Deep Learning for CIFAR-10 Image Classification

Exploring Convolutional Neural Networks (CNNs) and Transfer Learning

September 8, 2024

Project Overview

CIFAR-10 Image Classification using CNN and Transfer Learning

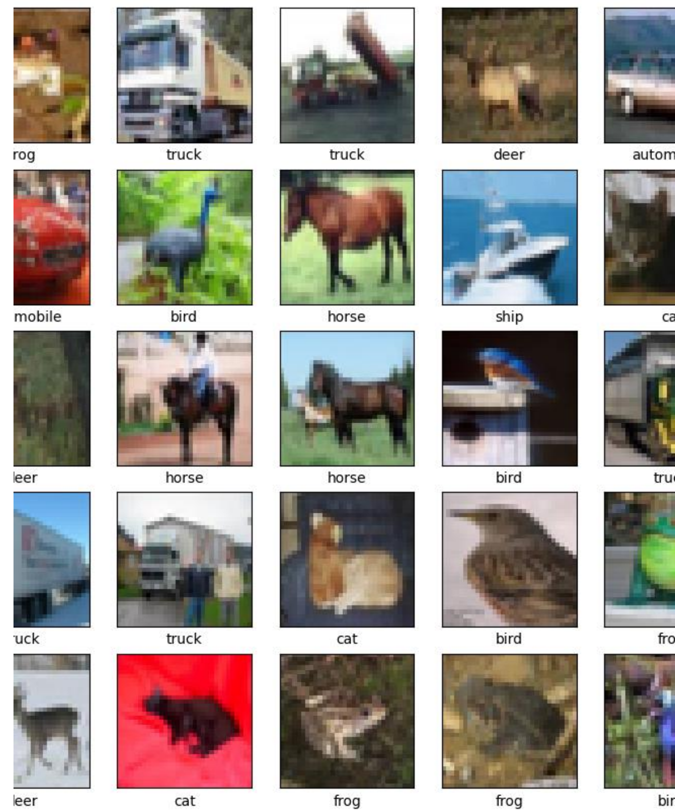
Objective: Classify images from CIFAR-10 into 10 categories using deep learning.

Dataset: 60,000 images (32x32) across 10 categories (airplane, car, bird, etc.).

Problem Statement

Challenge: Accurately classify low-resolution images with high variability.

Goal: Build a model that generalizes well, achieving high accuracy on unseen images.



ML Approach

CNN Architecture: Multiple convolutional layers with batch normalization and dropout.

Transfer Learning: Pre-trained models (ResNet18, Xception) to improve classification accuracy.

Techniques: Data normalization, Image augmentation (horizontal flipping), Hyperparameter tuning (Grid Search).

Exploratory Data Analysis (EDA)

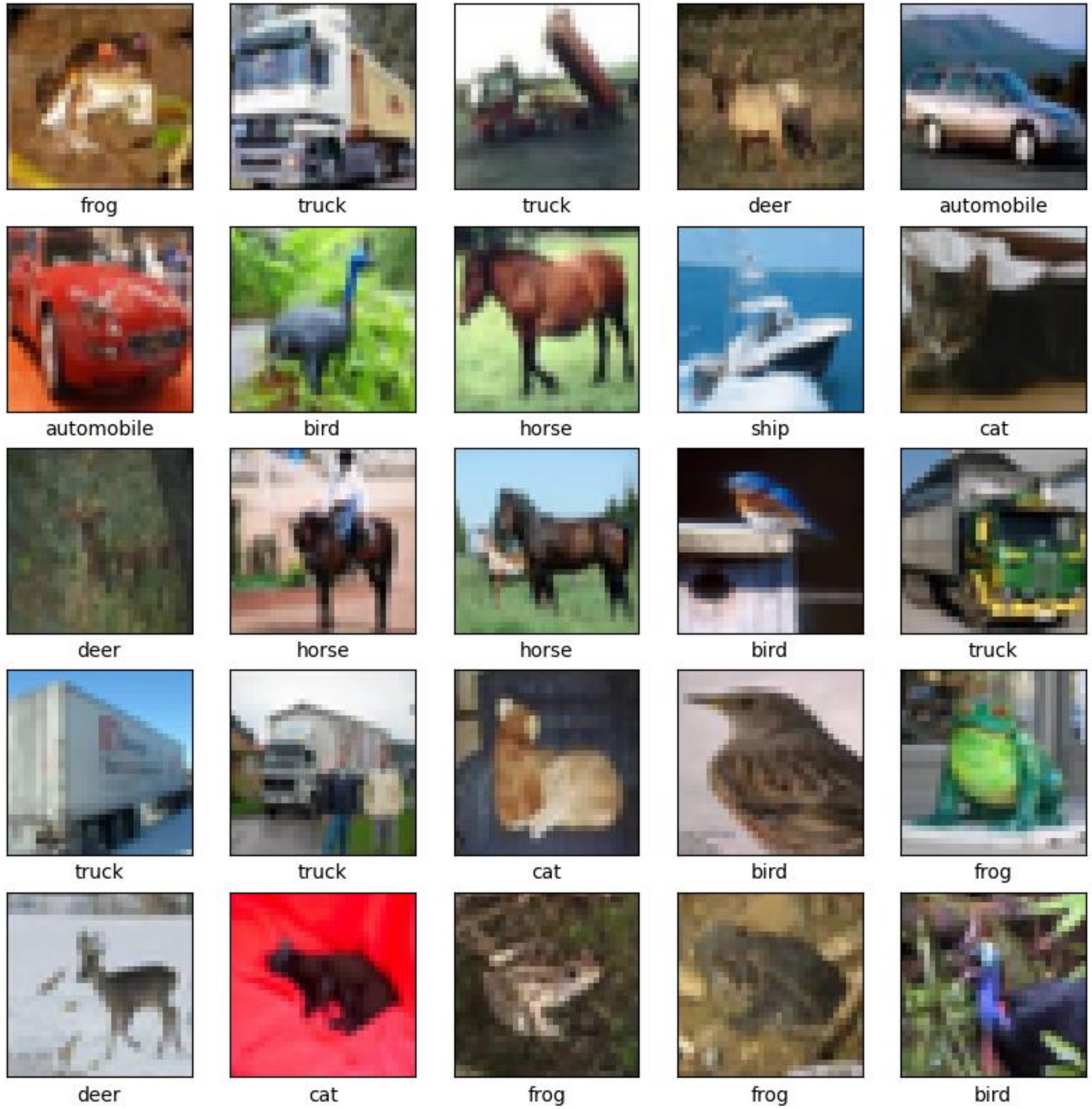
Class Distribution: Balanced across the 10 categories.

Image Characteristics: RGB channels, pixel intensity distributions.

Visualization: PCA and t-SNE for class separation, Edge detection using Canny.

Exploratory Data Analysis (EDA)

Example training images



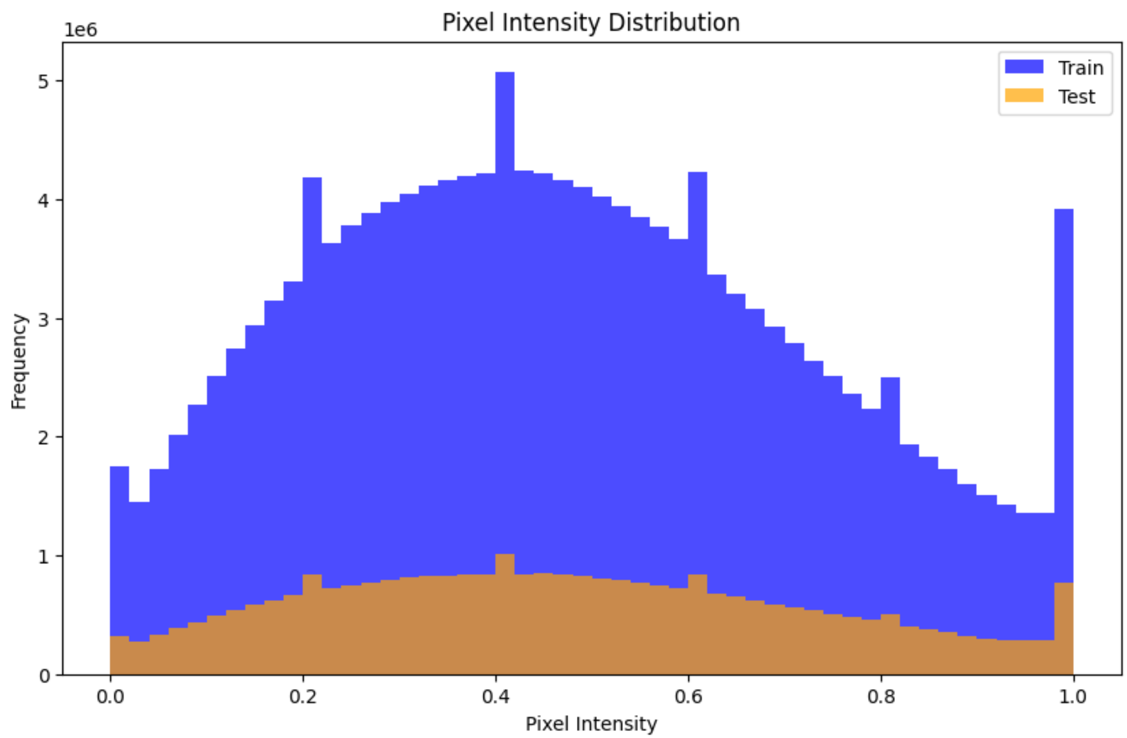
Exploratory Data Analysis (EDA)

Training data: 10 classes
Balanced at 5000 images each

	Class	Count
0	airplane	5000
1	automobile	5000
2	bird	5000
3	cat	5000
4	deer	5000
5	dog	5000
6	frog	5000
7	horse	5000
8	ship	5000
9	truck	5000

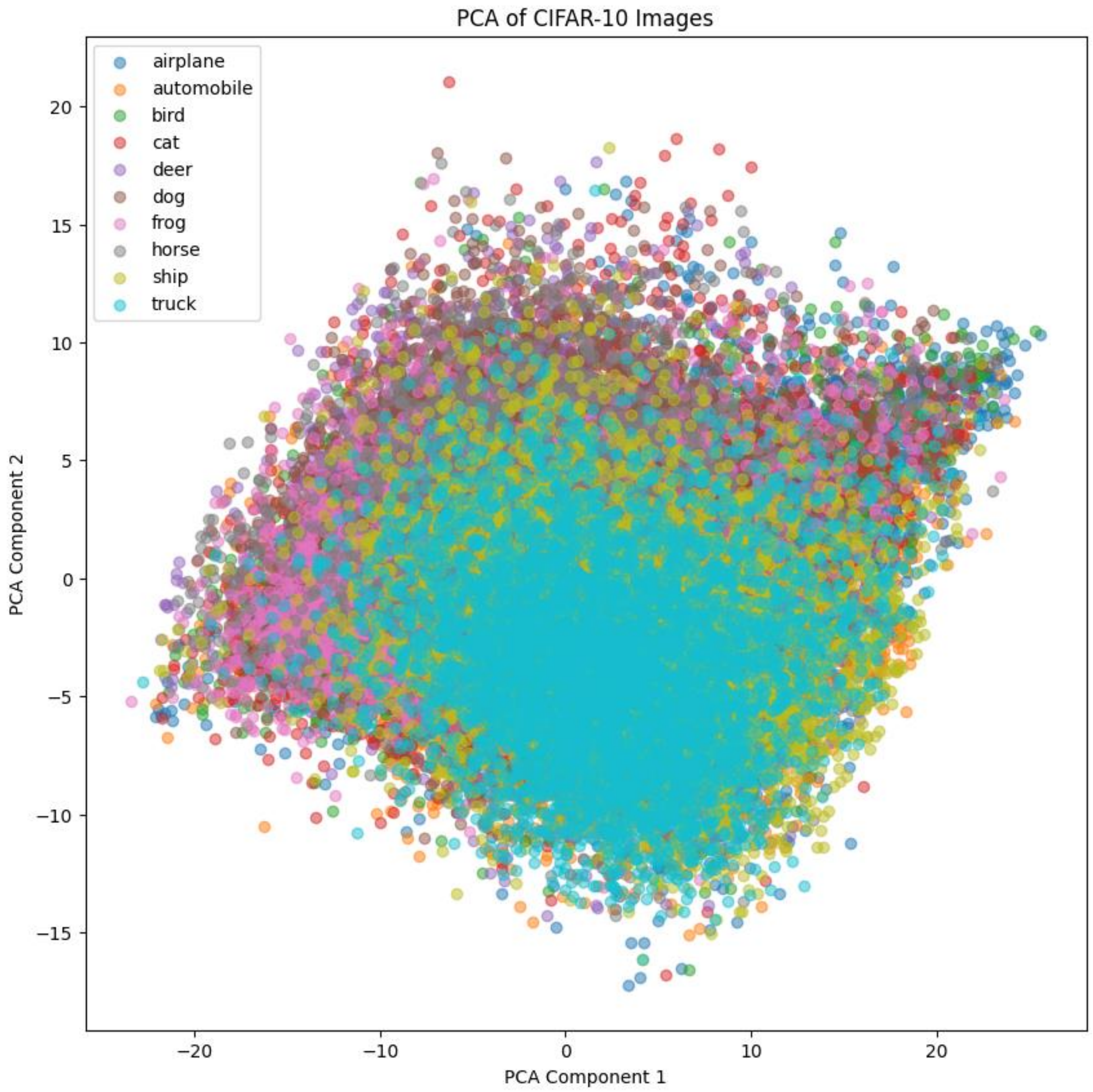
Exploratory Data Analysis (EDA)

Pixel Intensity Distribution



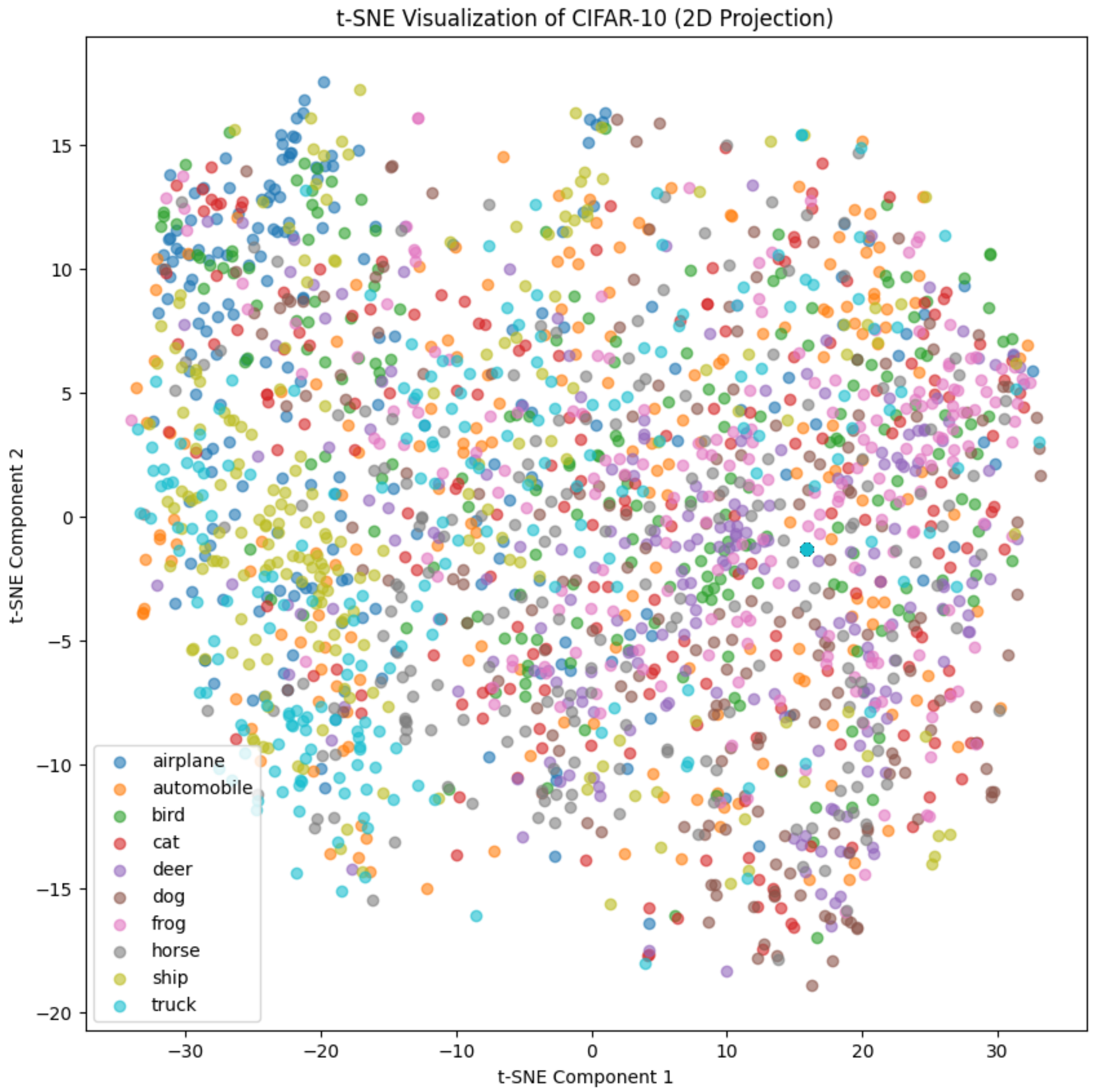
Exploratory Data Analysis (EDA)

PCA



Exploratory Data Analysis (EDA)

t-SNE



Model Architecture

CNN Model: Convolutional layers, Batch normalization, Dropout to prevent overfitting.

Transfer Learning: Fine-tuned Xception and ResNet18.

Training Results

CNN Performance: Best overall Test accuracy of 82.5%.

Xception Performance: After tuning, achieved ~80% test accuracy.

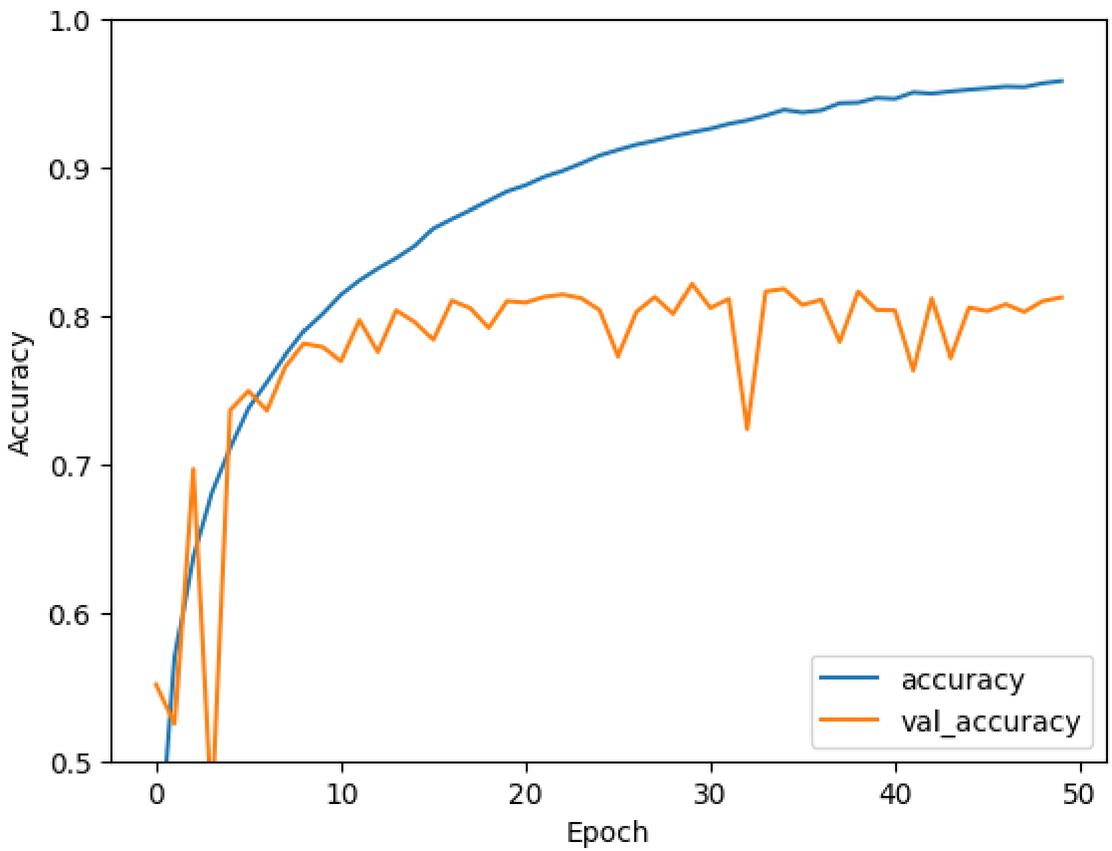
ResNet50: Lower performance (38.13%) on CIFAR-10.

Visualization of Model Performance

Accuracy & Loss Curves: Training and validation accuracy indicate overfitting.

CNN with 5 convolutional layers, 50 epochs

Training Results



CNN Model Architecture Summary

Input Layer:

- Shape: (32, 32, 3) RGB image

Convolutional Blocks (x5):

- Conv2D: Detects image patterns (32-256 filters)
- BatchNormalization: Stabilizes training
- MaxPooling (2x2): Reduces dimensions
- Dropout (25%): Prevents overfitting

GlobalAveragePooling2D:

- Converts feature maps to a 1D vector

Dense Layers:

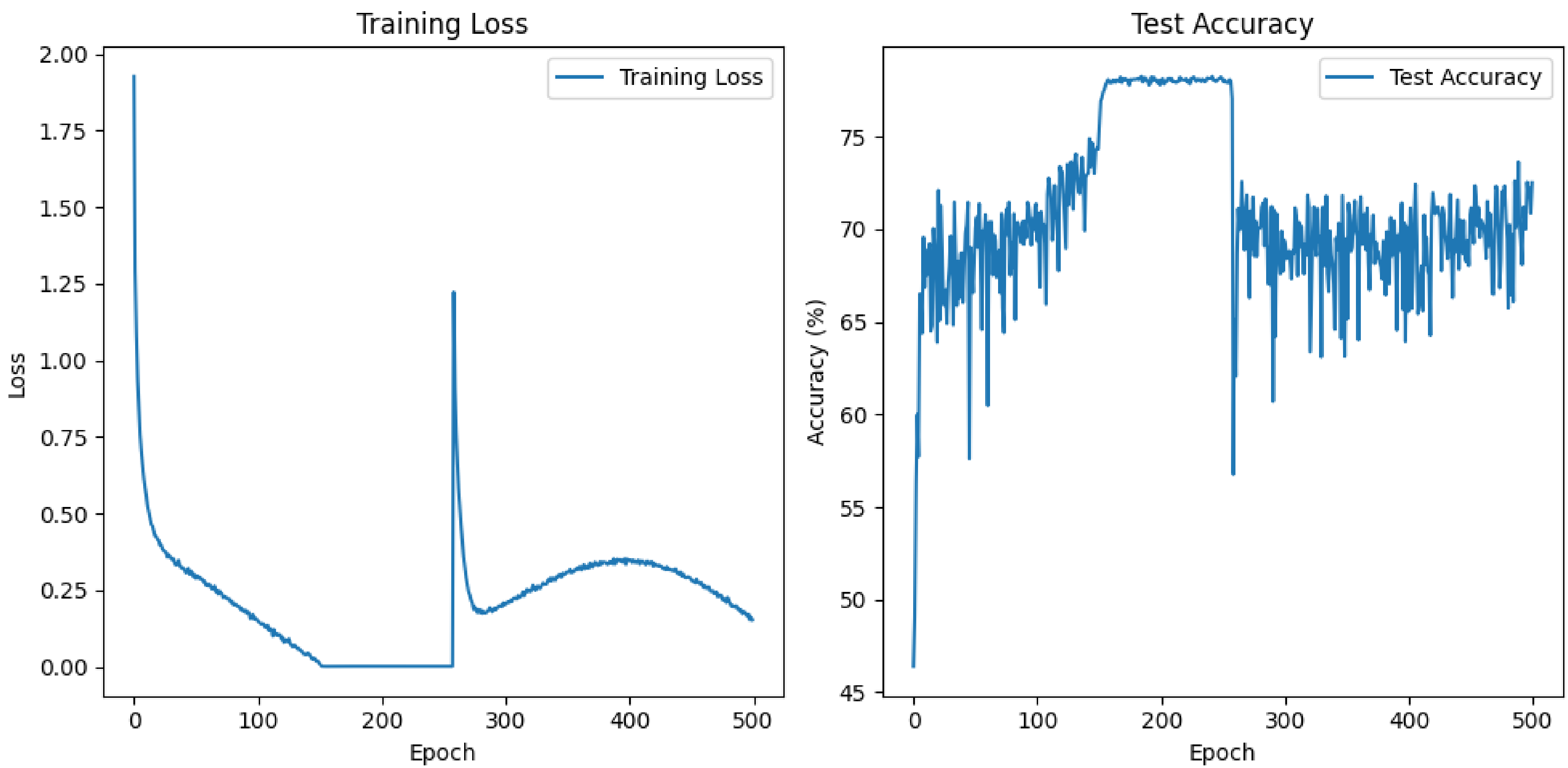
- Dense (64 units, ReLU): Learns complex patterns
- Dropout (50%): Regularization
- Dense (10 units): Output layer for classification

Compilation:

- Optimizer: Adam
- Loss: SparseCategoricalCrossentropy
- Metric: Accuracy

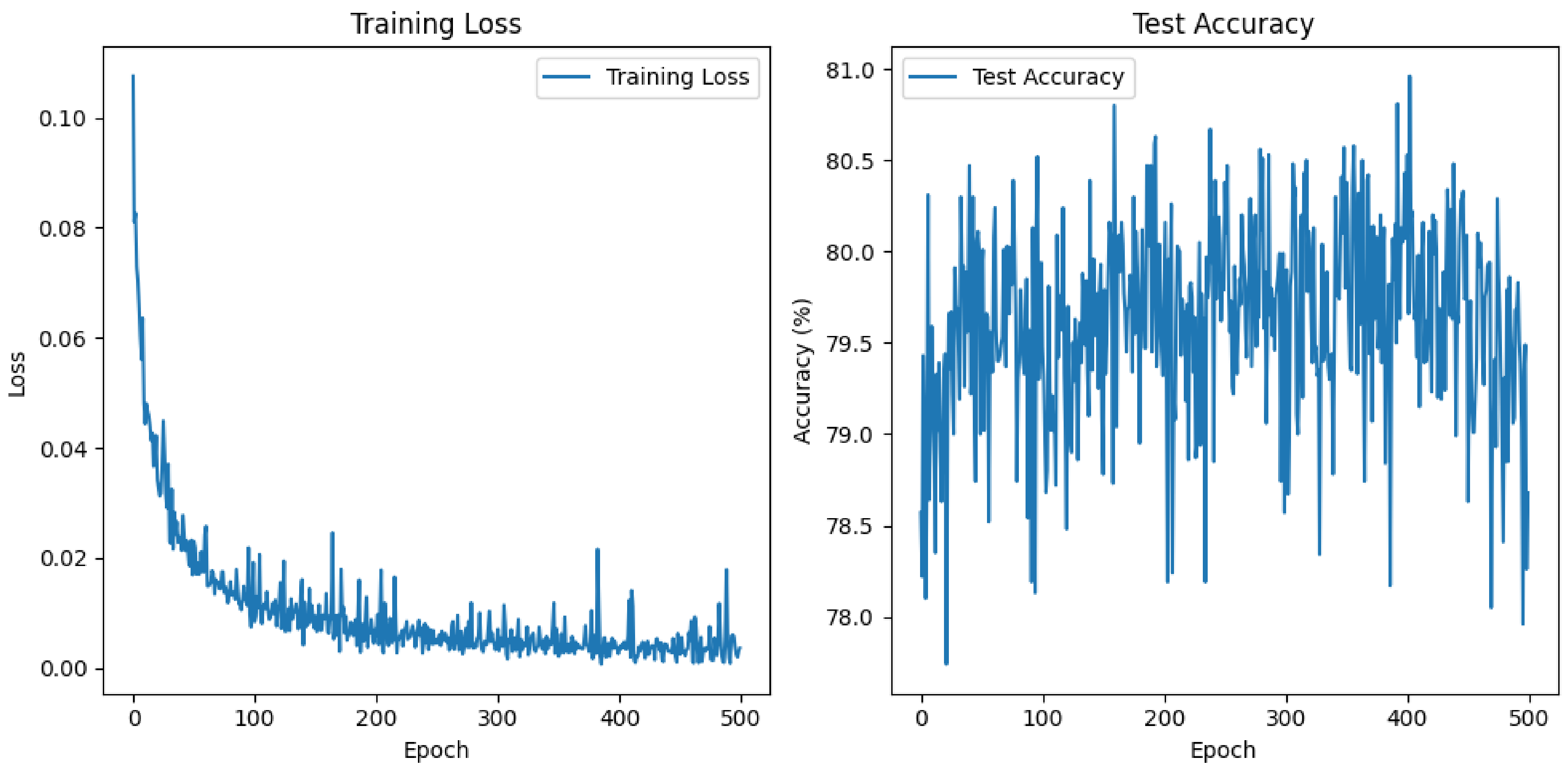
Training Results:

- Accuracy: Training vs Validation
- Test Accuracy: Final accuracy



Training Results

CNN with Random Horizontal Flip, 500 epochs



Training Results

Xception (pretrained) with best hyper parameters, 500 epochs

Optimization & Hyperparameter Tuning

Hyperparameters: Learning rate, batch size, optimizer choice.

Grid Search: Tested different combinations to maximize accuracy.

Optimization & Hyperparameter Tuning

Ran each of these 4 pretrained models:

- ResNet50 and ResNet18
- EfficientNetB0
- InceptionV3
- Xception

Optimization & Hyperparameter Tuning

Hyperparameter grid – best parameters found in **bold** below for pretrained Xception

lrs = [**0.001**, 0.01, 0.1]

momentums = [**0.8**, 0.9]

weight_decays = [**0**, 1e-4]

batch_sizes = [**64**, 128]

optimizers = ['SGD', **'Adam']**

Conclusion

Key Takeaways: CNNs are effective for image classification.

Transfer learning with Xception shows promising results.

Future Work: More data augmentation, regularization, deeper models.

Project Links:

- GitHub Link: https://github.com/LEBLAPI1/ImageClassifier-CNN-ResNet-Xception/
- Notebook Link (Google Colab): https://colab.research.google.com/drive/1muNErLFUuOIqb1wMnoUJ_zdv6eGhyTuW?usp=sharing
- Data set:

  https://www.tensorflow.org/datasets/catalog/cifar10