Introduction

CIFAR-10 was chosen as the image classification dataset for this assignment. CIFAR-10 comprises 60,000 color images of size 32x32 distributed among 10 classes (GeeksforGeeks, 2025). This dataset was chosen because its manageable size (50,000 training, 10,000 test images) demonstrates transfer learning principles while remaining computationally feasible within Google Colab's constraints. The low resolution and the 10-class structure create a difficult assignment that can be tackled with the help of pre-trained feature extraction.

The pre-trained ResNet18 model was chosen because it has a good track record on image classification and is also available in PyTorch Vision (PyTorch, n.d.). The use of transfer learning with pre-trained models is a major factor in achieving the desired performance level on smaller datasets where training from scratch would be impractical (Howard & Gugger, 2020).

Methods

A ResNet18 model pre-trained on ImageNet was employed (PyTorch, n.d.). The 1000-class output layer of the model was substituted with a 10-class layer intended for CIFAR-10. This transfer learning approach (Howard & Gugger, 2020) utilizes ImageNet’s learned feature representations while the classifier is adapted to the CIFAR-10 categories.

Training Configuration:

- Loss function: Cross-Entropy Loss
- Optimizer: SGD (learning rate: 0.001, momentum: 0.9)
- Batch size: 128
- Epochs: 50
- Data split: 80% training (40,000), 20% validation (10,000)
- Augmentation: Random horizontal flips and crops

Results

Question 1: Can you create and document a scenario where over training occurred?

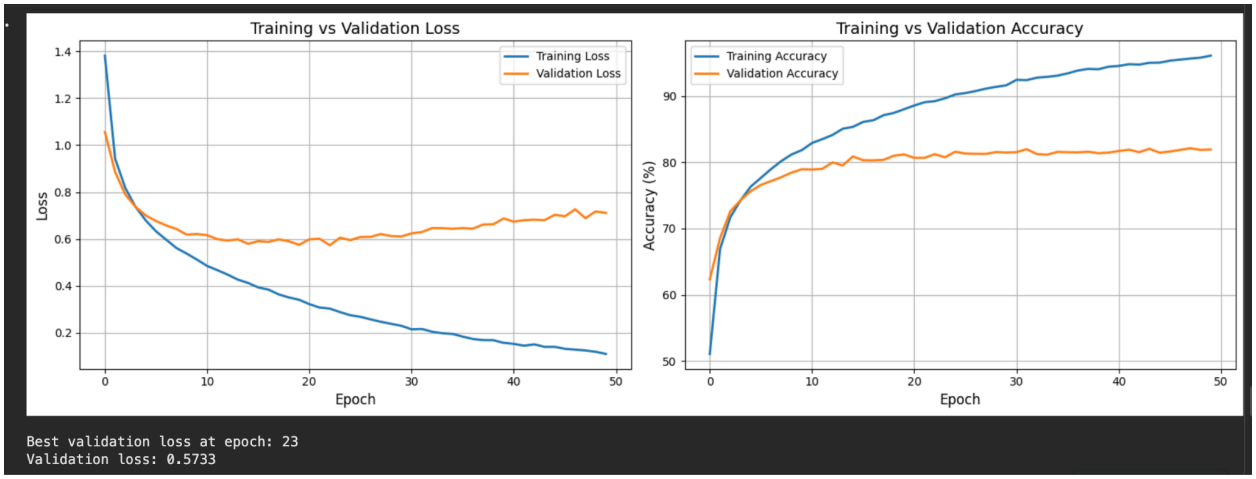

Figure 1. Training and validation loss (left) and accuracy (right) over 50 epochs, showing overfitting after epoch 23.

Overtraining was clearly observed. The validation loss reached its minimum of 0.5733 at epoch 23. After that, the training loss continued to decrease until it reached 0.1100, while on the other hand, the validation loss rose to 0.7110 by the end of epoch 50, representing a **24.02% increase** which is a clear sign of overfitting.

**Note: We have formula:*

*Percentage Increase = ((New Value - Old Value) / Old Value) * 100*

*-> ((0.7110 - 0.5733) / 0.5733) * 100 = 24.02*

```
Epoch 23/50
Training: 100%|          | 313/313 [00:18<00:00, 17.22it/s]
Validation: 100%|          | 79/79 [00:03<00:00, 23.12it/s]
Train Loss: 0.3032 | Train Acc: 89.21%
Val Loss: 0.5733 | Val Acc: 81.21%
```

```
Epoch 50/50
Training: 100%|██████████| 313/313 [00:19<00:00, 16.42it/s]
Validation: 100%|██████████| 79/79 [00:03<00:00, 23.48it/s]Train Loss: 0.1100 | Train Acc: 96.08%
Val Loss: 0.7110 | Val Acc: 81.92%
```

At epoch 50, the training accuracy was at **96.08%** meanwhile the validation accuracy was **81.92%**, thus making a performance gap of **14.16%**. This big difference indicates the model just memorized the training data rather than learning the patterns that can be applied to other data. The ratio of training to validation loss **increased** from **1.9** (0.5733 / 0.3032 = 1.89 ~ 1.9) at epoch 23 to **6.5** (0.7110 / 0.1100 = 6.46 ~ 6.5) at epoch 50, confirming severe overfitting.

Question 2: What training methods did you find helpful-useful to prevent overtraining and why?

I found **2 training methods** that are helpful to prevent overtraining:

1. **Stop training early**

Implementing early stopping at epoch 23 would prevent 27 epochs of unnecessary training. In this way, the model will be kept in its best condition with validation loss of 0.5733 by monitoring validation loss and stopping after 3-5 epochs without improvement, thus avoiding the 24% decline to 0.7110 by epoch 50. This simple technique requires no architectural changes and is highly effective (Howard & Gugger, 2020).

2. **Freeze layer**

Freezing pre-trained layers while training only the final classifier would reduce trainable parameters from 11.7 million to ~5,000 (99.96% reduction). This works as strong regularization because the model only learns the decision boundary for CIFAR-10 classes, while keeping all the useful features ResNet18 already learned from ImageNet. This method prevents overfitting by limiting the model's ability to learn the training-specific noise.

Question 3: Did you reach a point where it was clear that you should stop training?

Yes, I did. There are some signals that were clear to remind me to stop training:

- The validation loss dropped to the lowest point (0.5733) at epoch 23, after which it was consistently going up while training loss was coming down.
- The highest point of validation accuracy was 81.21% at epoch 23
- The difference between training and validation increased from 8.0% (at epoch 23) to 14.16% (at epoch 50)
- The loss ratio got worse from 1.9 to 6.5

Question 4: What image classes had the best-worst performance and for the worst performing classes, what is your recommended path for improvement?

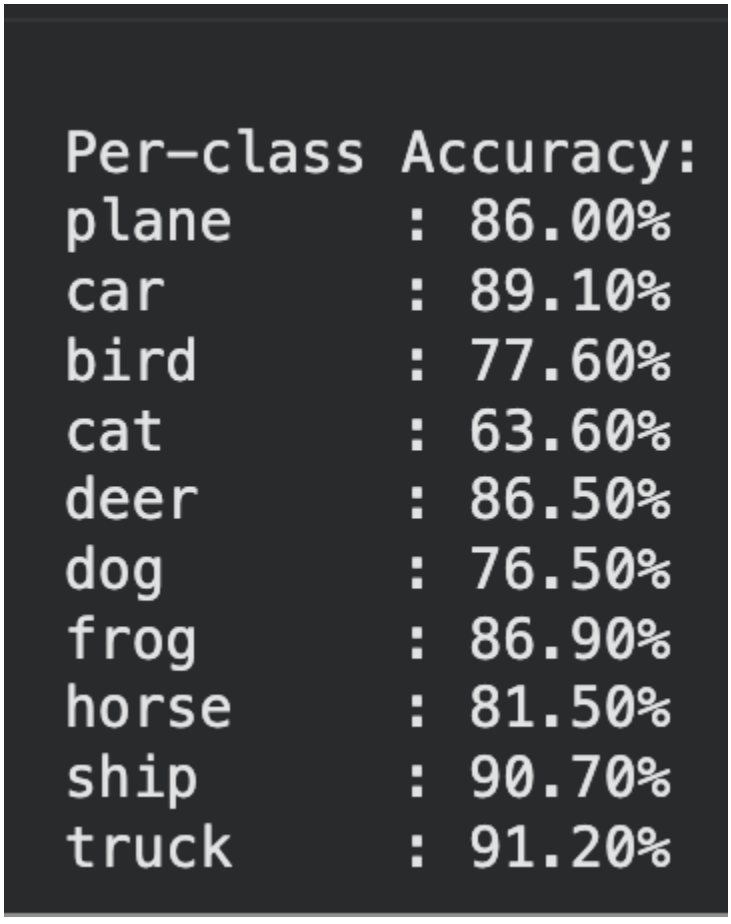

Figure 2. Per-class accuracy on test set

Best Performing Classes:

1. Truck: 91.20%
2. Ship: 90.70%
3. Car: 89.10%

Worst Performing Classes:

1. Cat: 63.60%
2. Dog: 76.50%
3. Bird: 77.60%

Analysis:

Vehicles (average 90.33%) significantly outperformed animals and bird (average 72.57%) demonstrate a 17.76-point gap. Vehicles have unique solid forms that can be identified at low resolution, while cats and dogs have comparable fur textures, attitudes, and body structures that become indistinguishable at 32×32 pixels. Birds have a significant degree of posture diversity and may be confused for aircraft.

Recommended Improvements:

1. Apply strong transformations like rotation (+30, -30), scale variations, color jittering, and occlusion to difficult classes. This not only increases the diversity of the training set but also makes the model learn from partial views, making it more effective in dealing with class imbalance (Howard & Gugger, 2020).
2. Implement two-stage classification: (1) broad categories (animals/vehicles), (2) specific classes. This leverages strong high-level distinction performance before fine-grained classification.
3. Penalize cat/dog/bird misclassifications more heavily, directing the learning capacity towards the difficult examples, and thus, perhaps cat accuracy can be improved from 63.60% up to 70-75%.

Conclusion

This experiment showed me how effective transfer learning can be, but also its limitations. The overfitting after epoch 23 was pretty dramatic. Validation loss jumped 24% while training loss kept improving. Looking back, stopping at epoch 23 would have been the smart move, saving 27 epochs of unnecessary training.

References

Howard, J., & Gugger, S. (2020). *Deep Learning for Coders with fastai and PyTorch*. O'Reilly Media.

Models and pre-trained weights — Torchvision 0.24 documentation. (n.d.). https://docs.pytorch.org/vision/stable/models.html

GeeksforGeeks. (2025, July 23). *Top PreTrained Models for Image Classification*. GeeksforGeeks. https://www.geeksforgeeks.org/computer-vision/top-pre-trained-models-for-image-classification/

GeeksforGeeks. (2025, July 23). *How to load CIFAR10 Dataset in Pytorch?* GeeksforGeeks. https://www.geeksforgeeks.org/python/how-to-load-cifar10-dataset-in-pytorch/