```
[1] from google.colab import drive
    drive.mount('/content/gdrive',force_remount=True)
```

```
Mounted at /content/gdrive
```

```
[2] pip install torch-lr-finder
```

```
Collecting torch-lr-finder
  Downloading torch_lr_finder-0.2.1-py3-none-any.whl (11 kB)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (21.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (1.19.5)
Requirement already satisfied: torch>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (1.10.0+cu111)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (4.62.3)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch>=0.4.1->torch-lr-finder) (3.10.0.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (3.0.6)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (0.11.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->torch-lr-finder) (1.15.0)
Installing collected packages: torch-lr-finder
Successfully installed torch-lr-finder-0.2.1
```

```
[3] pip install torch-lr-finder -v --global-option="apex"
```

```
Value for scheme.purelib does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/lib/python3.7/dist-packages
sysconfig: /usr/lib/python3.7/site-packages
Value for scheme.headers does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/include/python3.7/UNKNOWN
sysconfig: /usr/include/python3.7m/UNKNOWN
Value for scheme.scripts does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/bin
sysconfig: /usr/bin
Value for scheme.data does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local
sysconfig: /usr
Additional context:
user = False
home = None
root = None
prefix = None
Non-user install because site-packages writeable
Created temporary directory: /tmp/pip-ephem-wheel-cache-qqbtb3q2
Created temporary directory: /tmp/pip-req-tracker-3_a003f6
Initialized build tracking at /tmp/pip-req-tracker-3_a003f6
Created build tracker: /tmp/pip-req-tracker-3_a003f6
Entered build tracker: /tmp/pip-req-tracker-3_a003f6
Created temporary directory: /tmp/pip-install-svlyym4s
Requirement already satisfied: torch-lr-finder in /usr/local/lib/python3.7/dist-packages (0.2.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (21.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (4.62.3)
Requirement already satisfied: torch>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (1.10.0+cu111)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (1.19.5)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch>=0.4.1->torch-lr-finder) (3.10.0.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (1.3.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (3.0.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->torch-lr-finder) (1.15.0)
Created temporary directory: /tmp/pip-unpack-dvvw661m
Value for scheme.platlib does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/lib/python3.7/dist-packages
sysconfig: /usr/lib/python3.7/site-packages
Value for scheme.purelib does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/lib/python3.7/dist-packages
sysconfig: /usr/lib/python3.7/site-packages
Value for scheme.headers does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/include/python3.7/UNKNOWN
sysconfig: /usr/include/python3.7m/UNKNOWN
Value for scheme.scripts does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local/bin
sysconfig: /usr/bin
Value for scheme.data does not match. Please report this to <https://github.com/pypa/pip/issues/9617>
distutils: /usr/local
sysconfig: /usr
Additional context:
user = False
home = None
root = None
prefix = None
Removed build tracker: '/tmp/pip-req-tracker-3_a003f6'
```

```
[3]
```

```
[4] import torch, torchvision
    from torchvision import datasets, models, transforms
    import torch.nn as nn
    import torch.optim as optim
    from torch.optim import lr_scheduler
    from torch.utils.data import DataLoader
    import time
    from torchsummary import summary

    import numpy as np
    import matplotlib.pyplot as plt
    import os

    from PIL import Image
    from torch_lr_finder import LRFinder
```

```
[5] if torch.cuda.is_available():
      device = torch.device("cuda:0")
      print("GPU")
      torch.cuda.device_count()
      torch.cuda.get_device_name(0)
      !nvidia-smi
    else:
      device = torch.device("cpu")
```

```
GPU
Mon Jan 10 11:33:11 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   40C    P0    27W / 250W |      2MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
[6] #parameters
    Batch_Size = 8
    Number_of_Epoch = 30
    Init_LR_for_search = 1e-7
```

```
[7] # Applying Transforms to the Data

    image_transforms = {
        'train': transforms.Compose([
            transforms.RandomResizedCrop(size = 400, scale=(0.8, 1.0)), #size=256
            transforms.RandomRotation(degrees=15),
            transforms.RandomHorizontalFlip(),
            transforms.CenterCrop(size = 300), #size=224
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406],
                                 [0.229, 0.224, 0.225])
        ])
```

```
    ]),
    'valid': transforms.Compose([
        transforms.Resize(size = 400), #size=256
        transforms.CenterCrop(size = 300), #size=224
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize(size = 400), #size=256
        transforms.CenterCrop(size = 300), #size=224
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ])
}
```

```
# Load the Data

# Set train and valid directory paths

dataset = '/content/gdrive/MyDrive/c2'

#train_path = ("/content/gdrive/MyDrive/CAD_Project_Data/challenge1/train")
#val_path = ("/content/gdrive/MyDrive/CAD_Project_Data/challenge1/val")

train_directory = os.path.join(dataset, 'train')
valid_directory = os.path.join(dataset, 'val')

# Batch size
bs = Batch_Size

# Number of classes
num_classes = len(os.listdir(valid_directory))  #10#2#257
print(num_classes)

# Load Data from folders
data = {
    'train': datasets.ImageFolder(root=train_directory, transform=image_transforms['train']),
    'val': datasets.ImageFolder(root=valid_directory, transform=image_transforms['valid'])
}

# Get a mapping of the indices to the class names, in order to see the output classes of the test images.
idx_to_class = {v: k for k, v in data['train'].class_to_idx.items()}
print(idx_to_class)

# Size of Data, to be used for calculating Average Loss and Accuracy
train_data_size = len(data['train'])
valid_data_size = len(data['val'])

# Create iterators for the Data loaded using DataLoader module
train_data_loader = DataLoader(data['train'], batch_size=bs, shuffle=True)
valid_data_loader = DataLoader(data['val'], batch_size=bs, shuffle=True)
```

```
3
{0: 'bcc', 1: 'bkl', 2: 'mel'}
```

```
[9] train_data_size, valid_data_size
```

```
(2000, 500)
```

```
[10] model = models.resnet18(pretrained=True).to(device)
     model
```

```
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (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=1000, bias=True)
)
```

```
[11] # Freeze model parameters
     for param in model.parameters():
         param.requires_grad = False
```

```
[12] # Change the final layer of AlexNet Model for Transfer Learning
     #model.classifier[6] = nn.Linear(4096, num_classes)
     #model.classifier.add_module("7", nn.LogSoftmax(dim = 1))
     num_ftrs = model.fc.in_features
     model.fc = nn.Linear(num_ftrs, 3)
     model.to(device)
     model
```

```
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

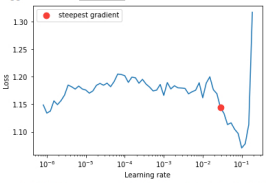
```
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (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=3, bias=True)
)
```

```
summary(model, (3, 224, 224))
```

```
              ReLU-21          [-1, 128, 28, 28]               0
            Conv2d-22          [-1, 128, 28, 28]         147,456
       BatchNorm2d-23          [-1, 128, 28, 28]             256
            Conv2d-24          [-1, 128, 28, 28]           8,192
       BatchNorm2d-25          [-1, 128, 28, 28]             256
              ReLU-26          [-1, 128, 28, 28]               0
        BasicBlock-27          [-1, 128, 28, 28]               0
            Conv2d-28          [-1, 128, 28, 28]         147,456
       BatchNorm2d-29          [-1, 128, 28, 28]             256
              ReLU-30          [-1, 128, 28, 28]               0
            Conv2d-31          [-1, 128, 28, 28]         147,456
       BatchNorm2d-32          [-1, 128, 28, 28]             256
              ReLU-33          [-1, 128, 28, 28]               0
        BasicBlock-34          [-1, 128, 28, 28]               0
            Conv2d-35          [-1, 256, 14, 14]         294,912
       BatchNorm2d-36          [-1, 256, 14, 14]             512
              ReLU-37          [-1, 256, 14, 14]               0
            Conv2d-38          [-1, 256, 14, 14]         589,824
       BatchNorm2d-39          [-1, 256, 14, 14]             512
            Conv2d-40          [-1, 256, 14, 14]          32,768
       BatchNorm2d-41          [-1, 256, 14, 14]             512
              ReLU-42          [-1, 256, 14, 14]               0
        BasicBlock-43          [-1, 256, 14, 14]               0
            Conv2d-44          [-1, 256, 14, 14]         589,824
       BatchNorm2d-45          [-1, 256, 14, 14]             512
              ReLU-46          [-1, 256, 14, 14]               0
            Conv2d-47          [-1, 256, 14, 14]         589,824
       BatchNorm2d-48          [-1, 256, 14, 14]             512
              ReLU-49          [-1, 256, 14, 14]               0
        BasicBlock-50          [-1, 256, 14, 14]               0
            Conv2d-51            [-1, 512, 7, 7]       1,179,648
       BatchNorm2d-52            [-1, 512, 7, 7]           1,024
              ReLU-53            [-1, 512, 7, 7]               0
            Conv2d-54            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-55            [-1, 512, 7, 7]           1,024
            Conv2d-56            [-1, 512, 7, 7]         131,072
       BatchNorm2d-57            [-1, 512, 7, 7]           1,024
              ReLU-58            [-1, 512, 7, 7]               0
        BasicBlock-59            [-1, 512, 7, 7]               0
            Conv2d-60            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-61            [-1, 512, 7, 7]           1,024
              ReLU-62            [-1, 512, 7, 7]               0
            Conv2d-63            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-64            [-1, 512, 7, 7]           1,024
              ReLU-65            [-1, 512, 7, 7]               0
        BasicBlock-66            [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-67            [-1, 512, 1, 1]               0
            Linear-68                  [-1, 3]           1,539
================================================================
Total params: 11,178,051
Trainable params: 1,539
Non-trainable params: 11,176,512
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 62.79
Params size (MB): 42.64
Estimated Total Size (MB): 106.00
----------------------------------------------------------------
```

```
[14] # Define Optimizer and Loss Function
loss_func = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr = Init_LR_for_search, weight_decay = 1e-2)
lr_finder = LRFinder(model, optimizer, loss_func, device = "cuda:0")
lr_finder.range_test(train_data_loader, end_lr = 100, num_iter = 100)
lr_finder.plot()
#optimizer
```

```
74% 74/100 [01:42<00:31, 1.23s/it]
Stopping early, the loss has diverged
Learning rate search finished. See the graph with {finder_name}.plot()
LR suggestion: steepest gradient
Suggested LR: 2.85E-02
```




```
(<matplotlib.axes._subplots.AxesSubplot at 0x7f97362a8650>,
 0.028480358684358022)
```

```
[15] optimizer = optim.Adam(model.parameters(), lr = 2.85E-02)
```

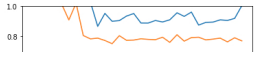
```
[16] decaying_lr = lr_scheduler.StepLR(optimizer, step_size = 4, gamma = 0.1)
```

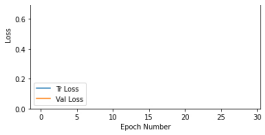
```
[17] def train_and_validate(model, loss_criterion, optimizer, scheduler, epochs):
        '''
        Function to train and validate
        Parameters
            :param model: Model to train and validate
            :param loss_criterion: Loss Criterion to minimize
            :param optimizer: Optimizer for computing gradients
            :param epochs: Number of epochs (default=25)

        Returns
            model: Trained Model with best validation accuracy
            history: (dict object): Having training loss, accuracy and validation loss, accuracy
        '''

        start = time.time()
        history = []
        best_acc = 0.0

        for epoch in range(epochs):
            epoch_start = time.time()
            print("Epoch: {}/{}".format(epoch+1, epochs))

            # Set to training mode
            model.train()

            # Loss and Accuracy within the epoch
            train_loss = 0.0
            train_acc = 0.0

            valid_loss = 0.0
```

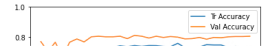
```
            valid_acc = 0.0

            for i, (inputs, labels) in enumerate(train_data_loader):

                inputs = inputs.to(device)
                labels = labels.to(device)

                # Clean existing gradients
                optimizer.zero_grad()

                # Forward pass - compute outputs on input data using the model
                outputs = model(inputs).to(device)

                # Compute loss
                loss = loss_criterion(outputs, labels)

                # Backpropagate the gradients
                loss.backward()

                # Update the parameters
                optimizer.step()

                # Compute the total loss for the batch and add it to train_loss
                train_loss += loss.item() * inputs.size(0)

                # Compute the accuracy
                ret, predictions = torch.max(outputs.data, 1)
                correct_counts = predictions.eq(labels.data.view_as(predictions))

                # Convert correct_counts to float and then compute the mean
                acc = torch.mean(correct_counts.type(torch.FloatTensor))

                # Compute total accuracy in the whole batch and add to train_acc
                train_acc += acc.item() * inputs.size(0)

                #print("Batch number: {:03d}, Training: Loss: {:.4f}, Accuracy: {:.4f}".format(i, loss.item(), acc.item()))
            scheduler.step()


            # Validation - No gradient tracking needed
            with torch.no_grad():

                # Set to evaluation mode
                model.eval()

                # Validation loop
                for j, (inputs, labels) in enumerate(valid_data_loader):
                    inputs = inputs.to(device)
                    labels = labels.to(device)

                    # Forward pass - compute outputs on input data using the model
                    outputs = model(inputs).to(device)

                    # Compute loss
                    loss = loss_criterion(outputs, labels).to(device)

                    # Compute the total loss for the batch and add it to valid_loss
                    valid_loss += loss.item() * inputs.size(0)

                    # Calculate validation accuracy
                    ret, predictions = torch.max(outputs.data, 1)
                    correct_counts = predictions.eq(labels.data.view_as(predictions))

                    # Convert correct_counts to float and then compute the mean
                    acc = torch.mean(correct_counts.type(torch.FloatTensor))

                    # Compute total accuracy in the whole batch and add to valid_acc
                    valid_acc += acc.item() * inputs.size(0)

                    #print("Validation Batch number: {:03d}, Validation: Loss: {:.4f}, Accuracy: {:.4f}".format(j, loss.item(), acc.item()))

            # Find average training loss and training accuracy
            avg_train_loss = train_loss/train_data_size
            avg_train_acc = train_acc/train_data_size

            # Find average training loss and training accuracy
            avg_valid_loss = valid_loss/valid_data_size
            avg_valid_acc = valid_acc/valid_data_size

            history.append([avg_train_loss, avg_valid_loss, avg_train_acc, avg_valid_acc])

            epoch_end = time.time()

            print("Epoch : {:03d}, Training: Loss: {:.4f}, Accuracy: {:.4f}%, \n\t\tValidation : Loss : {:.4f}, Accuracy: {:.4f}%, Time: {:.4f}s".format(epoch+1, avg_train_loss, avg_train_acc*100, avg_valid_loss, avg_v

            # Save if the model has best accuracy till now
            torch.save(model, dataset+'_model_'+str(epoch)+'.pt')

        return model, history
```

```
[18] num_epochs = Number_of_Epoch
     trained_model, history = train_and_validate(model, loss_func, optimizer, decaying_lr, num_epochs)

     torch.save(history, dataset+'_history.pt')


     history = np.array(history)
     plt.plot(history[:,0:2])
     plt.legend(['Tr Loss', 'Val Loss'])
     plt.xlabel('Epoch Number')
     plt.ylabel('Loss')
     plt.ylim(0,1)
     plt.savefig(dataset+'_loss_curve.png')
     plt.show()


     plt.plot(history[:,2:4])
     plt.legend(['Tr Accuracy', 'Val Accuracy'])
     plt.xlabel('Epoch Number')
     plt.ylabel('Accuracy')
     plt.ylim(0,1)
     plt.savefig(dataset+'_accuracy_curve.png')
     plt.show()
```

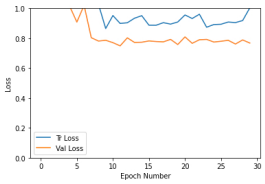
```
Epoch : 018, Training: Loss: 0.9044, Accuracy: 73.6500%,
                Validation : Loss : 0.7759, Accuracy: 79.6000%, Time: 46.8093s
Epoch: 19/30
Epoch : 019, Training: Loss: 0.8947, Accuracy: 73.5000%,
                Validation : Loss : 0.7924, Accuracy: 80.2000%, Time: 46.9615s
Epoch: 20/30
Epoch : 020, Training: Loss: 0.9090, Accuracy: 75.6500%,
                Validation : Loss : 0.7580, Accuracy: 79.8000%, Time: 46.9416s
Epoch: 21/30
Epoch : 021, Training: Loss: 0.9560, Accuracy: 72.8000%,
                Validation : Loss : 0.8092, Accuracy: 78.6000%, Time: 47.0617s
Epoch: 22/30
Epoch : 022, Training: Loss: 0.9322, Accuracy: 73.2500%,
                Validation : Loss : 0.7661, Accuracy: 78.8000%, Time: 46.8114s
Epoch: 23/30
Epoch : 023, Training: Loss: 0.9607, Accuracy: 73.3000%,
                Validation : Loss : 0.7901, Accuracy: 79.6000%, Time: 46.9446s
Epoch: 24/30
Epoch : 024, Training: Loss: 0.8739, Accuracy: 74.7500%,
                Validation : Loss : 0.7924, Accuracy: 78.4000%, Time: 47.8557s
Epoch: 25/30
Epoch : 025, Training: Loss: 0.8918, Accuracy: 74.5500%,
                Validation : Loss : 0.7747, Accuracy: 79.6000%, Time: 48.0402s
Epoch: 26/30
Epoch : 026, Training: Loss: 0.8938, Accuracy: 74.5500%,
                Validation : Loss : 0.7799, Accuracy: 79.4000%, Time: 46.6367s
Epoch: 27/30
Epoch : 027, Training: Loss: 0.9087, Accuracy: 73.0000%,
                Validation : Loss : 0.7862, Accuracy: 80.0000%, Time: 45.4838s
Epoch: 28/30
Epoch : 028, Training: Loss: 0.9049, Accuracy: 73.6500%,
                Validation : Loss : 0.7613, Accuracy: 80.2000%, Time: 45.5587s
Epoch: 29/30
Epoch : 029, Training: Loss: 0.9185, Accuracy: 73.3000%,
                Validation : Loss : 0.7887, Accuracy: 80.2000%, Time: 45.3660s
Epoch: 30/30
Epoch : 030, Training: Loss: 1.0040, Accuracy: 73.6000%,
                Validation : Loss : 0.7679, Accuracy: 80.4000%, Time: 45.6235s
```

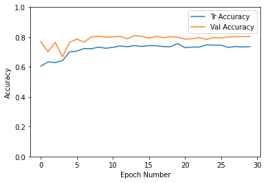







```
[19] history = np.array(history)
     plt.plot(history[:,0:2])
     plt.legend(['Tr Loss', 'Val Loss'])
     plt.xlabel('Epoch Number')
     plt.ylabel('Loss')
     plt.ylim(0,1)
     plt.savefig(dataset+'_loss_curve.png')
     plt.show()


     plt.plot(history[:,2:4])
     plt.legend(['Tr Accuracy', 'Val Accuracy'])
     plt.xlabel('Epoch Number')
     plt.ylabel('Accuracy')
     plt.ylim(0,1)
     plt.savefig(dataset+'_accuracy_curve.png')
     plt.show()
```






```
[ ] def predict(model, test_image_name):
        '''
        Function to predict the class of a single test image
        Parameters
            :param model: Model to test
            :param test_image_name: Test image

        '''

        transform = image_transforms['test']

        test_image = Image.open(test_image_name)
        plt.imshow(test_image)

        test_image_tensor = transform(test_image)

        if torch.cuda.is_available():
            test_image_tensor = test_image_tensor.view(1, 3, 224, 224).cuda()
        else:
            test_image_tensor = test_image_tensor.view(1, 3, 224, 224)

        with torch.no_grad():
            model.eval()
            # Model outputs log probabilities
            out = model(test_image_tensor)
            ps = torch.exp(out)
            topk, topclass = ps.topk(3, dim=1)
            for i in range(3):
                print("Predcition", i+1, ":", idx_to_class[topclass.numpy()[0][i]], ", Score: ", topk.numpy()[0][i])
```

```
[ ] # predict(trained_model, 'caltec256subset/test/009_0098.jpg')
    # predict(trained_model, 'caltec256subset/test/090_0107.jpg')
    # predict(trained_model, 'caltec256subset/test/150_0044.jpg')
```