MACHINE LEARNING-LAB-WEEK-14

Name	**G S S Surya Prakash**
SRN	**PES2UG23CS192**
Course	**Machine Learning Lab**
Date	**18 November 2025**
Week	**14**

Introduction

This lab involved building and training a Convolutional Neural Network (CNN) using PyTorch to classify images of rock, paper, and scissors. The dataset from Kaggle was pre-organized into class folders, and the task required completing the notebook, implementing preprocessing, defining a CNN, and evaluating its accuracy. Overall, the model performed well, though challenges included preventing overfitting and handling variations in image backgrounds. Accuracy could be improved by adding data augmentation or using a deeper architecture with batch normalization.

Model Architecture

The CNN starts with three convolution layers that help the model learn different features from the images, such as shapes and edges. Each convolution layer uses a 3x3 filter and increases the number of channels for richer feature extraction. MaxPooling is used after each layer to reduce the image size and make the model faster. Once the

feature maps are created, they are flattened and passed into a fully connected network. The classifier has a dense layer with 256 units, followed by ReLU and dropout to avoid overfitting. The final output layer predicts one of the three classes: rock, paper, or scissors. This design allows the model to steadily learn better representations at each layer.

Training And Performance

- Optimizer: Adam optimizer
- Loss Function: Categorical Cross-Entropy (for multi-class classification)
- Learning Rate: 0.001
- Number of Epochs: 10

```
EPOCHS = 10

for epoch in range(EPOCHS):
    model.train() # Set the model to training mode
    total_loss = 0

    for images, labels in train_loader:
        # Move data to the correct device
        images, labels = images.to(device), labels.to(device)

        # TODO: Implement the training steps
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    print(f"Epoch {epoch+1}/{EPOCHS}, Loss = {total_loss/len(train_loader):.4f}")

print("Training complete!")
```

```
Epoch 1/10, Loss = 0.7153
Epoch 2/10, Loss = 0.2124
Epoch 3/10, Loss = 0.0956
Epoch 4/10, Loss = 0.0459
Epoch 5/10, Loss = 0.0412
Epoch 6/10, Loss = 0.0130
Epoch 7/10, Loss = 0.0139
Epoch 8/10, Loss = 0.0103
Epoch 9/10, Loss = 0.0141
Epoch 10/10, Loss = 0.0117
Training complete!
```

```
model.eval() # Set the model to evaluation mode
correct = 0
total = 0

# TODO: Use torch.no_grad()
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)

        # TODO: Get model predictions
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)

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

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

```
Test Accuracy: 98.86%
```

The model achieved a Final Test Accuracy of 98.86%.

Conclusion

The CNN gave good results and was able to classify the gestures with high accuracy. Even though the dataset was not very large, the model still learned the features well. The main challenge was avoiding overfitting, but using dropout helped. To improve performance further, increasing the dataset with augmentation or using more advanced layers could help.