ML Lab Report Week 14: CNN Image Classification

NAME: Deepthi J

SRN: PES2UG23CS164

SECTION :5C

1)Introduction

The objective of this lab was to design, implement, and train a Convolutional Neural Network (CNN) capable of classifying images of hand gestures into three categories: rock, paper, and scissors. Using the Rock–Paper–Scissors dataset, the task involved data preprocessing, model construction in PyTorch, training, testing, and evaluating real predictions. The lab provided hands-on experience with convolutional architectures, activation functions, pooling mechanisms, and end-to-end deep learning workflows for image classification.

2. Model Architecture

The CNN model developed in this lab is composed of three sequential convolutional blocks, each designed to extract increasingly complex spatial features from the input images.

Convolutional Block 1:

- Conv2d layer: 3 → 16 channels
- Kernel size: 3 × 3, padding 1
- Activation: ReLU
- Downsampling: MaxPool2d (2 × 2)

Convolutional Block 2:

- Conv2d layer: 16 → 32 channels
- Kernel size: 3 × 3, padding 1
- Activation: ReLU
- Downsampling: MaxPool2d (2 × 2)

Convolutional Block 3:

- Conv2d layer: 32 → 64 channels
- Kernel size: 3 × 3, padding 1
- Activation: ReLU
- Downsampling: MaxPool2d (2 × 2)

Each MaxPooling layer reduces the spatial dimensions by half, transforming the original 128 × 128 input into 16 × 16 feature maps after three pooling operations. These features are then flattened and passed into a fully connected classifier, which includes:

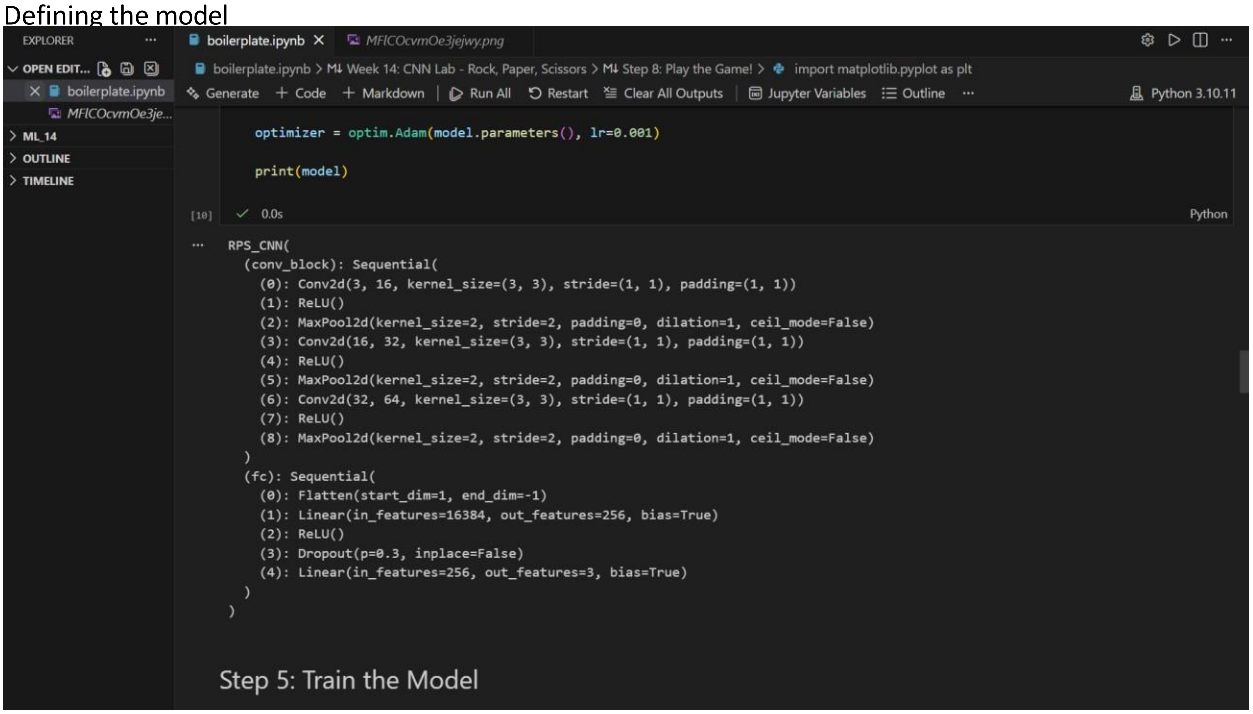
- A Linear layer mapping 64 × 16 × 16 → 256
- ReLU activation for non-linearity
- Dropout (p = 0.3) to reduce overfitting
- A final Linear layer mapping 256 → 3, producing logits for the three gesture classes. This architecture balances depth and computational efficiency, enabling effective feature extraction and robust classification performance for the Rock–Paper–Scissors dataset.

3. Training and Performance

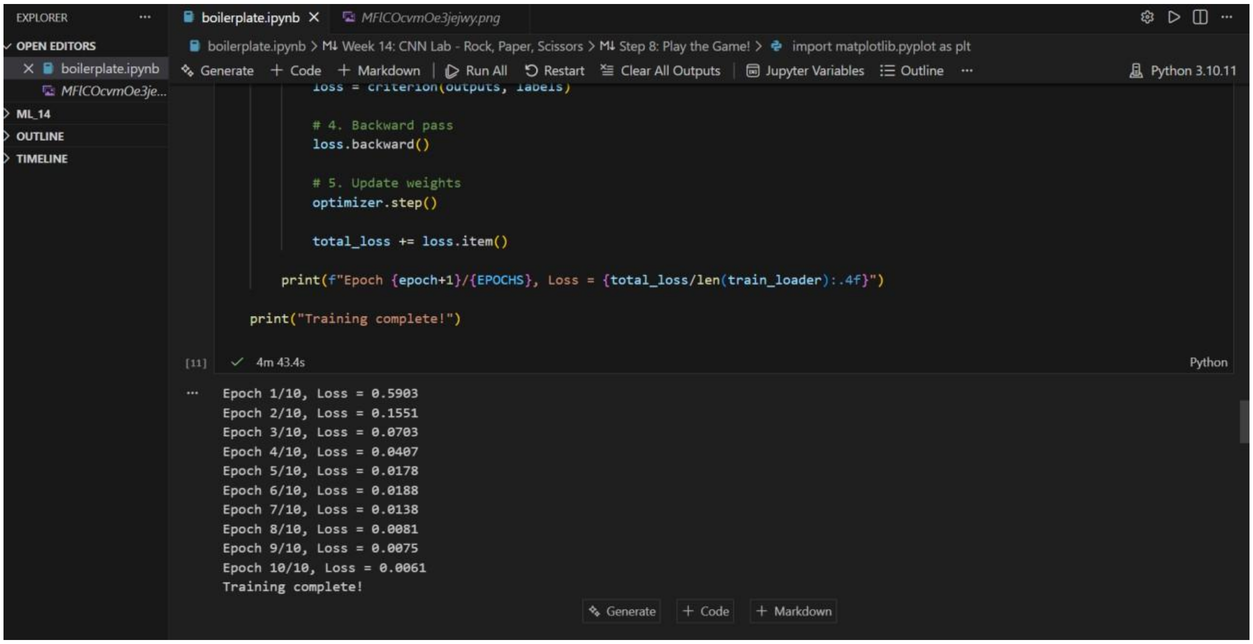
The model was trained using the following hyperparameters:

• Optimizer: Adam
• Loss Function: CrossEntropyLoss
• Learning Rate: 0.001
• Epochs: 10
• Batch Size: 32

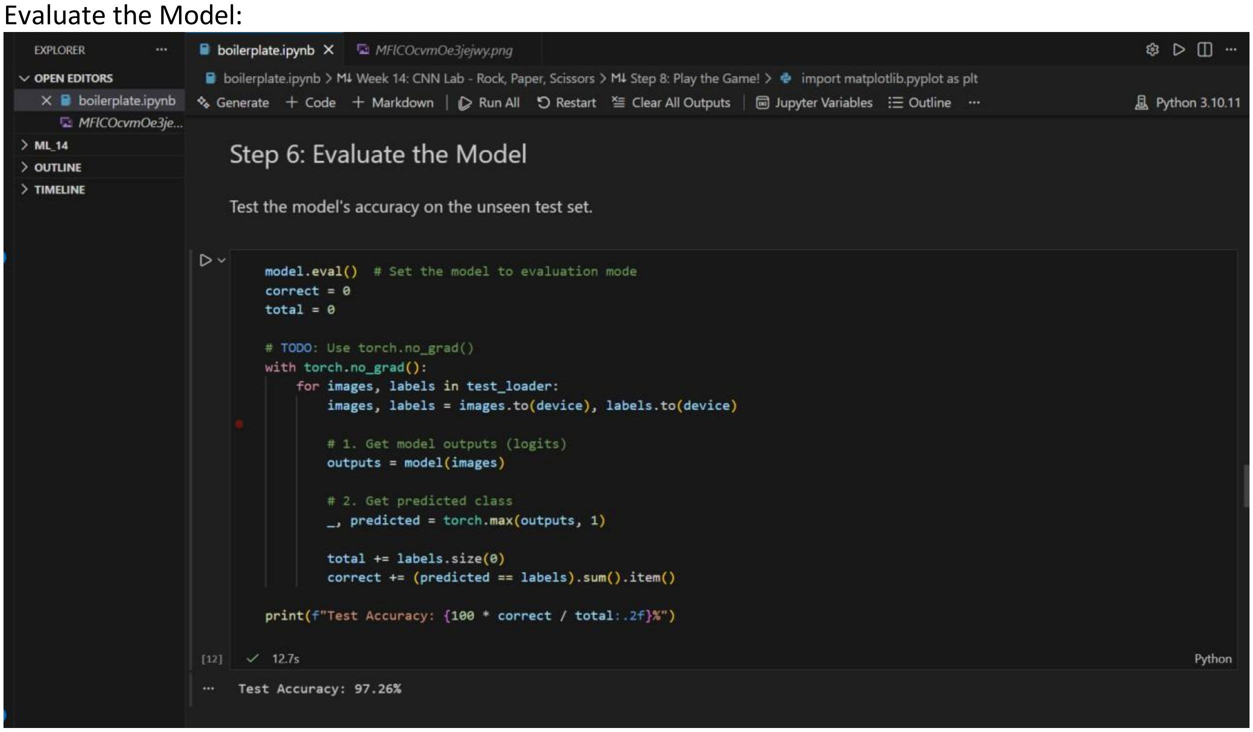
Defining the model



Training the model:



Evaluate the Model:



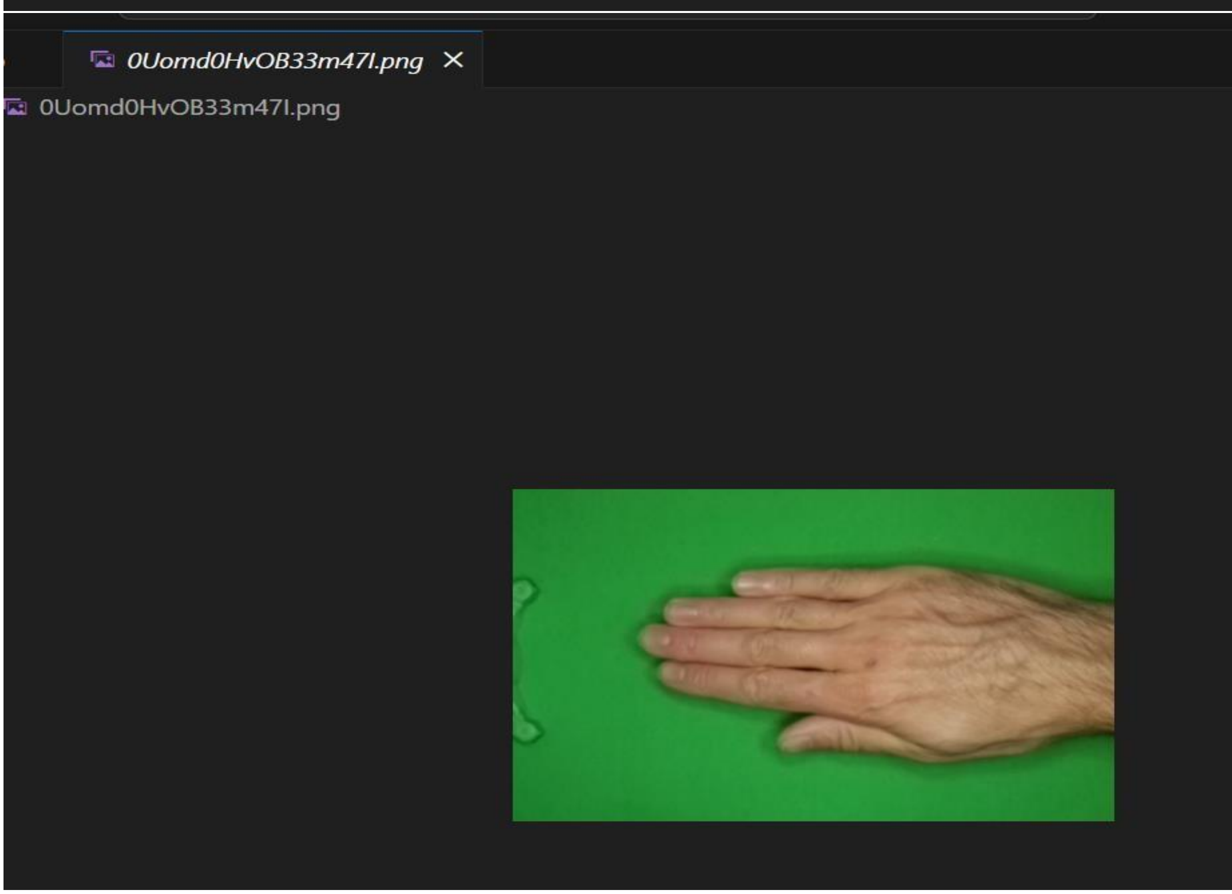
The dataset was split into 80% training and 20% testing. After training, the model achieved a final test accuracy of 97.26%, demonstrating strong generalization to unseen images.

Testing the model:

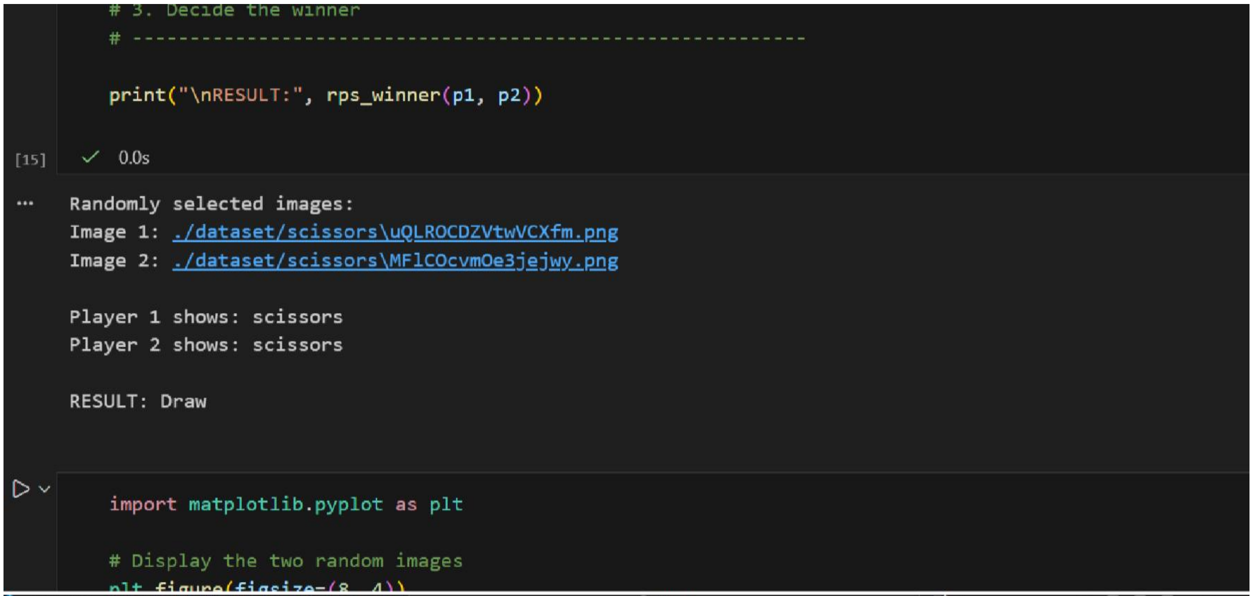
```python
def predict_image(model, img_path):
    model.eval()

    img = Image.open(img_path).convert("RGB")
    img = transform(img).unsqueeze(0).to(device)  # Apply transforms + batch dimension

    with torch.no_grad():
        # 1. Get raw model outputs (logits)
        output = model(img)

        # 2. Get predicted class index
        _, pred = torch.max(output, 1)

    return class_names[pred.item()]
# test the prediction function
test_img_path = "./dataset/paper/0Uomd0HvOB33m47I.png"
prediction = predict_image(model, test_img_path)
print(f"Model prediction for {test_img_path}: {prediction}")
```

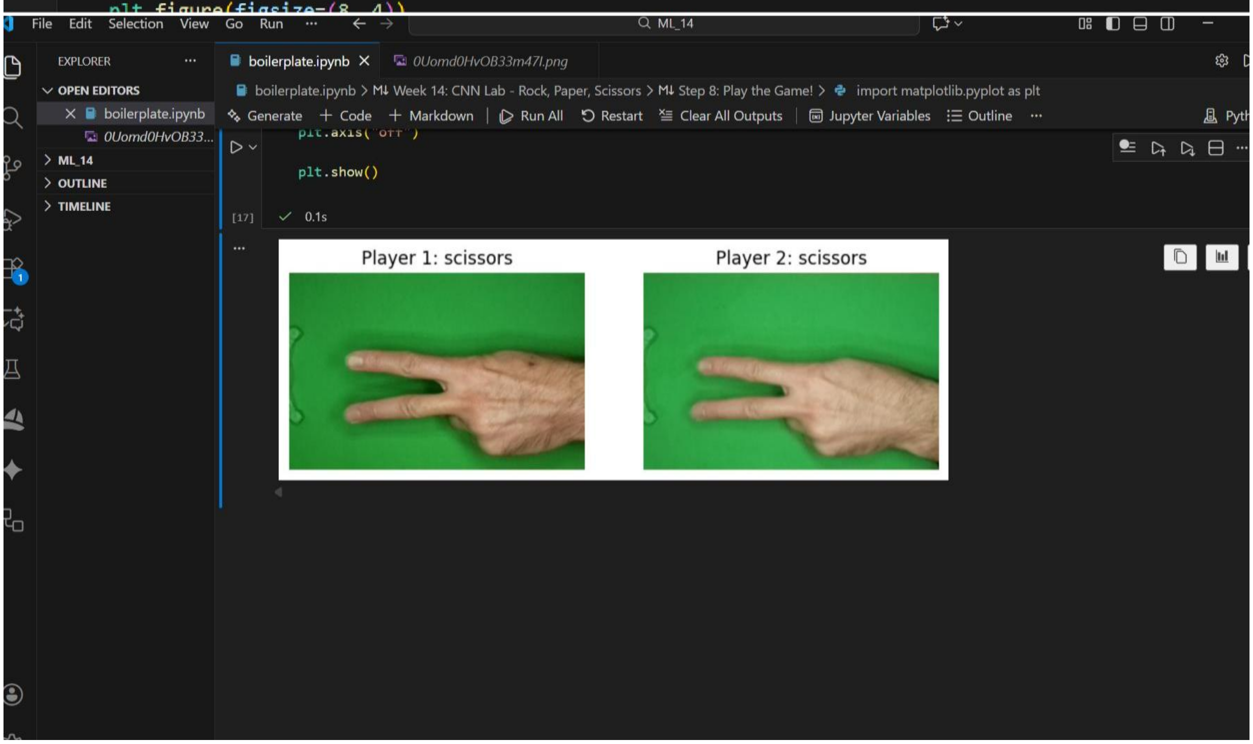
[13] ✓ 0.0s

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



Play the game:





4. Conclusion and Analysis

The model performed extremely well, achieving 97.26% accuracy on the test set. The CNN successfully learned to distinguish among the three gesture classes, validating the effectiveness of the chosen architecture. Challenges included setting up the dataset paths correctly and ensuring consistent preprocessing. Future enhancements that could further improve performance include applying data augmentation techniques or expanding the network with additional layers or batch normalization to improve feature extraction and stability