Assignment 5

Basic CNN:

Model:

In the architecture, 2 convolutional layers are used, followed by 3 fully connected layers. Third layer provides the output. Used Relu for all the layers. Last layer output param should be 101 that's the classes. Used CrossEntropyLoss.

The following code was used:

```
class basic_cnn(pl.LightningModule):
    def __init__(self):
        # Define Model Here
        super(basic_cnn,self).__init__()
        #First 2d convolutional layer
        # 1 is input channel 32 is outputing feature size, 3 is kernel size
        self.conv1 = nn.Conv2d(3, 32, 3)
        # Se 2d convolutional layer, takes 32 input from previous layer
        self.conv2 = nn.Conv2d(32, 64, 3)

        # First fully connected layer
        self.fc1 = nn.Linear(57600,256)
        # Second fully connected layer
        self.fc2 = nn.Linear(256,128)
        # Third fully connect layer that outputs 101 labels
        self.out = nn.Linear(128,101)
        self.lr = 0.01
        self.loss = nn.CrossEntropyLoss()
        #32x57600 and 512x256


    def forward(self, x):
        # Define Forward Pass Here
        batch_size, _, _, _ = x.size()
        #print(x.size())

        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = x.view(batch_size,-1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return self.out(x)
```

Preparing the data:

The following code was used to prepare data:

```
from pl_bolts.transforms.dataset_normalizations import cifar10_normalization

class Food101(pl.LightningDataModule):
    def prepare_data(self):
        transform=transforms.Compose([
            transforms.RandomResizedCrop(128),
            transforms.RandomCrop(64),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.5, 0.5, 0.5],
                                 [0.229, 0.224, 0.225])])
            #cifar10_normalization()

```

Results:

Training Loss: 4.3747

Validation Loss: 2.9199

Evaluating test data:

Below image shows accuracy and loss results for basic cnn on testing set:

```
trainer.test(clf_basic_cnn, Food101)
```

```
[INFO] loading the FOOD 101 dataset...
[INFO] data loaded successfully
C:\Users\aadit\anaconda3\envs\cse6363\lib\site-packages\pytorch_lightning\trainer\connectors\data_connector.py:466: PossibleUserWarning: Your `test_dataloader`'s sampler has shuffling enabled, it is strongly recommended that you turn shuffling off for val/test/predict dataloaders.
  rank_zero_warn(
Testing DataLoader 0: 100% 790/790 [01:21<00:00, 9.64it/s]
```

Test metric	DataLoader 0
test_accuracy	3.869307041168213
test_loss	4.420573711395264

```
]: [{'test_accuracy': 3.869307041168213, 'test_loss': 4.420573711395264}]
```

All Convolutional Net:

Model:

In the architecture, 3 convolutional layers are used, followed by a pooling layer. Flatten layer provides the output. Used Relu for all the conv2d layers. Last layer output param should be 101 that's the classes. Used CrossEntropyLoss.

The following code was used:

```
class all_conv_net(pl.LightningModule):
    def __init__(self):
        # Define Model Here
        super(all_conv_net,self).__init__()
        #First 2d convolutional layer
        # 1 is input channel 32 is outputing feature size, 3 is kernel size
        self.conv1 = nn.Conv2d(3, 32, 3)
        # Se 2d convolutional layer, takes 32 input from previous layer
        self.conv2 = nn.Conv2d(32, 64, 3)

        self.conv3 = nn.Conv2d(64, 101, 3)
        # First fully connected layer
        #self.fc1 = nn.Linear(57600,256)
        # Second fully connected layer
        #self.fc2 = nn.Linear(256,128)
        # Third fully connect layer that outputs 101 labels
        #self.out = nn.Linear(128,101)
        self.lr = 0.01
        self.loss = nn.CrossEntropyLoss()
        #32x57600 and 512x256


    def forward(self, x):
        # Define Forward Pass Here
        batch_size, _, _, _ = x.size()
        #print(x.size())

        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = F.relu(self.conv3(x))
        x = F.max_pool2d(x, 2)
        #x = x.view(batch_size,-1)
        #x = F.relu(self.fc1(x))
        #x = F.relu(self.fc2(x))
        x = x.flatten(1)
        return x
```

Preparing the data:

The same data is used from the basic CNN model.

Results:

Training Loss: 6.7058

Validation Loss: 6.6397

Evaluating test data:

Below image shows accuracy and loss results for all convolutional network on testing set:

```
7]:  1 trainer.test(clf_all_conv_net, Food101)

[INFO] loading the FOOD 101 dataset...
[INFO] data loaded successfully

Testing DataLoader 0: 100% 790/790 [02:29<00:00, 5.27it/s]

──────────────────────────────────────────────────────────────────────────────
       Test metric             DataLoader 0
──────────────────────────────────────────────────────────────────────────────
      test_accuracy         0.23366336524486542
        test_loss            6.738712787628174
──────────────────────────────────────────────────────────────────────────────

t[17]: [{'test_accuracy': 0.23366336524486542, 'test_loss': 6.738712787628174}]
```

Regularization:

In the architecture, 2 convolutional layers are used, followed by 3 fully connected layers. Third layer provides the output. Used Relu for all the layers. Added 2 dropout layers with 25% dropout. Last layer output param should be 101 that's the classes. Used CrossEntropyLoss.

Model: The following code was used:

```
class reg(pl.LightningModule):
    def __init__(self):
        # Define Model Here
        super(reg,self).__init__()
        #First 2d convolutional layer
        # 1 is input channel 32 is outputing feature size, 3 is kernel size
        self.conv1 = nn.Conv2d(3, 32, 3)
        # Se 2d convolutional layer, takes 32 input from previous layer
        self.conv2 = nn.Conv2d(32, 64, 3)
        # First fully connected layer
        self.fc1 = nn.Linear(57600,256)
        # Second fully connected layer
        self.fc2 = nn.Linear(256,128)
        # Third fully connect layer that outputs 101 labels
        self.out = nn.Linear(128,101)

        #Define proportion or neurons to dropout
        self.dropout = nn.Dropout(0.25)


        self.lr = 0.01
        self.loss = nn.CrossEntropyLoss()
        #32x57600 and 512x256


    def forward(self, x):
        # Define Forward Pass Here
        batch_size, _, _, _ = x.size()
        #print(x.size())

        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = x.view(batch_size,-1)
        x = self.dropout(x)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = F.relu(self.fc2(x))
        return self.out(x)
```

Preparing the data:

Applied RandAugment function to augment the data. Added 2 dropout layers with 25% dropout. The following code was used to prepare data:

```

class aug_Food101(pl.LightningDataModule):
    def prepare_data(self):
        transform=transforms.Compose([
            transforms.RandomResizedCrop(128),
            transforms.RandAugment(),
            transforms.RandomCrop(64),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.5, 0.5, 0.5],
                                 [0.229, 0.224, 0.225])])
            #cifar10_normalization()

```

Results:

Training Loss: 4.6841

Validation Loss: 4.5177

Evaluating test data:

Below image shows accuracy and loss results after applying regularization to basic cnn on testing set:

```
: 1 trainer.test(clf_reg, aug_Food101)
[INFO] loading the FOOD 101 dataset...
[INFO] data loaded successfully

Testing DataLoader 0: 100% 790/790 [01:02<00:00, 12.55it/s]

       Test metric             DataLoader 0
      test_accuracy         2.8039603233337402
        test_loss            4.498032569885254

[22]: [{'test_accuracy': 2.8039603233337402, 'test_loss': 4.498032569885254}]
```

Transfer Learning:

Model:

Used the pre-trained Resnet18 model. Resnet 50 is a very good model for image classification but as the model is very big takes a lot of time and resources to train the backbone. So I took the resnet18 which has less number of layers which decreased training time by good extent. The following code was used to get the pre-trained model.

```
        # init a pretrained resnet
        backbone = models.resnet18(pretrained=True)
        num_filters = backbone.fc.in_features
        layers = list(backbone.children())[:-1]
        self.feature_extractor = nn.Sequential(*layers)

        # use the pretrained model to classify food101
        num_target_classes = 101
        self.classifier = nn.Linear(num_filters, num_target_classes)

    def forward(self, x):
        self.feature_extractor.eval()
        with torch.no_grad():
            representations = self.feature_extractor(x).flatten(1)
        x = self.classifier(representations)

        return x
```

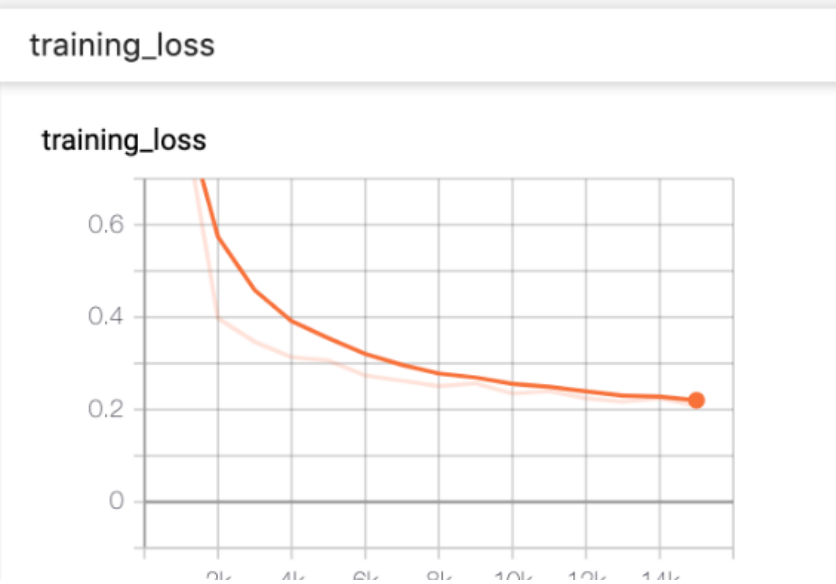
Preparing the data:

Changed the input to RandomCrop function in transforms to 56. The following code was used to prepare data:

```
        self.batch_size = batch_size
        self.lr = lr

        # Data preparation
        dataset = Food101(data_path, transform=transforms.Compose([
            transforms.RandAugment(),
            transforms.RandomCrop(56),
            transforms.ToTensor(),
        ]), download=True)

        dataset_size = len(dataset)
        train_size = int(dataset_size * .95)
        val_size = dataset_size - train_size

        self.train_dataset, self.val_dataset = torch.utils.data.random_split(dataset, [train_size, val_size])

        # Loss function
        self.loss_fn = nn.CrossEntropyLoss()
```

Results:

Training Loss:



Validation Loss: 4.43

Evaluating test data:

Below image shows accuracy and loss results for resnet18 model used for transfer learning on testing set:

```
1 trainer.test(model)
```

```
Testing DataLoader 0: 100% 119/119 [00:11<00:00, 10.69it/s]
```

Test metric	DataLoader 0
test_accuracy	4.963041305541992
test_loss	4.55490255355835

```
66]: [{'test_accuracy': 4.963041305541992, 'test_loss': 4.55490255355835}]
```

Bonus:

Model:

Used only 1 conv2d layer in the model. The following code was used for defining model:

```
class bonus(pl.LightningModule):
    def __init__(self):
        # Define Model Here
        super(bonus,self).__init__()
        #First 2d convolutional layer
        # 1 is input channel 32 is outputing feature size, 3 is kernel size
        #self.conv1 = nn.Conv2d(3, 32, 3)
        # Se 2d convolutional layer, takes 32 input from previous layer
        #self.conv2 = nn.Conv2d(32, 64, 3)

        self.conv1 = nn.Conv2d(3, 101, 3)
        # First fully connected layer
        #self.fc1 = nn.Linear(57600,256)
        # Second fully connected layer
        #self.fc2 = nn.Linear(256,128)
        # Third fully connect layer that outputs 101 labels
        #self.out = nn.Linear(128,101)
        self.lr = 0.01
        self.loss = nn.CrossEntropyLoss()
        #32x57600 and 512x256


    def forward(self, x):
        # Define Forward Pass Here
        batch_size, _, _, _ = x.size()
        #print(x.size())

        x = F.relu(self.conv1(x))
        #x = F.relu(self.conv2(x))
        #x = F.relu(self.conv3(x))
        x = F.max_pool2d(x, 2)
        #x = x.view(batch_size,-1)
        #x = F.relu(self.fc1(x))
        #x = F.relu(self.fc2(x))
        x = x.flatten(1)
        return x
```

Preparing the data:

Changed the input to RandomCrop function in transforms to 56. The following code was used to prepare data:

```python
class Food101_bonus(pl.LightningDataModule):
    def prepare_data(self):
        transform=transforms.Compose([
            transforms.RandomResizedCrop(128),
            transforms.RandomCrop(56),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.5, 0.5, 0.5],
                                 [0.229, 0.224, 0.225])])
            #cifar10_normalization()

```

Results:

Training Loss: 6.6256

Validation Loss: 6.5264

Evaluating test data:

Below image shows accuracy and loss results for shortest model created for bonus task on testing set:

```
: 1 trainer.test(clf_bonus, Food101_bonus)
[INFO] loading the FOOD 101 dataset...
[INFO] data loaded successfully

Testing DataLoader 0: 100% 790/790 [01:11<00:00, 11.05it/s]

       Test metric             DataLoader 0
      test_accuracy         0.22574257850646973
        test_loss            6.749154567718506

[74]: [{'test_accuracy': 0.22574257850646973, 'test_loss': 6.749154567718506}]

: 1
```

References:

https://www.geeksforgeeks.org/training-neural-networks-using-pytorch-lightning/