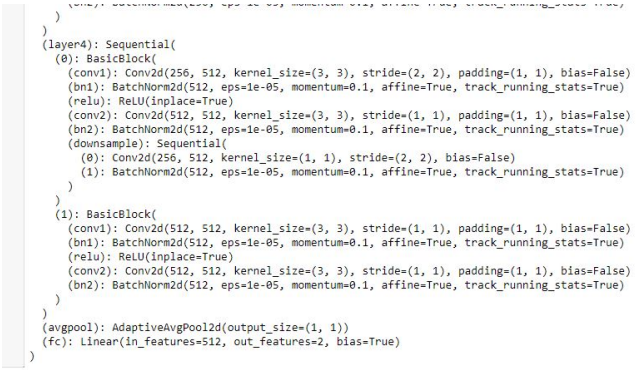
PROMPT

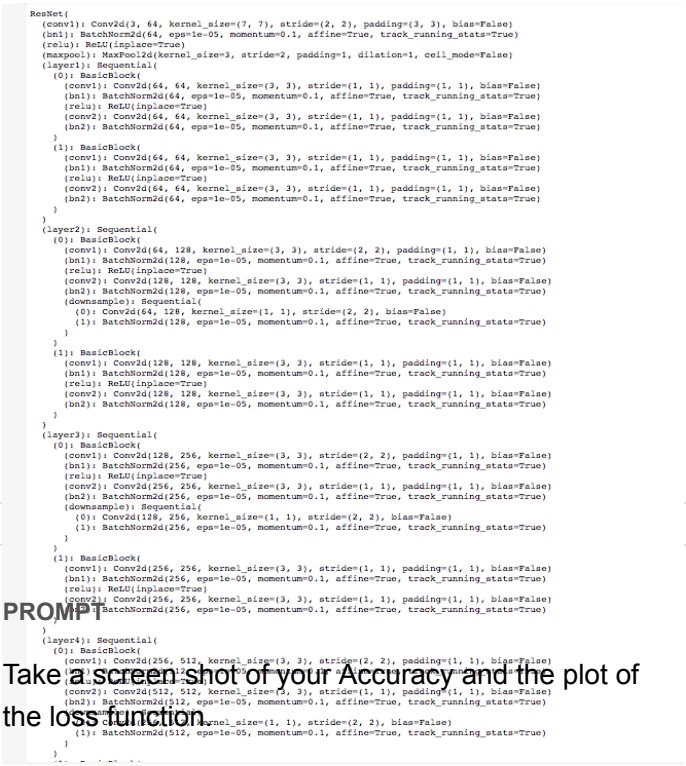
After you change the last linear layer of the resnet18 take a screen shot. make sure you include the last linear layer of the model **.**



```
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=2, bias=True)
)
```

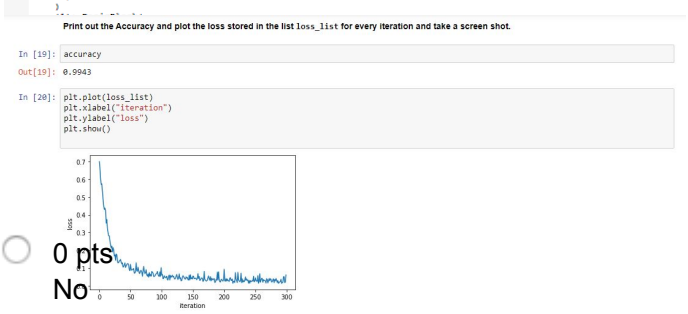
RUBRIC

The image of the ResNet object should be included, if some of the image is cutoff its fine just make sure the last layer is a linear layer with an output dimension of 2 and an input dimension of 512 :



PROMPT

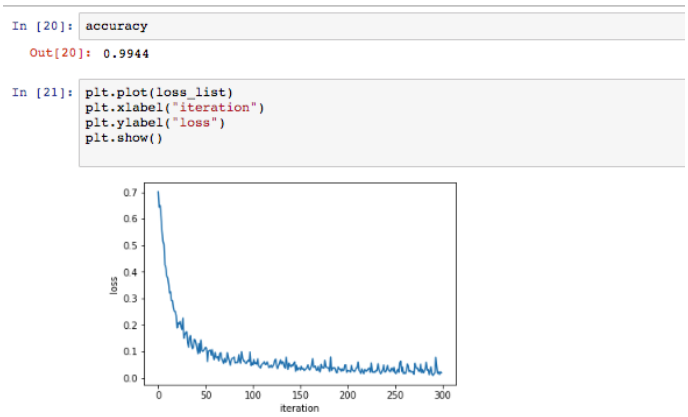
Take a screen shot of your Accuracy and the plot of the loss function.



- 0 pts
  No
- 2 pts
  Yes

RUBRIC

The user produces the accuracy and following plot. The accuracy should be at least 98% and plot should be deceasing.



- 0 pts
  no output
- 1 pt
  the accuracy or plot is missing
- 2 pts
  the accuracy or plot is incorrect
- 4 pts
  perfect

PROMPT

Take a screen shot for your first four misclassified samples:

sample { } predicted value: {} actual value:{}

for example

sample10 predicted value: tensor([0]) actual value:tensor([1])

```
In [22]: count = 0
         max_num_of_items = 4  # first four mis-classified samples
         validation_loader_batch_one = torch.utils.data.DataLoader(dataset=validation_dataset, batch_size=1)

         for i, (x_test, y_test) in enumerate(validation_loader_batch_one):
             # set model to eval
             model.eval()

             # make a prediction
             z = model(x_test)

             # find max
             _, yhat = torch.max(z.data, 1)

             # print mis-classified samples
             if yhat != y_test:
                 print("Sample : {}; Expected Label: {}; Obtained Label: {}".format(str(i), str(y_test), str(yhat)))
                 count += 1
                 if count >= max_num_of_items:
                     break
             # end if
         # end for

         Sample : 22; Expected Label: tensor([1]); Obtained Label: tensor([0])
         Sample : 101; Expected Label: tensor([0]); Obtained Label: tensor([1])
         Sample : 182; Expected Label: tensor([1]); Obtained Label: tensor([0])
         Sample : 213; Expected Label: tensor([0]); Obtained Label: tensor([1])
```

RUBRIC

this should be the output

```
sample 134 yhat tensor([0]) y tensor(1)
sample 364 yhat tensor([0]) y tensor(1)
sample 452 yhat tensor([0]) y tensor(1)
sample 546 yhat tensor([0]) y tensor(1)
```

- 0 pts
  no output
- 1 pt
  some correct
- 4 pts
  all correct