Image Classification using Convolutional Neural Network

Mohammed Shurrab (40323793), Oleksandr Yasinovskyy (40241188), Huzaifa Mohammed (40242080)

Abstract— **This project addresses the task of indoor scene classification using a custom Convolutional Neural Network (CNN) implemented in PyTorch. While Phase I explored traditional machine learning models such as Support Vector Machines (SVM), Random Forests (RF), and Decision Trees (DT) relying on hand-engineered features, these methods struggled to capture complex spatial patterns. In this second phase, we designed and trained a compact CNN to classify RGB images into three categories: library, museum, and shopping mall. The dataset comprises 15,000 training and 300 test images, with preprocessing and augmentation techniques applied to improve generalization. Extensive experiments were conducted using hyperparameter tuning via Ray Tune. The final CNN model achieved superior performance across all evaluation metrics compared to classical models. Ablation studies further highlight the impact of learning rate, batch size, and architecture depth. The results confirm that deep learning significantly enhances classification accuracy and robustness for high-dimensional image data.**

Index Terms **— Machine Learning, Classification, Deep learning, convolutional neural networks (CNN), hyperparameters-tuning.**

I. Introduction

IMAGE classification remains a cornerstone problem in computer vision, where popular machine learning (ML) algorithms have been utilized to tackle such a problem. However, the classical ML techniques explored in Phase I, such as support-vector machines (SVM) [1], and random forests (RF) [2] rely on hand-engineered features (e.g., color histogram and HOG) and show relatively poor performance, since they are incapable of capturing complex and non-linear patters in high dimensional images. Recent advances in GPU hardware have given rise to deep learning (DL), which made convolutional neural networks (CNN) [3] the de-facto standard for visual tasks. CNNs are a foundational architecture in deep learning, particularly effective for image analysis tasks. Unlike traditional machine learning models that rely on manual feature extraction, CNNs automatically learn multi-scale spatial features directly from raw pixel data through layers of convolutions, non-linearities, and pooling operations. This ability to extract hierarchical representations makes them ideal for classification, detection, and segmentation problems in fields such as medical imaging, autonomous driving, facial recognition, and scene understanding. The widespread adoption of CNNs is further supported by advances in hardware acceleration (GPUs), open-source frameworks (PyTorch, TensorFlow), and availability of large, labeled datasets [3]. This project investigates the viability of a compact, custom CNN model, based on PyTorch [4], for classifying indoor scenes into three categories, aiming to compare its performance with previously used classical machine learning models and assess whether deep learning offers measurable improvements in generalization and robustness.

II. Model Overview and Problem Formulation

A. Dataset preprocessing

The available dataset [5] consists of RGB images across three distinct classes, namely, *library, museum,* and *shopping mall.* Each class includes 5000 images for training, and 100 images for testing, resulting in a total of 15000 training samples and 300 testing samples. Each image is 256x256 pixels in resolution and stored in a structured directory format that supports automated label extraction. To improve generalization and simulate real-world variability, a series of data augmentation techniques during training were applied. These included random horizontal flipping ($p = 0.5$), random rotations up to ±15°, and color jittering (0.2) to introduce controlled variation in brightness, contrast, and saturation. After augmentation, images were converted to tensors and normalized channel-wise using the dataset's computed mean and standard deviation values. This normalization technique uses efficient streaming algorithm (Welford's algorithm) [6] that handles large datasets without the need to load the entire dataset, thus optimizing resource/memory usage. It also ensures stable convergence during training by centering the input distribution. For evaluation, only resizing and normalization were applied to maintain consistency. The dataset was split into training, validation, and testing subsets: 90% of the training images were used for training, and 10% were held out for validation using PyTorch's random_split. To ensure robust performance and break correlation between samples, the training dataset is shuffled randomly. The test set remained untouched throughout model development and was used only for final performance evaluation.

B. CNN architecture

CNN-based models consist of convolutional layers, pooling layers, flatten layers, and fully connected dense layers. The ability of CNN to learn and extract local patterns and features to make an informed decision makes them well suited for computer vision tasks. Hence, allowing for more robust and accurate analysis compared to traditional ML approaches that may struggle to capture complex relationships within the data.

The input of the proposed CNN consists of the RGB image with size HxW (256x256 in this case). The CNN input consists of a 2D convolutional layer with a kernel size of (3,3), a filter

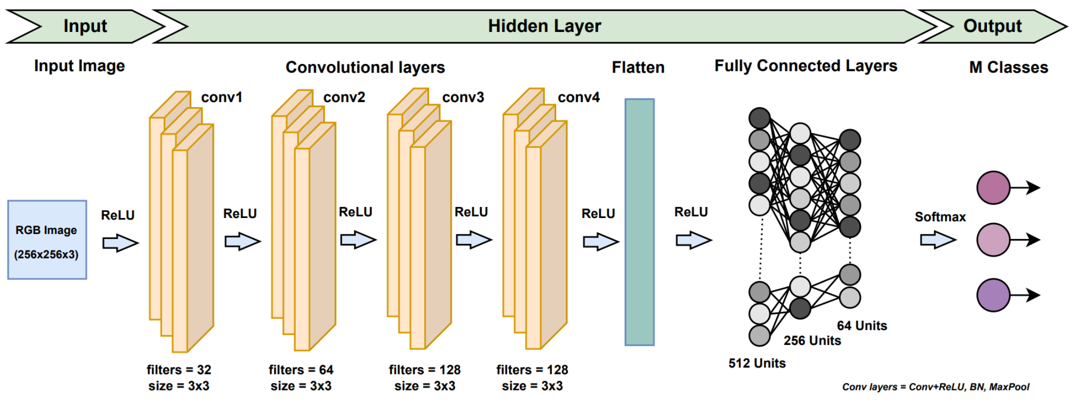


Fig. 1. CNN model architecture with 4 convolutional layers and 4 fully connected layers.

TABLE I
CNN MODEL ARCHITECTURE DETAILS

Layer	Kernel Size	Filter Depth	Activation
Convolution 1	(3,3)	32	ReLU
Convolution 2	(3,3)	64	ReLU
Convolution 3	(3,3)	128	ReLU
Convolution 4	(3,3)	128	ReLU
Flatten 1	-	-	-
Dense 1	-	512	ReLU
Dense 2	-	256	ReLU
Dense 4	-	64	ReLU
Output (Dense)	-	3	Softmax

TABLE II
CNN OPTIMIZATION HYPERPARAMETERS SETTINGS

Parameter	Values Tested	Purpose
Batch Size	64, 128, 256	Controls the number of training samples per iteration
Learning Rate	10^{-2}, 10^{-3}, 10^{-4}	Step size for optimizer weight updates
FC1 Size (Dense Layer)	128, 256, 512	Dimensionality of the first fully connected layer
Epochs	50	Maximum training cycles per configuration
Early Stopping	3 epochs	Stops training if no improvement in validation performance
Loss Function	Cross-Entropy	Measures multi-class classification error

depth of 32 and ReLU as the activation function , which is given as [7]:

$$ReLU(z) = \max(0, z) \tag{1}$$

The input layer is followed by three other 2D convolutional layers with the same kernel size of (3,3) and the ReLU activation function, but with different filter depths of 32, and 64, 128, and 128, respectively. The convolutional layers are essential to extract spatial features and local patterns from the images. The number of feature maps increases progressively across layers (starting from a base filter size of 32), enabling the network to capture increasingly abstract visual patterns from low-level edges to high-level scene semantic. Each convolutional layer has a padding of 1 to retain the original image size, a 2D batch normalization (BN) layer which normalizes feature distribution to maintain zero mean and unit variance per channel, thus stabilizing training and speeding up convergence, and finally a 2x2 max pooling layer is applied which is beneficial for spatial dimension reduction and focusing on strongest activations (generalization capability). Afterwards, a flatten layer is used to convert the 2D feature map into a 1D vector that is fed to the next layers, which consist of fully connected dense layers. A total of three hidden dense layers are employed with filter depths of 512, 256, and 64, respectively, and a dropout layer with p=0.2 for regularization. The activation function for all the layers is ReLU. The final layer is the output which consists of a dense layer with 3 possible classes with a softmax activation function. The softmax function produces probability values for each class, indicating the likelihood that the image belonging to a particular class. The softmax (σ) activation function is given as [7]:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{m=1}^{M} e^{z_m}} \tag{2}$$

where z is the input for the layer, and M is the number of classes. The CNN is trained based on a backpropagation algorithm with the objective of minimizing the categorical cross entropy loss function which is represented as:

$$Loss\ (\gamma, \gamma') = -\sum_i \gamma_i \log(\gamma_i') \tag{3}$$

where γ is the actual category that the image belongs to (ground truth), whereas γ' is the predicted output from the CNN.
To improve convergence and numerical stability, all convolutional and linear weights were initialized using Kaiming-normal initialization [8], and batch normalization was applied after each convolution.

The architecture of the employed CNN model is summarized in Fig. 1, where it consists of the input and output layers, 4 two-dimensional convolutional layers (Conv+ReLU, BN and maxPool), a flatten layer, and 4 fully connected dense layers. The hyperparameters and the details of the CNN architecture are summarized in Table I.

C. Hyperparameters-tuning Optimization

The objective of hyperparameters tuning is to maximize validation accuracy while preventing over-fitting and minimizing training time. The hyperparameters that are explored in this phase include but not limited to number of layers, number of filters in each layer, the learning rate, and batch size. The hyperparameters for the model were optimized through a tuning process, based on Ray Tune [9], focusing on the learning rate, batch size and the architecture of the CNN layers. The loss function used for training was categorical

TABLE III
PERFORMANCE COMPARISON ACROSS DIFFERENT MODELS

Metric	CNN	RF	SVM	Semi-DT
Train Accuracy	94.1	85	91	57
Val Accuracy	86.1	-	-	-
Test Accuracy	89	66	70	58
Precision	89	67	70	58
Recall	89	66	70	58
F1-score	89	66	70	58

cross-entropy and the model accuracy was compared across validation and test sets. Table II summarizes the configurations that were explored, including batch sizes (64, 128, 256), number of convolution and fully connected layers (3,4,5), number of filters in the layers, and the learning rate (10^{-2}, 10^{-3}, 10^{-4}). All models were trained for a maximum of 50 epochs using a dataset split of 90% training and 10% validation. Early stopping [10] was implemented to limit runtime and prevent overfitting. If the model showed no improvement in training loss or validation accuracy for three consecutive epochs (defined by patience), then the training would be stopped to reduce computation and maintain efficient convergence when evaluating multiple model configurations. During training, performance was evaluated using accuracy, precision, recall, F1-score and confusion matrix. These metrics allowed to assess the correctness in classification. The final model configuration was selected based on the best validation accuracy across all configurations and later verified on the independent test dataset.

III. RESULTS

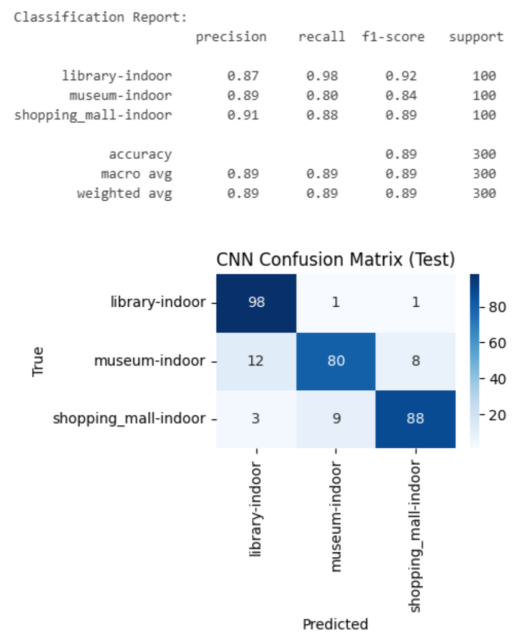
A. Model Training and Experimental Setup

In this phase, the efficacy of the proposed CNN model is evaluated. First, the optimum model is selected through extensive number of experiments, where the hyperparameters are varied across the different configurations discussed in section II.C. Each experiment is trained for a maximum of 50 epochs with early stopping enabled (patience = 3) based on the training accuracy. The best performing checkpoint in each trial is selected based on validation accuracy. An Adam optimizer is used for all sets of training since it uses adaptive learning rates for each weight thus, resulting in faster convergence while requiring no manual momentum tuning. The training is conducted on a macOS system equipped with an Apple M4 MPS backend and 36 GB unified memory.

The best model that exhibited the highest validation accuracy has the following configuration: 1) 4 Conv layers, 2) 4 FC layers, 3) LR = 10^{-3}, 4) Batch size = 64, and 5) Number of epochs = 50. This model is evaluated fully in terms of testing accuracy, precision, recall and F1 score, and compared with the models generated in phase I.

B. Performance Evaluation

A summary of the results is shown in Table III, where the CNN model is compared to Random Forest (RF), SVM, and semi-supervised decision tree (DT) [11]. The hyperparameters for the best RF model were number of estimators of 100, entropy criterion, tree depth of 9, min split of 5 and min leaf of 5. The best hyperparameters for the SVM model were RBF kernel with C=1, whereas the decision tree structure comprised of entropy



```
Classification Report:
                       precision    recall  f1-score   support

       library-indoor       0.87      0.98      0.92       100
        museum-indoor       0.89      0.80      0.84       100
 shopping_mall-indoor       0.91      0.88      0.89       100

             accuracy                           0.89       300
            macro avg       0.89      0.89      0.89       300
         weighted avg       0.89      0.89      0.89       300
```

Fig. 2. CNN classification report and confusion matrix for the test dataset.

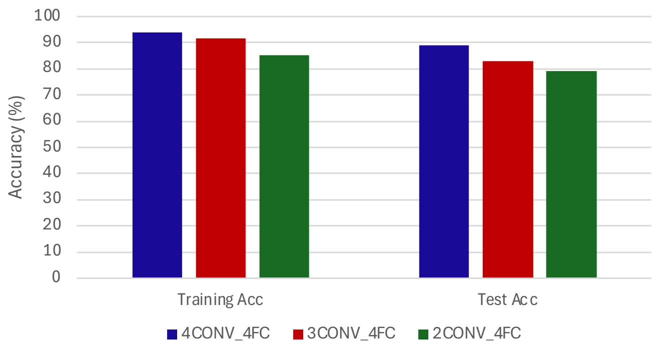


Fig. 3. Training and test accuracy comparison for different number of convolution layers

criterion, max depth of 10, while min split and min leaf were both set to 5.

It can be seen that CNN is outperforming all classical ML models across all performance metrics. This is due to the fact that CNN is capable of capturing and extracting spatial features and local patters within the images without the need for any external feature extraction techniques. It also has the capability of processing high dimensional inputs with their raw data. Unlike the other models that require manual feature engineering and can only accept 1 dimensional data.
Figure 2 shows the performance metrics along with the confusion matrix for the CNN model on the testing dataset.

C. Ablative Study

The model's performance sensitivity to key hyperparameter variations was verified using adjustments to learning rate, batch size and number of layers. Each configuration was evaluated using validation and test accuracy to verify the impact of the modification. The first set of experiments were done using changes in the number of convolution layers. By removing the fourth block, it is possible to see a notable drop in both training and test accuracy (Fig. 3). By varying the batch size (keeping the rest of the parameters the same at learning rate 10^{-3}, 4 conv layers and 4 FC layers) it was possible to observe the best

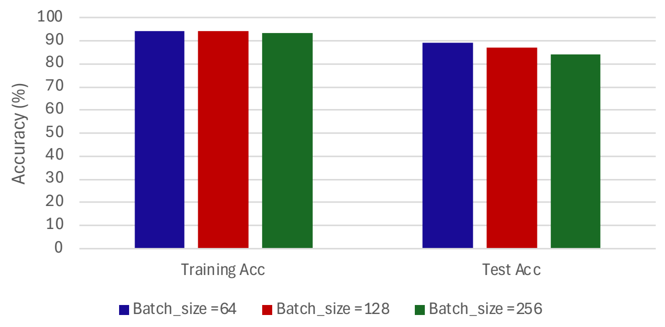


Fig. 4. Training and test accuracy comparison for different batch sizes

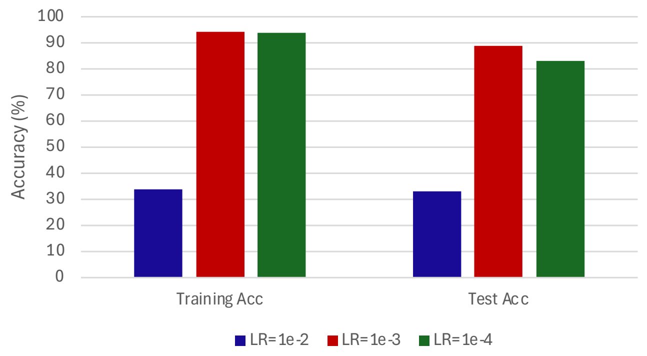


Fig. 5. Training and test accuracy comparison for different learning rates

generalization (batch size = 64). Increasing batch size to 128 caused a slight reduction in performance (training accuracy = 94.2%; test accuracy= 87%). while increasing the batch size further (256) the results decreased even further (training accuracy = 93.5%; test accuracy= 84%) as seen in Figure 4.

By varying the learning rate from 10^{-2} to 10^{-4} while keeping the other hyperparameters constant (batch size= 64, 4 conv layers, and 4 FC layers), the results show that model generalization and stability are largely affected by choice of learning rate (Fig. 5). A high learning rate (10^{-2}) caused unstable learning and severely reduced the performance (training accuracy = 33.8%; test accuracy = 33%). A low learning rate (10^{-4}) decreased the performance slightly (training accuracy = 94%; test accuracy = 83%). The configurations that had 10^{-3} as a learning rate produced the highest and most stable results.

IV. Future Improvements

Future work will focus on building on the results of the custom Convolutional Neural Network (CNN), the focus will be on further enhancing model performance and robustness. Potential improvements can include exploring deeper and more advanced CNN architectures such as ResNet or EfficientNet, leveraging transfer learning with pre-trained models to reduce computational overhead and training time, and expanding the dataset to cover additional indoor scene categories. Additionally, incorporating techniques like attention mechanisms, ensemble learning, and advanced data augmentation strategies could further boost classification accuracy and generalization. Finally, optimizing the model for deployment on resource-constrained devices and real-time inference will be valuable for practical applications.

V. Conclusion

During this phase, we aimed to evaluate the effectiveness of a custom Convolutional Neural Network (CNN) in comparison to classical machine learning models for the task of indoor scene classification. Through the design and implementation of a compact CNN using PyTorch, we demonstrated substantial improvements over traditional models like SVM, Random Forest, and Decision Tree, which rely on hand-crafted features and struggle with high-dimensional image data. Our model, trained on raw RGB images, achieved superior results across accuracy, precision, recall, and F1-score by learning spatial features automatically through stacked convolutional layers. Hyperparameter tuning using Ray Tune revealed that four convolution layers, a learning rate of 10^{-3} and batch size of 64 provided optimal performance, while ablation studies confirmed the sensitivity of the model to these parameters. The system successfully supports model saving, loading, and inference on individual images, fulfilling all practical implementation requirements. Overall, our results confirm that a well-designed CNN offers a powerful, scalable, and generalizable solution for visual classification tasks.

References

1. C. Cortes and V. Vapnik, “Support-vector networks”, *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
2. L. Breiman, “Random Forests”, *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
3. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
4. A. Paszke *et al*., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 8024–8035, 2019.
5. B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 Million Image Database for Scene Recognition”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.
6. B. P. Welford, "Note on a Method for Calculating Corrected Sums of Squares and Products," *Technometrics*, vol. 4, no. 3, pp. 419–420, Aug. 1962.
7. I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016.
8. K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 1026–1034.
9. R. Liaw *et al*., "Tune: A Research Platform for Distributed Model Selection and Training," in *Proc. 33rd Conf. Neural Information Processing Systems (NeurIPS), Workshop on Systems for ML*, Vancouver, Canada, 2018.
10. L. Prechelt, "Early stopping — But when?" in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer, 1998, pp. 55–69.
11. J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986.