UNIVERSITY OF TRIESTE
Department of Engineering and Architecture

Master's Degree in Electronic and Computer Engineering

Project 3 (CNN classifier)

A.Y. 2024-2025

Author	Student Number
Stefano Chen	IN2000246

Abstract

This report explores the implementation of a Convolutional Neural Network (CNN) to classify images into 15 distinct categories, such as “office”, “kitchen”, and “mountain”. Using the provided dataset, this project demonstrates the process of training a shallow CNN from scratch and enhancing its performance through data augmentation, batch normalization, and transfer learning. The baseline model achieved a modest accuracy of 30% on the test set, while enhancements improved performance up to 60%. Leveraging transfer learning with AlexNet, the classification accuracy exceeded 85%, highlighting the effectiveness of pre-trained networks for small datasets.

1. Introduction

Image classification has been a cornerstone problem in computer vision, with applications ranging from autonomous vehicles to content organization. This project focuses on using CNNs, a class of deep learning models, to classify grayscale images into one of 15 categories. Challenges addressed include the limited size of the training dataset, the risk of overfitting, and ensuring robust generalization. By combining a shallow CNN architecture with advanced techniques like data augmentation and transfer learning, this work aims to balance simplicity and performance. The source code for this project can be found here.

2. Methodology

2.1 Network Architecture

The primary CNN architecture (baseline) implemented for this study follows a shallow design with three convolutional layers. This baseline architecture is summarized in the following table.

Table 1: Layout of the CNN to be used in Project 3

#	type	size
1	Image Input	64×64×1 images
2	Convolution	8 3×3 convolutions with stride 1
3	ReLU	
4	Max Pooling	2×2 max pooling with stride 2
5	Convolution	16 3×3 convolutions with stride 1
6	ReLU	
7	Max Pooling	2×2 max pooling with stride 2
8	Convolution	32 3×3 convolutions with stride 1
9	ReLU	
10	Fully Connected	15
11	Softmax	softmax
12	Classification Output	crossentropyex

In addition to the baseline, the following advanced models and technique were explored to enhance performance:

1. **Batch Normalization Model**: Batch normalization layers were introduced before each ReLU activation to reduce internal covariate shift, leading to faster convergence and improved stability during training.

2. **Dropout Regularization Model**: Dropout layers were added with a 0.5 probability before the fully connected layer to prevent overfitting by randomly deactivating neurons during training.

3. **Kernel Variations**: The convolutional filters were adjusted to use larger kernel size (3x3, 5x5 and 7x7) in deeper layers to capture more complex spatial relationships.

4. **Adam Optimizer**: The baseline was also trained using the Adam optimizer which is known for leading to faster convergence.

5. **Ensemble of Networks**: An ensemble approach was employed, when five independently trained networks were combined by averaging their output. This ensemble strategy is used to mitigate individual network biases and improve classification accuracy.

6. **Transfer Learning with AlexNet**: Two distinct approaches were used to leverage AlexNet:
    a. **Fine-tuning**: The pre-trained AlexNet model was fine-turned by freezing all layers except the last fully connected one. And trained using the provided dataset.
    b. **Feature Extraction + SVM**: Intermediate layer activations of AlexNet were used for extracting features, those features were then used to train a multiclass SVM.

Note:

Since we are using PyTorch for this project, the nn.CrossEntropyLoss class already incorporates a softmax operation. Therefore the Softmax layer was not implemented.

2.2 Data Preprocessing

The provided dataset (from Lazebnik et al., 2006), contains images divided in 15 categories (office, kitchen, living room, bedroom, store, industrial, tall building, inside city, street, highway, coast, open country, mountain, forest and suburb), and was already splitted in training set (1500 images) and test set (2985 images).
Before training our neural network we need to prepare our dataset.
The following preprocessing steps were performed:

1. **Image Resize**: since the dataset's images are of different sizes (267x200, 220x286, etc.) , every image were resized to 64x64 using an anisotropic rescaling (rescaling independently along x and y axis)

2. **Normalization**: the input values we scaled to the range [0,255] to improve the model training efficiency

3. **Data Splitting**: the training dataset was divided into 85% training (1275 images) and 15% validation (225 images) subsets

4. **Data Augmentation**: Horizontal flipping was applied to the training images to improve generalization.

2.3 Training Configuration

We employed the "stochastic gradient descent with momentum" as optimization algorithm, the learning rate is set to 0.001 and the momentum at 0.75 (both values were determined experimentally through multiple training).
For all models, the Cross Entropy Loss is used as loss function.
The minibatches size is set to 32 and the number of epochs is 30 (with early stopping based on validation loss).
For each network layer, the initial weights are drawn from a Gaussian distribution having mean of 0 and a standard deviation of 0.01, while the initial biases are set to 0.

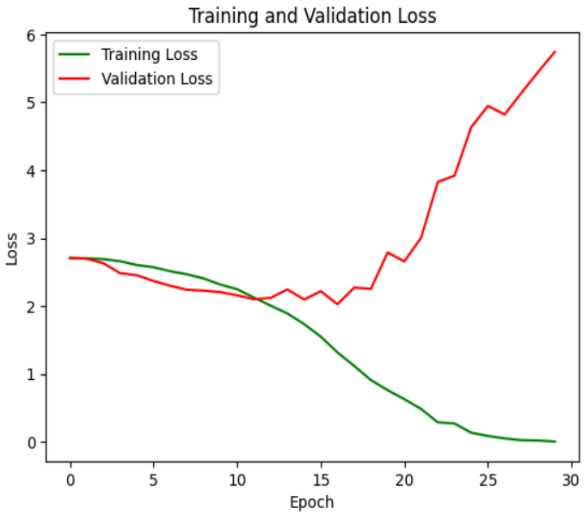
3. Results & Discussion

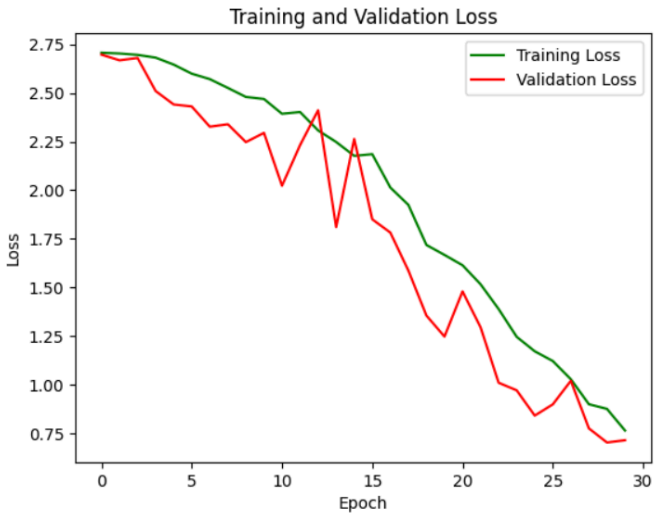
The training process consisted of two phases. First, Each network was trained using the original dataset to establish a baseline. Then, it was retrained using the augmented dataset, allowing for a comparative evaluation of the benefits introduced by data augmentation in improving generalization and robustness.

The stopping criteria used for each model is based on early stopping technique, the training is terminated when the performance on the validation set starts to degrade, indicating that the model is beginning to overfit the training data. In our case we only save the model that archives the lowest validation loss

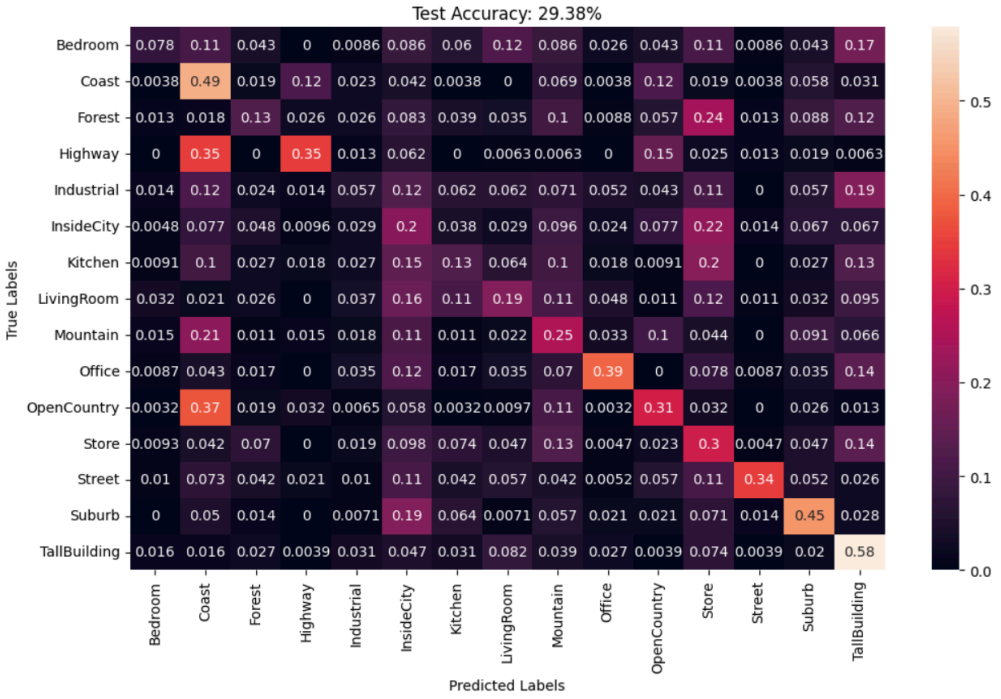
3.1 Baseline Model

The baseline model achieved limited success, with signs of overfitting emerging after approximately 12 epochs (left figure). When trained using the augmented data (right figure), the validation loss was more stable and exhibited less overfitting.



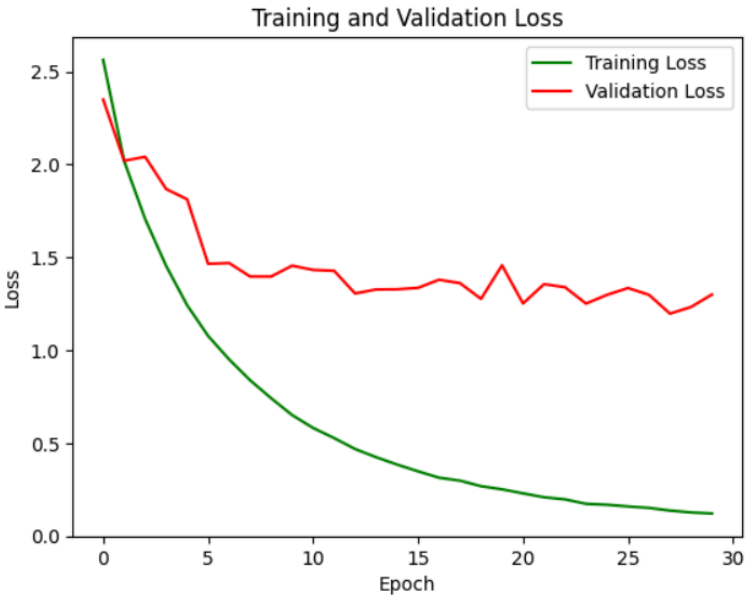


The test accuracy for the baseline model trained without the augmented data is just little less than 30%. Using the augmented data the accuracy goes to 38%.

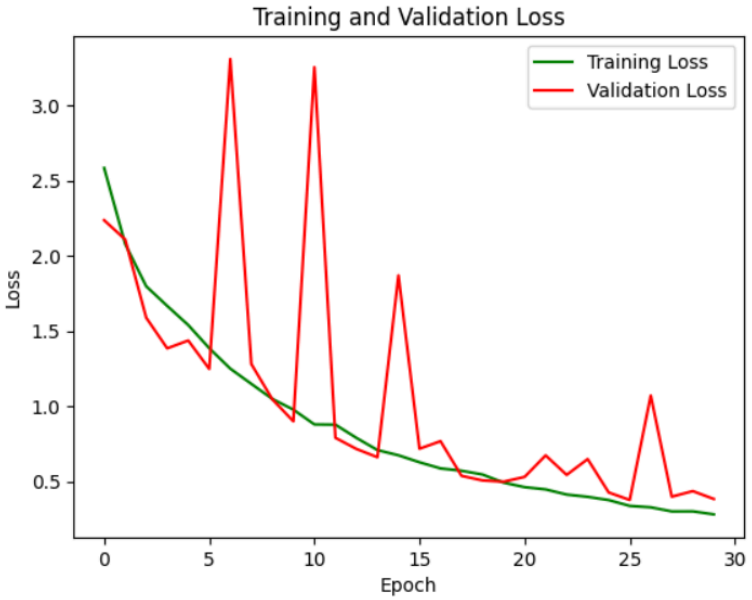


3.2 Batch Normalization Model

The incorporation of batch normalization layers led to an overall enhancement in performance, with the training process converging rapidly. However, the validation loss did not exhibit a corresponding improvement (left figure). When utilizing augmented data, both the training loss and validation loss demonstrated similar trends, although the validation loss exhibited occasional spikes (right figure).



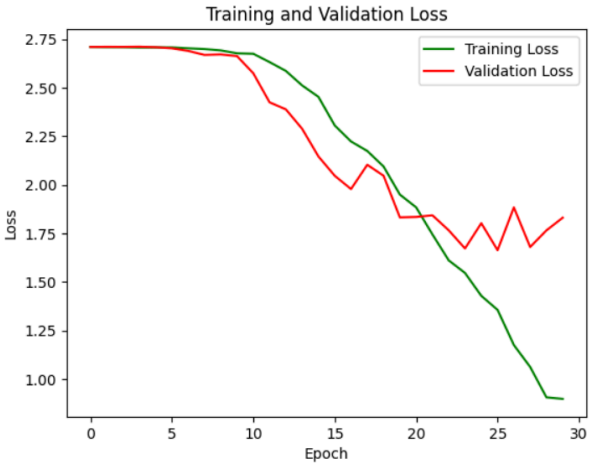
Test Accuracy: 51.26%



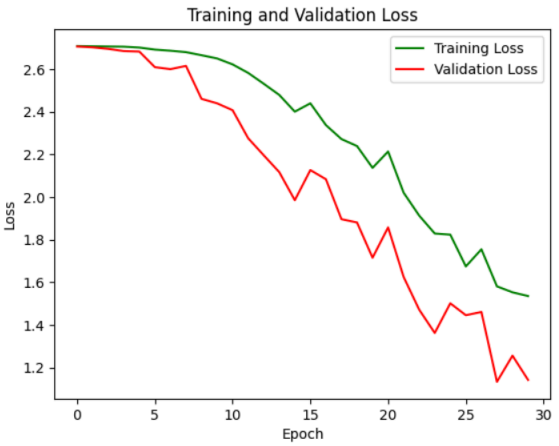
Test Accuracy: 58.46%

3.3 Dropout Model

The dropout layer was added with a probability of 0.5. This modification resulted in improved resistance to overfitting, as evidenced by a more stable validation loss and a modest increase in test accuracy.
Given the minimal difference in test accuracy between training with and without augmented data, it can be inferred that the dropout model demonstrate greater efficacy in scenarios where the training data is limited.



Test Accuracy: 36.25%

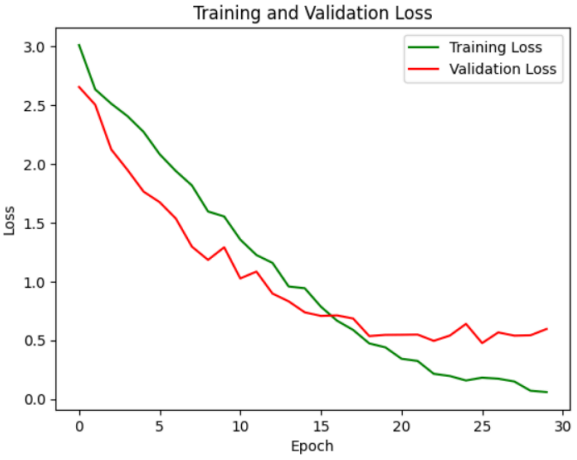


Test Accuracy: 37.52%

3.4 Kernel Variation Model

In this model, the convolutional filters were adjusted to utilize larger kernel sizes (3x3, 5x5, and 7x7) in the deeper layers to capture more complex spatial relationships. However, this modification did not result in a significant improvement in the overall performance of the network, as evidenced by the plots below, which closely resemble those of the baseline model. This lack of improvement may be attributed to the possibility that the selected kernel sizes were not optimal or that the network architecture was too simplistic to effectively capture the most salient features.

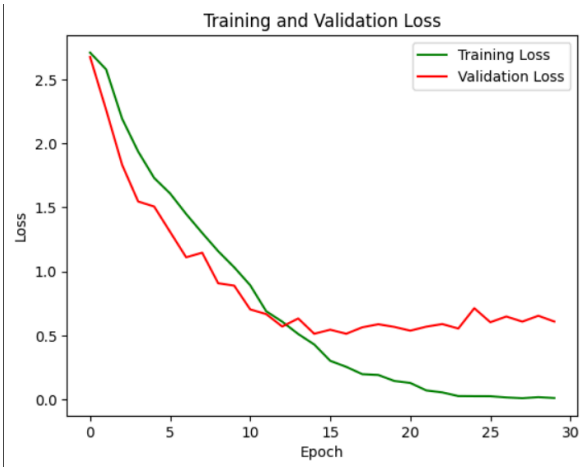
Test Accuracy: 31.73%



Test Accuracy: 36.48%

3.5 Adam Optimizer

This time, we employed the Adam optimizer to train the baseline model. The Adam optimizer facilitated faster convergence during training. However, the model exhibited rapid overfitting, likely due to the limited size of the training dataset.

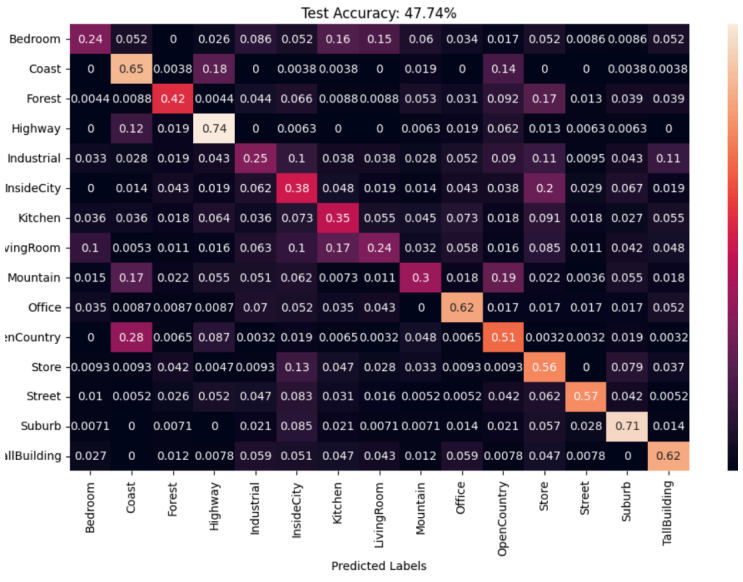


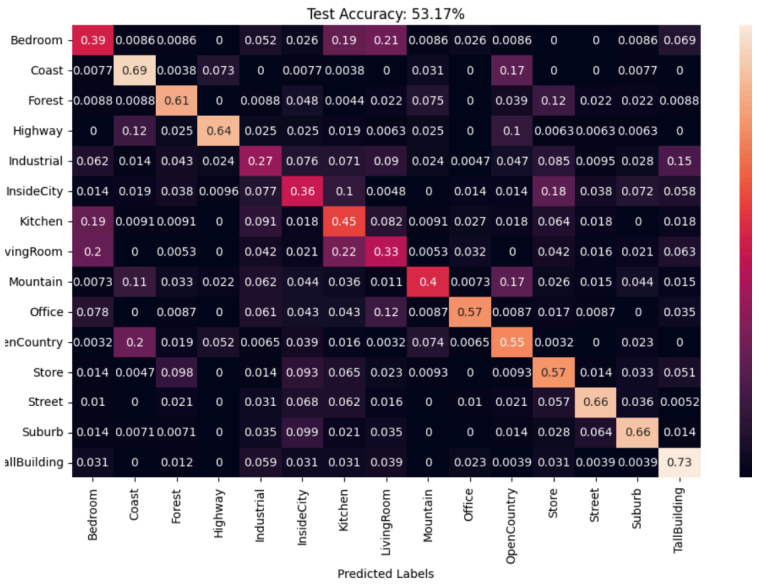
Test Accuracy: 32.09%

Test Accuracy: 42.35%

3.6 Ensemble of Networks

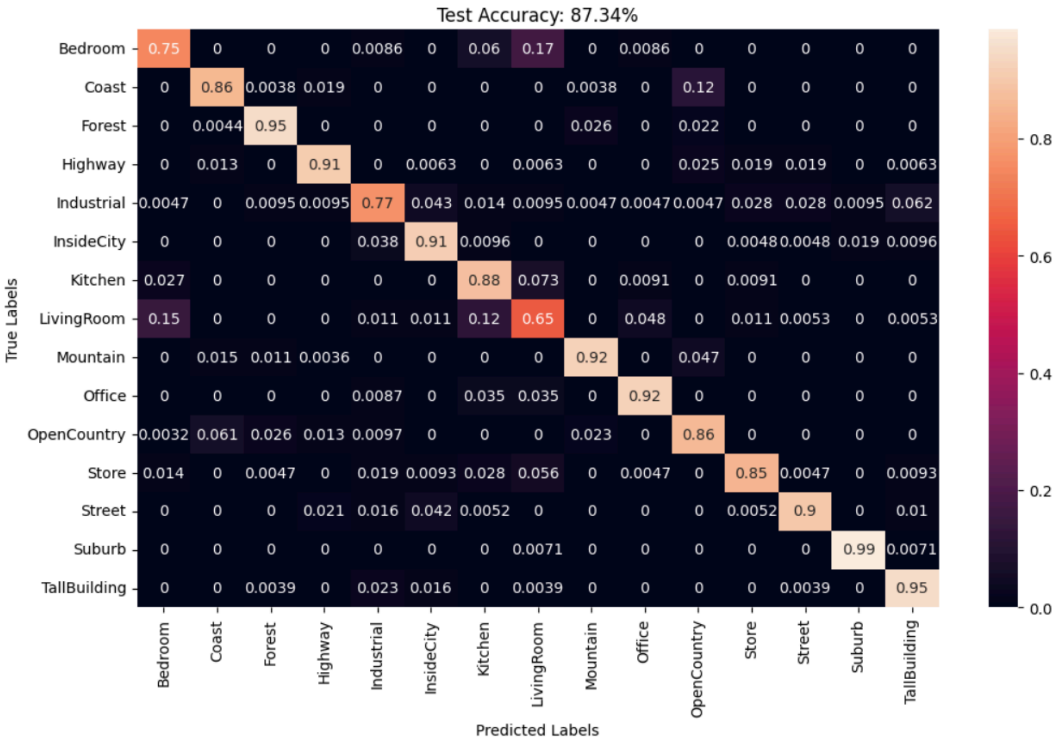
An ensemble approach was employed, when five independently trained networks were combined by averaging their output.
The five network are the models presented above
This ensemble strategy is used to mitigate individual network biases and improve classification accuracy. Notably, the version trained on the augmented data performed better, highlighting the importance of having a large and diversified dataset.





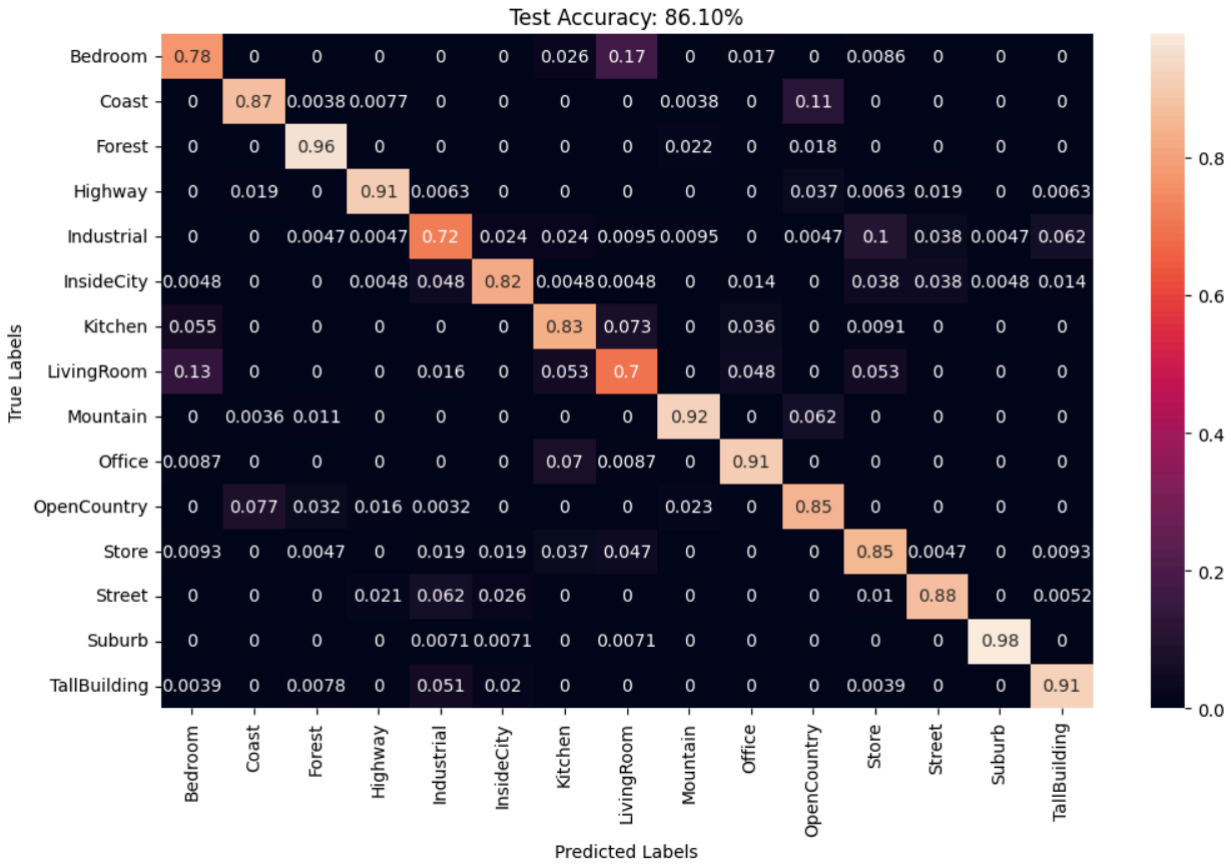
3.7 Fine-Tuned AlexNet

The pre-trained AlexNet model was fine-tuned by freezing all layers except the final fully connected layer. In PyTorch, the pre-trained AlexNet requires input images of size 224x224. Consequently, prior to training, it is necessary to resize the dataset to meet this requirement. During training the training loss converges rapidly, while the validation loss remains nearly constant.



3.8 AlexNet as Feature Extractor

Intermediate layer activations of AlexNet were utilized for feature extraction, and these features were subsequently used to train a linear multiclass Support Vector Machine (SVM). For each input image, 4096 features were extracted and employed to train the SVM. This strategy resulted in an SVM with a test accuracy of 86.10%



4. Conclusion

This project showcased the implementation and enhancement of CNN architectures for the image classification task involving 15 categories. Starting with a shallow baseline model, we explored various improvements, including data augmentation, batch normalization, dropout layers, kernel size variations, and the application of ensemble strategies. While the baseline model achieved limited accuracy, progressive modifications and transfer learning approaches substantially elevated performance.

These findings highlight the importance of robust preprocessing, effective training strategies, and leveraging transfer learning when working with small datasets. The successful application of advanced techniques like batch normalization and ensemble learning suggests their pivotal role in stabilizing training and improving generalization.

This report was written with the assistance of ChatGPT-4o to enhance the clarity and conciseness of the text.