PES UNIVERSITY EC,

ML Lab Week-14

CNN Image Classification

Name : Sharath Gowda GR

SRN : PES2UG24CS823

Section : F

1. Introduction

The objective of this lab was to design, build, and train a Convolutional Neural Network (CNN) using the PyTorch framework to accurately classify images of hand gestures into one of three categories: 'rock', 'paper', or 'scissors'. The model was trained on the "Rock Paper Scissors" dataset, which contains over 2,000 images, and its performance was evaluated on an unseen test set. The final trained model was then used to make predictions on test images.

2. Model Architecture

The model, named RPS_CNN, is a sequential Convolutional Neural Network followed by a Fully-Connected (FC) classifier.

Convolutional Block (conv_block)

The convolutional block is designed to automatically extract hierarchical features from the input images and consists of three repeating sequences of layers:

Layer Type	Input Channels	Output Channels	Kernel Size	Stride/Pooling	Description
Conv2d	3	16	3x3	1	Processes the 3-channel (RGB) image.
ReLU	-	-	-	-	Non-linear activation.
MaxPool2d	-	-	2x2	2	Halves the spatial dimensions (e.g., 128X128 TO 64X64).

Conv2d	16	32	3x3	1	Increases feature complexity.
MaxPool2d	-	-	2x2	2	Halves the spatial dimensions
Conv2d	32	64	3x3	1	Further increases feature complexity.
MaxPool2d	-	-	2x2	2	Final reduction to a 16x16x64 tensor.

The key parameters are a **kernel size of 3x3** for all convolutional layers, and **Max Pooling with a size of 2 and stride of 2**.

Fully-Connected Classifier

The output of the convolutional block (a tensor of size 64x16x16 , which flattens to 16,384 features) is fed into the classifier:

1. **Flatten**: Converts the 16x16x64 output into a 16,384 element vector.
2. **Linear**: A fully connected layer mapping **16,384 input features to 256 output features**.
3. **ReLU**: Non-linear activation.
4. **Dropout**: A **dropout rate of p=0.3** is applied for regularization to prevent overfitting.
5. **Linear**: The final layer mapping **256 features to 3 output features** (corresponding to the 'rock', 'paper', 'scissors' classes).

3. Training and Performance

Key Hyperparameters

Hyperparameter	Value	Description
Optimizer	Adam	Adaptive Moment Estimation.
Loss Function	nn.CrossEntropyLoss()	Used for multi-class classification.
Learning Rate	0.001	Rate at which the model weights are updated.
Number of Epochs	10	The number of full passes over the training dataset.
Batch Size	32	Number of samples processed before the model's internal parameters are updated.
Device	cuda	The model was trained on a GPU for faster processing.

Final Test Accuracy

The model was evaluated on the unseen test dataset (438 images) after 10 epochs of training.

- **Final Test Accuracy: 97.95%**

4. Conclusion and Analysis

Discussion of Results

The model performed exceptionally well, achieving a **97.95% Test Accuracy**. This high accuracy suggests the CNN architecture was highly effective at learning the necessary features to distinguish between the 'rock', 'paper', and 'scissors' hand gestures. The combination of 3x3 convolutions, ReLU activations, and 2x2 Max Pooling layers successfully extracted robust and scale-invariant features. The use of the Adam optimizer and a small learning rate allowed for efficient convergence of the training loss. The presence of the Dropout layer in the classifier likely contributed to the model's strong generalization ability, preventing it from over-fitting to the 1,750 training images.

Potential Improvements

While the performance is already very strong, potential future improvements could include:

1. **Data Augmentation:** Implement more aggressive data augmentation techniques such as random rotations, shifting, or shearing during training. This would further increase the variability of the training data, making the model even more robust to different hand orientations and lighting conditions in real-world scenarios.
2. **Transfer Learning:** Instead of training a simple CNN from scratch, use a pre-trained state-of-the-art model (like ResNet or VGG) on a much larger dataset (e.g., ImageNet) and fine-tune its final layers. This is a common practice for smaller datasets and could potentially push the accuracy closer to 100%.