Lab 14 - CNN

Name: Muskan Goenka | SRN : PES2UG23CS355 | Section: F

1. Introduction

The objective of this lab was to design, implement, and train a Convolutional Neural Network (CNN) capable of classifying hand gesture images into three classes: **rock**, **paper**, and **scissors**. Using PyTorch, the task involved completing a partially implemented Jupyter Notebook, performing data preprocessing, building a CNN model, training it on the Rock-Paper-Scissors dataset, evaluating its accuracy, and finally demonstrating predictions using the trained model.

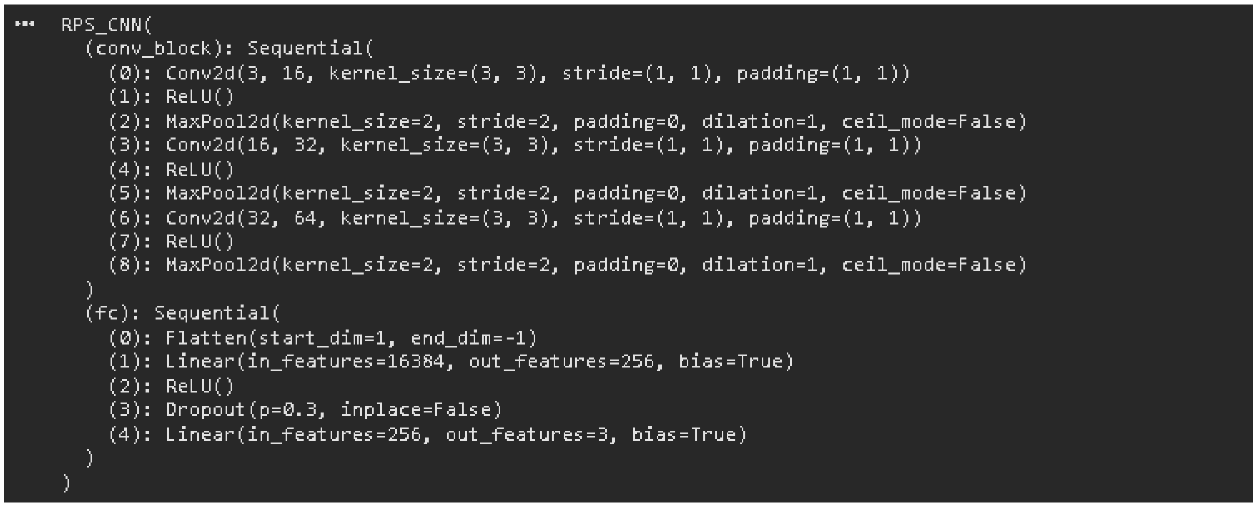
2. Model Architecture

The CNN model designed for this task consists of **three convolutional blocks** followed by a **fully connected classifier**.

Convolutional Blocks

Each block contains:

- **Conv2d layer**
    - Kernel size: **3 × 3**
    - Padding: **1**
    - Channels:
        - Block 1: 3 → 16
        - Block 2: 16 → 32
        - Block 3: 32 → 64
- **ReLU activation**
- **MaxPool2d(2)** which halves the spatial dimensions


```
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)
  )
)
```

Fully-Connected Classifier

The classifier consists of:

1. **Flatten layer**
2. **Linear layer:**

   64 × 16 × 16 → 256
3. **ReLU activation**
4. **Dropout (p = 0.3)**
5. **Final Linear layer:**

   256 → 3 (for rock, paper, scissors)

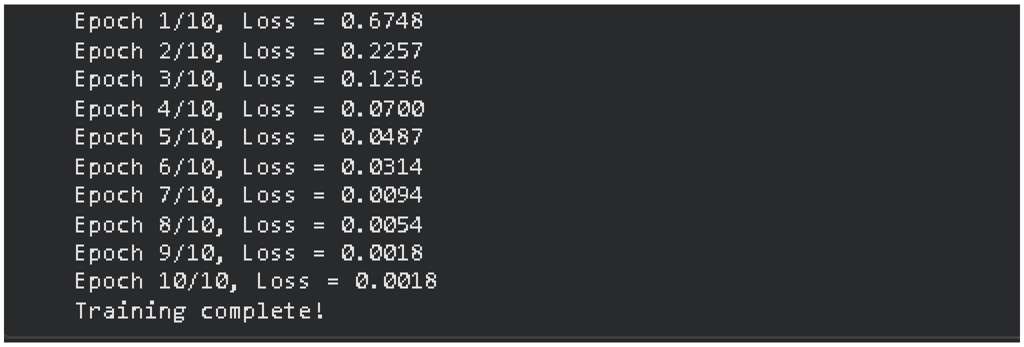
This architecture allows the model to progressively learn spatial features and map them into class scores.

3. Training and Performance

Hyperparameters Used

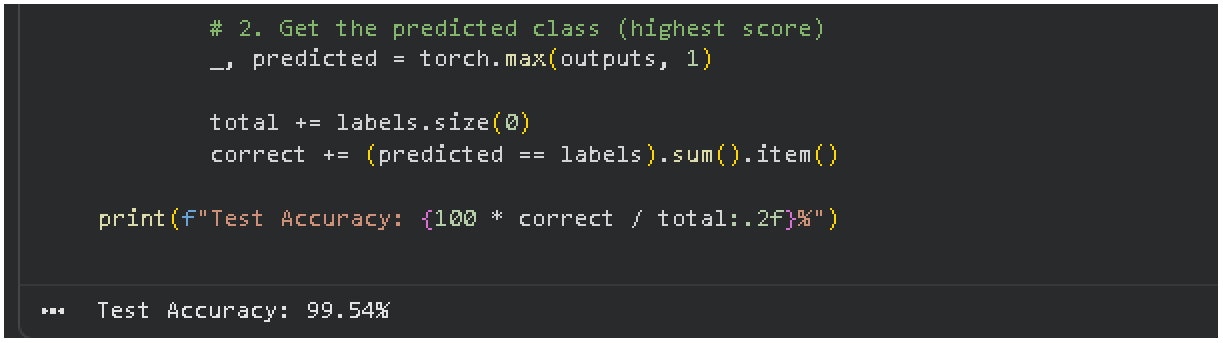
- **Optimizer:** Adam
- **Learning Rate:** 0.001
- **Loss Function:** CrossEntropyLoss
- **Batch Size:** 32

- **Epochs:** 10
- **Device:** GPU if available, otherwise CPU



```
Epoch 1/10, Loss = 0.6748
Epoch 2/10, Loss = 0.2257
Epoch 3/10, Loss = 0.1236
Epoch 4/10, Loss = 0.0700
Epoch 5/10, Loss = 0.0487
Epoch 6/10, Loss = 0.0314
Epoch 7/10, Loss = 0.0094
Epoch 8/10, Loss = 0.0054
Epoch 9/10, Loss = 0.0018
Epoch 10/10, Loss = 0.0018
Training complete!
```

Final Test Accuracy



```
            # 2. Get the predicted class (highest score)
            _, predicted = torch.max(outputs, 1)

            total += labels.size(0)
            correct += (predicted == labels).sum().item()

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

```
Test Accuracy: 99.54%
```

Final Test Accuracy: 99.54%

4. Conclusion and Analysis:

This CNN achieved strong performance on the Rock-Paper-Scissors dataset. The model successfully learned visual features of hand gestures and generalized well on the test set. The results indicate that even a relatively lightweight CNN architecture can classify natural images effectively when trained on a clean and structured dataset.

Challenges Faced

- Ensuring correct dataset folder structure in Colab
- Handling image transforms uniformly during training and prediction

- Understanding shape reductions through convolution and pooling layers

Possible Improvements

1. **Data Augmentation**

   Techniques like random rotations, flips, and brightness adjustments would help improve robustness.

2. **Deeper Model / More Filters**

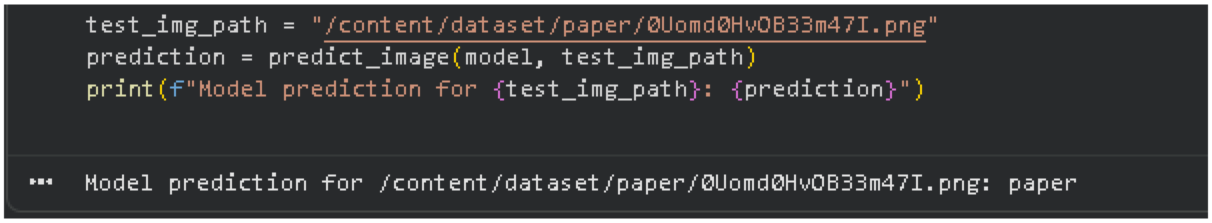
   Adding more convolutional layers or increasing channel counts could improve feature extraction.

3. **More Epochs & Learning Rate Scheduling**

   Training for longer or using schedulers like `StepLR` or `ReduceLROnPlateau` may improve accuracy.

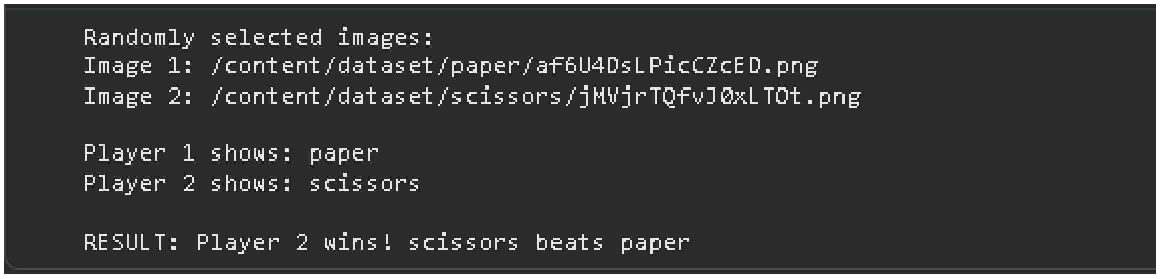
4. **Regularization Enhancements**

   Increasing dropout or using weight decay could prevent overfitting.



```
test_img_path = "/content/dataset/paper/0Uomd0HvOB33m47I.png"
prediction = predict_image(model, test_img_path)
print(f"Model prediction for {test_img_path}: {prediction}")
```

```
Model prediction for /content/dataset/paper/0Uomd0HvOB33m47I.png: paper
```



```
Randomly selected images:
Image 1: /content/dataset/paper/af6U4DsLPicCZcED.png
Image 2: /content/dataset/scissors/jMVjrTQfvJ0xLTOt.png

Player 1 shows: paper
Player 2 shows: scissors

RESULT: Player 2 wins! scissors beats paper
```