CSE 512 Spring 2021 – Machine Learning – Homework 6

Name: Irfan Ahmed

Solar ID: 113166464

NetID Email: irfan.ahmed@stonybrook.edu

1.1)a)

X =
```
tensor([[0.3868, 0.7403],

            [0.6417, 0.5854],

            [0.1191, 0.1833]], dtype=torch.float64)
```

```
b) Y = tensor([[1., 1.],

        [1., 1.],
        [1., 1.]], dtype=torch.float64) torch.Size([3, 2])
```

```
c) out = tensor([[1.3868, 1.7403],

        [1.6417, 1.5854],
        [1.1191, 1.1833]], dtype=torch.float64)
```

```
In-place addition :
```

```
Y = tensor([[1.3868, 1.7403],
        [1.6417, 1.5854],
        [1.1191, 1.1833]], dtype=torch.float64)
```

```
d) Randomly generated numpy array :
```

```
X = [[0.35990055 0.82630721]

 [0.25775084 0.98181074]
 [0.46863991 0.85011467]]
```

```
Converted to tensor :
```

```
tensor([[0.3599, 0.8263],
        [0.2578, 0.9818],
        [0.4686, 0.8501]], dtype=torch.float64)
```

```
Converted back to numpy array:
```

```
[[0.35990055 0.82630721]
 [0.25775084 0.98181074]
 [0.46863991 0.85011467]]
```

1.2) Requires_grad = true:

```
tensor([[0.0891, 0.2565],
        [0.9623, 0.7975],
        [0.9608, 0.4749]], requires_grad=True)
```

b)Multiplied by 10:

```
tensor([[0.8914, 2.5654],
        [9.6231, 7.9753],
        [9.6077, 4.7492]], grad_fn=<MulBackward0>)
```

Added 0.1:

```
tensor([[0.9914, 2.6654],
        [9.7231, 8.0753],
        [9.7077, 4.8492]], grad_fn=<AddBackward0>)
```

Taking max:

```
tensor(9.7231, grad_fn=<MaxBackward1>)
```

c) Gradient:

```
tensor([[ 0.,  0.],
        [10.,  0.],
        [ 0.,  0.]])
```

d) After multiply:

```
tensor([[0.8914, 2.5654],
        [9.6231, 7.9753],
        [9.6077, 4.7492]])
```

After addition:

```
tensor([[0.9914, 2.6654],
        [9.7231, 8.0753],
        [9.7077, 4.8492]])
```

Max:

```
tensor(9.7231)
```

Since no grad(), these cannot be traced back to do backpropagation.

1.3) a)

Model :

```
TNet(
  (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1))
  (layer1): Sequential(
    (0): ReLU()
    (1): MaxPool2d(kernel_size=4, stride=4, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (layer2): Sequential(
    (0): Sigmoid()
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=512, out_features=10, bias=True)
  )
  (dropout): Dropout(p=0.5, inplace=False)
)

No.of parameters: 6
```

b) Output and size:

```
tensor([[-0.2261,  0.3337, -0.1659, -0.1669, -0.4315,  0.0406, -0.2984,
0.0986,-0.3152,  0.0104]], grad_fn=<AddmmBackward>) torch.Size([1, 10])
```

c)

Gradients of intermediate layer:

```
layer 2.bias.grad after backward
tensor([-0.0018,0.0000,0.0269,0.0092,0.0123,0.0231,0.0108,-0.0168,0.0000,
0.0000,  0.0080, -0.0099, -0.0008,  0.0065,  0.0089,  0.0280,-0.0008,
0.0023,  0.0046, -0.0200, -0.0044, -0.0173,  0.0164, -0.0146,0.0158, -
0.0067,  0.0062,  0.0109, -0.0244,  0.0067,  0.0000, -0.0122,0.0011,
0.0000, -0.0169, -0.0143,  0.0087,  0.0038,  0.0048,  0.0074,-0.0180, -
0.0006, -0.0098,  0.0000,  0.0000,  0.0025,  0.0065, -0.0015,
0.0064,  0.0112,  0.0213, -0.0004,  0.0027,  0.0147,  0.0146, -0.0004,
-0.0121,  0.0274, -0.0044,  0.0045,  0.0000,  0.0007, -0.0164,  0.0093,
0.0000,  0.0110, -0.0063, -0.0286, -0.0018,  0.0142, -0.0017, -0.0054,
-0.0013,  0.0041,  0.0275,  0.0161,  0.0000, -0.0291,  0.0017,  0.0107,
0.0000,  0.0098,  0.0006, -0.0077, -0.0089,  0.0022,  0.0000,  0.0113,
0.0135,  0.0082,  0.0109, -0.0034,  0.0082, -0.0023,  0.0000, -0.0034,
0.0105, -0.0014,  0.0001, -0.0102,  0.0129,  0.0035, -0.0030, -0.0134,
-0.0071, -0.0041,  0.0000,  0.0022, -0.0043, -0.0311,  0.0045,  0.0000,
0.0142, -0.0071, -0.0028, -0.0153,  0.0127, -0.0080,  0.0080, -0.0109,
0.0001,  0.0063, -0.0054,  0.0193, -0.0018,  0.0124, -0.0141, -0.0186])
```

1.4) Accuracy of the network on the 10000 test images: 45 %

Accuracy per class:

```
Accuracy for class plane is: 50.1 %
Accuracy for class car   is: 52.0 %
Accuracy for class bird  is: 46.3 %
Accuracy for class cat   is: 25.4 %
Accuracy for class deer  is: 28.6 %
Accuracy for class dog   is: 42.2 %
Accuracy for class frog  is: 58.9 %
Accuracy for class horse is: 40.6 %
Accuracy for class ship  is: 46.5 %
Accuracy for class truck is: 63.9 %
```

1.5) a)

Accuracy of the network on the 10000 test images: 36 %

```
Accuracy for class plane is: 0.0 %
Accuracy for class car   is: 26.0 %
Accuracy for class bird  is: 0.0 %
Accuracy for class cat   is: 25.2 %
Accuracy for class deer  is: 0.0 %
Accuracy for class dog   is: 0.0 %
Accuracy for class frog  is: 22.8 %
Accuracy for class horse is: 25.6 %
Accuracy for class ship  is: 0.0 %
Accuracy for class truck is: 0.0 %
```

b)

Accuracy of the network on the 10000 test images: 20 %

```
Accuracy for class plane is: 0.0 %
Accuracy for class car   is: 26.0 %
Accuracy for class bird  is: 0.0 %
Accuracy for class cat   is: 25.2 %
Accuracy for class deer  is: 0.0 %
Accuracy for class dog   is: 0.0 %
Accuracy for class frog  is: 22.8 %
Accuracy for class horse is: 25.6 %
Accuracy for class ship  is: 0.0 %
Accuracy for class truck is: 0.0 %
```

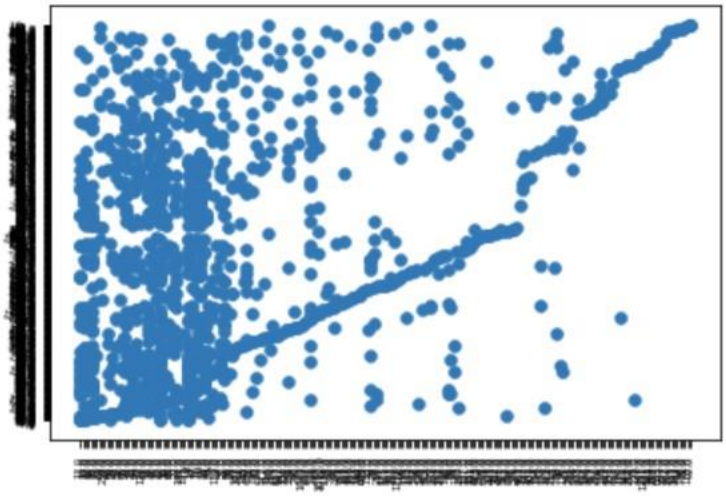
2)

2.1) FamNet:

This network architecture has been developed to learn from exemplary in the image itself and count them. It consists of two modules - 1) Feature Extraction 2) Density Map Prediction. Test-time adaption is an important feature where it can count novel classes based on the exemplaries provided in the image. It uses two loss functions 1)Min-count loss 2) Perturbation Loss and combines them to form the Adaptation Loss. We need to tune hyperparameters to make sure combined loss is similar to the training loss so that the gradients get updated properly.

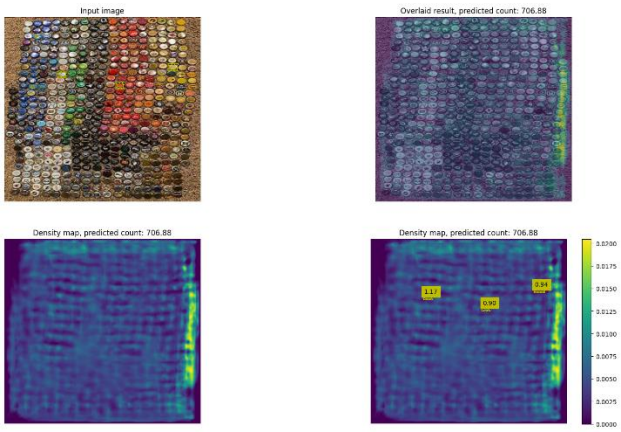
2.2)

Ground truth on X- axis and Predicted count on Y-axis for whole Val Set

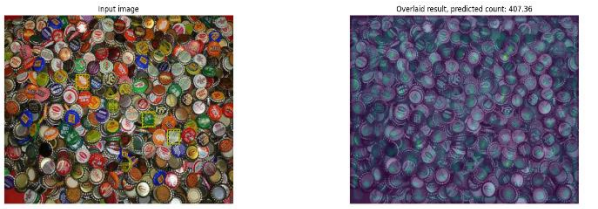


On val data, MAE: 24.32, RMSE: 70.94

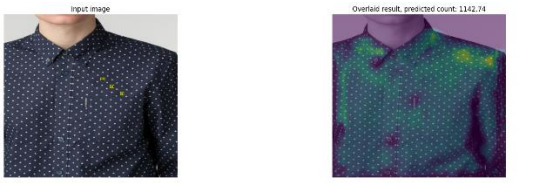
<u>Over Count images:</u>



Ground truth : 501, Predicted count: 706.88

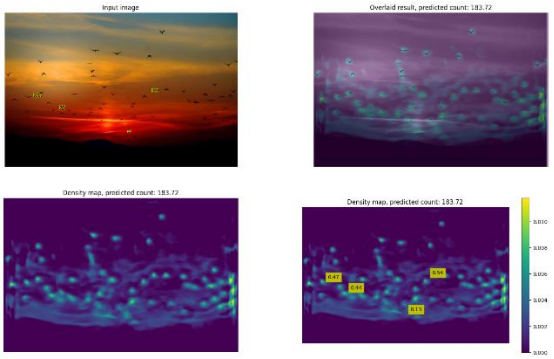


Ground truth : 313, Predicted count:407.36



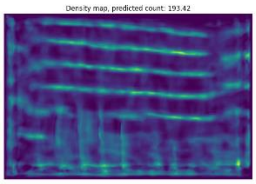
Ground truth:885 , Predicted Count: 1142.74

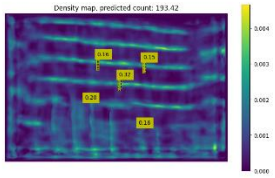
Ground truth:501 , Predicted Count: 657.34



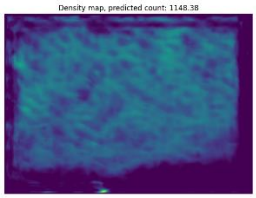
Ground truth:60 , Predicted Count: 182.72

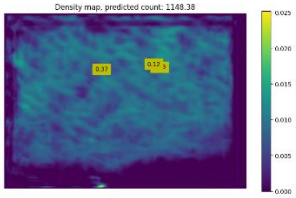
Under-Count images:



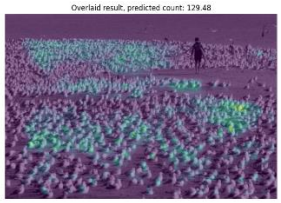


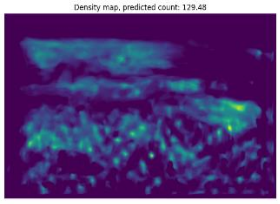
Ground truth:1022 , Predicted Count: 193.42

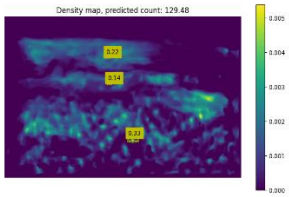




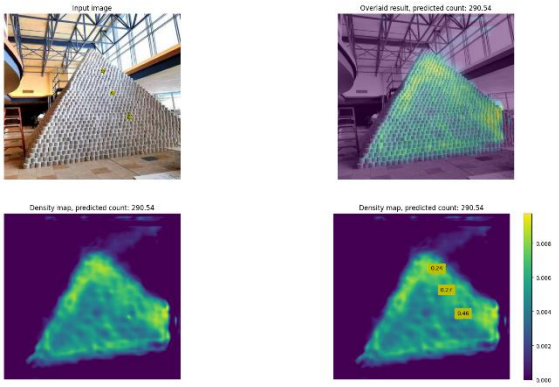
Ground truth:2092 , Predicted Count: 1148.38



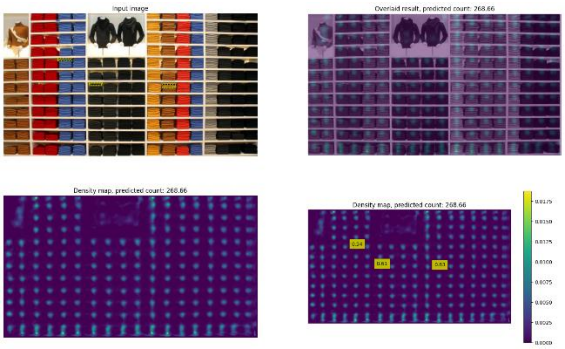




Ground truth:1092 , Predicted Count: 129.48



Ground truth:907 , Predicted Count: 290.54



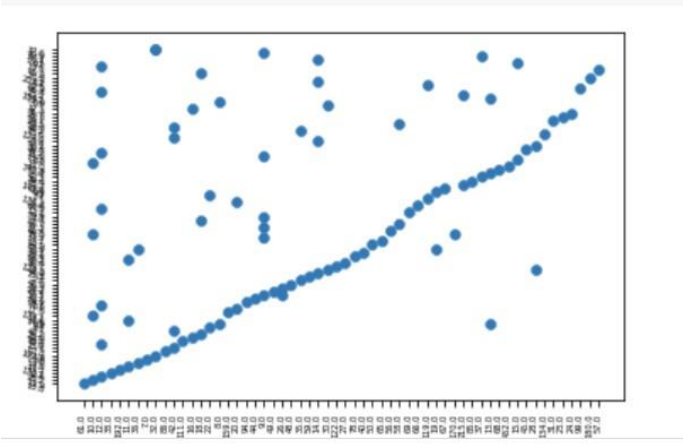
Ground truth:1231 , Predicted Count: 268.66

2.3) With Adaptation,

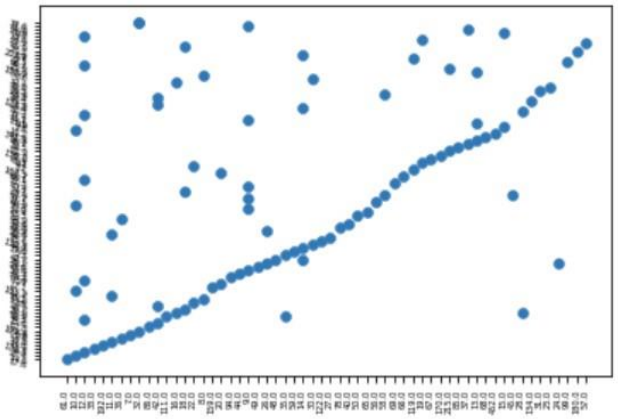
On val_PartA data, MAE: 15.70, RMSE: 25.07

Scatter Plot – GT on X-axis, PredCount on Y-axis

1) With Test-time adaptation:



2) Without Test-time Adaptation:



Attached utils.py and test.py since I modified those files only. Changed the MinLossCount function to adapt to binary masks information.