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

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

```
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: numpy in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (1.19.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (3.2.2)
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: 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: 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: 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: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (0.11.0)
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: 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-yakor88j
Created temporary directory: /tmp/pip-req-tracker-fbftwcis
Initialized build tracking at /tmp/pip-req-tracker-fbftwcis
Created build tracker: /tmp/pip-req-tracker-fbftwcis
Entered build tracker: /tmp/pip-req-tracker-fbftwcis
Created temporary directory: /tmp/pip-install-x9z0jh4g
Requirement already satisfied: torch-lr-finder in /usr/local/lib/python3.7/dist-packages (0.2.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (1.19.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (4.62.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (3.2.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from torch-lr-finder) (21.3)
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: 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: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->torch-lr-finder) (0.11.0)
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: 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-v8sqfdlz
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-fbftwcis'
```

```
[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 13:05:41 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   34C    P0    26W / 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])
    ])
}
```

[8]
```
# Load the Data

# Set train and valid directory paths

dataset = '/content/gdrive/MyDrive/c1'

#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)
```

```
2
{0: 'les', 1: 'nv'}
```

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

```
(4800, 1200)
```

[13]
```
model = models.alexnet(pretrained=True)
model.to(device)
model
```

```
Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to /root/.cache/torch/hub/checkpoints/alexnet-owt-7be5be79.pth
100% 233M/233M [00:01<00:00, 233MB/s]
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

[14]
```
# Freeze model parameters
for param in model.parameters():
    param.requires_grad = False
```

[15]
```
# 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))
model.to(device)
model
```

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=2, bias=True)
    (7): LogSoftmax(dim=1)
  )
)
```

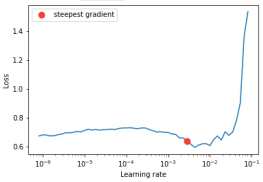
[16]
```
summary(model, (3, 224, 224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 55, 55]          23,296
              ReLU-2          [-1, 64, 55, 55]               0
         MaxPool2d-3          [-1, 64, 27, 27]               0
            Conv2d-4         [-1, 192, 27, 27]         307,392
              ReLU-5         [-1, 192, 27, 27]               0
         MaxPool2d-6         [-1, 192, 13, 13]               0
            Conv2d-7         [-1, 384, 13, 13]         663,936
              ReLU-8         [-1, 384, 13, 13]               0
            Conv2d-9         [-1, 256, 13, 13]         884,992
             ReLU-10         [-1, 256, 13, 13]               0
           Conv2d-11         [-1, 256, 13, 13]         590,080
             ReLU-12         [-1, 256, 13, 13]               0
        MaxPool2d-13           [-1, 256, 6, 6]               0
AdaptiveAvgPool2d-14           [-1, 256, 6, 6]               0
          Dropout-15                [-1, 9216]               0
           Linear-16                [-1, 4096]      37,752,832
             ReLU-17                [-1, 4096]               0
          Dropout-18                [-1, 4096]               0
```

```
           Linear-19                [-1, 4096]      16,781,312
             ReLU-20                [-1, 4096]               0
           Linear-21                   [-1, 2]           8,194
       LogSoftmax-22                   [-1, 2]               0
================================================================
Total params: 57,012,034
Trainable params: 8,194
Non-trainable params: 57,003,840
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 8.37
Params size (MB): 217.48
Estimated Total Size (MB): 226.43
----------------------------------------------------------------
```

[17]
```
# 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
```

```
70% 70/100 [08:29<03:16, 6.56s/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-03
```




```
(<matplotlib.axes._subplots.AxesSubplot at 0x7fa72a77c090>,
 0.002848035868435803)
```

[18]
```
optimizer = optim.Adam(model.parameters(), lr = 2.85E-03)
```

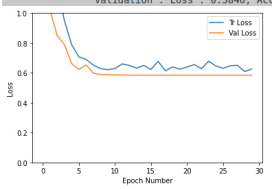
[19]
```
decaying_lr = lr_scheduler.StepLR(optimizer, step_size = 4, gamma = 0.1)
```

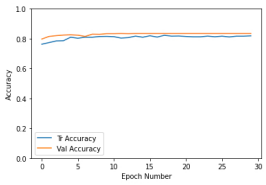
```
def train_and_validate(model, loss_criterion, optimizer, scheduler, epochs):
    '''
    Function to train and validate
    Parameters
        :param model: Model to train and validate
        :param loss_criterion: Loss Criterion to minimize
        :param optimizer: Optimizer for computing gradients
        :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
```

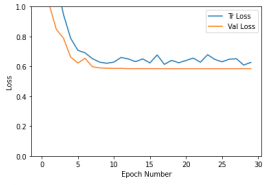
```
num_epochs = Number_of_Epoch
trained_model, history = train_and_validate(model, loss_func, optimizer, decaying_lr, num_epochs)

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()
```

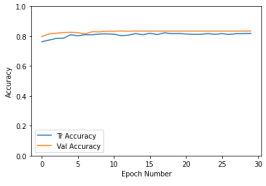
```
Epoch : 022, Training: Loss: 0.6556, Accuracy: 81.2292%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.5055s
Epoch: 23/30
Epoch : 023, Training: Loss: 0.6277, Accuracy: 81.2292%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.2805s
Epoch: 24/30
Epoch : 024, Training: Loss: 0.6781, Accuracy: 81.7083%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.9448s
Epoch: 25/30
Epoch : 025, Training: Loss: 0.6466, Accuracy: 81.2708%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.9627s
Epoch: 26/30
Epoch : 026, Training: Loss: 0.6303, Accuracy: 81.6667%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.8035s
Epoch: 27/30
Epoch : 027, Training: Loss: 0.6479, Accuracy: 81.1458%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.8874s
Epoch: 28/30
Epoch : 028, Training: Loss: 0.6508, Accuracy: 81.6667%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.7168s
Epoch: 29/30
Epoch : 029, Training: Loss: 0.6089, Accuracy: 81.6667%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 92.6678s
Epoch: 30/30
Epoch : 030, Training: Loss: 0.6263, Accuracy: 81.8750%,
                Validation : Loss : 0.5846, Accuracy: 83.4167%, Time: 93.4204s
```






```
[22] 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')
```