Machine Learning

ML Lab Week 14: CNN Image Classification

Name: Veera Mahendra Reddy

SRN: PES2UG23CS391

Section: F

Introduction

The objective of this laboratory exercise was to design, train, and evaluate a Convolutional Neural Network (CNN) capable of classifying hand gesture images into three categories: rock, paper, or scissors. The Rock-Paper-Scissors dataset from Kaggle contains labeled hand gesture images with natural variations such as background, hand size, skin tone, and orientation. The experiment demonstrates the effectiveness of CNNs for image-based supervised classification tasks using PyTorch.

Dataset Description

The dataset contains 2,188 labeled images distributed across three folders: rock, paper, and scissors. Each class corresponds to real-world hand gesture images. The dataset was downloaded using the kagglehub API, and then structured into a format compatible with PyTorch's ImageFolder.

Class	Count
Rock	~730
Papers	~730
Scissors	~730
Total	2,188

```
Classes: ['paper', 'rock', 'scissors']
Total images: 2188
Training images: 1750
Test images: 438
```

The dataset was randomly split into 80% training (1,750 images) and 20% testing (438 images).

Model Architecture

A custom CNN architecture was designed consisting of three convolutional blocks followed by a fully connected classifier. Each block included Conv2D → ReLU → MaxPool2D, gradually extracting hierarchical spatial features

Layers Overview

Component	Details
Input	3-channel RGB image resized to 128×128
Conv Block 1	Conv2D(3→16, kernel=3, padding=1), ReLU, MaxPool(2)
Conv Block 2	Conv2D(16→32, kernel=3, padding=1), ReLU, MaxPool(2)
Conv Block 3	Conv2D(32→64, kernel=3, padding=1), ReLU, MaxPool(2)
Flatten	Output reshaped to 64×16×16
FC Layer 1	Linear → 256 neurons, ReLU, Dropout(0.3)
Output Layer	Linear → 3 neurons (rock, paper, scissors)

Activation Function

- ReLU after each Conv and FC layer for non-linearity

Regularization

- Dropout(0.3) applied before the final output layer

Training Configuration

Parameter	Value
Optimizer	Adam
Loss Function	CrossEntropyLoss
Learning Rate	0.001
Epochs	10
Batch Size	32
Device	CPU/GPU Auto Detection

The model was trained for 10 epochs, showing progressive reduction in loss— which indicates successful learning.

```
Epoch 1/10, Loss = 0.6206
Epoch 2/10, Loss = 0.2054
Epoch 3/10, Loss = 0.0805
Epoch 4/10, Loss = 0.0683
Epoch 5/10, Loss = 0.0200
Epoch 6/10, Loss = 0.0384
Epoch 7/10, Loss = 0.0226
Epoch 8/10, Loss = 0.0038
Epoch 9/10, Loss = 0.0028
Epoch 10/10, Loss = 0.0017
Training complete!
```

Evaluation Results

After training, the model was evaluated on the **testing set**

- **Test Accuracy Achieved: 99.32%**

This high accuracy indicates that the CNN is able to correctly distinguish between gestures with strong generalization performance.

The model was further tested on a single image, and predictions were validated using random samples via a simulated Rock-Paper-Scissors game.

```
print(f"Test Accuracy: {100 * correct / total:.2f}%")
```

```
Test Accuracy: 99.32%
```

```
Randomly selected images:
Image 1: /content/dataset/rock/whv9ZooPZNEjStCk.png
Image 2: /content/dataset/scissors/17HZDUFSVPxcar99.png

Player 1 shows: rock
Player 2 shows: scissors

RESULT: Player 1 wins! rock beats scissors
```

Conclusion

The designed Convolutional Neural Network successfully learned to classify rock, paper, and scissors hand gestures with **99.32%** accuracy, demonstrating that CNNs are highly effective for image classification tasks. This lab strengthened understanding of deep learning workflows, image preprocessing, CNN design, training loops, evaluation, and inference using PyTorch.

Discussion & Analysis

The model achieved excellent accuracy due to:

- Sufficient dataset size and class balance
- Effective convolutional feature extraction
- Appropriate normalization and architecture depth
- Proper learning rate and optimizer choice However, potential limitations include:
- No data augmentation — performance may degrade with unseen backgrounds
- Dataset contains mostly controlled environment samples
- Risk of slight overfitting since training loss approached zero

Possible Improvements

To further improve performance and robustness, the following enhancements are suggested:

1. Add Data Augmentation
    - Random rotation, flip, brightness shift, noise
2. Use Transfer Learning
    - Replace custom CNN with ResNet18, MobileNet, or EfficientNet
3. Add Learning Rate Scheduler
    - torch.optim.lr_scheduler.StepLR / ReduceLROnPlateau
4. Evaluate Using Confusion Matrix & Precision/Recall
    - Provides class-wise insights