Introduction to Machine Learning and Artificial Neural Networks

Deep Neural Networks

22 December 2021

Part 1

In this homework I have created three convolutional. Below you can find my first Network. It has three convolutional layers.

```python
class Network1(nn.Module):
    # get input and output dimensions as input
    def __init__(self):
        # all derived classes must call __init__ method of super class
        super(Network1, self).__init__()

        self.conv1 = nn.Conv2d(
            in_channels=1,
            out_channels=16,
            kernel_size=3,
            stride=2,
            padding=1
        )

        # [(10 − 3 + 2) / 2] + 1 = 5.5 = 5

        self.conv2 = nn.Conv2d(
            in_channels=16,
            out_channels=32,
            kernel_size=3,
            stride=2,
            padding=1
        )

        # [(5 − 3 + 2) / 2] + 1 = 3

        self.conv3 = nn.Conv2d(
            in_channels=32,
            out_channels=64,
            kernel_size=3,
            stride=2,
            padding=1
        )

        # [(3 − 3 + 2) / 2] + 1 = 2

        self.fcl = nn.Linear(64*2*2, 10)
        self.model = nn.Sequential(
            self.conv1,
            self.conv2,
            self.conv3,
            self.fcl
        )

    # forward method should get the input and return the output
    def forward(self,x):
        x = torch.relu(self.conv1(x))
        x = torch.relu(self.conv2(x))
        x = torch.relu(self.conv3(x))
        x = torch.flatten(x, 1)
        x = self.fcl(x)
        return torch.log_softmax(x, dim=1)
```

Below is my second network that has 2 convolutional layers.

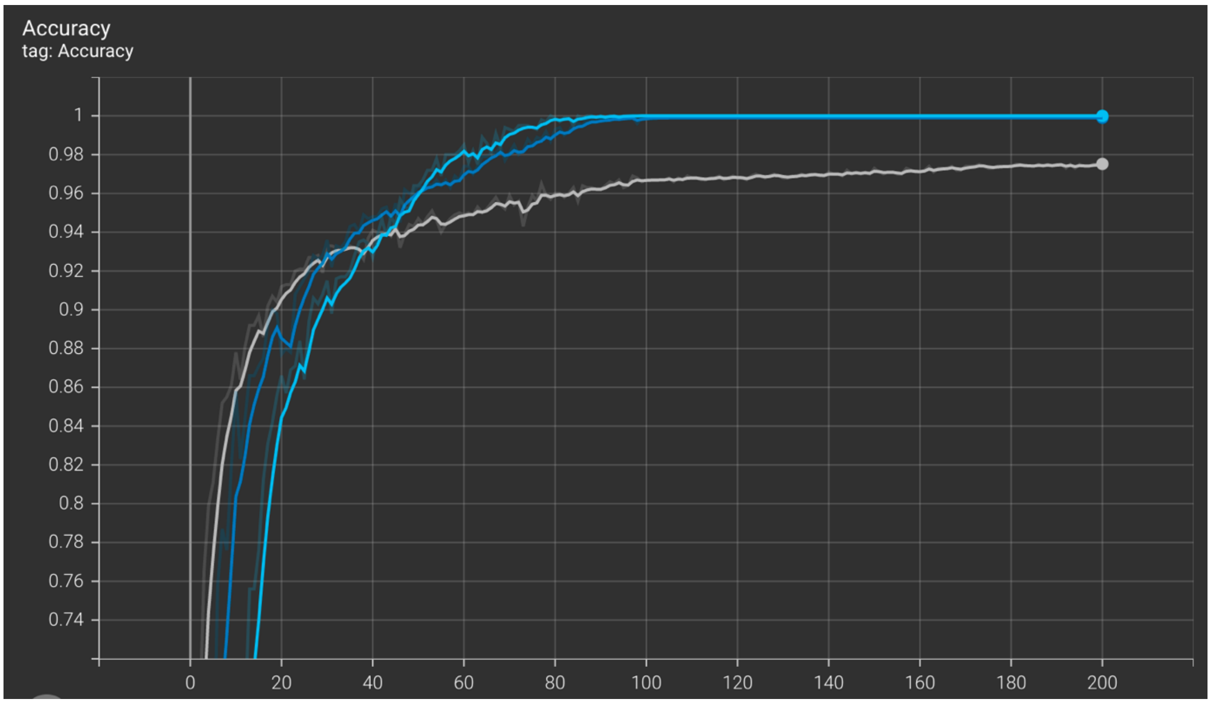
```
class Network2(nn.Module):
    # get input and output dimensions as input
    def __init__(self):
        # all derived classes must call __init__ method of super class
        super(Network2, self).__init__()

        self.conv1 = nn.Conv2d(
            in_channels=1,
            out_channels=16,
            kernel_size=3,
            stride=2,
            padding=1
        )

        # [(10 − 3 + 2) / 2] + 1 = 5.5 = 5

        self.conv2 = nn.Conv2d(
            in_channels=16,
            out_channels=32,
            kernel_size=3,
            stride=2,
            padding=1
        )

        # [(5 − 3 + 2) / 2] + 1 = 3

        self.fcl = nn.Linear(32*3*3, 10)
        self.model = nn.Sequential(
            self.conv1,
            self.conv2,
            self.fcl
        )

    # forward method should get the input and return the output
    def forward(self,x):
        x = torch.relu(self.conv1(x))
        x = torch.relu(self.conv2(x))
        x = torch.flatten(x, 1)
        x = self.fcl(x)
        return torch.log_softmax(x, dim=1)
```

Finally, my third network consists of a single convolutional layer.

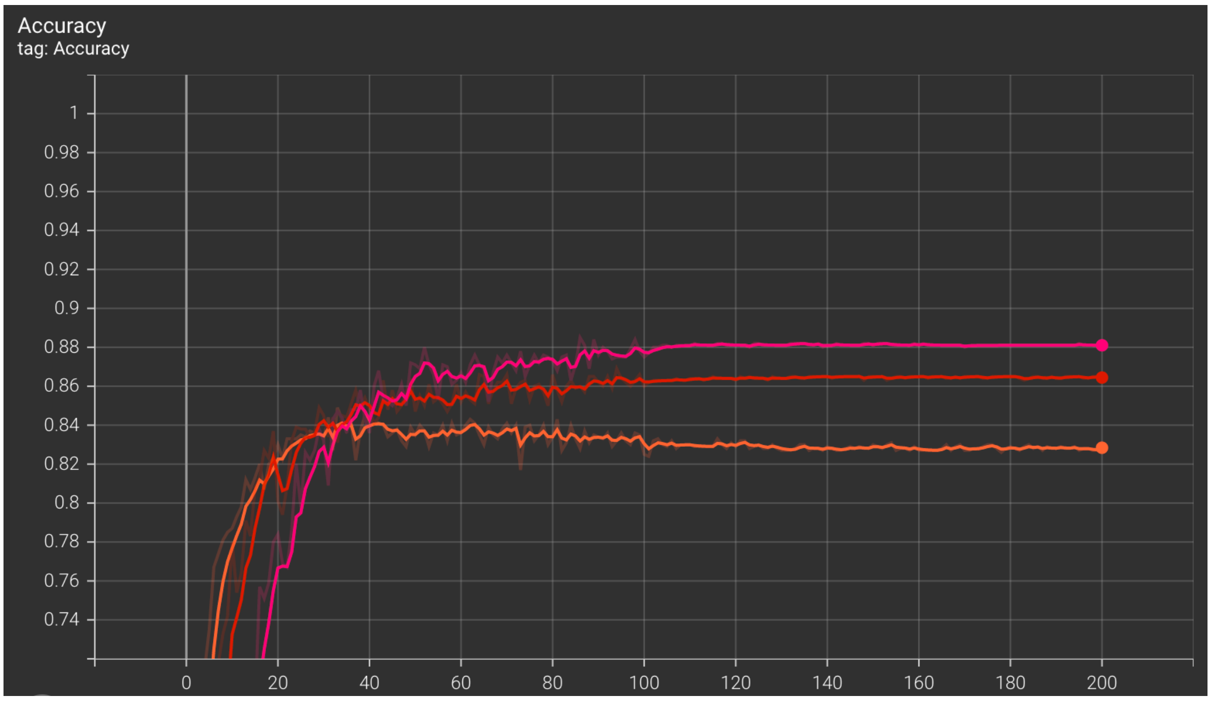
```
class Network3(nn.Module):
    # get input and output dimensions as input
    def __init__(self):
        # all derived classes must call __init__ method of super class
        super(Network3, self).__init__()

        self.conv1 = nn.Conv2d(
            in_channels=1,
            out_channels=16,
            kernel_size=3,
            stride=2,
            padding=1
        )

        # [(10 - 3 + 2) / 2] + 1 = 5.5 = 5

        self.fcl = nn.Linear(16*3*3, 10)
        self.model = nn.Sequential(
            self.conv1,
            self.fcl
        )

    # forward method should get the input and return the output
    def forward(self,x):
        x = torch.relu(self.conv1(x))
        x = torch.flatten(x, 1)
        x = self.fcl(x)
        return torch.log_softmax(x, dim=1)
```

Then I have trained the network and below you can see the results of accuracy for training and testing datasets.

Below is the accuracy for the training on all three networks. Light Blue represents network 1 (3 conv), Dark Blue represents network 2 (2 conv) and White represents network 3 (1 conv).



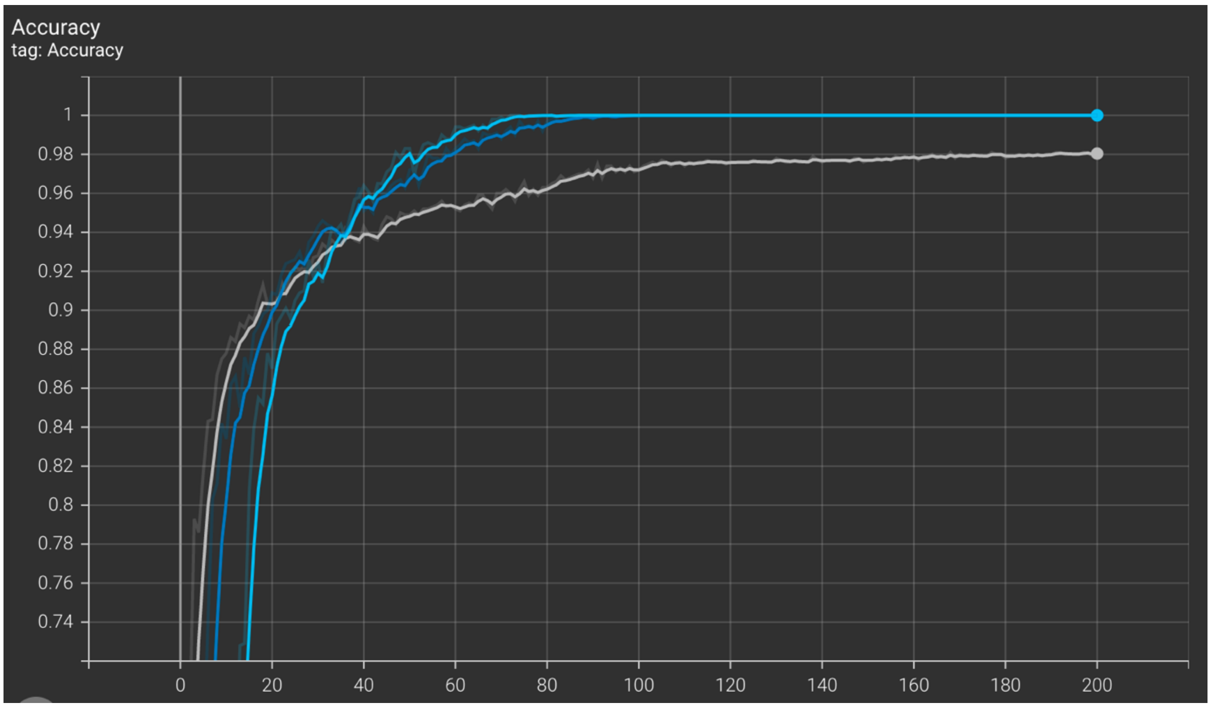
Below is the accuracy for the testing on all three networks. Purple network 1, Red network 2, Orange network 3.



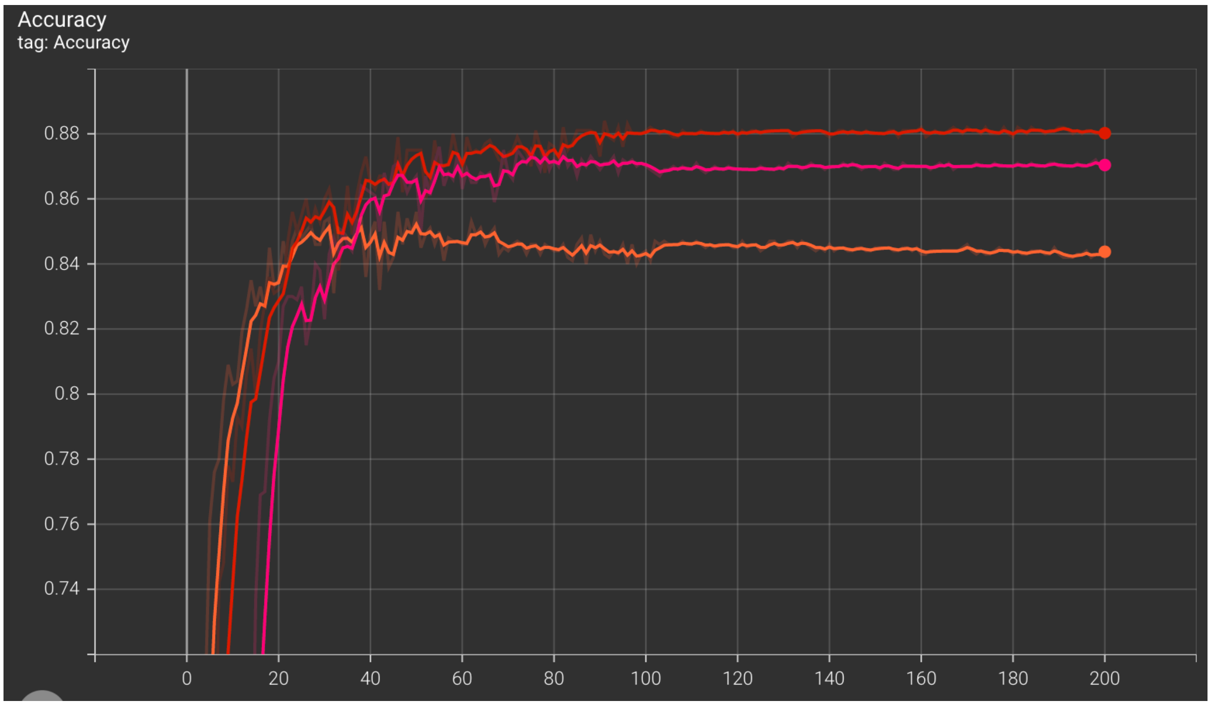
Part 2

In this part I made modifications and changed kernel size = 4, stride = 2, and padding= 2.

Below is the accuracy for the training on all three networks. Light Blue represents network 1 (3 conv), Dark Blue represents network 2 (2 conv) and White represents network 3 (1 conv).

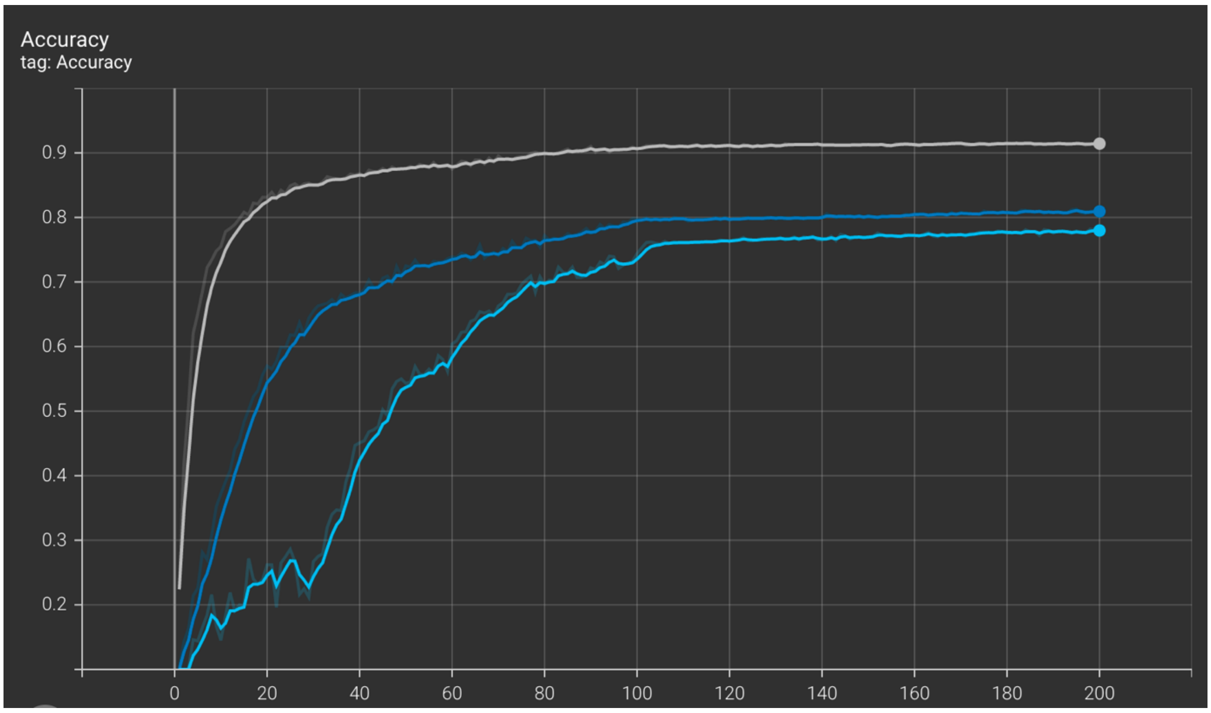


Below is the accuracy for the testing on all three networks. Purple network 1, Red network 2, Orange network 3.

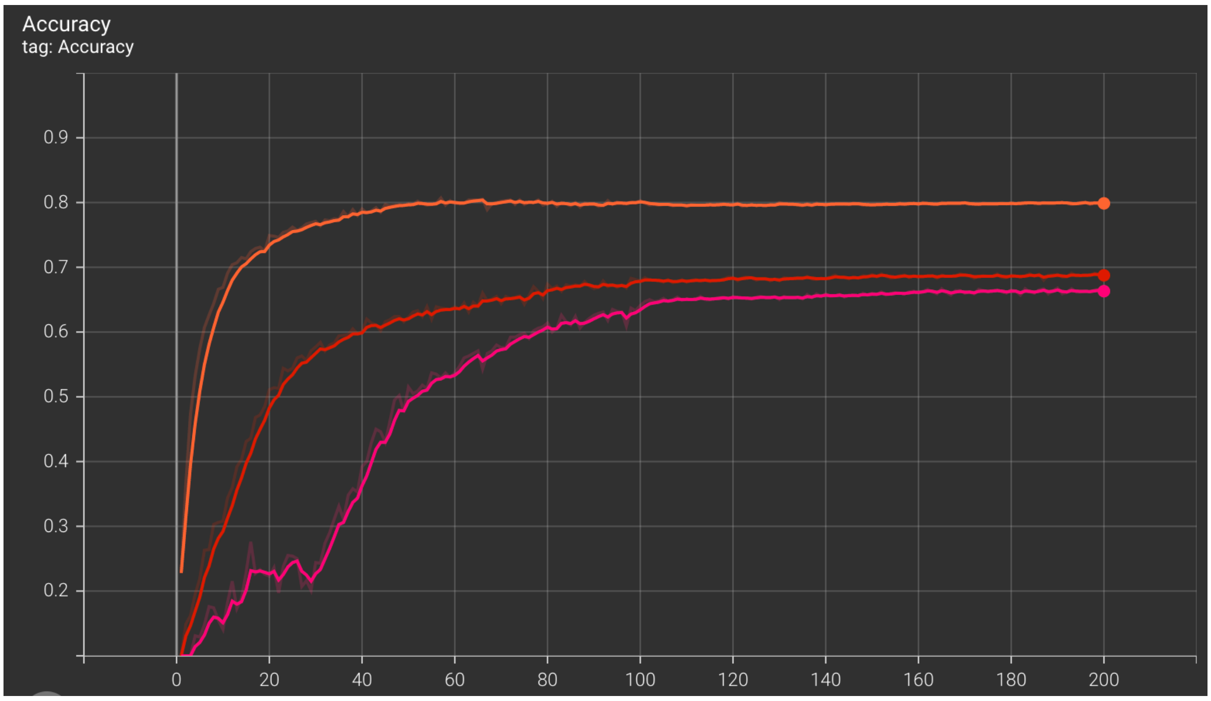


Finally, I tested with kernel size = 3, stride = 4, and padding = 1. As you can see the results have changed completely since the number of the stride is higher than our kernel size. In this case the accuracy has decreased since during the training process some of the pixels were omitted for training. Network 3 showed best accuracy rate followed by Network 2 then Network 1.

Below is the accuracy for the training on all three networks. Light Blue represents network 1 (3 conv), Dark Blue represents network 2 (2 conv) and White represents network 3 (1 conv).



Below is the accuracy for the testing on all three networks. Purple network 1, Red network 2, Orange network 3.


Accuracy
tag: Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140
160
180
200