Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

INTELLIGENT SYSTEMS DESIGN LABORATORY
(EC69210)

Experiment - 2

Ashutosh Naik (20EC39049)
Arman Atibudhi (20EC39048)

Model: Resnet 50

```
def conv_block(in_channels, out_channels, pool=False):
    layers = [
                nn.Conv2d(in_channels, out_channels, kernel_size=3,
padding=1),
                nn.BatchNorm2d(out_channels),
                nn.ReLU(inplace=True)
    ]
    if pool: layers.append(nn.MaxPool2d(2))
    return nn.Sequential(*layers)



class Net(nn.Module):
    def __init__(self):
        super().__init__()



        self.conv1 = conv_block(1, 64)
        self.conv2 = conv_block(64, 128, pool=True)
        self.res1 = nn.Sequential(conv_block(128, 128), conv_block(128,
128))



        self.conv3 = conv_block(128, 256, pool=True)
        self.conv4 = conv_block(256, 512, pool=True)
```

```
        self.res2 = nn.Sequential(conv_block(512, 512), conv_block(512,
512))



        self.classifier = nn.Sequential(nn.MaxPool2d(2),
                                        nn.Flatten(),
                                        nn.Dropout(0.2),
                                        nn.Linear(512, 10),
                                        nn.Softmax(dim=1))
    def forward(self,x):
        out=self.conv1(x)
        out=self.conv2(out)
        out=self.res1(out)+out
        out=self.conv3(out)
        out=self.conv4(out)
        out=self.res2(out)+out
        out=self.classifier(out)
        return out
```

Optimizer:

```
criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Part 1:

```
Accuracy of the network on the 10000 test images: 99 %
[10,  2000] loss: 1.465
[10,  4000] loss: 1.464
[10,  6000] loss: 1.464
[10,  8000] loss: 1.465
[10, 10000] loss: 1.466
[10, 12000] loss: 1.465
[10, 14000] loss: 1.466
Accuracy of the network on the 10000 test images: 99 %
Finished Training
```

Part 2:

- Used Hindi language to create the custom dataset of 1000 images. 80 training images for each digit and 20 testing images for each digit.
- The dataset is divided into 2 zip files *train_data.zip* and *test_data.zip*

```
Epoch: 0
Accuracy of the network on the 200 test images: 35 %
Epoch: 1
Accuracy of the network on the 200 test images: 64 %
Epoch: 2
Accuracy of the network on the 200 test images: 86 %
Epoch: 3
Accuracy of the network on the 200 test images: 91 %
Epoch: 4
Accuracy of the network on the 200 test images: 92 %
Epoch: 5
Accuracy of the network on the 200 test images: 91 %
Epoch: 6
Accuracy of the network on the 200 test images: 94 %
Epoch: 7
Accuracy of the network on the 200 test images: 94 %
Epoch: 8
Accuracy of the network on the 200 test images: 92 %
Epoch: 9
Accuracy of the network on the 200 test images: 95 %
Finished Training
```

Part 3:

- Augmentation includes random flipping in horizontal and vertical directions and rotation in any angle possible.

The following addition in code will happen:

```
# Define transformations

transform = transforms.Compose([

    transforms.Resize((28, 28)),

    transforms.RandomHorizontalFlip(),

    transforms.RandomVerticalFlip(),

    transforms.RandomRotation(180),

    transforms.ToTensor(),

    transforms.Normalize((0.5,), (0.5,))

])
```

```
Epoch: 0
Accuracy of the network on the 200 test images: 93 %
Epoch: 1
Accuracy of the network on the 200 test images: 93 %
Epoch: 2
Accuracy of the network on the 200 test images: 94 %
Epoch: 3
Accuracy of the network on the 200 test images: 94 %
Epoch: 4
Accuracy of the network on the 200 test images: 93 %
Epoch: 5
Accuracy of the network on the 200 test images: 94 %
Epoch: 6
Accuracy of the network on the 200 test images: 94 %
Epoch: 7
Accuracy of the network on the 200 test images: 95 %
Epoch: 8
Accuracy of the network on the 200 test images: 96 %
Epoch: 9
Accuracy of the network on the 200 test images: 93 %
Finished Training
```

Part 4:

The following changes will happen in the code from part 2:

```
model_params_file_path =
"/content/drive/MyDrive/checkpoints/best_model.pth"



net = Net().to(device)

checkpoint = torch.load(model_params_file_path,
map_location=torch.device('cpu'))  # Adjust map_location if using GPU

net.load_state_dict(checkpoint['model_state_dict'])

#Freeze the parameters

for param in net.parameters():

    param.requires_grad = False



# Add a fully connected layer at the end

num_ftrs = net.classifier[3].in_features

net.classifier[3] = nn.Linear(num_ftrs, 10)

import torch.optim as optim



criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.classifier[3].parameters(), lr=0.001,
momentum=0.9)
```

```
Epoch: 0
Accuracy of the network on the 200 test images: 67 %
Epoch: 1
Accuracy of the network on the 200 test images: 68 %
Epoch: 2
Accuracy of the network on the 200 test images: 63 %
Epoch: 3
Accuracy of the network on the 200 test images: 62 %
Epoch: 4
Accuracy of the network on the 200 test images: 67 %
Epoch: 5
Accuracy of the network on the 200 test images: 69 %
Epoch: 6
Accuracy of the network on the 200 test images: 65 %
Epoch: 7
Accuracy of the network on the 200 test images: 59 %
Epoch: 8
Accuracy of the network on the 200 test images: 68 %
Epoch: 9
Accuracy of the network on the 200 test images: 71 %
Finished Training
```

At 140 epochs

Observations:

- The MNIST dataset was trained in **2 epochs** to an accuracy of **98%**. It was possibly due to the larger size (60000 images) of the dataset.
- Our custom dataset took **10 epochs**, a lot more than MNIST dataset to become >90% accurate (**95%** accuracy). It is possibly due to less number of images (800 training images).
- Data augmentation further increases the training time **(400 epochs**) to reach a point of reasonable accuracy (**96%**) . It is because existing features may not be enough to label new images.
- While Transfer Learning increases the efficiency by reducing the computation time, the accuracy is reduced drastically to a maximum of **70%**. Reached within **50** epochs.