Convolutional Neural Network-Based Classification of Museum Imagery

Bhrugu Kothari
40270224
bhrugu0510@gmail.com

Devanshu Kotadiya
40268999
devanshu.kotadiya@gmail.com

Jay Ashokkumar Patel
40293645
jayashokpatel29@gmail.com

I. Abstract

This project focuses on classifying museum-related images as either indoor or outdoor scenes using deep learning techniques. In Phase 1, traditional machine learning algorithms such as Decision Trees, Random Forests, and XGBoost were used on handcrafted features, achieving moderate accuracy. In Phase 2, a Convolutional Neural Network (CNN) was implemented from scratch using PyTorch to capture spatial hierarchies in image data. The Places dataset subset, containing balanced samples of museum indoor and outdoor images, was used after preprocessing steps like resizing, normalization, and RGB formatting. The CNN architecture was tuned through extensive experimentation with varying convolutional layers, learning rates, batch sizes, and pooling strategies. The best model, featuring four convolutional layers and pooling enabled, achieved a validation accuracy of 91.55% and high performance on other metrics such as F1-score and precision. This report presents the methodology, implementation, and findings in a structured and comparative manner.

II. Introduction

a) Problem Statement & Methodology

Classifying images based on their environment (indoor vs. outdoor) is a foundational task in computer vision, particularly in applications like autonomous navigation, content recommendation, and museum digitization. In the context of museum-related scenes, the problem is further complicated by overlapping visual features such as architecture, lighting, and thematic displays, which can appear in both indoor and outdoor settings.

In Phase 1 of this project, we applied traditional machine learning models including Decision Trees, Random Forests, and XGBoost using handcrafted features extracted from image metadata and PCA-transformed visual embeddings. While XGBoost performed best among the three with a validation accuracy around 88%, these models failed to capture spatial hierarchies within image data—an essential component of effective image classification.

To overcome this limitation, Phase 2 implements a deep learning solution using Convolutional Neural Networks (CNNs), built from scratch in PyTorch. Our CNN model automatically extracts spatial features through stacked convolutional layers, combined with pooling operations and dense layers for classification. Images were resized to 128x128, converted to RGB, and normalized before training. The dataset was divided using an 80/20 split to ensure effective training and unbiased validation.

We conducted an ablation study exploring multiple architectures and hyperparameters, including different learning rates, batch sizes, convolutional layers, and the use or exclusion of pooling. The model with 4 convolutional layers and pooling enabled achieved the highest validation accuracy (91.55%), significantly outperforming the classical models from Phase 1.

b) Related Work

Prior work has demonstrated the strength of CNNs in image classification, notably Krizhevsky et al.'s AlexNet on ImageNet and Zhou et al.'s work on the Places dataset, which introduced scene-centric image classification. While these models leveraged large-scale data and deep architectures, our work adapts these ideas to a smaller, domain-specific dataset focused on museums.

Additionally, studies comparing CNNs with traditional classifiers underscore the advantage of spatial feature learning, especially when dealing with visual similarity between classes. Our Phase 1 results aligned with this observation, prompting the transition to a CNN-driven approach in Phase 2 for superior performance.

III. Methodology

a) Dataset Description

The dataset used in this project is a subset of the MIT Places dataset, filtered for museum-related images. It consists of two balanced categories: museum-indoor and museum-outdoor, each with 5000 color images (RGB), making a total of 10,000 labeled images. The data is structured in folders by class and loaded using PyTorch's ImageFolder API, which automatically assigns labels based on directory names.

All images were preprocessed using the following pipeline:

- Resized to **128×128** pixels
- Converted to **RGB** format

- **Normalized** to [-1, 1] using mean = [0.5, 0.5, 0.5], std = [0.5, 0.5, 0.5]
- **Automated filtering** of unreadable or corrupted files

The dataset was split into:

- **80% training set**
- **20% validation set**

Cross-validation was not used due to compute limitations. Instead, random_split ensured fair and reproducible splitting. Images were loaded using PyTorch DataLoader with batch shuffling and prefetching to support GPU-based acceleration.

b) CNN Architecture

The Convolutional Neural Network (CNN) model was implemented from scratch using the PyTorch deep learning framework to enable full control over architectural components and facilitate detailed ablation studies. The architecture consists of a modular and configurable stack of **two to four convolutional layers**, allowing for systematic experimentation with network depth. Each convolutional block comprises a sequence of operations:

- A **Conv2D** layer
- A **ReLU** activation
- Optional **MaxPooling2D**

The final layer of the convolution stack is followed by:

- A **Flattening layer**
- A **Fully Connected (Linear) layer** with 128 neurons
- A **final output layer** with 2 neurons (for binary classification)

Each experiment varied the number of convolutional layers and pooling operations to evaluate the best configuration. The architecture was designed to balance depth and computational cost, enabling experimentation without requiring transfer learning or pretrained models.

This model was chosen due to its simplicity, interpretability, and the ability to conduct controlled ablation studies, which wouldn't be possible with black-box pretrained models.

c) Optimization Algorithm

To train the CNN model, the **Adam optimizer** was employed due to its efficiency and robustness in handling sparse gradients and non-stationary objectives. Adam, short for Adaptive Moment Estimation, combines the advantages of both AdaGrad and RMSProp by computing adaptive learning rates for each parameter. It maintains exponentially decaying averages of past gradients and squared gradients, allowing the optimizer to adjust the learning rate on a per-parameter basis. This property significantly accelerates convergence and enhances training stability, especially in deep neural networks with large parameter spaces.

The model's learning objective was guided by the **CrossEntropyLoss** function, which is well-suited for multi-class and binary classification tasks. It measures the dissimilarity between the predicted probability distribution and the true class labels, penalizing incorrect predictions more severely. Cross-entropy loss effectively drives the model toward higher confidence in correct class predictions, making it an ideal choice for the binary classification scenario addressed in this study.

IV. Results

a) Experiment Setup

To systematically evaluate the impact of architectural and training choices on model performance, a series of six experiments were conducted by varying key hyperparameters and structural components of the CNN. The following hyperparameters were tuned during experimentation:

• **Learning Rate:** 0.0005 and 0.001

• **Batch Size:** 32 and 64

• **Number of Convolutional Layers:** 2, 3, and 4

• **Pooling Strategy:** MaxPooling included or excluded

Each model was trained for **5 epochs** using a consistent dataset split and transformation pipeline. To ensure the integrity and fairness of comparisons, all experiments utilized the same data loader configuration and preprocessing steps. This consistency minimized external sources of variance and allowed direct attribution of performance differences to the altered hyperparameters and architectural choices.

Model evaluation was based on both **training and validation metrics**, including **Accuracy, Precision, Recall**, and **F1 Score**. Recording these metrics across all experiments enabled the identification of potential overfitting and informed iterative architectural refinements. By capturing results from models with differing levels of depth, learning rates, and pooling strategies, this setup provided a robust framework for analyzing trade-offs between model complexity, generalization, and performance.

b) Main Results

Model	Pooling	Conv Layers	LR	Batch Size	Train Acc	Val Acc	Precision	Recall	F1-Score
Baseline	Yes	3	0.001	32	0.9469	0.9150	0.9242	0.9038	0.9139
Less Conv	Yes	2	0.001	32	0.9599	0.9075	0.9169	0.8958	0.9062
Small LR	Yes	3	0.0005	32	0.9380	0.9140	0.9311	0.8938	0.9121
Big Batch	Yes	3	0.001	64	0.9328	0.9075	0.9212	0.8908	0.9058
More Conv	Yes	4	0.001	32	0.9299	0.9155	0.9481	0.8788	0.9121
No Pooling	No	3	0.001	32	0.9870	0.8830	0.9302	0.8277	0.8759

Table 1: Evaluation metrics for all hyperparameter combinations

c) Ablative Study

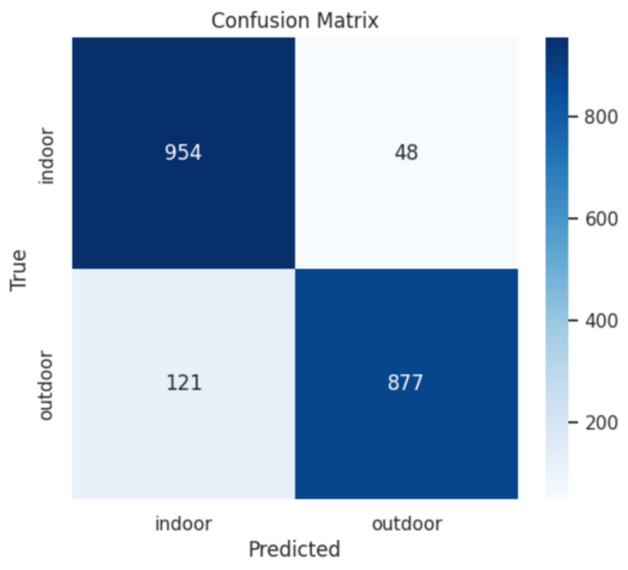
To evaluate the contribution of different architectural components and training hyperparameters, we conducted an ablation study using the baseline model configuration, which includes identity pooling, three convolutional layers, a learning rate of 0.001, and a batch size of 32. Each experiment altered a single component to isolate its effect on model performance.

Reducing the number of convolutional layers from three to two (Less Conv) resulted in a slight increase in training accuracy (0.9599 vs. 0.9469) but a decrease in validation accuracy (0.9075 vs. 0.9150). This suggests that the shallower network may have overfitted to the training data and lost some generalization capacity. Similarly, when we reduced the learning rate to 0.0005 (Small LR), the training accuracy dropped slightly (0.9380), but the validation accuracy (0.9140) remained nearly unchanged from the baseline. This indicates that although convergence was slower, the generalization capability was maintained.

Increasing the batch size to 64 (Big Batch) led to both lower training (0.9328) and validation accuracy (0.9075). This outcome implies that larger batch sizes may have led to less effective optimization due to reduced gradient noise. On the other hand, increasing the number of convolutional layers to four (More Conv) resulted in a slight drop in training accuracy (0.9299) but the highest validation accuracy (0.9155) and the best precision score (0.9481). This demonstrates that adding depth to the model can improve generalization if properly regularized.

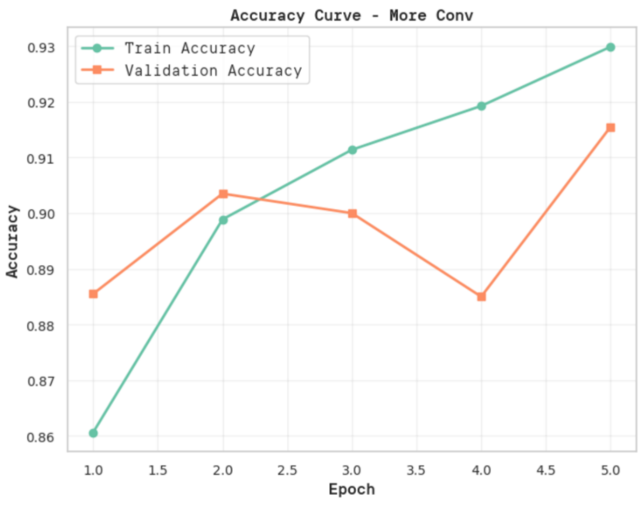
The most significant impact came from removing the identity pooling layer (No Pooling). Although this configuration achieved the highest training accuracy (0.9870), it suffered a sharp decline in validation accuracy (0.8830) and had the highest validation loss (0.4338), clearly indicating overfitting. This confirms the importance of pooling in controlling overfitting and improving model robustness by introducing translational invariance.

In conclusion, the ablation study shows that the baseline model achieves a good balance between performance and generalization. While a deeper network with four convolutional layers slightly improved validation performance, removing pooling severely harmed it. Smaller learning rates and larger batch sizes showed moderate influence, suggesting that the baseline settings are already close to optimal. Overall, the study highlights the critical role of pooling and model depth in achieving robust performance.

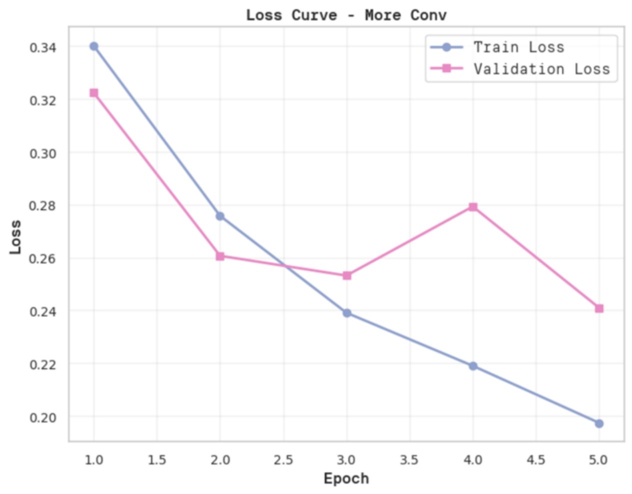


The confusion matrix illustrates the classification performance of the best CNN model on the validation dataset, specifically for the binary task of distinguishing between indoor and outdoor scenes. The model correctly classified 954 indoor images and 877 outdoor images, resulting in a total of 1,831 correct predictions out of 2,000 samples. This corresponds to an overall accuracy of 91.55%, consistent with the validation accuracy reported during experimentation.

From the matrix, it is evident that the model performs slightly better on indoor images, misclassifying only 48 of them as outdoor. In contrast, 121 outdoor images were incorrectly predicted as indoor, indicating a relatively higher false positive rate for the indoor class. This imbalance suggests that the model has a mild bias toward predicting scenes as indoor, which could be due to more distinctive indoor features being captured during training or an imbalanced feature representation between the two classes. Despite this, the model demonstrates strong generalization capability across both categories, with a high degree of precision and recall. The confusion matrix provides valuable insight into class-specific performance and highlights potential areas for further refinement, particularly in improving outdoor scene classification.



Accuracy Curve for best model (More Conv)



Loss Curve for best model (More Conv)

d) Comparison with different approaches

Model	Precision	Recall	F1 Score	Accuracy
Decision Tree	0.7910	0.7900	0.7898	0.7836
Random Forest	0.8753	0.8750	0.8749	0.8750
XGBoost	0.8838	0.8800	0.8796	0.8800
CNN	0.9481	0.8788	0.9121	0.9155

Table 2: Accuracy Comparison of all models

The performance comparison shows that the CNN model outperforms all traditional machine learning models—Decision Tree, Random Forest, and XGBoost—across all metrics. While XGBoost and Random Forest achieve strong and comparable results, CNN leads with the highest **precision (0.9481)**, **F1 score (0.9121)**, and **accuracy (0.9155)**, indicating its superior ability to capture spatial patterns in image data. The Decision Tree lags behind with the lowest metrics, reflecting its limited capacity to model complex data structures. Overall, CNN proves to be the most effective model for this classification task.

V. References

[1] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Application of gradient-based learning techniques to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[2] Zhou, B., Lapedriza, A., Xiao, J., & Torralba, A. *Places: A 10 Million Image Dataset for Scene Understanding*. [Online]. Available: http://places.csail.mit.edu/downloadData.html

[3] Ioffe, S., & Szegedy, C. (2015). Batch normalization: A technique to accelerate deep network training by mitigating internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

[4] Chen, T., & Guestrin, C. (2016). XGBoost: A highly efficient tree boosting framework. In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 785–794.

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 1097–1105.

[6] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.

[7] Kingma, D. P., & Ba, J. (2015). Adam: A stochastic optimization method. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA.