Rutuja Dukhande
1001730748
CSE 6363: Machine Learning

Assignment 5

Image classification with Image Learning

1) A Basic CNN

- **Transformers**

```
#Transforms
transformer=transforms.Compose([
    transforms.Resize((150,150)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),  #0-255 to 0-1, numpy to tensors
    transforms.Normalize([0.5,0.5,0.5], # 0-1 to [-1,1] , formula (x-mean)/std
                         [0.5,0.5,0.5])
])
```

- **Data Loaders:**

It helps in reading the data and training in batches.

```
#Dataloader

#Path for training and testing directory
train_path= r'train'
test_path= r'test'

train_loader=DataLoader(
    torchvision.datasets.ImageFolder(train_path,transform=transformer),
    batch_size=64, shuffle=True
)
test_loader=DataLoader(
    torchvision.datasets.ImageFolder(test_path,transform=transformer),
    batch_size=32, shuffle=True
)
```

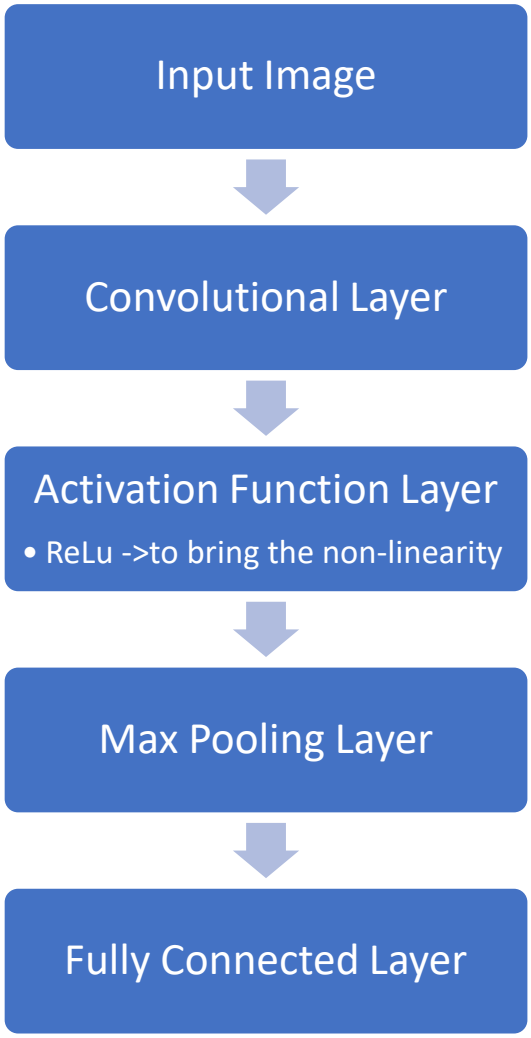
- **Hyperparameters:**

```
num_epochs=10
```

```
class ConvNet(nn.Module):
    def __init__(self,num_classes=6):
        super(ConvNet,self).__init__()

        #Output size after convolution filter
        #((w-f+2P)/s) +1

        #Input shape= (256,3,150,150)

        self.conv1=nn.Conv2d(in_channels=3,out_channels=12,kernel_size=3,stride=1,padding=1)
        #Shape= (256,12,150,150)
        self.bn1=nn.BatchNorm2d(num_features=12)
        #Shape= (256,12,150,150)
        self.relu1=nn.ReLU()
        #Shape= (256,12,150,150)

        self.pool=nn.MaxPool2d(kernel_size=2)
        #Reduce the image size be factor 2
        #Shape= (256,12,75,75)
```

```
self.pool=nn.MaxPool2d(kernel_size=2)
#Reduce the image size be factor 2
#Shape= (256,12,75,75)


self.conv2=nn.Conv2d(in_channels=12,out_channels=20,kernel_size=3,stride=1,padding=1)
#Shape= (256,20,75,75)
self.relu2=nn.ReLU()
#Shape= (256,20,75,75)



self.conv3=nn.Conv2d(in_channels=20,out_channels=32,kernel_size=3,stride=1,padding=1)
#Shape= (256,32,75,75)
self.bn3=nn.BatchNorm2d(num_features=32)
#Shape= (256,32,75,75)
self.relu3=nn.ReLU()
#Shape= (256,32,75,75)


self.fc=nn.Linear(in_features=75 * 75 * 32,out_features=num_classes)
```

- **Architecture:**



- **Training Loss, Validation Loss, Testing Accuracy**

```
100%|██████████| 10/10 [1:56:00<00:00, 696.10s/it]Epoch: 9 Train Loss: tensor(4.6541) Train Accuracy: 0.0099009900990099901

Validation Loss: tensor(1.5478) Test Accuracy: 0.009267326732673267
```

2) All Convolutional Net

- **Transformers**

```
#Transforms
transformer=transforms.Compose([
    transforms.Resize((150,150)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),  #0-255 to 0-1, numpy to tenso Loading...
    transforms.Normalize([0.5,0.5,0.5], # 0-1 to [-1,1] , formula (x-mean)/std
                         [0.5,0.5,0.5])
])
```

- **Data Loaders:**

It helps in reading the data and training in batches.

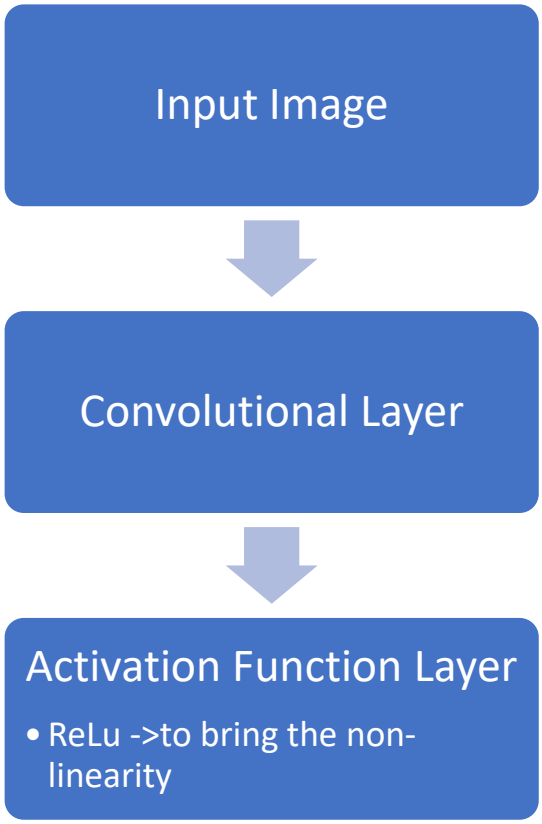
```
#Dataloader

#Path for training and testing directory
train_path= r'train'
test_path= r'test'

train_loader=DataLoader(
    torchvision.datasets.ImageFolder(train_path,transform=transformer),
    batch_size=64, shuffle=True
)
test_loader=DataLoader(
    torchvision.datasets.ImageFolder(test_path,transform=transformer),
    batch_size=32, shuffle=True
)
```

- **Hyperparameters used in this model compared to basic CNN**

```
num_epochs=10
```

```
AllConvNet(
  (net): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (7): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU(inplace=True)
    (15): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (16): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (17): ReLU(inplace=True)
    (18): Conv2d(64, 101, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (19): BatchNorm2d(101, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (20): ReLU(inplace=True)
    (21): AdaptiveAvgPool2d(output_size=1)
    (22): Flatten(start_dim=1, end_dim=-1)
  )
)
```

- **Architecture:**



- **Training Loss, Validation Loss, Testing Accuracy**

```
100%|██████████| 10/10 [2:01:17<00:00, 727.78s/it]Epoch: 9 Train Loss: tensor(4.6203) Train Accuracy: 0.010693069306930694

Test Accuracy: 0.009267326732673267
```

3) Regularization:

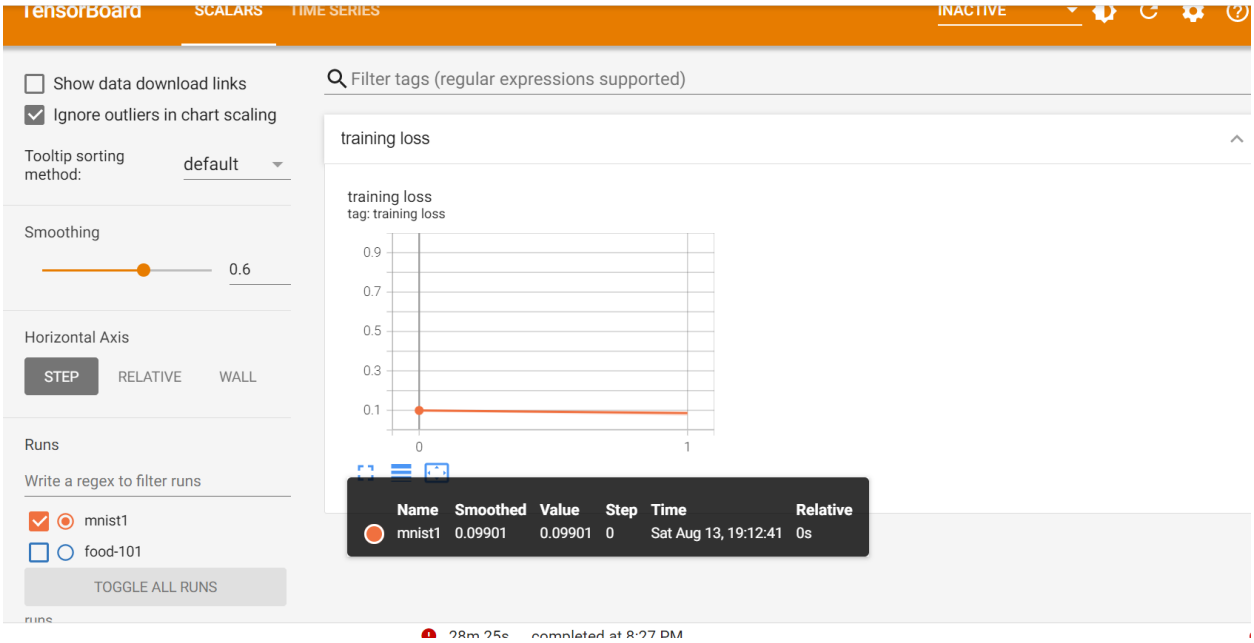
Data Augmentation:

The technique of modifying your data in order to artificially improve the diversity of your training set is known as data augmentation.

Data Augmentations

```
[ ]  train_transforms = transforms.Compose([
         transforms.Resize((224, 224)),
         transforms.TrivialAugmentWide(num_magnitude_bins=31), # how intense
         transforms.ToTensor() # use ToTensor() last to get everything between 0 & 1
     ])

     # Don't need to perform augmentation on the test data
     test_transforms = transforms.Compose([
         transforms.Resize((224, 224)),
         transforms.ToTensor()
     ])
```

For visualization, I have used TensorBoard, I tried to visualize using 2 epochs for testing loss.



4) Transfer Learning:

Pretrained model used-> resnet50

Fine tuning

```
num_filters = pre_model.fc.in_features
layers = list(pre_model.children())[:-1]
self.feature_extractor = nn.Sequential(*layers)
num_classes = 101
self.classifier = nn.Linear(num_filters, num_classes)
```

Result: