

MACHINE LEARNING LAB WEEK 14

Name: Daksh Yadav

SRN: PES2UG23CS926

Section: C

1. Introduction

The primary objective of this lab is to design, implement, and train a Convolutional Neural Network (CNN) using the PyTorch framework to perform image classification. Specifically, the model is tasked with recognizing and categorizing images of hand gestures into three distinct classes: 'rock', 'paper', and 'scissors'. The experiment involves preprocessing the dataset, constructing a deep learning architecture comprising convolutional and fully connected layers, and evaluating the trained model's accuracy on an unseen test set to verify its ability to generalize to new data.

2. Model Architecture

The model, defined as the class `RPS_CNN`, utilizes a sequential Convolutional Neural Network (CNN) architecture designed for image classification. The architecture consists of two main components: a convolutional feature extractor and a fully connected classifier.

Feature Extraction (Convolutional Blocks)

The feature extraction stage is composed of three sequential convolutional blocks. Each block performs the following operations to progressively extract higher-level features while reducing the spatial dimensions of the input image:

- **Convolutional Layers:** The network increases the depth of the feature maps across the three blocks.

- **Block 1:** Transforms the input (3 RGB channels) to **16 channels**.
- **Block 2:** Increases depth from 16 to **32 channels**.
- **Block 3:** Increases depth from 32 to **64 channels**.
- **Kernel Parameters:** All convolutional layers utilize a **kernel size of 3x3** with a padding of 1, preserving spatial dimensions before pooling.
- **Activation Function:** A Rectified Linear Unit (**ReLU**) activation follows each convolution to introduce non-linearity.
- **Pooling:** Each block concludes with a **Max Pooling** layer (kernel size 2x2), which halves the spatial dimensions of the feature maps (reducing the 128x128 input down to 16x16 by the final block) to reduce computational complexity and control overfitting.

Fully Connected Classifier

Following feature extraction, the 3D feature maps (64 channels × 16 × 16) are flattened into a 1D vector to be processed by the dense layers:

- **Flattening:** The output is flattened into a vector of size **16,384** (64*16*16).
- **Hidden Layer:** This vector is passed through a linear layer with **256 neurons**, followed by a ReLU activation.
- **Regularization:** A **Dropout** layer with a probability of **0.3** is applied to prevent overfitting by randomly zeroing out neurons during training.
- **Output Layer:** The final linear layer maps the 256 features to **3 output classes** corresponding to 'rock', 'paper', and 'scissors'.

3. Training and Performance

Hyperparameters The model was trained using the following hyperparameters to optimize the weights and minimize classification error:

- **Optimizer:** Adam (Adaptive Moment Estimation)
- **Loss Function:** CrossEntropyLoss (suited for multi-class classification)

- **Learning Rate:** 0.001
- **Number of Epochs:** 10

Performance After training for 10 epochs, the model was evaluated on the held-out test set (20% of the total dataset) to determine its generalization capability.

- **Final Test Accuracy:** 98.86%

4. Conclusion and Analysis

Results Discussion

The Convolutional Neural Network (CNN) implemented in this lab successfully learned to classify hand gestures into the categories of 'rock', 'paper', and 'scissors'. With a final test accuracy of **98.86%**, the model demonstrated strong performance. This indicates that the three-block convolutional architecture was effective in extracting relevant spatial features from the 128x128 input images and distinguishing between the different hand shapes.

Challenges Faced

Some of the potential challenges observed during the experiment were:

- **Overfitting:** The model might memorize the training data rather than generalizing, where training accuracy increases while test accuracy stagnates. The inclusion of the Dropout layer ($p=0.3$) helped mitigate this.
- **Similarity in Gestures:** Certain angles of 'rock' and 'paper' can look similar, occasionally leading to misclassification by the model.

Future Improvements

To further improve the model's accuracy and robustness, the following strategies could be implemented:

1. **Data Augmentation:** Currently, the preprocessing only involves resizing and normalization. Applying random transformations such as *RandomHorizontalFlip*, *RandomRotation*, or *ColorJitter* would artificially expand the training set and help the model generalize better to unseen variations.
2. **Hyperparameter Tuning:** Increasing the number of epochs beyond 10 or experimenting with a dynamic learning rate scheduler could allow the model to converge to a better minimum, potentially squeezing out higher accuracy.