ML Lab Week 14: CNN Image Classification

Name: Najmus Seher

SRN:PES2UG23CS359

Section:F

Course: Machine Learning

1. Introduction

This lab builds, trains, and evaluates a Convolutional Neural Network (CNN) to classify images of hands showing Rock, Paper, or Scissors. The objective is to prepare image data, design a small CNN, train it on the provided dataset, and measure its performance on unseen test images.

2. Model Architecture

Convolutional feature extractor (conv_block):

- The network uses three convolutional blocks. Each block contains:
    1. Conv2d with kernel size = 3, padding = 1, then ReLU, then MaxPool2d(2).
    2. Channel progression:
        - Block 1: input channels = 3 -> output channels = 16
        - Block 2: 16 -> 32
        - Block 3: 32 -> 64
- Because each MaxPool2d halves spatial dimensions, an input image resized to 128×128 becomes 64×64 after the first pool, 32 x 32 after the second, and 16×16 after the third.

Fully-connected classifier (fc):

- After the conv blocks the feature map shape is (batch_size, 64, 16, 16).
- The classifier is:
    1. Flatten()
    2. Linear(64 * 16 * 16 → 256)
    3. ReLU()
    4. Dropout(p=0.3)
    5. Linear(256 → 3) — final logits for the three classes (rock, paper, scissors).

Design rationale: compact architecture keeps model small (fast to train) while providing progressive feature abstraction (increasing channels + downsampling). ReLU activation and dropout chosen for nonlinearity and basic regularization.

3. Training and Performance

Data processing & loader details

- Image transforms:

  - Resize all images to 128 × 128
  - ToTensor()
  - Normalize with mean = 0.5, std = 0.5 (applied per channel)
- Dataset loading:
  - Used torchvision.datasets.ImageFolder(DATA_DIR, transform=transform)
  - Split: 80% training, 20% test using random_split
  - Batch size = 32
  - train_loader shuffled, test_loader not shuffled

Training hyperparameters

- Optimizer: Adam
- Loss function: CrossEntropyLoss
- Learning rate: 0.001
- Epochs: 10
- Batch size: 32
- Device: torch.device("cuda" if torch.cuda.is_available() else "cpu")-training uses GPU if available, otherwise CPU.

  In my case I used the cpu

Final test accuracy

Test Accuracy: 98.17%

.

4. Conclusion and Analysis

Overall Performance

- The model performed **very well**, achieving a **98.17% test accuracy**, which shows that the CNN learned to distinguish Rock, Paper, and Scissors images almost perfectly.
- The high accuracy also suggests that the model generalized well to unseen test images.

Challenges Faced

- One challenge was making sure the dataset paths and folder structure were correct so the images could load without errors.
- Another challenge was understanding how the CNN layers work together, especially calculating output sizes for the fully connected layer.
- Training also took some time, especially when running on CPU, so monitoring the progress was important.

Possible Improvements

- I could add **data augmentation** (like rotations, flips, brightness changes) to make the model even more robust to different hand positions or lighting conditions.
- Another improvement would be using **transfer learning** (e.g., ResNet18), which could slightly boost accuracy and reduce training time.
- Increasing the number of epochs or tuning hyperparameters (like learning rate or batch size) could also help push the accuracy even higher.