```
import torch
import torch.nn as nn
import torch.nn.functional as F

import numpy as np


 import shutil

try:
  shutil.rmtree("./EVA5")

except:
  pass


!git clone https://github.com/abishek-raju/EVA5.git
```

```
Cloning into 'EVA5'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 177 (delta 24), reused 85 (delta 13), pack-reused 81
Receiving objects: 100% (177/177), 160.49 MiB | 34.85 MiB/s, done.
Resolving deltas: 100% (65/65), done.
Checking out files: 100% (73/73), done.
```

```
import os
os.chdir('/content/EVA5/S11')


os.listdir()
```

```
['main_onecycle_gradcam.ipynb',
 'run.py',
 'plots.py',
 'lr_finder_fast_ai.py',
 'main.py',
 'resnet_build.py',
 'loss.png',
 'model.py',
 'gradcam.py',
 'test',
 'trained.pt',
 'Gradcam_out',
 'data.py',
 'utils.py',
 'device.py',
 'Incorrect_GC',
 'gbn.py',
 'cyclic_learning_rate.ipynb',
 'trained50ep.pt',
 'trained_quiz.pt']
```

```
import pkg_resources
import sys
if ((pkg_resources.get_distribution('albumentations').version == "0.5.2")):
  pass
else:
  !pip install -U git+https://github.com/albu/albumentations --no-cache-dir
  exit()


if 'torch_lr_finder' not in sys.modules:
  !pip install torch_lr_finder
else:
  pass
```

```
Requirement already satisfied: torch_lr_finder in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from tor
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: torch>=0.4.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from to
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
```

```
# !pip install -U git+https://github.com/albu/albumentations --no-cache-dir


############ DATA & TRANSFORMS

from data import get_data
from device import get_device

device = get_device(force_cpu=False)


horizontal_flip_prob = 0.2
vertical_flip_prob = 0.0
gaussian_blur_prob = 0.0
rotate_degree = 20
cutout = 0.3
#
transform_args = {}


transform_args['horizontal_flip_prob'] =  0.2
transform_args['vertical_flip_prob'] = 0.0
transform_args['gaussian_blur_prob'] = 0.0
transform_args['rotate_degree'] = 20
```

```
transform_args['rotate_degree'] = 20
transform_args['cutout'] = 0.3
transform_args['cutout_height'] = 16
transform_args['cutout_width'] = 16


train_loader, test_loader = get_data(device,transform_args,batch_size=512 )
```

```
/usr/local/lib/python3.7/dist-packages/albumentations/pytorch/transforms.py:58: FutureW
  "ToTensor is deprecated and will be replaced by ToTensorV2 in albumentations 0.7.0",
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-
                                                                    170499072/? [00:26<00:00, 6428945.13it/s]

Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

```
train_loader.dataset.classes
```

```
['airplane',
 'automobile',
 'bird',
 'cat',
 'deer',
 'dog',
 'frog',
 'horse',
 'ship',
 'truck']
```

```
train_loader.dataset
```

```
Dataset CIFAR10
    Number of datapoints: 50000
    Root location: ./data
    Split: Train
    StandardTransform
Transform: <data.Transforms object at 0x7f9245b1b410>
```

```
# !pip install torch_lr_finder


from model import main11


from torchsummary import summary
model = main11().to("cuda")


summary(model, input_size=(3,32,32))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
```

```
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,792
       BatchNorm2d-2           [-1, 64, 32, 32]             128
              ReLU-3           [-1, 64, 32, 32]               0
            Conv2d-4          [-1, 128, 32, 32]          73,856
         MaxPool2d-5          [-1, 128, 16, 16]               0
       BatchNorm2d-6          [-1, 128, 16, 16]             256
              ReLU-7          [-1, 128, 16, 16]               0
            Conv2d-8          [-1, 128, 16, 16]         147,456
       BatchNorm2d-9          [-1, 128, 16, 16]             256
           Conv2d-10          [-1, 128, 16, 16]         147,456
      BatchNorm2d-11          [-1, 128, 16, 16]             256
       BasicBlock-12          [-1, 128, 16, 16]               0
           Conv2d-13          [-1, 256, 16, 16]         295,168
        MaxPool2d-14            [-1, 256, 8, 8]               0
      BatchNorm2d-15            [-1, 256, 8, 8]             512
             ReLU-16            [-1, 256, 8, 8]               0
           Conv2d-17            [-1, 512, 8, 8]       1,180,160
        MaxPool2d-18            [-1, 512, 4, 4]               0
      BatchNorm2d-19            [-1, 512, 4, 4]           1,024
             ReLU-20            [-1, 512, 4, 4]               0
           Conv2d-21            [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-22            [-1, 512, 4, 4]           1,024
           Conv2d-23            [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-24            [-1, 512, 4, 4]           1,024
       BasicBlock-25            [-1, 512, 4, 4]               0
        MaxPool2d-26            [-1, 512, 1, 1]               0
           Linear-27                  [-1, 10]           5,130
================================================================
Total params: 6,574,090
Trainable params: 6,574,090
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 6.13
Params size (MB): 25.08
Estimated Total Size (MB): 31.22
----------------------------------------------------------------
/content/EVA5/S11/model.py:122: UserWarning: Implicit dimension choice for log_softmax
  return F.log_softmax(outX)
```

```
!pip install pytorch_lr_finder
```

```
Requirement already satisfied: pytorch_lr_finder in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from py
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from p
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (from py
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/l
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
```

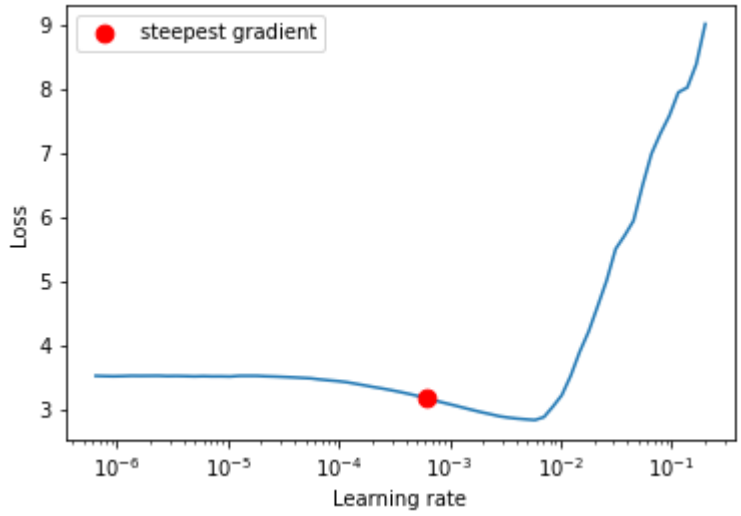
```
from torch_lr_finder import LRFinder


criterion = nn.NLLLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-7, weight_decay=1e-2)
lr_finder = LRFinder(model, optimizer, criterion, device="cuda")
lr_finder.range_test(train_loader, end_lr=10, num_iter=100, step_mode="exp")
```

```
 83%                                                 83/100 [00:12<00:02, 6.75it/s]
/content/EVA5/S11/model.py:122: UserWarning: Implicit dimension choice for log_softmax
  return F.log_softmax(outX)
Stopping early, the loss has diverged
Learning rate search finished. See the graph with {finder_name}.plot()
```

```
lr_finder.plot()
lr_finder.reset()
```

```
LR suggestion: steepest gradient
Suggested LR: 6.28E-04
```




```
incorrectSamples = []
correctSamples = []
correctLabels = []
learningRates = []
model = main11().to(device)
EPOCHS = 24

criterion = nn.NLLLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = 0.018, momentum=0.95, weight_decay = 1e-
scheduler = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr = 0.02, total_steps=2400, e
                                                pct_start=5/EPOCHS, anneal_strategy='linear',
                                                base_momentum=0.85, max_momentum=0.95, div_fa
```

```
model.train()
```

```
(
(conv1): Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): Sequential(
    (0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats
    (1): ReLU()
  )
)
(conv2): Sequential(
  (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mo
  (2): Sequential(
    (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stat
    (1): ReLU()
  )
)
(res1): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
    (bn1): Sequential(
      (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_st
    )
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
    (bn2): Sequential(
      (0): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_st
    )
  )
)
(conv3): Sequential(
  (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mo
  (2): Sequential(
    (0): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stat
    (1): ReLU()
  )
)
(conv4): Sequential(
  (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mo
  (2): Sequential(
    (0): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stat
    (1): ReLU()
  )
)
(res2): Sequential(
  (0): BasicBlock(

    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
    (bn1): Sequential(
      (0): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
    )
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), b
    (bn2): Sequential(
      (0): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
    )
  )
```

```
    )
    (pool4): MaxPool2d(kernel_size=(4, 4), stride=(4, 4), padding=0, dilation=1, ceil_
    (linear): Linear(in_features=512, out_features=10, bias=True)
  )
```

```
train_acc = []
train_losses = []
def train(model,trainloader,
          epoch,num_epochs,device,optimizer,criterion,scheduler,
          L1lambda=None):
  """
  Args:-
  trainLoader: Dataloader for Train Dataset
  epoch: Number of Epochs
  L1lambda: L1lambda Value, by default set to None
  """


  model.train()    # prepare model for training
  pbar = tqdm(enumerate(trainloader),total=len(trainloader),leave = True)
  correct = 0
  processed = 0
  loss = 0
  for batch_idx, (data, target) in pbar: # passing on data & target values to device
    data, target = data.to(device), target.to(device)
    optimizer.zero_grad()    # clear the gradients of all optimized variables

    # Predict
    y_pred = model(data)   # forward pass


    # Calculate loss
    c_loss = criterion(y_pred, target)

    #Implementing L1 Regularization
    if L1lambda:
      with torch.enable_grad():
        l1_loss = 0.
        for param in model.parameters():
          l1_loss += torch.sum(param.abs())
        loss = c_loss +  (L1lambda * l1_loss)
    else:
      loss = c_loss
    loss.backward()   # backward pass: compute gradient of the loss with respect to model par
    optimizer.step()   # perform a single optimization step (parameter update)
    scheduler.step()

    # Update pbar-tqdm

    pred = y_pred.argmax(dim=1, keepdim=True)  # get the index of the max log-probability
    correct += pred.eq(target.view_as(pred)).sum().item()
```

```
        processed += len(data)

        pbar.set_description(desc= f'Epoch [{epoch}/{num_epochs}]')
        pbar.set_postfix(loss_ = loss.item(),acc_ = 100*correct/processed)
      train_acc.append(100*correct/processed)
      train_losses.append(loss)


    import numpy as np


    correct_samples = []
    incorrect_samples = []
    correctLabels = []
    test_acc = []
    test_losses = []
    test_loss_min = np.inf
    def test(model,
            testloader,
            filename,
            correct_samples,
            correctLabels,
            incorrect_samples):

      model.eval()  # prep model for evaluation
      test_loss = 0
      correct = 0
      processed = 0
      pbar = tqdm(enumerate(testloader),total=len(testloader),leave = True)
      with torch.no_grad(): # setting gradients back to zero
          for batch_idx,(data, target) in pbar:

            img_batch = data # this is done to store data
            data, target = data.to(device), target.to(device)

            # forward pass: compute predicted outputs by passing inputs to the model
            output = model(data)

            # sum up batch loss
            # test_loss += F.nll_loss(output, target, reduction='sum').item()
            test_loss = criterion(output, target).item()



            # get the index of the max log-probability
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()

            # storing the entire result data as binary
            result = pred.eq(target.view_as(pred))
            # scheduler.step()

            # # This is to extract incorrect samples/misclassified images
            # if len(incorrect_samples) < 25:
```

```
        # if len(incorrect_samples) < 25:
        #   for i in range(0, testloader.batch_size):
        #     if not list(result)[i]:
        #       incorrect_samples.append({'prediction': list(pred)[i], 'label': list(target.v


        # # this is to extract correct samples/classified images
        # if len(correct_samples) < 25:
        #   for i in range(0, testloader.batch_size):
        #     if list(result)[i]:
        #       correct_samples.append({'prediction': list(pred)[i], 'label': list(target.vie
        #       correctLabels.append(list(target.view_as(pred))[i]) # this is for gradcam



  # save model if validation loss has decreased
  # if test_loss <= test_loss_min:
  #     print('Validation loss has  decreased ({:.4f} --> {:.4f}).  Saving model ...'.format(
  #     torch.save(model.state_dict(), filename)
  #     test_loss_min = test_loss


  # print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.2f}%)\n'.format(
  #     test_loss, correct, len(testloader.dataset),
  #     100. * correct / len(testloader.dataset)))
        # pbar.set_description(desc= f'Epoch [{epoch}/{num_epochs}]')
        processed += len(data)
        pbar.set_postfix(loss_ = test_loss,acc_ = 100. * correct / processed)

  test_acc.append(100. * correct / len(testloader.dataset))
  test_losses.append(test_loss)


from tqdm.notebook import tqdm



num_of_epochs = 24
for epoch in range(1, num_of_epochs+1):
  print('EPOCH: ',epoch)
  train(model,train_loader, epoch,num_of_epochs,"cuda",optimizer,criterion,scheduler, L1lambd
  scheduler.step()
  test(model,test_loader, "model_tut.pt", correct_samples, correctLabels, incorrect_samples)
  for param_group in optimizer.param_groups:
    print('Learning Rate = {a} for EPOCH {e}'.format(a = round(param_group['lr'],5), e=epoch+
    learningRates.append(param_group['lr'])
```

```
EPOCH:  1
```

Epoch [1/24]: 100% 98/98 [00:59<00:00, 1.66it/s, acc_=43.3, loss_=2.29]

```
/content/EVA5/S11/model.py:122: UserWarning: Implicit dimension choice for log_softmax
  return F.log_softmax(outX)
```

100% 20/20 [00:01<00:00, 10.31it/s, acc_=44.9, loss_=2.55]

```
Learning Rate = 0.00464 for EPOCH 2
EPOCH:  2
```

Epoch [2/24]: 100% 98/98 [00:41<00:00, 2.34it/s, acc_=58.3, loss_=1.52]

100% 20/20 [00:26<00:00, 1.34s/it, acc_=64.8, loss_=0.966]

```
Learning Rate = 0.00844 for EPOCH 3
EPOCH:  3
```

Epoch [3/24]: 100% 98/98 [00:24<00:00, 3.93it/s, acc_=66.7, loss_=1.33]

100% 20/20 [00:01<00:00, 10.88it/s, acc_=66.5, loss_=1.08]

```
Learning Rate = 0.01224 for EPOCH 4
EPOCH:  4
```

Epoch [4/24]: 100% 98/98 [00:15<00:00, 6.52it/s, acc_=72.9, loss_=1.23]

100% 20/20 [01:04<00:00, 3.25s/it, acc_=71.5, loss_=0.869]

```
Learning Rate = 0.01604 for EPOCH 5
EPOCH:  5
```

Epoch [5/24]: 100% 98/98 [01:02<00:00, 1.56it/s, acc_=76.4, loss_=1.21]

100% 20/20 [00:47<00:00, 2.39s/it, acc_=72, loss_=0.933]

```
Learning Rate = 0.01985 for EPOCH 6
EPOCH:  6
```

Epoch [6/24]: 100% 98/98 [00:45<00:00, 2.13it/s, acc_=81.4, loss_=0.894]

100% 20/20 [00:30<00:00, 1.54s/it, acc_=77.9, loss_=0.697]

```
Learning Rate = 0.01904 for EPOCH 7
EPOCH:  7
```

Epoch [7/24]: 100% 98/98 [00:28<00:00, 3.40it/s, acc_=84.5, loss_=0.742]

100% 20/20 [00:13<00:00, 1.47it/s, acc_=81.7, loss_=0.553]

```
Learning Rate = 0.01804 for EPOCH 8
EPOCH:  8
```

Epoch [8/24]: 100% 98/98 [00:15<00:00, 6.51it/s, acc_=86.7, loss_=0.627]

100% 20/20 [01:08<00:00, 3.42s/it, acc_=83.8, loss_=0.486]

```
Learning Rate = 0.01704 for EPOCH 9
EPOCH:  9
```

Epoch [9/24]: 100% 98/98 [01:06<00:00, 1.47it/s, acc_=88.5, loss_=0.545]

100% 20/20 [00:02<00:00, 9.68it/s, acc_=83.7, loss_=0.558]

```
Learning Rate = 0.01605 for EPOCH 10
EPOCH:  10
```

Epoch [10/24]: 100% 98/98 [00:49<00:00, 1.99it/s, acc_=89.5, loss_=0.535]

100% 20/20 [00:01<00:00, 10.08it/s, acc_=84.2, loss_=0.494]

```
Learning Rate = 0.01505 for EPOCH 11
EPOCH:  11
```

Epoch [11/24]: 100% 98/98 [00:32<00:00, 3.03it/s, acc_=90.3, loss_=0.548]

100% 20/20 [00:17<00:00, 1.16it/s, acc_=83.4, loss_=0.43]

```
Learning Rate = 0.01405 for EPOCH 12
EPOCH:  12
```

Epoch [12/24]: 100% 98/98 [00:15<00:00, 6.49it/s, acc_=91.4, loss_=0.431]

100% 20/20 [00:01<00:00, 10.66it/s, acc_=85.3, loss_=0.391]

```
Learning Rate = 0.01305 for EPOCH 13
EPOCH:  13
```

Epoch [13/24]: 100% 98/98 [01:08<00:00, 1.42it/s, acc_=91.7, loss_=0.407]

100% 20/20 [00:01<00:00, 11.13it/s, acc_=87.8, loss_=0.449]

```
Learning Rate = 0.01205 for EPOCH 14
EPOCH:  14
```

Epoch [14/24]: 100% 98/98 [00:51<00:00, 1.89it/s, acc_=92.5, loss_=0.431]

100% 20/20 [00:01<00:00, 10.04it/s, acc_=83.9, loss_=0.482]

```
Learning Rate = 0.01105 for EPOCH 15
EPOCH:  15
```

Epoch [15/24]: 100% 98/98 [00:34<00:00, 2.82it/s, acc_=93.1, loss_=0.358]

100% 20/20 [00:19<00:00, 1.02it/s, acc_=86.7, loss_=0.494]

```
Learning Rate = 0.01005 for EPOCH 16
EPOCH:  16
```

Epoch [16/24]: 100% 98/98 [00:17<00:00, 5.57it/s, acc_=93.3, loss_=0.382]

100% 20/20 [00:02<00:00, 8.07it/s, acc_=86, loss_=0.358]

```
Learning Rate = 0.00905 for EPOCH 17
EPOCH:  17
```

Epoch [17/24]: 100% 98/98 [00:15<00:00, 6.49it/s, acc_=93.9, loss_=0.332]

100% 20/20 [00:54<00:00, 2.71s/it, acc_=88.7, loss_=0.359]

```
Learning Rate = 0.00806 for EPOCH 18
EPOCH:  18
```

Epoch [18/24]: 100% 98/98 [00:52<00:00, 1.88it/s, acc_=94.4, loss_=0.336]

100% 20/20 [00:36<00:00, 1.85s/it, acc_=85.4, loss_=0.436]

```
Learning Rate = 0.00706 for EPOCH 19
EPOCH:  19
```

Epoch [19/24]: 100% 98/98 [00:35<00:00, 2.79it/s, acc_=94.7, loss_=0.259]

100% 20/20 [00:19<00:00, 1.01it/s, acc_=89.9, loss_=0.272]

```
Learning Rate = 0.00606 for EPOCH 20
EPOCH:  20
```

Epoch [20/24]: 100% 98/98 [00:15<00:00, 6.50it/s, acc_=95.2, loss_=0.274]

100% 20/20 [00:02<00:00, 7.27it/s, acc_=89.7, loss_=0.343]

```
Learning Rate = 0.00506 for EPOCH 21
EPOCH:  21
```

Epoch [21/24]: 100% 98/98 [00:15<00:00, 6.50it/s, acc_=96, loss_=0.274]

100% 20/20 [00:53<00:00, 2.66s/it, acc_=90.6, loss_=0.259]

```
Learning Rate = 0.00406 for EPOCH 22
EPOCH:  22
```

Epoch [22/24]: 100% 98/98 [00:51<00:00, 1.91it/s, acc_=96.2, loss_=0.217]

100% 20/20 [00:01<00:00, 10.44it/s, acc_=91.3, loss_=0.214]

```
Learning Rate = 0.00306 for EPOCH 23
```

```
Learning Rate = 0.00206 for EPOCH 23
EPOCH:  23
```

```
Epoch [23/24]: 100%  98/98 [00:34<00:00, 2.87it/s, acc_=97.1, loss_=0.236]
100%  20/20 [00:18<00:00, 1.06it/s, acc_=91.5, loss_=0.232]
```

```
Learning Rate = 0.00206 for EPOCH 24
EPOCH:  24
```

```
Epoch [24/24]: 100%  98/98 [00:15<00:00, 6.51it/s, acc_=97.7, loss_=0.179]
```

```
import matplotlib.pyplot as plt

# line 1 points
x1 = range(1,len(test_acc)+1)
y1 = test_acc
# plotting the line 1 points
plt.plot(x1, y1, label = "test_acc")

# line 2 points
x2 = range(1,len(train_acc)+1)
y2 = train_acc
# # plotting the line 2 points
plt.plot(x2, y2, label = "train_acc")

# naming the x axis
plt.xlabel('Epochs')
# naming the y axis
plt.ylabel('Accuracy')
# giving a title to my graph
plt.title('ACCURACY')

# show a legend on the plot
plt.legend()

# function to show the plot
plt.show()
```

```
# line 1 points
x1 = range(1,len(test_losses)+1)
y1 = test_losses
# plotting the line 1 points
plt.plot(x1, y1, label = "test_loss")

# line 2 points
x2 = range(1,len(train_losses)+1)
y2 = train_losses
# # plotting the line 2 points
plt.plot(x2, y2, label = "train_loss")

# naming the x axis
plt.xlabel('Epochs')
# naming the y axis
plt.ylabel('Loss')
# giving a title to my graph
plt.title('Loss')

# show a legend on the plot
plt.legend()

# function to show the plot
plt.show()
```