Week 14 Lab Report: CNN for Rock, Paper, Scissors Classification

Name: PES1UG23CS917

Date: 17 November 2025

1. Introduction

This lab focused on building, training, and evaluating a Convolutional Neural Network (CNN) to classify images of hands playing Rock, Paper, or Scissors. The objective was to implement a deep learning model using PyTorch that could accurately distinguish between the three hand gestures from RGB images. The dataset was obtained from Kaggle and consisted of images across three classes, which were preprocessed, split into training and test sets, and used to train a custom CNN architecture.

2. Model Architecture

The CNN architecture implemented for this task consists of two main components: a convolutional feature extraction block and a fully-connected classification block.

Convolutional Block

The convolutional block is composed of three sequential layers, each following a pattern of convolution → activation → pooling:

1. First Layer:
    - Conv2d(3 → 16 channels, kernel_size=3, padding=1)
    - ReLU activation
    - MaxPool2d(kernel_size=2)
2. Second Layer:
    - Conv2d(16 → 32 channels, kernel_size=3, padding=1)
    - ReLU activation
    - MaxPool2d(kernel_size=2)
3. Third Layer:
    - Conv2d(32 → 64 channels, kernel_size=3, padding=1)
    - ReLU activation
    - MaxPool2d(kernel_size=2)

Key Design Choices:

- Kernel Size: A 3×3 kernel size was used throughout, which is a standard choice that effectively captures local spatial patterns while keeping computational costs manageable.
- Padding: Padding of 1 was applied to maintain spatial dimensions after convolution operations.
- Channel Progression: The number of channels progressively increases (3 → 16 → 32 → 64), allowing the network to learn increasingly complex features at different levels of abstraction.
- Max Pooling: Each convolutional layer is followed by 2×2 Max Pooling, which reduces spatial dimensions by half at each stage (128 → 64 → 32 → 16), providing translation invariance and reducing computational complexity.

Fully-Connected Classifier

After the convolutional feature extraction, the output is flattened and passed through a fully-connected classifier:

1. Flatten Layer: Converts the 3D feature maps (64 × 16 × 16) into a 1D vector of size 16,384
2. First Dense Layer: Linear(16,384 to 256) followed by ReLU activation
3. Dropout Layer: Dropout(p=0.3) for regularization to prevent overfitting
4. Output Layer: Linear(256 to 3) producing logits for the three classes (rock, paper, scissors)

The classifier uses dropout with a probability of 0.3 to improve generalization by randomly deactivating neurons during training.

3. Training and Performance

Hyperparameters

The following hyperparameters were used during training:

- Optimizer: Adam optimizer
- Learning Rate: 0.001
- Loss Function: CrossEntropyLoss (standard for multi-class classification)
- Number of Epochs: 10
- Batch Size: 32
- Train-Test Split: 80% training, 20% testing
- Image Size: 128×128 pixels
- Normalization: Mean=0.5, Std=0.5 for all three RGB channels

Training Process

The model was trained for 10 epochs with the training loss progressively decreasing, indicating effective learning. The training loop implemented standard practices including:

- Gradient zeroing before each forward pass
- Forward propagation through the network
- Loss calculation using CrossEntropyLoss
- Backpropagation to compute gradients
- Weight updates using the Adam optimizer

Performance Results

Final Test Accuracy: 97.72%

The model achieved an excellent test accuracy of 97.72% on the unseen test set, demonstrating strong generalization capabilities. This high accuracy indicates that the CNN successfully learned discriminative features to distinguish between rock, paper, and scissors gestures.

4. Conclusion and Analysis

Results Discussion

The model performed exceptionally well, achieving a test accuracy of 97.72%. This strong performance can be attributed to several factors:

1. Appropriate Architecture: The three-layer convolutional design with progressive channel expansion effectively captured hierarchical features from simple edges to complex hand shapes.
2. Sufficient Training Data: The 80-20 train-test split provided adequate data for both learning and evaluation.
3. Effective Regularization: The 0.3 dropout rate helped prevent overfitting, as evidenced by the high test accuracy.
4. Proper Preprocessing: Resizing all images to a uniform 128×128 size and normalizing pixel values ensured consistent input to the network.

Challenges Faced

1. Dataset Preparation: Initial setup required downloading the Kaggle dataset and organizing it into the appropriate directory structure.
2. Device Compatibility: Ensuring the code worked on both CPU and GPU required careful device management throughout the implementation.
3. Hyperparameter Selection: Choosing appropriate values for learning rate, batch size, and dropout probability required consideration of the dataset size and complexity.

Future Improvements

To potentially improve the model's accuracy further, the following approaches could be explored:

1. Data Augmentation: Implementing random transformations such as rotation, horizontal flipping, brightness adjustment, and zoom could increase the diversity of training samples and improve model robustness to variations in hand positions and lighting conditions.
2. Transfer Learning: Utilizing pre-trained models such as ResNet or EfficientNet and fine-tuning them on this specific task could leverage features learned from larger image datasets (like ImageNet) and potentially achieve even higher accuracy with fewer training epochs.
3. Ensemble Methods: Combining predictions from multiple models with different architectures or initialization could reduce variance and improve overall prediction reliability.
4. Deeper Architecture: Adding more convolutional layers or implementing skip connections (as in ResNet) could allow the model to learn even more complex feature representations.
5. Learning Rate Scheduling: Implementing a learning rate scheduler that reduces the learning rate over time could help the model converge to a better optimum.