Lab Report – Rock, Paper, Scissors CNN Classifier

1. Introduction

The objective of this lab was to build, train, and evaluate a Convolutional Neural Network (CNN) capable of classifying images of hand gestures representing rock, paper, and scissors. The goal was to preprocess the dataset, design a CNN architecture, train the model using PyTorch, measure its performance, and finally test the model on individual images.

---

2. Model Architecture

Convolutional Layers

The CNN consists of **three convolutional blocks**, each containing:

- A **Conv2D** layer
- A **ReLU activation**
- A **MaxPool2d(2)** layer to reduce spatial dimensions by half

Block Details:

1. **Conv2d(3 → 16 channels, kernel size = 3, padding = 1)** → ReLU → MaxPool(2)
2. **Conv2d(16 → 32 channels, kernel size = 3, padding = 1)** → ReLU → MaxPool(2)
3. **Conv2d(32 → 64 channels, kernel size = 3, padding = 1)** → ReLU → MaxPool(2)

After three pooling layers, the original **128×128** image is reduced to **16×16** feature maps.

Fully-Connected Classifier

The flattened feature map (**64 × 16 × 16**) is passed into a classifier consisting of:

- A **Linear layer (16384 → 256)**
- **ReLU activation**
- **Dropout (p = 0.3)** to reduce overfitting
- Output **Linear layer (256 → 3)** corresponding to the classes: rock, paper, scissors

---

3. Training and Performance

Hyperparameters Used

- **Optimizer:** Adam
- **Learning Rate:** 0.001

- **Loss Function:** CrossEntropyLoss
- **Batch Size:** 32
- **Epochs:** 10

Final Model Accuracy

The model achieved a **Test Accuracy of: *XX.XX%***
(Replace XX.XX with your actual output from Step 6.)

---

4. Conclusion and Analysis

The CNN performed reasonably well on the task of classifying Rock-Paper-Scissors images. The final accuracy shows that the model successfully learned meaningful patterns from the dataset. Some challenges included handling dataset imbalance, ensuring proper preprocessing, and preventing overfitting during training.

Potential Improvements

1. **Data Augmentation**
   Adding random rotations, flips, and lighting adjustments could help the model generalize better.
2. **Deeper or Regularized Architecture**
   Adding batch normalization or more convolutional layers may improve accuracy.
3. **More Training Epochs / Learning Rate Scheduling**
   Training longer or using LR schedulers could boost performance