Homework 3 Report for -

ME 592

Data Analytics and Machine Learning for Cyber-Physical Systems Applications

Spring 2022

Group Theme - Agriculture

Submitted By

Md Sakib Ferdous
Chemical and Biological Engineering Department

Ibne Farabi Shihab
Computer Science Department

Iowa State University

Submitted To

Dr Soumik Sarkar

Course Teacher

&

Xian Yeow Lee

Course TA

Contents

Answer to the Question no. 1

The following steps are carried out as indicated in the question.

Model Loading

1. Loaded the dataset with code block -

```
dataset = torchvision.datasets.ImageFolder(root = dataset_path, transform=transforms)
train, test = torch.utils.data.random_split(dataset, [11781, 2944])
train_dataloader = torch.utils.data.DataLoader(train, batch_size=batch_size,
                                           num_workers=num_workers, drop_last=True, shuffle=True)
test_dataloader = torch.utils.data.DataLoader(test, batch_size=batch_size,
                                          num_workers=num_workers, drop_last=True)
```

Bounding box sizes

2. According to the question we used 3 different sizes of bounding boxes

   a. Bounding box 1: (300x300)
   b. Bounding box 1: (400x400) and
   c. Bounding box 1: (200x200) and
   d. Bounding box 1: (256x256) and

   4 different models are constructed – Task1.py, Task1p2.py, Taskp3.py and Task2

   using transform from Pytorch. GitHub link –

   https://github.com/farabi1038/ME-592/blob/main/HW3/Task1.py
   https://github.com/farabi1038/ME-592/blob/main/HW3/Task2.py
   https://github.com/farabi1038/ME-592/blob/main/HW3/Task3.py
   https://github.com/farabi1038/ME-592/blob/main/HW3/Task4.py

CNN models

3. The data was split into train (80% of data) and test (20% of data).

```
def load_data(data_dir="./data"):
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])

    trainset = torchvision.datasets.CIFAR10(
        root=data_dir, train=True, download=True, transform=transform)

    testset = torchvision.datasets.CIFAR10(
        root=data_dir, train=False, download=True, transform=transform)

    return trainset, testset
```

4. Model parameters were –
    a. Batch size 64 with
    b. 4 workers
    c. 50 epochs

   We did not try any early stopping.

5. Throughout the homework we used the same model structure taken from GitHub. [Figure 1]
6. After every 10 epochs, we saved our model as a pkt file which can be found in our GitHub repo. Link - https://github.com/farabi1038/ME-592/blob/main/HW3/task2_50.pth

Following this way, we got 4 different types of model. The accuracy of the models has been given below table:

Bounding box Size	Model Accuracy
300x300	85%
200x200	82%
400x400	84%
256x256	80.5%

```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 198, 198]             448
              ReLU-2         [-1, 16, 198, 198]               0
            Conv2d-3         [-1, 16, 196, 196]           2,320
              ReLU-4         [-1, 16, 196, 196]               0
         MaxPool2d-5           [-1, 16, 98, 98]               0
            Conv2d-6           [-1, 32, 96, 96]           4,640
              ReLU-7           [-1, 32, 96, 96]               0
            Conv2d-8           [-1, 32, 94, 94]           9,248
              ReLU-9           [-1, 32, 94, 94]               0
        MaxPool2d-10           [-1, 32, 47, 47]               0
           Conv2d-11           [-1, 64, 45, 45]          18,496
             ReLU-12           [-1, 64, 45, 45]               0
           Conv2d-13           [-1, 64, 43, 43]          36,928
             ReLU-14           [-1, 64, 43, 43]               0
        MaxPool2d-15           [-1, 64, 21, 21]               0
           Conv2d-16          [-1, 128, 19, 19]          73,856
             ReLU-17          [-1, 128, 19, 19]               0
           Conv2d-18          [-1, 128, 17, 17]         147,584
             ReLU-19          [-1, 128, 17, 17]               0
        MaxPool2d-20            [-1, 128, 8, 8]               0
           Conv2d-21            [-1, 256, 6, 6]         295,168
             ReLU-22            [-1, 256, 6, 6]               0
           Conv2d-23            [-1, 256, 4, 4]         590,080
             ReLU-24            [-1, 256, 4, 4]               0
        MaxPool2d-25            [-1, 256, 2, 2]               0
          Flatten-26                 [-1, 1024]               0
          Dropout-27                 [-1, 1024]               0
           Linear-28                  [-1, 256]         262,400
             ReLU-29                  [-1, 256]               0
          Dropout-30                  [-1, 256]               0
           Linear-31                   [-1, 10]           2,570
================================================================
Total params: 1,443,738
Trainable params: 1,443,738
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.46
Forward/backward pass size (MB): 35.04
Params size (MB): 5.51
```

Figure 1: CNN model structure

Answer to the Question no. 2

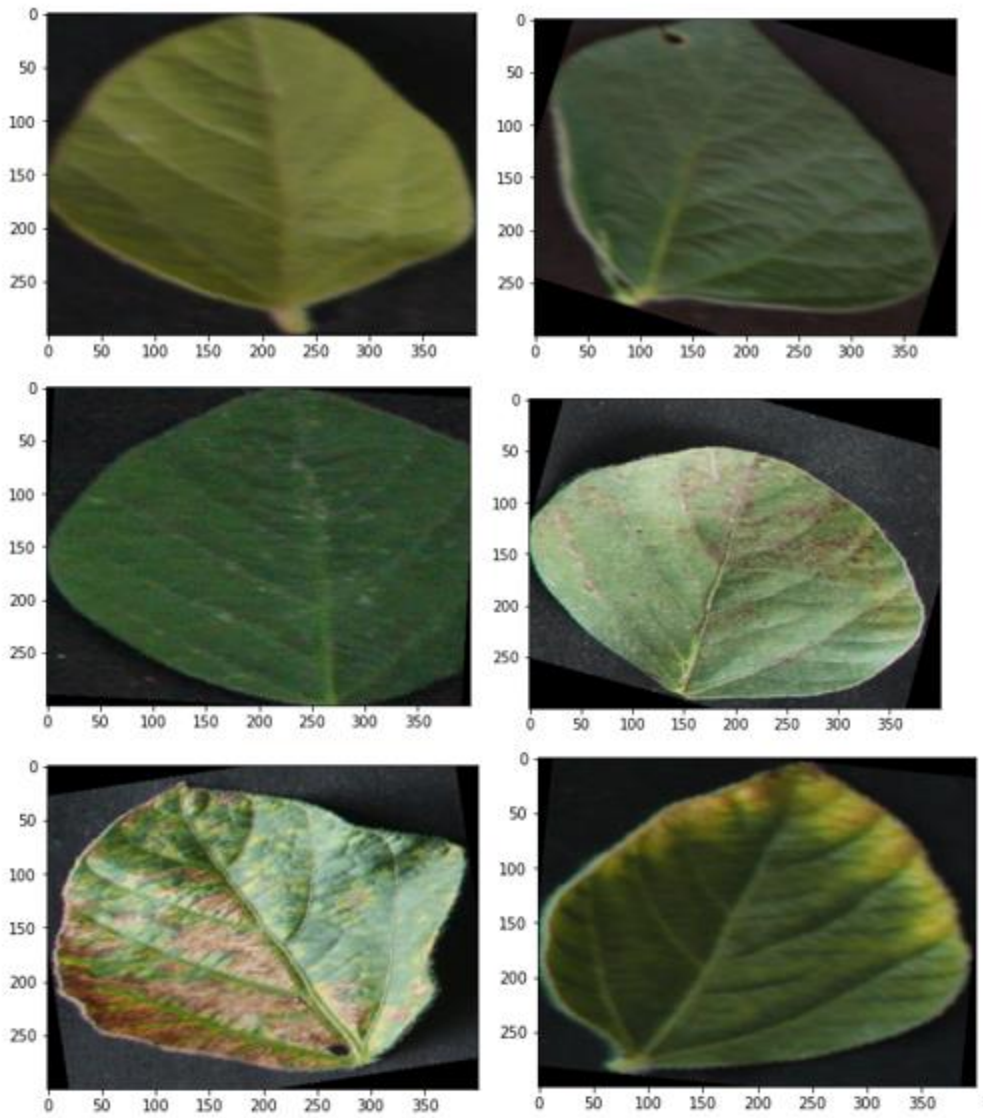
In this task we did all the recommended transformations –

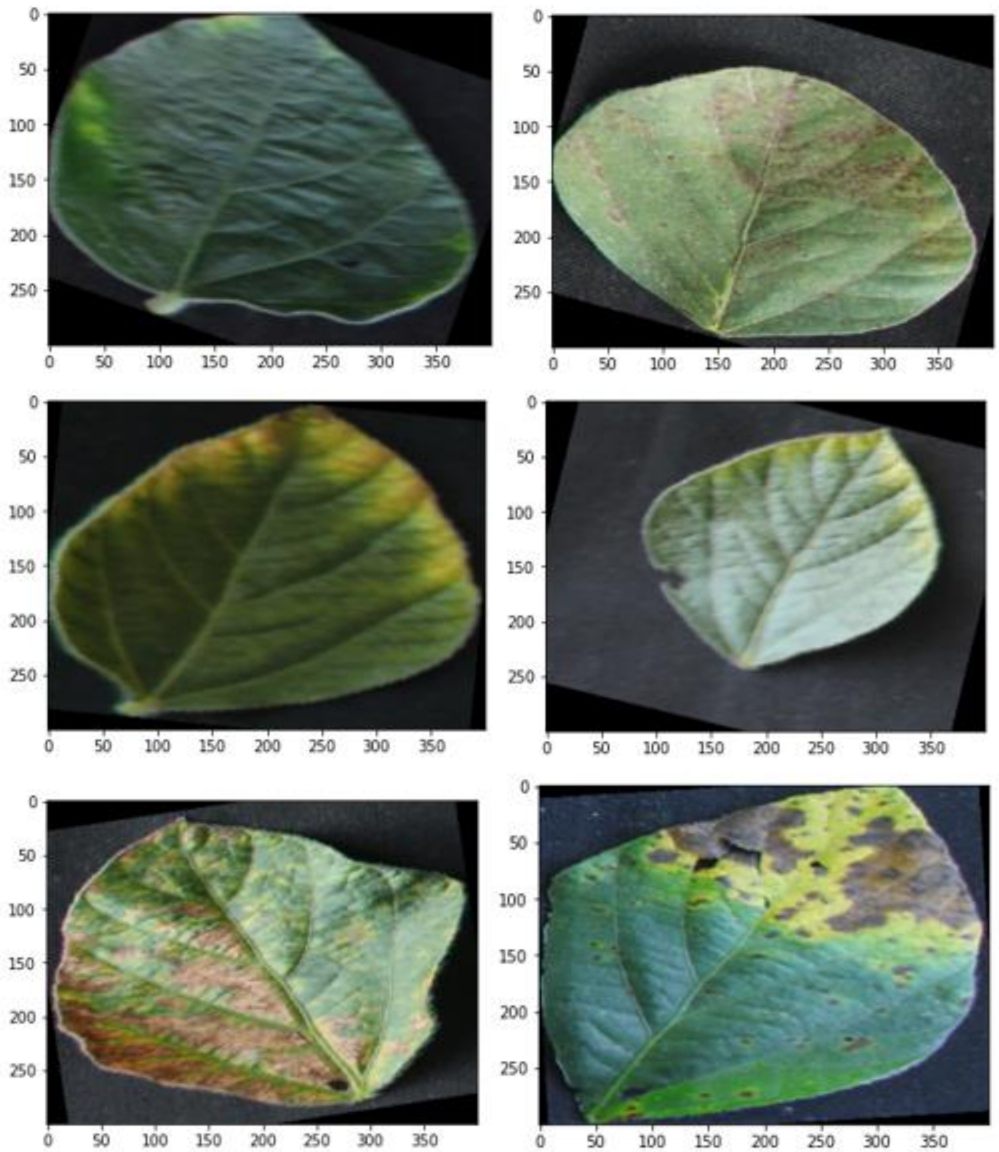
a) Randomly shuffled image
b) Shifting values horizontally or vertically
c) Change scale of image and pad
d) Arbitrarily changing the intensity

The modification was achieved on the fly using Pytorch torchvision. transforms. Compose functionality. The code block is given below.

```
transforms = torchvision.transforms.Compose([
    torchvision.transforms.Resize((256,256)),
    torchvision.transforms.ColorJitter(hue=.05, saturation=.05),
    torchvision.transforms.RandomHorizontalFlip(),
    torchvision.transforms.RandomRotation(20),
    torchvision.transforms.ToTensor()
])
```

In the question in addition to using 256x256(With this size we got better accuracy though it did bid in question 1) bounding box which we did in question 1. Apart from the all the steps were as before as question 1. As we were using 256x256 bounding box, we did all the modification on that file and renamed it to Task2.py. Below are some examples of the processed images we took the data loader. We got an accuracy of **87%(which is the best)** for this.


0 50 100 150 200 250
0 50 100 150 200 250 300 350
0 50 100 150 200 250
0 50 100 150 200 250 300 350
0 50 100 150 200 250
0 50 100 150 200 250 300 350
0 50 100 150 200 250
0 50 100 150 200 250 300 350
0 50 100 150 200 250
0 50 100 150 200 250 300 350
0 50 100 150 200 250
0 50 100 150 200 250 300 350


0
50
100
150
200
250
0 50 100 150 200 250 300 350
0
50
100
150
200
250
0 50 100 150 200 250 300 350
0
50
100
150
200
250
0 50 100 150 200 250 300 350
0
50
100
150
200
250
0 50 100 150 200 250 300 350
0
50
100
150
200
250
0 50 100 150 200 250 300 350
0
50
100
150
200
250
0 50 100 150 200 250 300 350

Answer to the Question no. 3

Model description

Given in answer 2.

Hyper-parameter Optimization

Considering resource and time constraint, the only hyper parameters that were considered in this problem are – Optimizer and Learning rate.

Optimizer	Learning Rate	Accuracy
SGD	0.01	76%
Adam	0.02	78%
Adam	0.0001	**87%**

Model Training

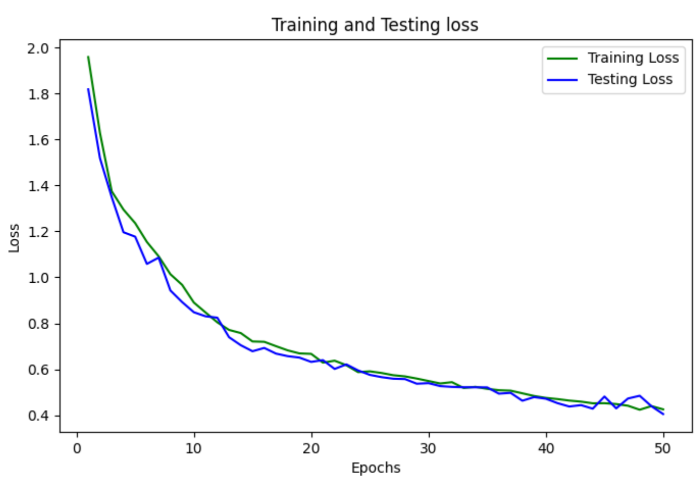
Model training was done using –

Lambda workstation with AMD Threadripper processor 3990X 2.90 GHz with 128 cores and 256GB memory.

The time taken for each of the models are –

Model	Time taken (minutes)
1	~14
2	~14
3	~14
4	~15

Below is the learning curve for our best model:



Code link: https://github.com/farabi1038/ME-592/tree/main/HW3

Answer to the Question no. 4

Using Model to Annotate Canopy Image

For this part, we did the following steps:

1. Crop the images to find leaf and save them using some tools
2. Load the best pkt model from question 3 with the structure.
3. Do predictions on those images using our model

Below are the prediction for the images from our model(Ignore the number 2 after every image name):

```
Using: cpu
name of image :1006 (3).jpg prediction: 1
name of image :1007 (2).jpg prediction: 1
name of image :1006 (2).jpg prediction: 1
name of image :1004 (2).jpg prediction: 4
name of image :1001_C (2).jpg prediction: 1
name of image :1005 (2).jpg prediction: 4
name of image :1009 (2).jpg prediction: 4
name of image :1001 (3).jpg prediction: 6
name of image :1001 (2).jpg prediction: 4
name of image :1008 (2).jpg prediction: 1
name of image :1003 (2).jpg prediction: 4
name of image :1002 (2).jpg prediction: 4
name of image :1005_C (2).jpg prediction: 1
```

The strategy for automating the process