Lab Report: CNN Image Classification (Week 14)

Name: Diya Prakash
SRN: PES2UG23CS184

Section: 5C CSE

1. Introduction

The objective of this lab was to design, implement, and train a Convolutional Neural Network (CNN) using the PyTorch framework to solve a multi-class image classification problem. The specific task was to classify images of hand gestures into three distinct categories: 'rock', 'paper', and 'scissors'. Using a dataset sourced from Kaggle, the project involved data preprocessing, defining a deep learning architecture, training the model using backpropagation, and evaluating its accuracy on a held-out test set.

2. Model Architecture

I implemented a custom CNN architecture class named RPS_CNN. The network processes input images of size pixels with 3 color channels (RGB).

Convolutional Blocks: The feature extraction component consists of three sequential convolutional blocks.

1. **Block 1:** Conv2d (3 input 16 output channels), followed by ReLU activation and MaxPool2d ().
2. **Block 2:** Conv2d (16 input 32 output channels), followed by ReLU activation and MaxPool2d ().
3. **Block 3:** Conv2d (32 input 64 output channels), followed by ReLU activation and MaxPool2d ().

Key Parameters: All convolutional layers used a kernel size of and padding of 1 to maintain spatial dimensions before pooling. Max pooling was used to downsample the feature maps, reducing computational complexity and inducing translation invariance.

Fully-Connected Classifier: After the third pooling layer, the output tensor (size) is flattened into a 1D vector of size 16,384.

1. **Hidden Layer:** A Linear layer mapping 16,384 inputs to 256 neurons, followed by ReLU activation.
2. **Regularization:** A Dropout layer with a probability of was applied to prevent overfitting.
3. **Output Layer:** A final Linear layer mapping 256 neurons to 3 output classes (Rock, Paper, Scissors).

3. Training and Performance

The dataset was split into a training set (80%, approx. 1,750 images) and a test set (20%, approx. 438 images). The training process utilized the following hyperparameters:

- **Optimizer:** Adam (Adaptive Moment Estimation)
- **Learning Rate:** 0.001
- **Loss Function:** CrossEntropyLoss (Combines LogSoftmax and NLLLoss)
- **Batch Size:** 32
- **Epochs:** 10

Results: The training loss decreased consistently from an initial value of **0.6361** in Epoch 1 to **0.0209** in Epoch 10.

- **Final Test Accuracy: 97.26%**

```
Test Accuracy: 97.26%
```

4. Conclusion and Analysis

Discussion of Results: The model performed exceptionally well, achieving over 97% accuracy on the test set. The rapid decrease in loss suggests that the architecture was well-suited for the complexity of the dataset. The validation script ("AI Referee") demonstrated that the model could correctly generalize to unseen images, distinguishing between the three hand shapes with high confidence.

Challenges Faced:

1. **Environment Setup:** Migrating the code from a Kaggle/Cloud environment to a local VS Code environment required adjusting file paths. The default Kaggle input paths (/kaggle/input) had to be replaced with dynamic paths returned by the kagglehub library.
2. **Dimension Mismatch:** Calculating the correct input size for the first fully connected layer required careful tracking of the image size reduction through the three MaxPool layers ().

Future Improvements: To further improve the model's robustness for real-world deployment:

1. **Data Augmentation:** Currently, we only resize and normalize. Adding random rotations, horizontal flips, or lighting adjustments during training would help the model handle hand gestures oriented in different directions (e.g., a hand coming from the side rather than straight up).
2. **Confusion Matrix:** Implementing a confusion matrix would allow us to see specifically which pairs of gestures are most likely to be confused (e.g., if 'rock' is frequently misclassified as 'paper'), allowing for targeted tuning.