Q4 Shoulders of Giants (15 points)

As we have already seen, deep networks can sometimes be hard to optimize. Often times they heavily overfit on small training sets. Many approaches have been proposed to counter this, eg, Krahenbuhl et al. (ICLR'16) (http://arxiv.org/pdf/1511.06856.pdf), self-supervised learning, etc. However, the most effective approach remains pre-training the network on large, well-labeled supervised datasets such as ImageNet.

While training on the full ImageNet data is beyond the scope of this assignment, people have already trained many popular/standard models and released them online. In this task, we will initialize a ResNet-18 model with pre-trained ImageNet weights (from `torchvision`), and finetune the network for PASCAL classification.

4.1 Load Pre-trained Model (7 pts)

Load the pre-trained weights up to the second last layer, and initialize last weights and biases from scratch.

The model loading mechanism is based on names of the weights. It is easy to load pretrained models from `torchvision.models` , even when your model uses different names for weights. Please briefly explain how to load the weights correctly if the names do not match (hint (https://discuss.pytorch.org/t/loading-weights-from-pretrained-model-with-different-module-names/11841)).

The key names of the model state dict could be changed to match the layer's key names.

In [1]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torchvision import models
import matplotlib.pyplot as plt
%matplotlib inline

import trainer
from utils import ARGS
from simple_cnn import SimpleCNN
from voc_dataset import VOCDataset


# Pre-trained weights up to second-to-last layer
# final layers should be initialized from scratcH!
class PretrainedResNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.pretrained_resnet = models.resnet18(pretrained=True)
#         self.pretrained_resnet.fc = nn.Linear(512,20,bias=True)
        num_classes= len(VOCDataset.CLASS_NAMES)
        self.fc = nn.Sequential(
                      nn.Linear(1000, num_classes,bias=True))
#                          nn.LogSoftmax(dim=1))


    def forward(self, x):
        x=self.pretrained_resnet(x)
        out=self.fc(x)
        return out

%env CUDA_VISIBLE_DEVICES=3
```

```
env: CUDA_VISIBLE_DEVICES=3
```

Use similar hyperparameter setup as in the scratch case. Show the learning curves (training loss, testing MAP) for 10 epochs. Please evaluate your model to calculate the MAP on the testing dataset every 100 iterations.

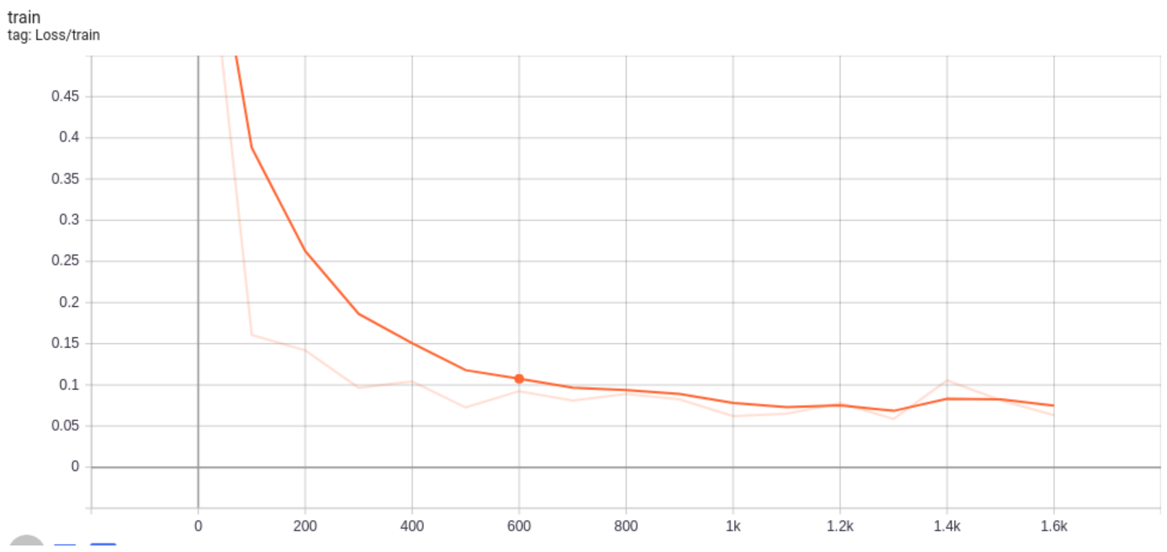
REMEMBER TO SAVE MODEL AT END OF TRAINING

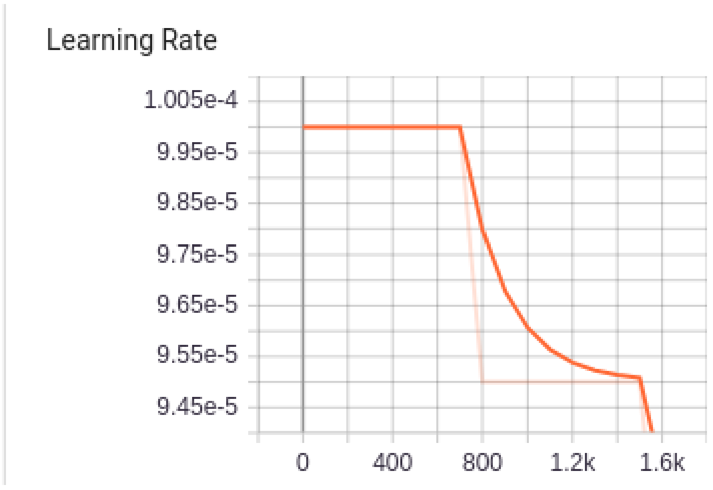
In [2]:
```
args = ARGS(epochs=50, batch_size=32,test_batch_size=32, lr=0.0001,val_e
model = PretrainedResNet()
optimizer = torch.optim.Adam(model.parameters(),lr=args.lr)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer,step_size=5,gamma=
test_ap, test_map = trainer.train(args, model, optimizer, scheduler=sche
print('test map:', test_map)
```

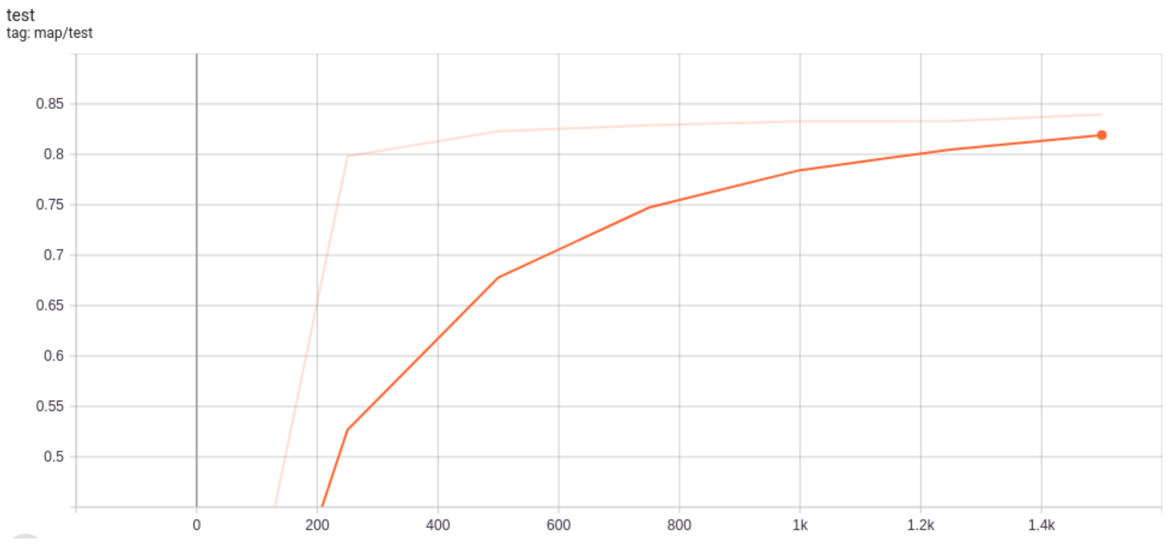
```
Train Epoch: 0 [0 (0%)] Loss: 0.767171
Test Epoch: 0 [0 (0%)]  mAP: 0.074844
Train Epoch: 0 [100 (64%)]      Loss: 0.160626
Train Epoch: 1 [200 (27%)]      Loss: 0.141839
Test Epoch: 1 [250 (59%)]       mAP: 0.798087
Train Epoch: 1 [300 (91%)]      Loss: 0.096467
Train Epoch: 2 [400 (55%)]      Loss: 0.104169
Train Epoch: 3 [500 (18%)]      Loss: 0.072544
Test Epoch: 3 [500 (18%)]       mAP: 0.822814
Train Epoch: 3 [600 (82%)]      Loss: 0.092371
Train Epoch: 4 [700 (46%)]      Loss: 0.081079
Test Epoch: 4 [750 (78%)]       mAP: 0.828885
Train Epoch: 5 [800 (10%)]      Loss: 0.089056
Train Epoch: 5 [900 (73%)]      Loss: 0.082280
Train Epoch: 6 [1000 (37%)]     Loss: 0.062066
Test Epoch: 6 [1000 (37%)]      mAP: 0.832790
Train Epoch: 7 [1100 (1%)]      Loss: 0.064998
Train Epoch: 7 [1200 (64%)]     Loss: 0.078141
Test Epoch: 7 [1250 (96%)]      mAP: 0.832912
Train Epoch: 8 [1300 (28%)]     Loss: 0.058550
Train Epoch: 8 [1400 (92%)]     Loss: 0.105378
Train Epoch: 9 [1500 (55%)]     Loss: 0.081284
Test Epoch: 9 [1500 (55%)]      mAP: 0.839711
Train Epoch: 10 [1600 (19%)]    Loss: 0.063042
```

In [ ]:
```
PATH = 'PretrainedResNet/models/15_.pt'
model.load_state_dict(torch.load(PATH))
```

YOUR TB SCREENSHOTS HERE




Learning Rate
1.005e-4
9.95e-5
9.85e-5
9.75e-5
9.65e-5
9.55e-5
9.45e-5
0
400
800
1.2k
1.6k


test
tag: map/test
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
200
400
600
800
1k
1.2k
1.4k

```
In [ ]:
```