Project Title :

Lab Week 14 - Convoluted Neural Networks

Name:

Nidhi Nitin Nag

SRN:

PES2UG23CS385

Course Name:

ML Lab

Submission Date:

November 19, 2025

Section :

F

1. INTRODUCTION

The goal of this lab was to build a Convolutional Neural Network (CNN) that can classify images of hands showing rock, paper, or scissors. Using a dataset pulled from Kaggle, I preprocessed the images, built a custom CNN from scratch in PyTorch, trained it, and then tested how well it could recognize each gesture. The main objective was to go through the full workflow: setting up the data, designing the model, training it, evaluating it, and finally trying it out on individual images.

2. MODEL ARCHITECTURE

CNN Architecture

The model is built using three convolutional blocks stacked one after another. Each block performs feature extraction at a deeper level, starting from basic edges and textures and moving toward more abstract patterns. After each block, the spatial dimensions are reduced through pooling, which lets the network progressively focus on higher-level features while keeping the number of parameters manageable.

Key Parameters

Every convolutional layer uses a 3×3 kernel with padding of 1, which helps preserve the width and height before pooling. The first block transforms 3 input channels into 16, the second expands 16 to 32, and the third increases it again from 32 to 64. Each block follows the same pattern: Conv2d → ReLU → MaxPool2d(2). The MaxPool2d layers downsample the image by a factor of 2 each time, taking the original 128×128 input down to 64×64, then 32×32, and finally 16×16.

Classifier

After the convolutional layers, the output feature map (64 channels at 16×16) is flattened and passed into a fully-connected classifier. The classifier starts with a Linear layer mapping the flattened size (64 × 16 × 16) to 256 units. This is followed by a ReLU activation and a dropout layer with p=0.3 to help reduce

overfitting. The final Linear layer maps the 256 units down to 3 outputs, corresponding to the three classes: rock, paper, and scissors.

3. TRAINING and PERFORMANCE

Hyperparameters

For training, I used the Adam optimizer with a learning rate of 0.001. The loss function was CrossEntropyLoss, which fits a multi-class classification problem like this. I trained the model for 10 epochs, using a batch size of 32 and shuffling only the training data.

Final Test Accuracy

```
Test Accuracy: 99.09%
```

4. CONCLUSION and ANALYSIS

Did the Model Perform Well?

Overall, the model performed very well, especially considering that the architecture was intentionally kept simple. It reached a strong accuracy on the test set and was able to correctly classify individual images when I tried them manually. For a custom CNN built from scratch, the performance was solid and consistent.

Were there any challenges you faced?

One of the main challenges was making sure that the transforms applied during training were exactly the same when predicting a single image later. Another small challenge was keeping track of the dimensions after each convolution and pooling layer so the fully-connected layer would receive the correct flattened size.

Suggest ways to improve model's accuracy in future

To push accuracy even further, I could add data augmentation so the model sees more varied images and learns to generalize better. Another upgrade would be inserting batch normalization layers to stabilize training. Beyond that, experimenting with a deeper CNN or even using transfer learning could also boost performance.