Prakeerti Misra
2022VST9521

ASSIGNMENT-4

Q1- Part A

Convolutional Neural Network

Aim: To implement a simple CNN which takes as input book cover images and returns the genre label. Implement a Convolutional Neural Network with the following structure:

• CONV1: Kernel Size → 5x5, Input Size → 3, Output Size → 32

• POOL1 : Kernel Size → 2x2

• CONV2 : Kernel Size → 5x5, Input Size → 32, Output Size → 64

• POOL2 : Kernel Size → 2x2

• CONV3 : Kernel Size → 5x5, Input Size → 64, Output Size → 128

• POOL3 : Kernel Size → 2x2

• FC1 : Fully Connected Layer with 128 outputs

• FC2 : Fully Connected Layer with 30 outputs

Method:

- We start by writing some transformations and then we scale the images, convert them to tensors, and normalize them using the mean and standard deviation of each band in the input image.
- We then create two data loaders (for training/testing) and set the batch size.
- We start by creating a new class that extends the nn.Module class from PyTorch.
- To define the layers in our neural network we define the __init__ method of the class.
- We simply name our layers, and then assign them to the appropriate layer that we want; e.g., convolutional layer, pooling layer, fully connected layer, etc.
- The last thing to do is define a method directly in our class. The purpose of this method is to determine the order in which input data passes through different layers
- Then we train our model:
    i) We start by iterating through the number of epochs, and then the batches in our training data

ii) In the forward pass we make predictions using our model and calculate loss based on those predictions and our actual labels

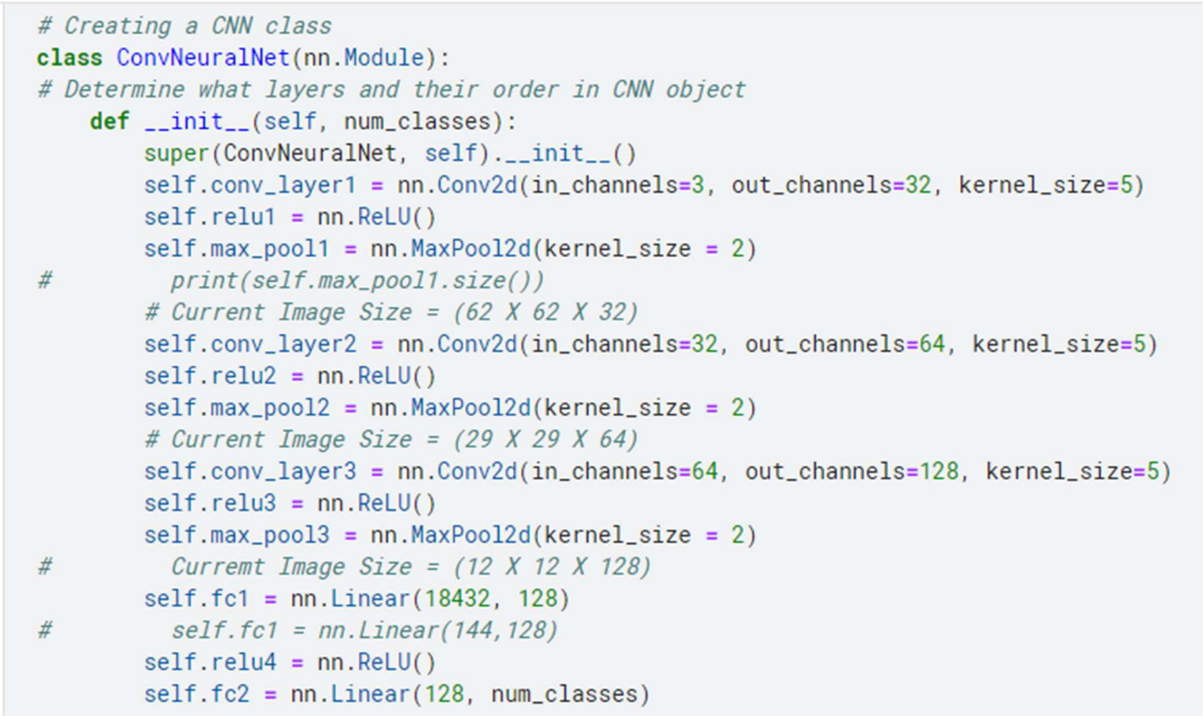
iii) Next, we do the backward pass where we actually update our weights to improve our model

iv) We then set the gradients to zero before every update using optimizer.zero_grad() function

v) And finally, we update the weights with the optimizer.step() function.

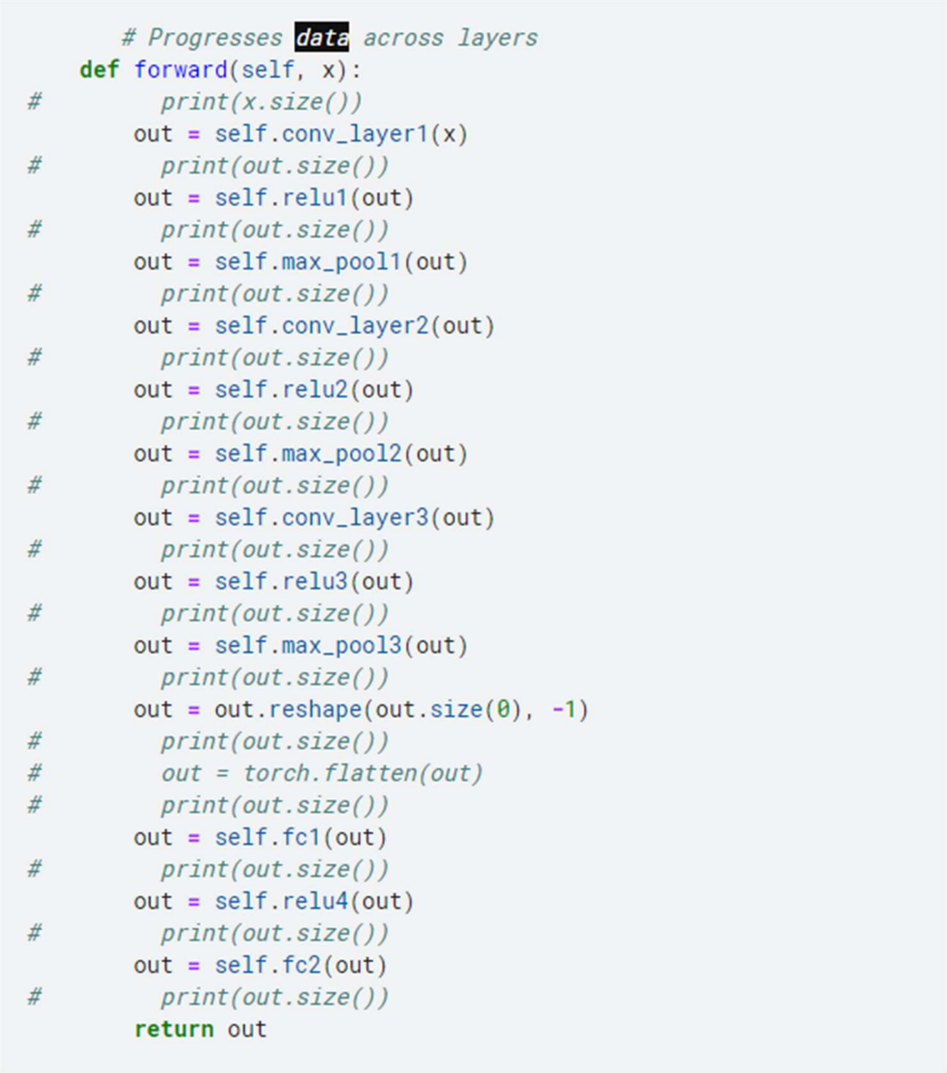
- We wrap the code inside torch.no_grad() as there is no need to calculate any gradients. We then predict each batch using our model and calculate how many it predicts correctly. We get the final result of ~75.5% test accuracy:

Model design:



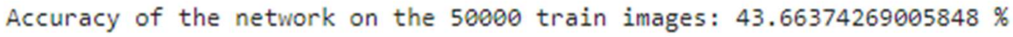
```
# Creating a CNN class
class ConvNeuralNet(nn.Module):
# Determine what layers and their order in CNN object
    def __init__(self, num_classes):
        super(ConvNeuralNet, self).__init__()
        self.conv_layer1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=5)
        self.relu1 = nn.ReLU()
        self.max_pool1 = nn.MaxPool2d(kernel_size = 2)
#         print(self.max_pool1.size())
        # Current Image Size = (62 X 62 X 32)
        self.conv_layer2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=5)
        self.relu2 = nn.ReLU()
        self.max_pool2 = nn.MaxPool2d(kernel_size = 2)
        # Current Image Size = (29 X 29 X 64)
        self.conv_layer3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=5)
        self.relu3 = nn.ReLU()
        self.max_pool3 = nn.MaxPool2d(kernel_size = 2)
#         Curremt Image Size = (12 X 12 X 128)
        self.fc1 = nn.Linear(18432, 128)
#         self.fc1 = nn.Linear(144,128)
        self.relu4 = nn.ReLU()
        self.fc2 = nn.Linear(128, num_classes)
```

Prakeerti Misra
2022VST9521



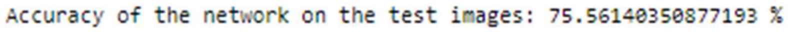
```
       # Progresses data across layers
    def forward(self, x):
#         print(x.size())
        out = self.conv_layer1(x)
#         print(out.size())
        out = self.relu1(out)
#         print(out.size())
        out = self.max_pool1(out)
#         print(out.size())
        out = self.conv_layer2(out)
#         print(out.size())
        out = self.relu2(out)
#         print(out.size())
        out = self.max_pool2(out)
#         print(out.size())
        out = self.conv_layer3(out)
#         print(out.size())
        out = self.relu3(out)
#         print(out.size())
        out = self.max_pool3(out)
#         print(out.size())
        out = out.reshape(out.size(0), -1)
#         print(out.size())
#         out = torch.flatten(out)
#         print(out.size())
        out = self.fc1(out)
#         print(out.size())
        out = self.relu4(out)
#         print(out.size())
        out = self.fc2(out)
#         print(out.size())
        return out
```

- The training set accuracy: 43.66374269005848 %



```
Accuracy of the network on the 50000 train images: 43.66374269005848 %
```

- The test set accuracy: 75.56140350877193 %



```
Accuracy of the network on the test images: 75.56140350877193 %
```