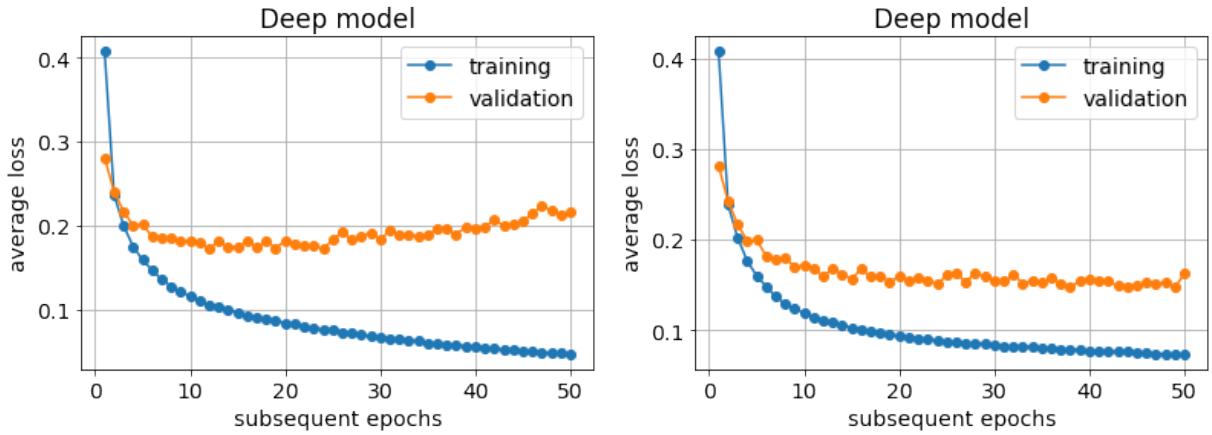
Handwritten digits classification using MNIST dataset with Pytorch

Mateusz Bulanda - Gorol

June 2022

1 Apply some regularization technique to Deep model in order to avoid overfitting (start with nonzero weight decay in optimizer).

To avoid overfitting I have add *weight_decay* parameter in optimizer. The best results I got to $weight\_decay = 1.e{-}3$.

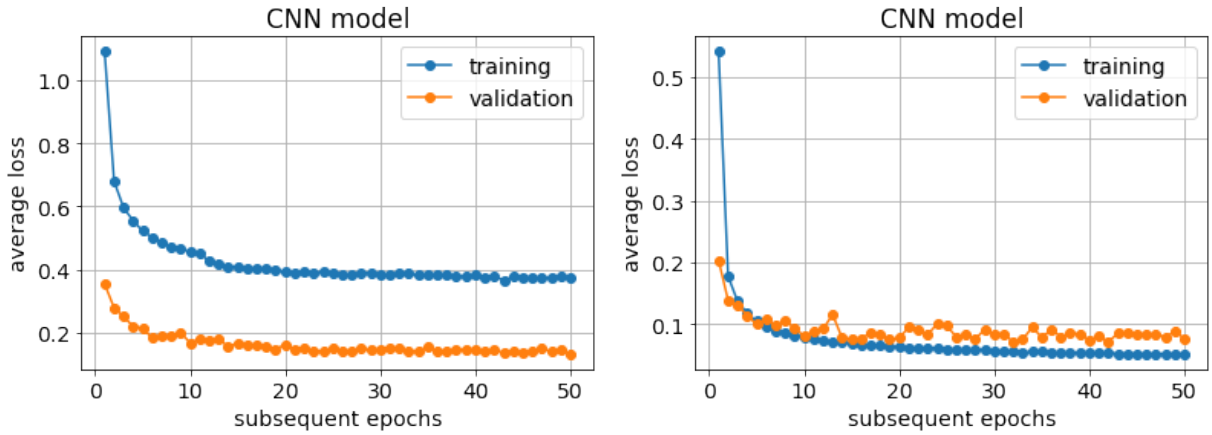


without regularization $with\ weight\_decay = 1.e-3$

Figure 1: Results of average trainig and validation loss for Deep model.

2 Try to figure out why the validation loss for CNN model turns to be lower than the train loss (hint: turn off regularization).

For standard CNN model I got avarage trainning loss equal $0,3720$ and validation loss eqeual $0,1310$. After removing dropout training losses I got trainning loss equal $0,0513$ and validation loss equal $0,0766$. The trainning loss increases because we include regularization in trainning loss calculations but not in the validation loss calculations. When we turn off regularization the trainning loss decresses.



with regularization *without regularization*

Figure 2: Results of average trainig and validation loss for CNN model.

3 Tune one of these models to get the Test Set Accuracy greater than 99%.

New CNN setup:

```
class CNN(nn.Module):
    # this defines the structure of the CNN model
    def __init__(self):
        super(CNN, self).__init__()
        # convolutional layer with 16 kernels of size 5x5
        self.conv1 = nn.Conv2d(1, 16, kernel_size=5)
        # 512 kernels of size 5x5
        self.conv2 = nn.Conv2d(16, 32, kernel_size=5)
        # 2D dropout
        self.conv2_drop = nn.Dropout2d()
        # fully connected layers
        self.fc1 = nn.Linear(512, 100)
        self.fc2 = nn.Linear(100, 10)

    def forward(self, x):
        # 1st layers group
        x = self.conv1(x) # resulting in 2 feature maps each of size 24x24
        x = F.max_pool2d(x, 2) # downsizing each map to 12x12
        x = F.relu(x) # standard (in CNNs) ReLU activation
        # 2nd group
        x = self.conv2(x) # resulting in 4 feature maps each of size 8x8
        # x = self.conv2_drop(x)
        x = F.max_pool2d(x, 2) # downsizing each map to 4x4
        x = F.relu(x)
        # fully connected layers
        x = x.view(-1, 512) # 4 maps of 4x4 size gives 64 numbers
        x = self.fc1(x) # 64 -> 20
        x = F.relu(x)
        x = F.dropout(x, training=self.training) # dropout is a type of regularization
        x = self.fc2(x) # 20 -> 10
        # softmax (multinomial classification) gives probabilities of each class
        return F.log_softmax(x, dim=1) # note that dim=0 is the number of samples in batch
```

Test set accuracy: 9934/10000(99

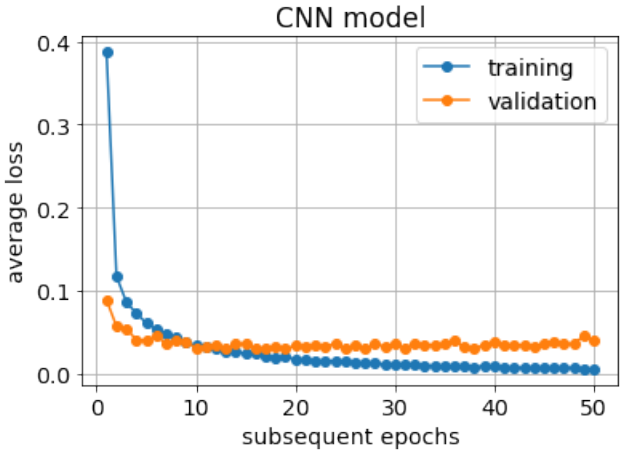


Figure 3: Results of average trainig and validation loss for modified CNN model.

4 Plot the confusion matrix among all of the classes—which of digits are mostly confused with each other?

Mostly confused with each other are pairs: $(5,3), (8,3), (8,5), (2,3)(4,9)$

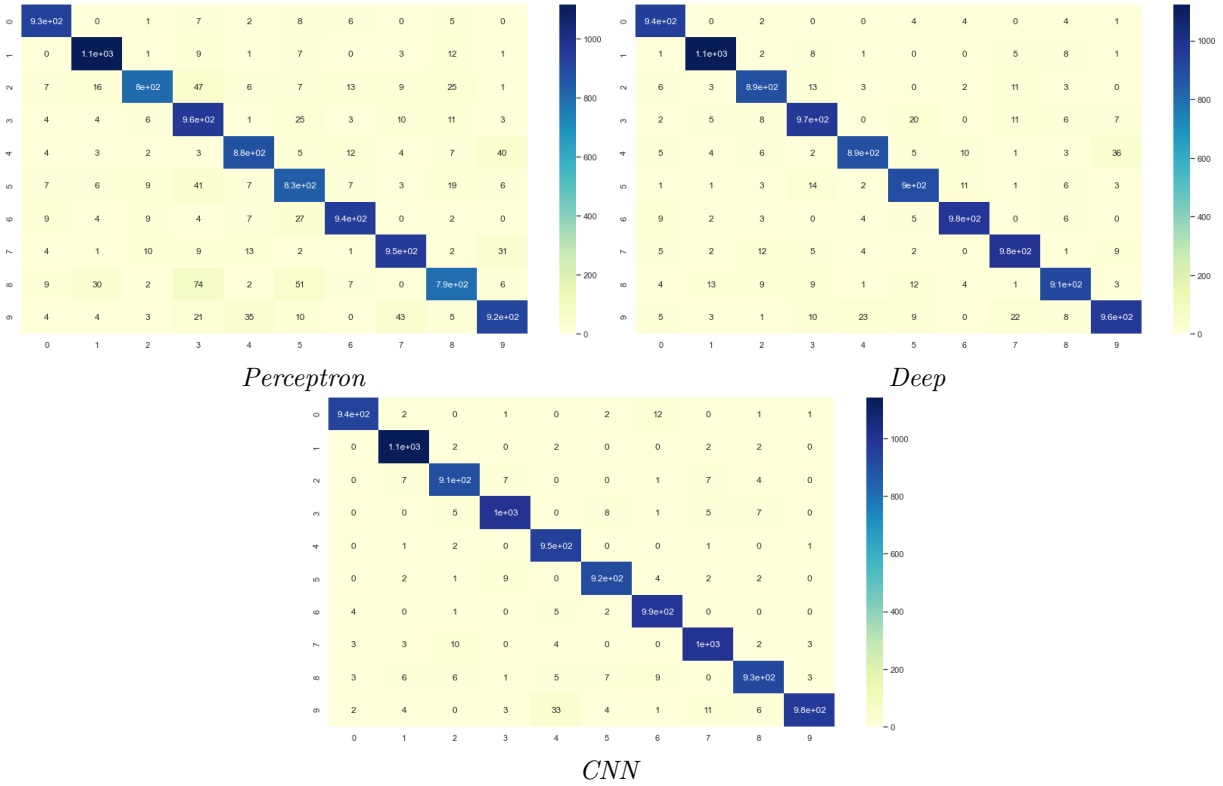


Figure 4: Confusion matrixes for all models.