91	0.88	1000
Car	0.96	0.91	0.94	1000
Bird	0.83	0.81	0.82	1000
Cat	0.74	0.90	0.84	1000
Deer	0.79	0.90	0.84	1000
Dog	0.89	0.74	0.81	1000
Frog	0.87	0.90	0.89	1000
Horse	0.91	0.89	0.90	1000
Ship	0.95	0.92	0.93	1000
Truck	0.91	0.93	0.92	1000
Accuracy			0.87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

Effects of Random Data Augmentation:

- Less overfitting than before
- Improvement to accuracy by 3% from 84% to **87%**
- Can be implemented into final implementation

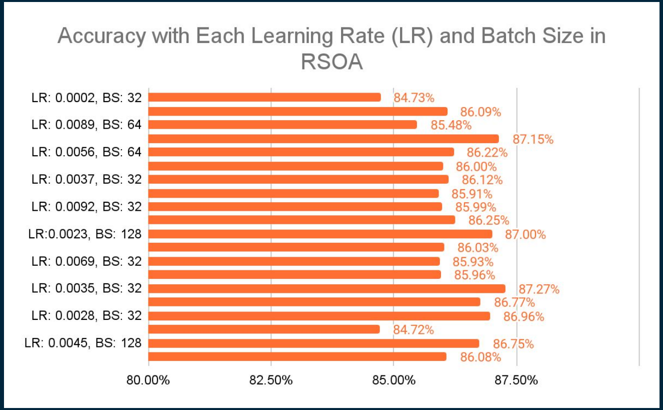
Model Improvement: Using RSOA for Hyperparameter Tuning

- The Random Search Optimization Algorithm (RSOA) was used to choose an optimal set of hyperparameters
- 20 random combinations of batch size and learning rate were tested to find optimal hyperparameters
    - Randomly select learning rate between 0.001 and 0.01
    - Randomly select batch size in [32, 64, 128]

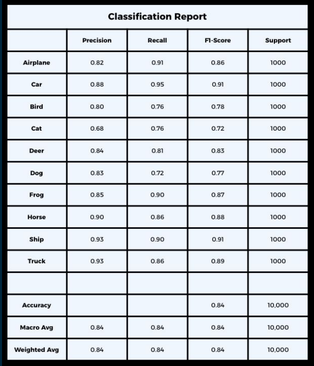


RSOA is more effective than other searches since it can cover a wider range of possible parameters (gray circles) in the same number of searches (blue circles)

- Findings:



- A learning rate of 0.0035 and batch size of 32 performed the best for the final accuracy of the CNN architecture on the validation set, but it then only achieved an accuracy of 84% on the unseen testset



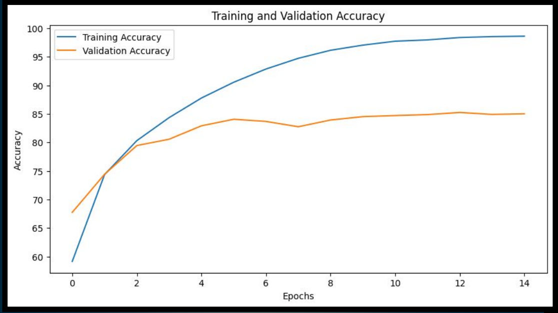
Classification Report

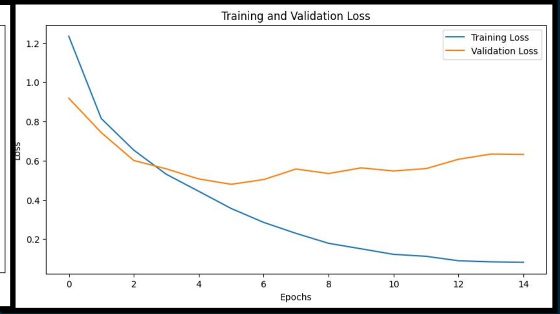
	Precision	Recall	F1-Score	Support
Airplane	0.82	0.91	0.86	1000
Car	0.88	0.95	0.91	1000
Bird	0.80	0.76	0.78	1000
Cat	0.68	0.76	0.72	1000
Deer	0.84	0.81	0.83	1000
Dog	0.83	0.72	0.77	1000
Frog	0.85	0.90	0.87	1000
Horse	0.90	0.86	0.88	1000
Ship	0.93	0.90	0.91	1000
Truck	0.93	0.86	0.89	1000
Accuracy			0.84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

Model Improvement: Reducing Learning Rate on Plateau Scheduler

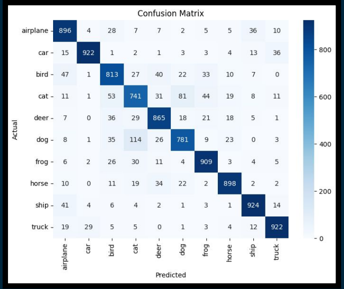
- The Reducing LR on Plateau Scheduler was implemented
- LR Reduces by ¼ every time that the validation loss increased during the 15 epochs

Findings:

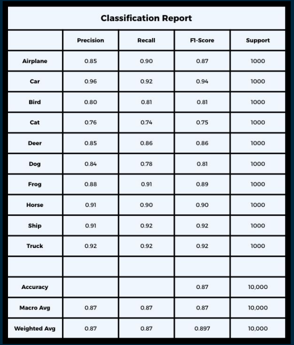




Findings Cont.:



	airplane	car	bird	cat	deer	dog	frog	horse	ship	truck
airplane	896	4	28	7	7	2	5	5	36	10
car	15	922	1	2	1	3	3	4	13	36
bird	47	1	813	27	40	22	33	10	7	0
cat	11	1	53	741	31	81	44	19	8	11
deer	7	0	36	29	865	18	21	18	5	1
dog	8	1	35	114	26	781	9	23	0	3
frog	6	2	26	30	11	4	909	3	4	5
horse	10	0	11	19	34	22	2	898	2	2
ship	41	4	6	4	2	1	3	1	924	14
truck	19	29	5	5	0	1	3	4	12	922



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

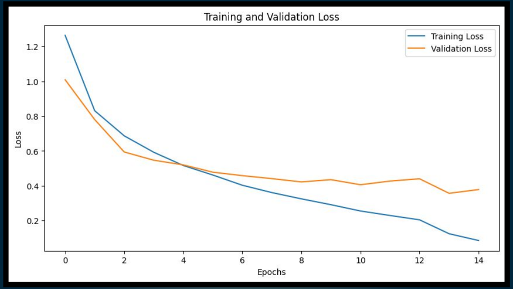
Effects of the Scheduler:

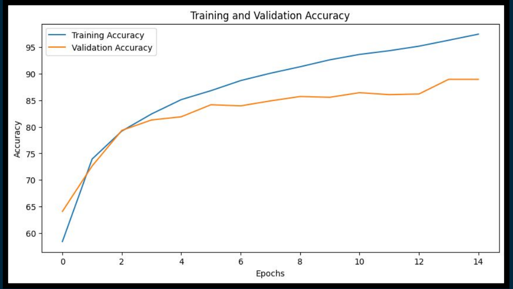
- Improvement to accuracy by 3% from 84% **87%**
- Can be implemented into final implementation

Final Implementation: Combining Random Data Augmentation with Reducing LR on Plateau Scheduler

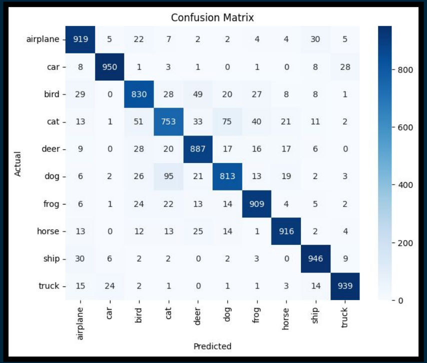
- Since Strategy 1, Random Data Augmentation and Strategy 3, Reduced Learning Rate on Plateau Scheduler, improved on the original CNN by the greatest margins, a CNN model using both of these strategies was tested

Findings:

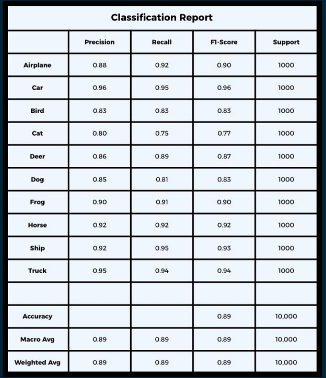




Findings Cont.:


Confusion Matrix

Actual \ Predicted	airplane	car	bird	cat	deer	dog	frog	horse	ship	truck
airplane	919	5	22	7	2	2	4	4	30	5
car	8	950	1	3	1	0	1	0	8	28
bird	29	0	830	28	49	20	27	8	8	1
cat	13	1	51	753	33	75	40	21	11	2
deer	9	0	28	20	887	17	16	17	6	0
dog	6	2	26	95	21	813	13	19	2	3
frog	6	1	24	22	13	14	909	4	5	2
horse	13	0	12	13	25	14	1	916	2	4
ship	30	6	2	2	0	2	3	0	946	9
truck	15	24	2	1	0	1	1	3	14	939


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

- Combining these two strategies yielded in the biggest improvement with an **accuracy of 89%** as compared to only 84% earlier!

Final Conclusion

- The CNN architecture with the Random Data Augmentations and Reduced LR on Plateau Scheduler performed very well with an accuracy of 89% on the CIFAR-10 Dataset!

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%**