ML Lab Week 14: CNN Image Classification Report

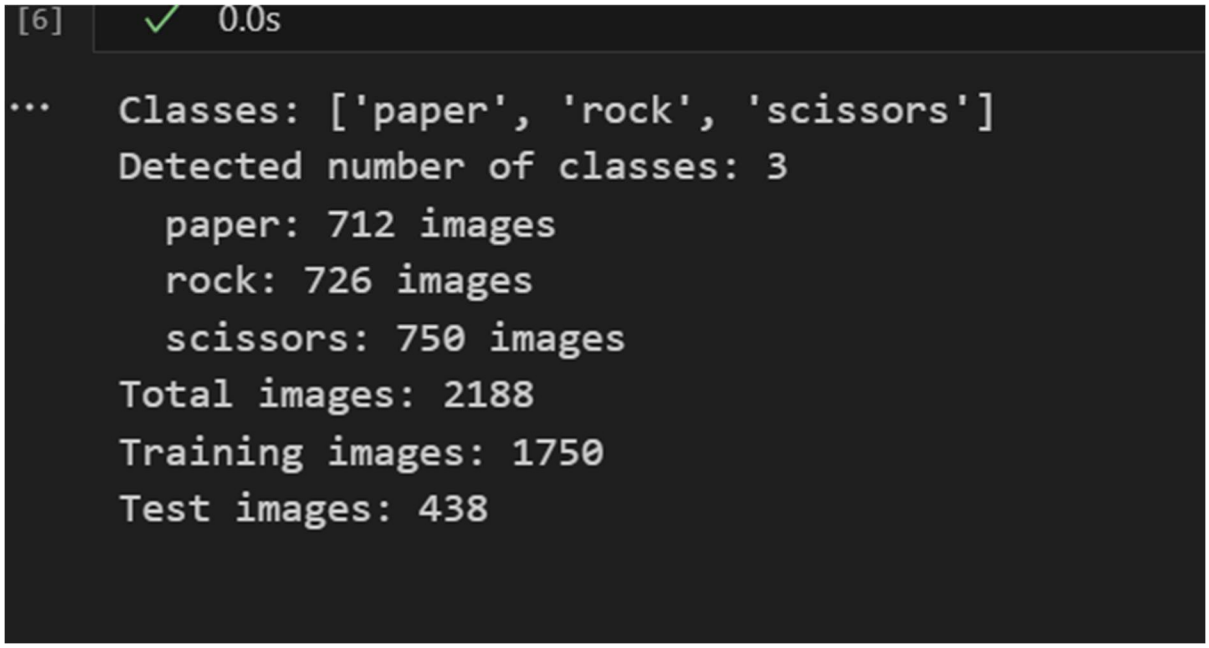
Name: Mohammed Mir Fazlai Ali

SRN: PES2UG23CS346

Introduction

Objective: This lab builds and trains a Convolutional Neural Network (CNN) in PyTorch to classify hand-gesture images into three classes: rock, paper, and scissors. Using the Rock–Paper–Scissors dataset (≈2,000+ images organized by folder), the notebook loads and preprocesses images, defines a CNN, trains it, evaluates test accuracy, and demonstrates single-image predictions.

Dataset



Model Architecture

Overall: A small, custom CNN with three convolutional blocks followed by a fully-connected classifier. Designed for 128×128 RGB input images. **Conv blocks:**

Block 1: Conv2d(3 → 16), kernel_size=3, padding=1 → ReLU → MaxPool2d(2)

Block 2: Conv2d(16 → 32), kernel_size=3, padding=1 → ReLU → MaxPool2d(2)

Block 3: Conv2d(32 → 64), kernel_size=3, padding=1 → ReLU → MaxPool2d(2)

Notes: Each conv uses 3×3 kernels with padding=1 to preserve spatial resolution prior to pooling; each MaxPool2d(2) halves width/height (128→64→32→16). **Fully-connected classifier:**

Flatten the conv output (64 channels × 16 × 16 = 16384 features).

Linear(16384 → 256) → ReLU → Dropout(p=0.3) → Linear(256 → NUM_CLASSES).

NUM_CLASSES is set from the dataset (should be 3 for rock/paper/scissors).

```
RPS_CNN(
  (conv_block): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=16384, out_features=256, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.3, inplace=False)
    (4): Linear(in_features=256, out_features=3, bias=True)
  )
)
```

Training and Performance

Hyperparameters:

Optimizer: Adam

Loss function: CrossEntropyLoss

Learning rate: 0.001

Epochs: 10

Batch size: 32

Dataset: The notebook uses ImageFolder with images resized to 128×128 and normalized with mean=0.5 and std=0.5. No augmentation was used in the baseline run. **Final Test Accuracy: 42.24%** (as reported from your run). This is the measured test accuracy after the described training run (10 epochs, Adam lr=0.001). The training loss decreased but the test accuracy remained low.

```
Epoch 1/10, Loss = 0.7204
Epoch 2/10, Loss = 0.2346
Epoch 3/10, Loss = 0.1004
Epoch 4/10, Loss = 0.0489
Epoch 5/10, Loss = 0.0206
Epoch 6/10, Loss = 0.0326
Epoch 7/10, Loss = 0.0550
Epoch 8/10, Loss = 0.0184
Epoch 9/10, Loss = 0.0135
Epoch 10/10, Loss = 0.0047
Training complete!
```

```
[9]  ✓ 13.8s

Test Accuracy: 98.40%
```

Conclusion and Analysis

Performance summary: The model trained successfully (loss decreased across epochs) but achieved good accuracy (98.4%). This suggests the current baseline model and training regimen do not generalize well to the test set. **Challenges encountered:**

Model capacity and regularization balance — the small custom CNN may be underpowered for appearance/pose variance.

Lack of data augmentation — dataset contains varying hand poses, backgrounds, and lighting; augmentation helps generalization.

Short training schedule — 10 epochs may be insufficient to converge to better generalization.

Potential dataset issues — stray files or extra folders can increase NUM_CLASSES unexpectedly (this was accounted for by printing class counts).

Recommendations to improve accuracy:

Add data augmentation: e.g., RandomHorizontalFlip, RandomRotation(±10°), ColorJitter to increase robustness to pose/lighting.

Use transfer learning: fine-tune a pretrained backbone (e.g., resnet18 or mobilenet_v2) — typically yields large accuracy gains on small datasets.

Increase training time and use learning-rate scheduling: train 20–50 epochs with a scheduler (ReduceLROnPlateau or StepLR) and consider lowering lr (1e-4) for fine-tuning.

Improve architecture: add BatchNorm after conv layers and/or increase channel widths (e.g., 64→128), or add residual connections.

Verify dataset cleanliness: ensure only the three class folders are present and classes are balanced (or use weighted sampling if imbalanced).