Image Classification Project Report

Matteo Munini

September 2025

1 Introduction

Image classification is one of the most fundamental tasks in computer vision — it allows machines to recognize and categorize what they see. In this project, it is explored this exciting area by building an image classifier using convolutional neural networks (CNNs), a type of deep learning model designed to process visual information effectively.

The dataset used in this project, introduced by Lazebnik et al. (2006), contains 15 distinct scene categories, ranging from natural environments such as forests, coasts, and mountains to man-made settings like kitchens, offices, and city streets. It has been done an encoding in the following way:

Table 1: Mapping of image categories to labels.

Class Name	**Class Index**
Office	0
Kitchen	1
Living Room	2
Bedroom	3
Store	4
Industrial	5
Tall Building	6
Inside City	7
Street	8
Highway	9
Coast	10
Open Country	11
Mountain	12
Forest	13
Suburb	14

The images are already organized into training and test sets, making it possible to focus on designing, training, and evaluating our model.

These are some of the images in the dataset:

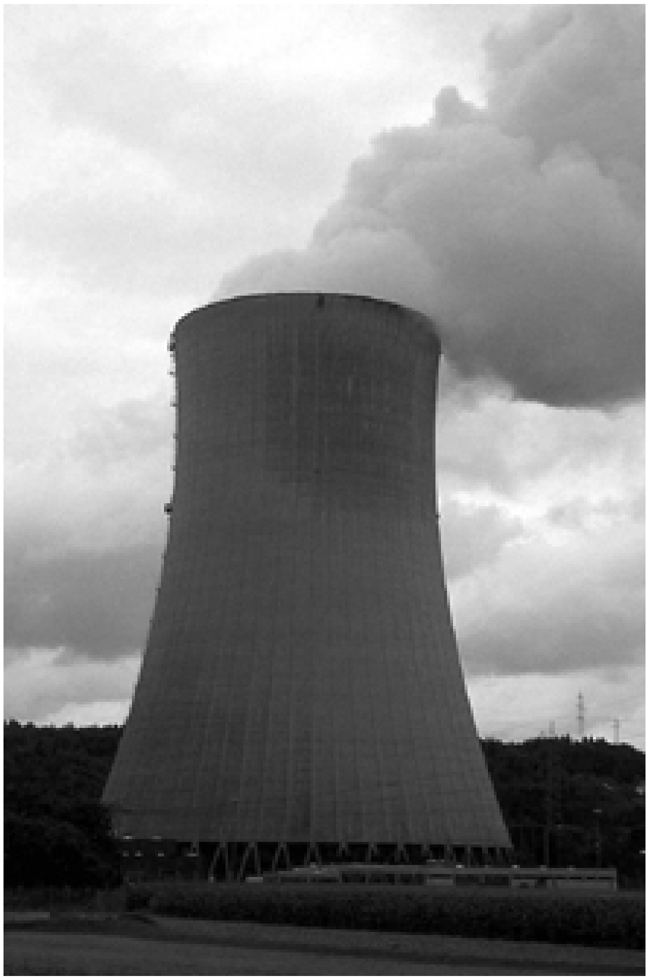


Figure 1: Example of the Industrial building

Figure 2: Example of the Tall Building

The goal of this project is to understand how changes in the architecture, in a low-epochs training regime, could affect the performance of the model.

2 First Experiment

In this first experiment, a shallow convolutional neural network (CNN) has been defined and trained from scratch, following the architecture specified in Table 2. Since the network expects input images of size 64×64, all images in the dataset must first be resized accordingly. A simple anisotropic rescaling strategy is employed, in which each image is independently rescaled along both the x and y-axes to match the target resolution. Image pixels are not normalized, so each element will belong to the range 0 to 255 (as required). This is due to the following definition of the network layes: the network parameters are initialized such that all bias terms are set to zero, and all weights are sampled from a $N(\mu = 0; \sigma = 0.1)$. Training is carried out using mini-batches of size 32.

The provided training set is partitioned into two subsets: 85% is used for actual model training, while the remaining 15% serves as a validation set to monitor model performance and tune hyperparameters. Model optimization is performed using stochastic gradient descent with momentum (SGD).

Since, it is asked to train the model for a few dozens of epochs, I choose for every experiment to train for 40 epochs. To accomplish the task in few epochs, I decided to use a cosine annealing scheduler for the learning rate in SGD (from

0.01 to 0.005) and also, an early stopping method is employed to monitor the progress in the validation. The patience parameter is set to 10 epochs.

Table 2: Network architecture used for image classification.

#	**Type**	**Size / Details**
1	Image Input	$64 \times 64 \times 1$ images
2	Convolution	8 filters of 3×3, stride $= 1$
3	ReLU	Activation function
4	Max Pooling	2×2 with stride $= 2$
5	Convolution	16 filters of 3×3, stride $= 1$
6	ReLU	Activation function
7	Max Pooling	2×2 with stride $= 2$
8	Convolution	32 filters of 3×3, stride $= 1$
9	ReLU	Activation function
10	Fully Connected	15 neurons
11	Softmax	Produces class probabilities
12	Classification Output	Cross-entropy loss

In the following plot it will be analyzed the behavior in the training, by comparing the loss and accuracy in both training and validation:

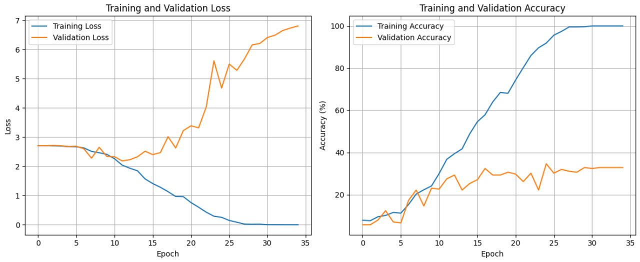


Figure 3: Training and Validation Loss and Accuracy

- Complete Overfitting: Training accuracy approaches 100%, while validation accuracy remains around 30%, indicating that the model memorized the training set without learning generalizable features.
- Validation Collapse: After approximately 20 epochs, validation loss increases sharply, signaling severe overfitting and inadequate regularization, even if Early Stopping acts as implicit regularization.
- Learning Behavior: Validation accuracy never exceeds 35%, suggesting that the current network architecture or hyperparameter configuration prevents effective generalization.

Predictions are highly scattered, with only a weak diagonal pattern in the confusion matrix, indicating poor classification and substantial misclassification across categories. Even the best-performing classes (6, 10, 11) show considerable error rates, resulting in an overall accuracy of roughly 34.47%, only marginally above random chance.

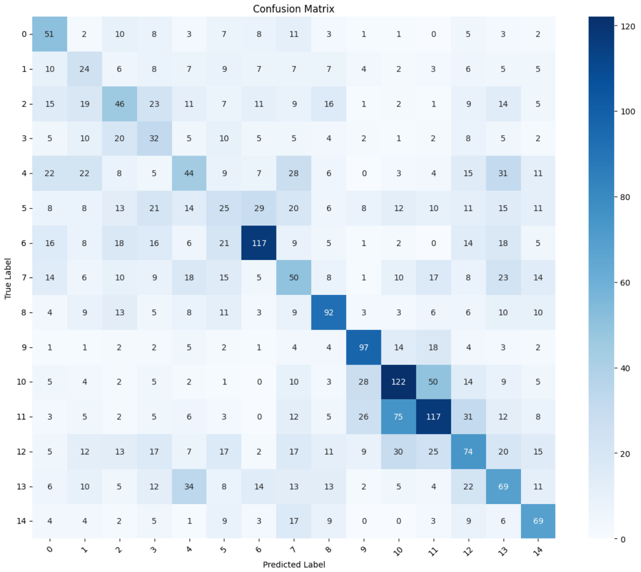


Figure 4: Confusion matrix for test prediction

3 Second Experiment

In this second experiment, some changes have been applied both on training dataset and the architecture of the NN. The new architecture is reported on Table 3.

Table 3: Simplified layer summary of ImprovedCNN architecture.

Layer #	**Layer Type**	**Operation Details**
Input	Input Layer	RGB image input
1	Conv2d	3→16 channels, 3×3 kernel, stride=1, padding=1
2	BatchNorm2d	Normalize 16 channels (,)
3	ReLU	Activation function
4	MaxPool2d	2×2 kernel, stride=2
5	Conv2d	16→32 channels, 5×5 kernel, stride=1, padding=2
6	BatchNorm2d	Normalize 32 channels
7	ReLU	Activation function
8	MaxPool2d	2×2 kernel, stride=2
9	Conv2d	32→64 channels, 7×7 kernel, stride=1, padding=3
10	BatchNorm2d	Normalize 64 channels
11	ReLU	Activation function
12	MaxPool2d	2×2 kernel, stride=2
13	Flatten	Reshape to 1D vector
14	Linear (FC1)	Fully connected layer, 4,096 → 128
15	ReLU	Activation function
16	Dropout	50% dropout rate
17	Linear (FC2)	Output layer for 15 classes
Output	Softmax*	Class probabilities (applied in loss function)

Then, from the image pre-processing side, they are still resized to 64×64 pixels to match the network input. A random horizontal flip with 50% probability is applied as a simple and cheap (since generated at the moment) form of data augmentation to increase model robustness to left-right variations.

Here in the learning rate scheduler , the learning rate could go from 0.005 to 0.0005.

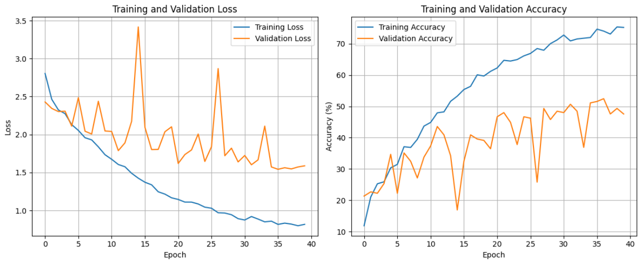
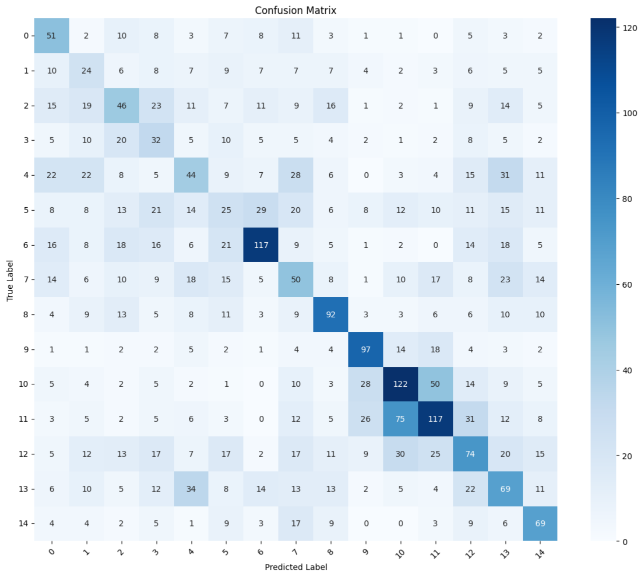


Figure 5: Training and Validation Loss and Accuracy

The training exhibits instability, with validation loss showing multiple spikes around epochs 15 and 25. The model demonstrates poor convergence, as validation accuracy plateaus near 50% and shows high variance throughout training. Clear signs of overfitting are present, indicated by a large gap between training accuracy (70%+) and validation accuracy (50%). The observed erratic patterns suggest that the learning rate may be too high or that regularization is insufficient.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	51	2	10	8	3	7	8	11	3	1	1	0	5	3	2
1	10	24	6	8	7	9	7	7	7	4	2	3	6	5	5
2	15	19	46	23	11	7	11	9	16	1	2	1	9	14	5
3	5	10	20	32	5	10	5	5	4	2	1	2	8	5	2
4	22	22	8	5	44	9	7	28	6	0	3	4	15	31	11
5	8	8	13	21	14	25	29	20	6	8	12	10	11	15	11
6	16	8	18	16	6	21	117	9	5	1	2	0	14	18	5
7	14	6	10	9	18	15	5	50	8	1	10	17	8	23	14
8	4	9	13	5	8	11	3	9	92	3	3	6	6	10	10
9	1	1	2	2	5	2	1	4	4	97	14	18	4	3	2
10	5	4	2	5	2	1	0	10	3	28	122	50	14	9	5
11	3	5	2	5	6	3	0	12	5	26	75	117	31	12	8
12	5	12	13	17	7	17	2	17	11	9	30	25	74	20	15
13	6	10	5	12	34	8	14	13	13	2	5	4	22	69	11
14	4	4	2	5	1	9	3	17	9	0	0	3	9	6	69

Figure 6: Confusion matrix for test prediction

Overall, the network's predictions are more scattered, with weak diagonal dominance. Several classes, 0, 1, 2, 3, 4, 5, 7, 12, 13, and 14, exhibit significant misclassification across multiple categories. In contrast, classes 6, 8, 9, 10, and 11, maintain relatively strong classification accuracy. The model appears to achieve moderate overall performance, with an estimated accuracy likely in the 58% range.

4 Third Experiment

We employ transfer learning using a pre-trained convolutional neural network, such as AlexNet [Krizhevsky et al., 2012], to leverage previously learned feature representations. Differntly from the previous experiemnts, here it is applied z-score normalization on the images using parameters defined in AlexNet paper. All layers of the network are frozen except for the last fully connected layer. The weights of this final layer are fine-tuned using the same training and validation sets as before. Here in the learning rate scheduler , the learning rate could go from 0.005 to 0.0005.

The corresponding training curves confirm successful convergence: both training and validation loss decrease smoothly, stabilizing around epochs 8–10. Training accuracy reaches approximately 95%, while validation accuracy stabilizes around 87–89%. The small but manageable gap between training and validation indicates good generalization, and the steep rise in accuracy during the first five epochs reflects the fast initial learning typical of transfer learning.

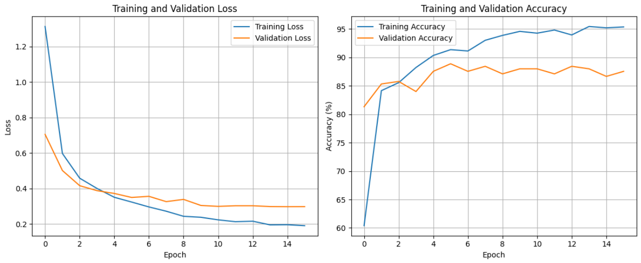


Figure 7: Confusion matrix for test prediction

The model demonstrates excellent overall performance, with strong diagonal values in the confusion matrix indicating high per-class accuracy. Almost all the classes exhibit very clean classification with minimal confusion, while class 3 shows some misclassification. The scarcity of off-diagonal entries suggests that this transfer learning approach achieved high test accuracy, likely exceeding 86%.

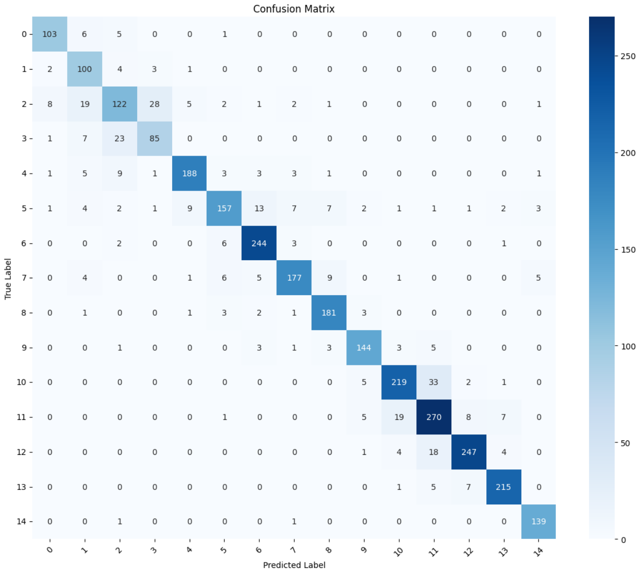


Figure 8: Confusion matrix for test prediction

5 Fourth Experiment

In this last approach, the pre-trained AlexNet is used as a feature extractor. The preprocessing is the same used in 3^{rd} experiment. Specifically, the activations of the last fully connected layer are extracted for each input image. These feature vectors serve as inputs to a multiclass linear Support Vector Machine (SVM), which is trained to perform the 15-category classification task. Directed Acyclic Graph (DAG) method is used for multi-class SVM. This method leverages the rich hierarchical representations learned by the pre-trained network while relying on a linear classifier to distinguish between classes. Once the features are extracted, new train, validation and test are defined and then z-score normalized. Also, it has been defined a DAG-SVM with Gaussian kernel.

The accuracy on the test for the linear is 86%, while the one with Gaussian kernel is 80%.

6 Conclusion

7 Conclusion

In this project, several approaches for scene image classification were explored, ranging from training a shallow CNN from scratch to employing transfer learning with a pre-trained AlexNet. The shallow network exhibited poor generalization, with significant overfitting and low validation accuracy around 30–35%. Introducing a deeper architecture with data augmentation improved performance moderately, achieving approximately 58% test accuracy, though some classes remained challenging. Transfer learning using AlexNet fine-tuned on the last fully connected layer substantially increased performance, with test accuracy exceeding 86%. Finally, using the pre-trained network as a feature extractor for a multiclass SVM also yielded high accuracy, demonstrating the effectiveness of leveraging pre-trained representations for this classification task.

8 Bibliography

S. Lazebnik, C. Schmid, and J. Ponce, *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories*, In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, In Advances in Neural Information Processing Systems (NeurIPS), 2012.

K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv preprint arXiv:1409.1556, 2014.

C. Cortes and V. Vapnik, *Support-Vector Networks*, Machine Learning, 20(3), 273–297, 1995.