ML Lab Week 14: CNN Image Classification

Name	Dhanya Prabhu
SRN	PES2U23CS169
Section	CSE – C
Date	20/11/2025

- **Introduction:** The objective of this lab was to design and train a Convolutional Neural Network capable of classifying images of hand gestures into three categories: rock, paper, and scissors. We used the Rock–Paper–Scissors dataset from Kaggle, performed preprocessing, built a custom CNN using PyTorch, trained it for 10 epochs, and finally evaluated the model's performance on unseen images. The goal was to understand the full workflow of image classification—right from data loading to model evaluation and prediction

- **Model Architecture:**
  - **Describe the CNN architecture you built.**

    I built a CNN with three convolutional blocks followed by a fully connected classifier. Each block contains a convolution layer, a ReLU activation, and a MaxPooling layer. After these three blocks, the feature maps are flattened and passed through two linear layers to generate the final class scores.
  - **Mention key parameters like kernel size, number of channels, and the use of Max Pooling.**

    The first convolution layer uses 3 input channels and produces 16 output channels, the second converts 16 to 32 channels, and the third converts 32 to 64 channels. All convolution layers use a 3×3 kernel with padding set to 1. Each block has a MaxPooling layer with kernel size 2, which reduces the height and width by half at every stage.
  - **Describe the fully-connected classifier.**

    After the final MaxPooling layer, the feature map size becomes 64 × 16 × 16. This flattened output is passed into a linear layer with 256 units. A ReLU activation and dropout of 0.3 are applied. The final linear layer outputs 3 values corresponding to the three gesture classes: rock, paper, and scissors.

- **Training and Performance:**
  - **State the key hyperparameters used for training: optimizer, loss function, learning rate, and number of epochs.**

    The model was trained using the Adam optimizer with a learning rate of 0.001. The loss function used was CrossEntropyLoss. Training was done for 10 epochs with a batch size of 32.
  - **Report the final Test Accuracy your model achieved.**

My model achieved an accuracy of 98.4%

- ## Conclusion and Analysis:
  - **Briefly discuss your results. Did the model perform well?**

    The model performed very well. The loss consistently decreased across epochs, and the test accuracy of 98.4% indicates that the model learned to distinguish between the gestures effectively.
  - **Were there any challenges you faced?**

    I faced minor challenges related to ensuring that the dataset was copied correctly into the working directory and confirming the correct flattened size before the first fully connected layer. Once these were fixed, the training process worked smoothly.
  - **Suggest one or two ways you could potentially improve the model's accuracy in the future.**

    Accuracy could be improved by applying data augmentation techniques such as random flips, rotations, or brightness changes to make the model more robust. Another improvement would be to use a deeper architecture or a pretrained model like ResNet18 to extract richer features.

- ## Screenshots
  - **Model Summary:**

```
print(model)
```

```
RPS_CNN(
  (conv_block): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=16384, out_features=256, bias=True)
    (2): ReLU()
    (3): Dropout(p=0.3, inplace=False)
    (4): Linear(in_features=256, out_features=3, bias=True)
  )
)
```

  - **Test Accuracy:**

```
Test Accuracy: 98.40%
```

- **Loss:**

```
Epoch 1/10, Loss = 0.7501
Epoch 2/10, Loss = 0.2220
Epoch 3/10, Loss = 0.1150
Epoch 4/10, Loss = 0.0638
Epoch 5/10, Loss = 0.0415
Epoch 6/10, Loss = 0.0199
Epoch 7/10, Loss = 0.0123
Epoch 8/10, Loss = 0.0092
Epoch 9/10, Loss = 0.0029
Epoch 10/10, Loss = 0.0049
Training complete!
```

- **Model prediction:**

```
Model prediction for /content/dataset/paper/0Uomd0HvOB33m47I.png: paper
```

- **Final output ss:**

```
Randomly selected images:
Image 1: /content/dataset/paper/qGf97ZWPmGYzlTpo.png
Image 2: /content/dataset/scissors/hyQDSSjNFHX7Wirc.png

Player 1 shows: paper
Player 2 shows: scissors

RESULT: Player 2 wins! scissors beats paper
```

```
Randomly selected images:
Image 1: /content/dataset/scissors/M6PgmHZxElc7AKM5.png
Image 2: /content/dataset/rock/IQNb5XBjdbYa7A2W.png

Player 1 shows: scissors
Player 2 shows: rock

RESULT: Player 2 wins! rock beats scissors
```

```
Randomly selected images:
Image 1: /content/dataset/paper/vFCd43peE8nozYko.png
Image 2: /content/dataset/rock/337ARHTZmhCSkoEM.png

Player 1 shows: paper
Player 2 shows: rock

RESULT: Player 1 wins! paper beats rock
```

```
Randomly selected images:
Image 1: /content/dataset/scissors/HJ3qSJKzOvM3IwZR.png
Image 2: /content/dataset/scissors/jy6fSFQ1yneCl3P7.png

Player 1 shows: scissors
Player 2 shows: scissors

RESULT: Draw
```