DAI Assignment-1 Report

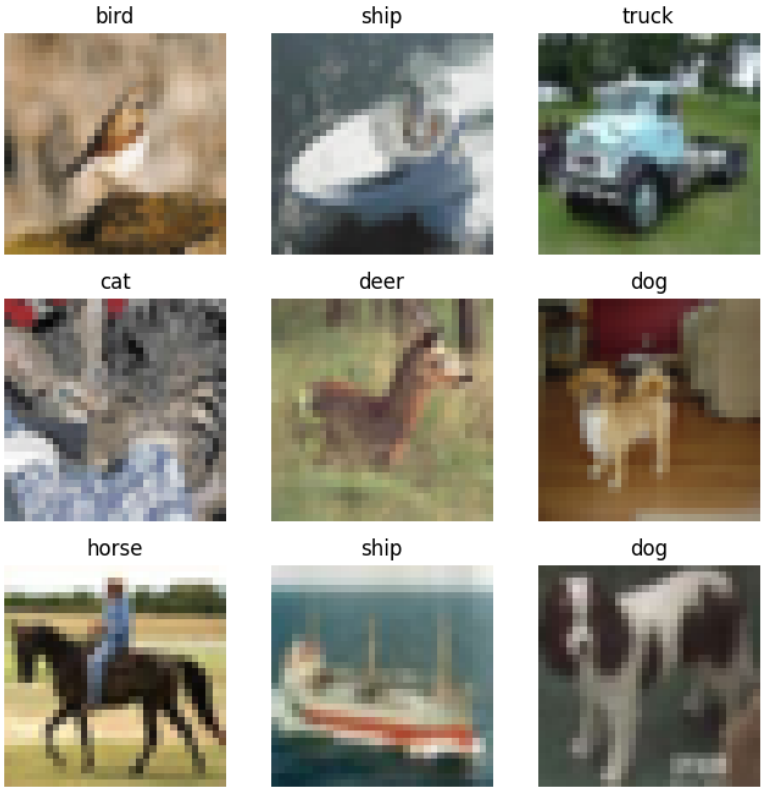
Task: Adversarial Perturbations: Using CIFAR10 dataset and protocol

Objectives:

- **implement FGSM on your own and using a basic CNN architecture with three convolution layers, you need to show the impact of the attack.**
- **Use two deep learning architectures of your choice, PGD attack and one other then FGSM.**
- **Using the SVHN dataset, you need to show the impact of the attacks and compare and contrast them**

Procedure:

- Import required packages. (PyTorch, NumPy, Matplotlib, torchmetrics …etc.)
- Setting up device-agnostic code for faster training.
- Downloading the Dataset using PyTorch vision datasets.
- Visualizing random data samples from the dataset.
- Making compose transform to convert image to tensors
- Converting the data to torch. Tensors



- Making data loaders from the datasets.
- Defining Basic CNN-Architecture Class Definition.
- Defining training and testing steps for training method.
- Creating the model object defined above, and printing model summary.

```
==========================================================================
==================================================
Layer (type (var_name))                  Input Shape          Output Shape          Param #
Trainable
==========================================================================
==================================================
BasicCNN (BasicCNN)                      [32, 3, 32, 32]      [32, 10]              --
True
├─Sequential (conv_block1)               [32, 3, 32, 32]      [32, 10, 16, 16]      --
True
│    └─Conv2d (0)                        [32, 3, 32, 32]      [32, 10, 32, 32]      280
True
│    └─ReLU (1)                          [32, 10, 32, 32]     [32, 10, 32, 32]      --
--
│    └─Conv2d (2)                        [32, 10, 32, 32]     [32, 10, 32, 32]      910
True
│    └─ReLU (3)                          [32, 10, 32, 32]     [32, 10, 32, 32]      --
--
│    └─MaxPool2d (4)                     [32, 10, 32, 32]     [32, 10, 16, 16]      --
--
├─Sequential (conv_block2)               [32, 10, 16, 16]     [32, 10, 8, 8]        --
True
│    └─Conv2d (0)                        [32, 10, 16, 16]     [32, 10, 16, 16]      910
True
│    └─ReLU (1)                          [32, 10, 16, 16]     [32, 10, 16, 16]      --
--
│    └─MaxPool2d (2)                     [32, 10, 16, 16]     [32, 10, 8, 8]        --
--
├─Sequential (classifier)                [32, 10, 8, 8]       [32, 10]              --
True
│    └─Flatten (0)                       [32, 10, 8, 8]       [32, 640]             --
--
│    └─Linear (1)                        [32, 640]            [32, 10]              6,410
True
│    └─ReLU (2)                          [32, 10]             [32, 10]              --
--
│    └─Linear (3)                        [32, 10]             [32, 10]              110
True
==========================================================================
==================================================
Total params: 8,620
Trainable params: 8,620
Non-trainable params: 0
Total mult-adds (M): 46.66
==========================================================================
==================================================
Input size (MB): 0.39
Forward/backward pass size (MB): 5.90
Params size (MB): 0.03
Estimated Total Size (MB): 6.33
==========================================================================
==================================================
```

- Training the model with cross CrossEntropyLoss and Adam optimizer.

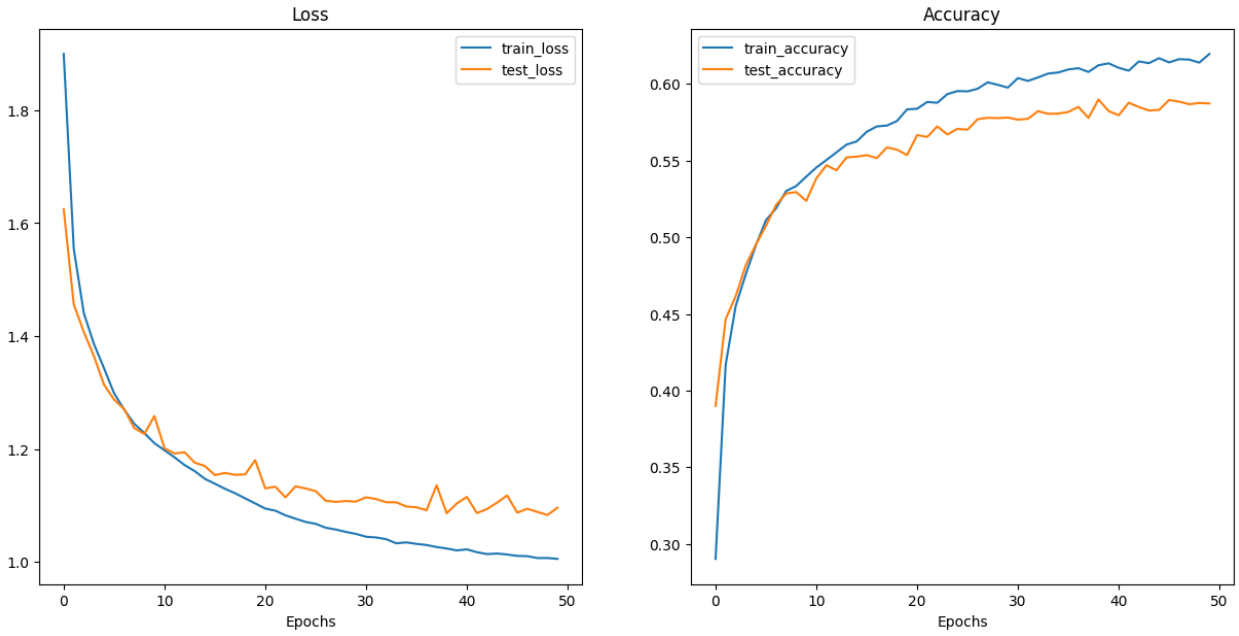
- Training Results:

Epoch: 1 | train_loss: 1.8998 | train_acc: 0.2903 | test_loss: 1.6249 | test_acc: 0.3900
Epoch: 2 | train_loss: 1.5543 | train_acc: 0.4167 | test_loss: 1.4560 | test_acc: 0.4466
Epoch: 3 | train_loss: 1.4405 | train_acc: 0.4552 | test_loss: 1.4071 | test_acc: 0.4616
Epoch: 4 | train_loss: 1.3858 | train_acc: 0.4759 | test_loss: 1.3644 | test_acc: 0.4818
Epoch: 5 | train_loss: 1.3429 | train_acc: 0.4948 | test_loss: 1.3139 | test_acc: 0.4952
Epoch: 6 | train_loss: 1.2986 | train_acc: 0.5111 | test_loss: 1.2874 | test_acc: 0.5075
Epoch: 7 | train_loss: 1.2697 | train_acc: 0.5188 | test_loss: 1.2707 | test_acc: 0.5210
Epoch: 8 | train_loss: 1.2446 | train_acc: 0.5301 | test_loss: 1.2369 | test_acc: 0.5285
Epoch: 9 | train_loss: 1.2282 | train_acc: 0.5334 | test_loss: 1.2266 | test_acc: 0.5294
Epoch: 10 | train_loss: 1.2102 | train_acc: 0.5395 | test_loss: 1.2583 | test_acc: 0.5237
Epoch: 11 | train_loss: 1.1976 | train_acc: 0.5455 | test_loss: 1.2013 | test_acc: 0.5384
Epoch: 12 | train_loss: 1.1851 | train_acc: 0.5503 | test_loss: 1.1918 | test_acc: 0.5469
Epoch: 13 | train_loss: 1.1714 | train_acc: 0.5554 | test_loss: 1.1942 | test_acc: 0.5437
Epoch: 14 | train_loss: 1.1607 | train_acc: 0.5604 | test_loss: 1.1757 | test_acc: 0.5521
Epoch: 15 | train_loss: 1.1471 | train_acc: 0.5625 | test_loss: 1.1699 | test_acc: 0.5525
Epoch: 16 | train_loss: 1.1384 | train_acc: 0.5688 | test_loss: 1.1537 | test_acc: 0.5535
Epoch: 17 | train_loss: 1.1294 | train_acc: 0.5722 | test_loss: 1.1575 | test_acc: 0.5516
Epoch: 18 | train_loss: 1.1216 | train_acc: 0.5728 | test_loss: 1.1542 | test_acc: 0.5586
Epoch: 19 | train_loss: 1.1125 | train_acc: 0.5758 | test_loss: 1.1549 | test_acc: 0.5571
Epoch: 20 | train_loss: 1.1035 | train_acc: 0.5834 | test_loss: 1.1800 | test_acc: 0.5536
Epoch: 21 | train_loss: 1.0944 | train_acc: 0.5838 | test_loss: 1.1302 | test_acc: 0.5666
Epoch: 22 | train_loss: 1.0907 | train_acc: 0.5882 | test_loss: 1.1331 | test_acc: 0.5654
Epoch: 23 | train_loss: 1.0824 | train_acc: 0.5877 | test_loss: 1.1142 | test_acc: 0.5723
Epoch: 24 | train_loss: 1.0765 | train_acc: 0.5933 | test_loss: 1.1337 | test_acc: 0.5670
Epoch: 25 | train_loss: 1.0707 | train_acc: 0.5953 | test_loss: 1.1300 | test_acc: 0.5707
Epoch: 26 | train_loss: 1.0674 | train_acc: 0.5951 | test_loss: 1.1253 | test_acc: 0.5702
Epoch: 27 | train_loss: 1.0604 | train_acc: 0.5968 | test_loss: 1.1083 | test_acc: 0.5770
Epoch: 28 | train_loss: 1.0573 | train_acc: 0.6010 | test_loss: 1.1062 | test_acc: 0.5779
Epoch: 29 | train_loss: 1.0530 | train_acc: 0.5994 | test_loss: 1.1077 | test_acc: 0.5777
Epoch: 30 | train_loss: 1.0495 | train_acc: 0.5976 | test_loss: 1.1067 | test_acc: 0.5780
Epoch: 31 | train_loss: 1.0444 | train_acc: 0.6038 | test_loss: 1.1144 | test_acc: 0.5767
Epoch: 32 | train_loss: 1.0432 | train_acc: 0.6019 | test_loss: 1.1112 | test_acc: 0.5772
Epoch: 33 | train_loss: 1.0402 | train_acc: 0.6042 | test_loss: 1.1056 | test_acc: 0.5822
Epoch: 34 | train_loss: 1.0329 | train_acc: 0.6067 | test_loss: 1.1055 | test_acc: 0.5805
Epoch: 35 | train_loss: 1.0345 | train_acc: 0.6074 | test_loss: 1.0981 | test_acc: 0.5806
Epoch: 36 | train_loss: 1.0319 | train_acc: 0.6094 | test_loss: 1.0967 | test_acc: 0.5817
Epoch: 37 | train_loss: 1.0298 | train_acc: 0.6102 | test_loss: 1.0914 | test_acc: 0.5850
Epoch: 38 | train_loss: 1.0262 | train_acc: 0.6078 | test_loss: 1.1356 | test_acc: 0.5778
Epoch: 39 | train_loss: 1.0236 | train_acc: 0.6122 | test_loss: 1.0860 | test_acc: 0.5898
Epoch: 40 | train_loss: 1.0203 | train_acc: 0.6134 | test_loss: 1.1033 | test_acc: 0.5823
Epoch: 41 | train_loss: 1.0221 | train_acc: 0.6105 | test_loss: 1.1150 | test_acc: 0.5795
Epoch: 42 | train_loss: 1.0172 | train_acc: 0.6086 | test_loss: 1.0864 | test_acc: 0.5877
Epoch: 43 | train_loss: 1.0137 | train_acc: 0.6146 | test_loss: 1.0935 | test_acc: 0.5849

Epoch: 44 | train_loss: 1.0148 | train_acc: 0.6135 | test_loss: 1.1049 | test_acc: 0.5827
Epoch: 45 | train_loss: 1.0129 | train_acc: 0.6167 | test_loss: 1.1177 | test_acc: 0.5830
Epoch: 46 | train_loss: 1.0106 | train_acc: 0.6140 | test_loss: 1.0872 | test_acc: 0.5895
Epoch: 47 | train_loss: 1.0101 | train_acc: 0.6161 | test_loss: 1.0941 | test_acc: 0.5884
Epoch: 48 | train_loss: 1.0068 | train_acc: 0.6158 | test_loss: 1.0885 | test_acc: 0.5867
Epoch: 49 | train_loss: 1.0068 | train_acc: 0.6139 | test_loss: 1.0830 | test_acc: 0.5875
Epoch: 50 | train_loss: 1.0053 | train_acc: 0.6195 | test_loss: 1.0692 | test_acc: 0.6029
total training time: 603.354 sec.

- Saving the trained model, loading, and re-evaluating the training.

test_loss: 1.0692 | test_acc: 0.6029



- Now, once the model has been trained begin, Attacking the Model.
- Setting epsilons for FGSM

$perturbed\_image = image + epsilon * sign(data\_grad) = x + \epsilon * sign(\nabla x J(\theta, \mathbf{x}, y))$

Finally, in order to maintain the original range of the data, the perturbed image is clipped to range [0,1].

- Defining the attack function.

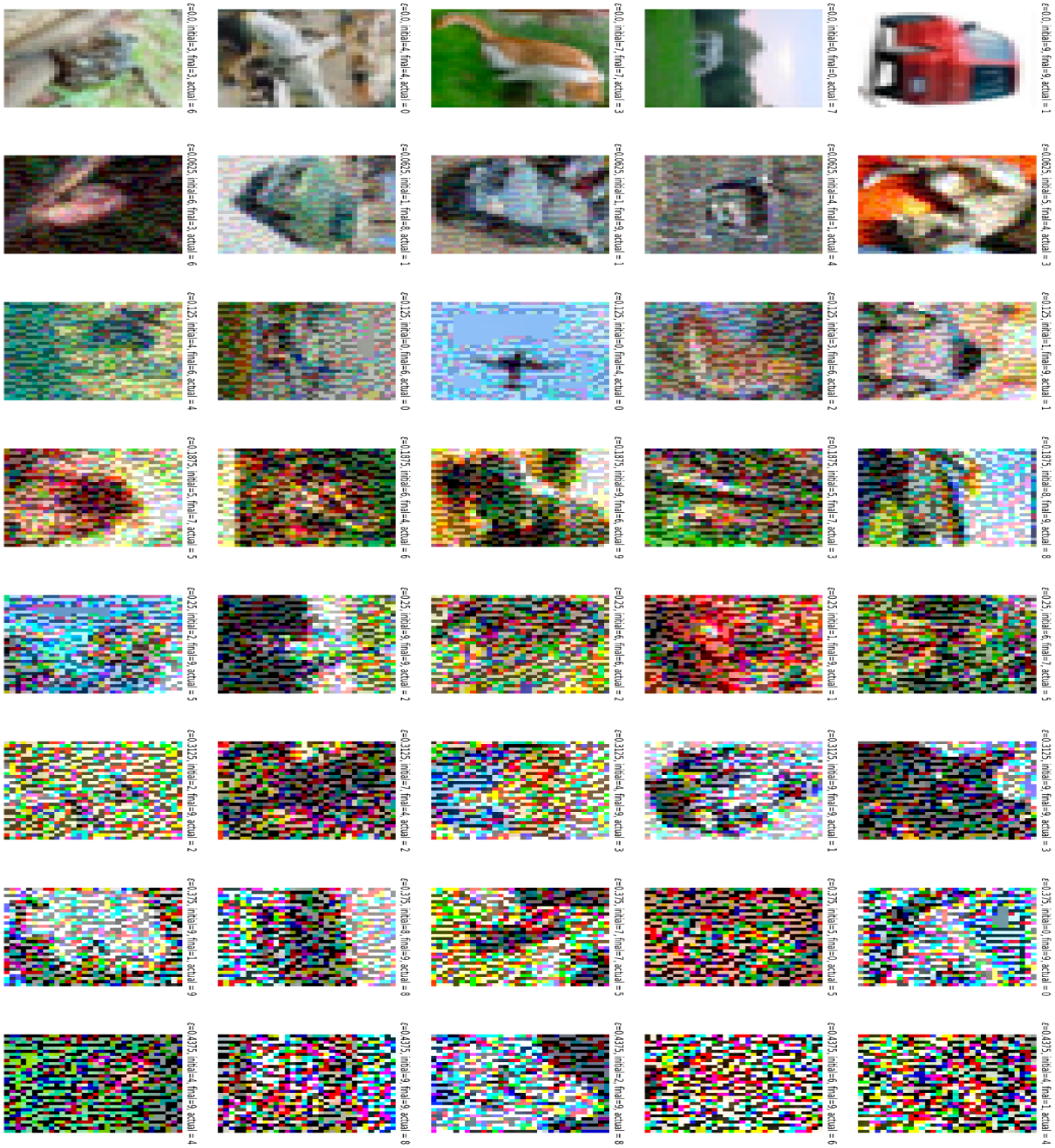
Python

```
# FGSM attack
def fgsm_attack(image: torch.Tensor, epsilon:torch.Tensor,
data_grad:torch.Tensor) -> torch.Tensor:
    # Collect the element-wise sign of the data gradient
    sign_data_grad = data_grad.sign()
    # Create the perturbed image by adjusting each pixel of the
input image
    perturbed_image = image + epsilon*sign_data_grad
    # Adding clipping to maintain [0,1] range
    perturbed_image = torch.clamp(perturbed_image, 0, 1)
    # Return the perturbed image
    return perturbed_image
```

- Testing the attack on the trained model, with different epsilons values.
- Testing Results:

Attacked Examples: 3752
test_loss: 1.0692 | test_acc: 0.6029 | epsilon: 0.0000
Attacked Examples: 9682
test_loss: 1.0692 | test_acc: 0.0310 | epsilon: 0.0625
Attacked Examples: 9466
test_loss: 1.0692 | test_acc: 0.0521 | epsilon: 0.1250
Attacked Examples: 9336
test_loss: 1.0692 | test_acc: 0.0641 | epsilon: 0.1875
Attacked Examples: 9296
test_loss: 1.0692 | test_acc: 0.0680 | epsilon: 0.2500
Attacked Examples: 9296
test_loss: 1.0692 | test_acc: 0.0689 | epsilon: 0.3125
Attacked Examples: 9265
test_loss: 1.0692 | test_acc: 0.0730 | epsilon: 0.3750
Attacked Examples: 9249
test_loss: 1.0692 | test_acc: 0.0737 | epsilon: 0.4375

- Ploting attacked Examples.



Procedure For task 2 (Attack 2 Arch.) :

- Import required packages. (PyTorch, NumPy, Matplotlib, torchmetrics …etc.)
- Setting up device-agnostic code for faster training.
- Downloading the Dataset using PyTorch vision datasets.
- Visualizing random data samples from the dataset.
- Making compose transform to convert image to tensors
- Converting the data to torch. Tensors

- Making data loaders from the datasets.
- Using trochvision models EfficientNet_B0 & ResNet34 for the task with pre-trained weights and pre-defined transforms.
- Model summaries for both the models are:

```
==========================================================================================
Layer (type (var_name))                  Input Shape          Output Shape         Param #
Trainable
==========================================================================================
EfficientNet (EfficientNet)              [32, 3, 224, 224]    [32, 10]             --
Partial
├─Sequential (features)                  [32, 3, 224, 224]    [32, 1280, 7, 7]     --
False
│    └─Conv2dNormActivation (0)          [32, 3, 224, 224]    [32, 32, 112, 112]   --
False
│    │    └─Conv2d (0)                   [32, 3, 224, 224]    [32, 32, 112, 112]   (864)
False
│    │    └─BatchNorm2d (1)              [32, 32, 112, 112]   [32, 32, 112, 112]   (64)
False
```

```
|    |    └─SiLU (2)                    [32, 32, 112, 112]  [32, 32, 112, 112]  --
--
|    └─Sequential (1)                   [32, 32, 112, 112]  [32, 16, 112, 112]  --
False
|    |    └─MBConv (0)                  [32, 32, 112, 112]  [32, 16, 112, 112]  (1,448)
False
|    └─Sequential (2)                   [32, 16, 112, 112]  [32, 24, 56, 56]    --
False
|    |    └─MBConv (0)                  [32, 16, 112, 112]  [32, 24, 56, 56]    (6,004)
False
|    |    └─MBConv (1)                  [32, 24, 56, 56]    [32, 24, 56, 56]    (10,710)
False
|    └─Sequential (3)                   [32, 24, 56, 56]    [32, 40, 28, 28]    --
False
|    |    └─MBConv (0)                  [32, 24, 56, 56]    [32, 40, 28, 28]    (15,350)
False
|    |    └─MBConv (1)                  [32, 40, 28, 28]    [32, 40, 28, 28]    (31,290)
False
|    └─Sequential (4)                   [32, 40, 28, 28]    [32, 80, 14, 14]    --
False
|    |    └─MBConv (0)                  [32, 40, 28, 28]    [32, 80, 14, 14]    (37,130)
False
|    |    └─MBConv (1)                  [32, 80, 14, 14]    [32, 80, 14, 14]    (102,900)
False
|    |    └─MBConv (2)                  [32, 80, 14, 14]    [32, 80, 14, 14]    (102,900)
False
|    └─Sequential (5)                   [32, 80, 14, 14]    [32, 112, 14, 14]   --
False
|    |    └─MBConv (0)                  [32, 80, 14, 14]    [32, 112, 14, 14]   (126,004)
False
|    |    └─MBConv (1)                  [32, 112, 14, 14]   [32, 112, 14, 14]   (208,572)
False
|    |    └─MBConv (2)                  [32, 112, 14, 14]   [32, 112, 14, 14]   (208,572)
False
|    └─Sequential (6)                   [32, 112, 14, 14]   [32, 192, 7, 7]     --
False
|    |    └─MBConv (0)                  [32, 112, 14, 14]   [32, 192, 7, 7]     (262,492)
False
|    |    └─MBConv (1)                  [32, 192, 7, 7]     [32, 192, 7, 7]     (587,952)
False
|    |    └─MBConv (2)                  [32, 192, 7, 7]     [32, 192, 7, 7]     (587,952)
False
|    |    └─MBConv (3)                  [32, 192, 7, 7]     [32, 192, 7, 7]     (587,952)
False
```

```
│    └─Sequential (7)                                  [32, 192, 7, 7]      [32, 320, 7, 7]      --
False
│    │    └─MBConv (0)                                 [32, 192, 7, 7]      [32, 320, 7, 7]      (717,232)
False
│    └─Conv2dNormActivation (8)                        [32, 320, 7, 7]      [32, 1280, 7, 7]     --
False
│    │    └─Conv2d (0)                                 [32, 320, 7, 7]      [32, 1280, 7, 7]     (409,600)
False
│    │    └─BatchNorm2d (1)                            [32, 1280, 7, 7]     [32, 1280, 7, 7]     (2,560)
False
│    │    └─SiLU (2)                                   [32, 1280, 7, 7]     [32, 1280, 7, 7]     --
--
├─AdaptiveAvgPool2d (avgpool)                          [32, 1280, 7, 7]     [32, 1280, 1, 1]     --
--
├─Sequential (classifier)                              [32, 1280]           [32, 10]             --
True
│    └─Dropout (0)                                     [32, 1280]           [32, 1280]           --                   --
│    └─Linear (1)                                      [32, 1280]           [32, 10]             12,810
True
==========================================================================
==========================================================================
Total params: 4,020,358
Trainable params: 12,810
Non-trainable params: 4,007,548
Total mult-adds (G): 12.31
==========================================================================
==========================================================================
Input size (MB): 19.27
Forward/backward pass size (MB): 3452.09
Params size (MB): 16.08
Estimated Total Size (MB): 3487.44
==========================================================================
==========================================================================


==========================================================================
================================================
Layer (type (var_name))                  Input Shape          Output Shape         Param #
Trainable
==========================================================================
================================================
ResNet (ResNet)                          [32, 3, 224, 224]    [32, 10]             --                   Partial
├─Conv2d (conv1)                         [32, 3, 224, 224]    [32, 64, 112, 112]   (9,408)              False
├─BatchNorm2d (bn1)                      [32, 64, 112, 112]   [32, 64, 112, 112]   (128)                False
```

```
├─ReLU (relu)                    [32, 64, 112, 112]   [32, 64, 112, 112]   --                  --
├─MaxPool2d (maxpool)            [32, 64, 112, 112]   [32, 64, 56, 56]     --                  --
├─Sequential (layer1)            [32, 64, 56, 56]     [32, 64, 56, 56]     --                  False
│    └─BasicBlock (0)            [32, 64, 56, 56]     [32, 64, 56, 56]     --                  False
│    │    └─Conv2d (conv1)       [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)            False
│    │    └─BatchNorm2d (bn1)    [32, 64, 56, 56]     [32, 64, 56, 56]     (128)               False
│    │    └─ReLU (relu)          [32, 64, 56, 56]     [32, 64, 56, 56]     --                  --
│    │    └─Conv2d (conv2)       [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)            False
│    │    └─BatchNorm2d (bn2)    [32, 64, 56, 56]     [32, 64, 56, 56]     (128)               False
│    │    └─ReLU (relu)          [32, 64, 56, 56]     [32, 64, 56, 56]     --                  --
│    └─BasicBlock (1)            [32, 64, 56, 56]     [32, 64, 56, 56]     --                  False
│    │    └─Conv2d (conv1)       [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)            False
│    │    └─BatchNorm2d (bn1)    [32, 64, 56, 56]     [32, 64, 56, 56]     (128)               False
│    │    └─ReLU (relu)          [32, 64, 56, 56]     [32, 64, 56, 56]     --                  --
│    │    └─Conv2d (conv2)       [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)            False
│    │    └─BatchNorm2d (bn2)    [32, 64, 56, 56]     [32, 64, 56, 56]     (128)               False
│    │    └─ReLU (relu)          [32, 64, 56, 56]     [32, 64, 56, 56]     --                  --
│    └─BasicBlock (2)            [32, 64, 56, 56]     [32, 64, 56, 56]     --                  False
│    │    └─Conv2d (conv1)       [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)            False
│    │    └─BatchNorm2d (bn1)    [32, 64, 56, 56]     [32, 64, 56, 56]     (128)               False
│    │    └─ReLU (relu)          [32, 64, 56, 56]     [32, 64, 56, 56]     --                  --
│    │    └─Conv2d (conv2)       [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)            False
│    │    └─BatchNorm2d (bn2)    [32, 64, 56, 56]     [32, 64, 56, 56]     (128)               False
│    │    └─ReLU (relu)          [32, 64, 56, 56]     [32, 64, 56, 56]     --                  --
├─Sequential (layer2)            [32, 64, 56, 56]     [32, 128, 28, 28]    --                  False
│    └─BasicBlock (0)            [32, 64, 56, 56]     [32, 128, 28, 28]    --                  False
│    │    └─Conv2d (conv1)       [32, 64, 56, 56]     [32, 128, 28, 28]    (73,728)            False
│    │    └─BatchNorm2d (bn1)    [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    │    └─ReLU (relu)          [32, 128, 28, 28]    [32, 128, 28, 28]    --                  --
│    │    └─Conv2d (conv2)       [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn2)    [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    │    └─Sequential (downsample)  [32, 64, 56, 56]  [32, 128, 28, 28]   (8,448)
False
│    │    └─ReLU (relu)          [32, 128, 28, 28]    [32, 128, 28, 28]    --                  --
│    └─BasicBlock (1)            [32, 128, 28, 28]    [32, 128, 28, 28]    --                  False
│    │    └─Conv2d (conv1)       [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn1)    [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    │    └─ReLU (relu)          [32, 128, 28, 28]    [32, 128, 28, 28]    --                  --
```

```
│    │    └─Conv2d (conv2)                [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn2)             [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    └─ReLU (relu)                        [32, 128, 28, 28]    [32, 128, 28, 28]    --                --
│    └─BasicBlock (2)                     [32, 128, 28, 28]    [32, 128, 28, 28]    --                False
│    │    └─Conv2d (conv1)                [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn1)             [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    └─ReLU (relu)                        [32, 128, 28, 28]    [32, 128, 28, 28]    --                --
│    │    └─Conv2d (conv2)                [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn2)             [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    └─ReLU (relu)                        [32, 128, 28, 28]    [32, 128, 28, 28]    --                --
│    └─BasicBlock (3)                     [32, 128, 28, 28]    [32, 128, 28, 28]    --                False
│    │    └─Conv2d (conv1)                [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn1)             [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    └─ReLU (relu)                        [32, 128, 28, 28]    [32, 128, 28, 28]    --                --
│    │    └─Conv2d (conv2)                [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn2)             [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    └─ReLU (relu)                        [32, 128, 28, 28]    [32, 128, 28, 28]    --                --
├─Sequential (layer3)                     [32, 128, 28, 28]    [32, 256, 14, 14]    --                False
│    └─BasicBlock (0)                     [32, 128, 28, 28]    [32, 256, 14, 14]    --                False
│    │    └─Conv2d (conv1)                [32, 128, 28, 28]    [32, 256, 14, 14]    (294,912)
False
│    │    └─BatchNorm2d (bn1)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    │    └─Conv2d (conv2)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
│    │    └─BatchNorm2d (bn2)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    │    └─Sequential (downsample)       [32, 128, 28, 28]    [32, 256, 14, 14]    (33,280)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    └─BasicBlock (1)                     [32, 256, 14, 14]    [32, 256, 14, 14]    --                False
│    │    └─Conv2d (conv1)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
```

```
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    └─BasicBlock (2)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             False
│    │    └─Conv2d (conv1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    └─BasicBlock (3)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             False
│    │    └─Conv2d (conv1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    └─BasicBlock (4)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             False
│    │    └─Conv2d (conv1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    └─BasicBlock (5)              [32, 256, 14, 14]   [32, 256, 14, 14]   --             False
│    │    └─Conv2d (conv1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
```

```
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │    └─ReLU (relu)                    [32, 256, 14, 14]   [32, 256, 14, 14]   --                --
│    │    └─Conv2d (conv2)                 [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │    └─ReLU (relu)                    [32, 256, 14, 14]   [32, 256, 14, 14]   --                --
├─Sequential (layer4)                      [32, 256, 14, 14]   [32, 512, 7, 7]     --                False
│    └─BasicBlock (0)                      [32, 256, 14, 14]   [32, 512, 7, 7]     --                False
│    │    └─Conv2d (conv1)                 [32, 256, 14, 14]   [32, 512, 7, 7]     (1,179,648)
False
│    │    └─BatchNorm2d (bn1)              [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)           False
│    │    └─ReLU (relu)                    [32, 512, 7, 7]     [32, 512, 7, 7]     --                --
│    │    └─Conv2d (conv2)                 [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)       False
│    │    └─BatchNorm2d (bn2)              [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)           False
│    │    └─Sequential (downsample)        [32, 256, 14, 14]   [32, 512, 7, 7]     (132,096)
False
│    │    └─ReLU (relu)                    [32, 512, 7, 7]     [32, 512, 7, 7]     --                --
│    └─BasicBlock (1)                      [32, 512, 7, 7]     [32, 512, 7, 7]     --                False
│    │    └─Conv2d (conv1)                 [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)       False
│    │    └─BatchNorm2d (bn1)              [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)           False
│    │    └─ReLU (relu)                    [32, 512, 7, 7]     [32, 512, 7, 7]     --                --
│    │    └─Conv2d (conv2)                 [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)       False
│    │    └─BatchNorm2d (bn2)              [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)           False
│    │    └─ReLU (relu)                    [32, 512, 7, 7]     [32, 512, 7, 7]     --                --
│    └─BasicBlock (2)                      [32, 512, 7, 7]     [32, 512, 7, 7]     --                False
│    │    └─Conv2d (conv1)                 [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)       False
│    │    └─BatchNorm2d (bn1)              [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)           False
│    │    └─ReLU (relu)                    [32, 512, 7, 7]     [32, 512, 7, 7]     --                --
│    │    └─Conv2d (conv2)                 [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)       False
│    │    └─BatchNorm2d (bn2)              [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)           False
│    │    └─ReLU (relu)                    [32, 512, 7, 7]     [32, 512, 7, 7]     --                --
├─AdaptiveAvgPool2d (avgpool)              [32, 512, 7, 7]     [32, 512, 1, 1]     --                --
├─Linear (fc)                              [32, 512]           [32, 10]            5,130             True
==========================================================================
==============================================
Total params: 21,289,802
Trainable params: 5,130
Non-trainable params: 21,284,672
Total mult-adds (G): 117.22
==========================================================================
==============================================
Input size (MB): 19.27
```

Forward/backward pass size (MB): 1913.92
Params size (MB): 85.16
Estimated Total Size (MB): 2018.34
==========
==========

- Training (fine tunning the models for the same) SVHN.
- Training Results for both:

Effnet:
Epoch: 1 | train_loss: 1.7303 | train_acc: 0.3565 | test_loss: 1.4631 | test_acc: 0.4466
Epoch: 2 | train_loss: 1.6190 | train_acc: 0.4052 | test_loss: 1.4546 | test_acc: 0.4525
Epoch: 3 | train_loss: 1.6110 | train_acc: 0.4099 | test_loss: 1.4096 | test_acc: 0.4650
Epoch: 4 | train_loss: 1.6074 | train_acc: 0.4114 | test_loss: 1.4169 | test_acc: 0.4521
Epoch: 5 | train_loss: 1.6020 | train_acc: 0.4132 | test_loss: 1.4224 | test_acc: 0.4500
Epoch: 6 | train_loss: 1.6007 | train_acc: 0.4158 | test_loss: 1.4077 | test_acc: 0.4611
Epoch: 7 | train_loss: 1.6063 | train_acc: 0.4113 | test_loss: 1.3802 | test_acc: 0.4613
Epoch: 8 | train_loss: 1.6057 | train_acc: 0.4110 | test_loss: 1.4265 | test_acc: 0.4570
Epoch: 9 | train_loss: 1.6013 | train_acc: 0.4137 | test_loss: 1.4317 | test_acc: 0.4498
Epoch: 10 | train_loss: 1.6027 | train_acc: 0.4115 | test_loss: 1.3979 | test_acc: 0.4638
Epoch: 11 | train_loss: 1.6032 | train_acc: 0.4131 | test_loss: 1.4110 | test_acc: 0.4630
Epoch: 12 | train_loss: 1.6024 | train_acc: 0.4127 | test_loss: 1.3908 | test_acc: 0.4544

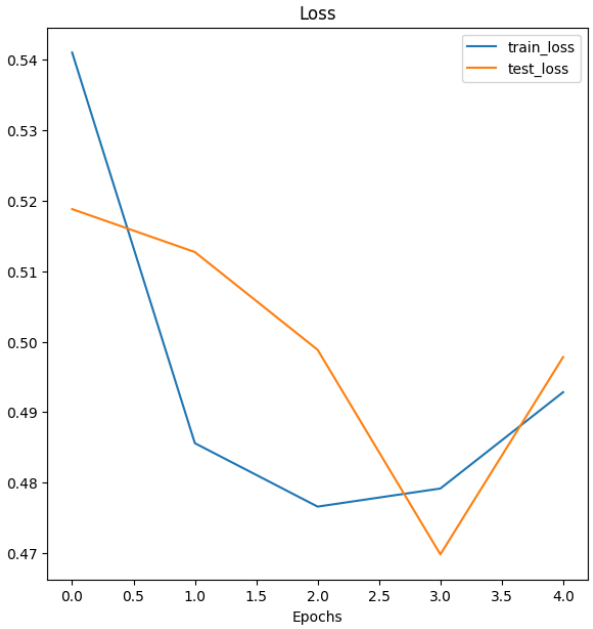
ResNet:
Epoch: 1 | train_loss: 1.8573 | train_acc: 0.2952 | test_loss: 1.7167 | test_acc: 0.3452
Epoch: 2 | train_loss: 1.7235 | train_acc: 0.3520 | test_loss: 1.6756 | test_acc: 0.3541
Epoch: 3 | train_loss: 1.6972 | train_acc: 0.3673 | test_loss: 1.6537 | test_acc: 0.3658
Epoch: 4 | train_loss: 1.6818 | train_acc: 0.3702 | test_loss: 1.6454 | test_acc: 0.3716
Epoch: 5 | train_loss: 1.6712 | train_acc: 0.3762 | test_loss: 1.6442 | test_acc: 0.3835
Epoch: 6 | train_loss: 1.6694 | train_acc: 0.3769 | test_loss: 1.6279 | test_acc: 0.3645
Epoch: 7 | train_loss: 1.6589 | train_acc: 0.3835 | test_loss: 1.6550 | test_acc: 0.3609
Epoch: 8 | train_loss: 1.6581 | train_acc: 0.3827 | test_loss: 1.6174 | test_acc: 0.3942
Epoch: 9 | train_loss: 1.6576 | train_acc: 0.3843 | test_loss: 1.6180 | test_acc: 0.3659
Epoch: 10 | train_loss: 1.6505 | train_acc: 0.3860 | test_loss: 1.6000 | test_acc: 0.3761
Epoch: 11 | train_loss: 1.6520 | train_acc: 0.3853 | test_loss: 1.6444 | test_acc: 0.3671
Epoch: 12 | train_loss: 1.6504 | train_acc: 0.3873 | test_loss: 1.6192 | test_acc: 0.3867
Epoch: 13 | train_loss: 1.6536 | train_acc: 0.3863 | test_loss: 1.5860 | test_acc: 0.3891
Epoch: 14 | train_loss: 1.6505 | train_acc: 0.3853 | test_loss: 1.6426 | test_acc: 0.3721
Epoch: 15 | train_loss: 1.6458 | train_acc: 0.3879 | test_loss: 1.6095 | test_acc: 0.3886
total training time: 1414.568 sec.

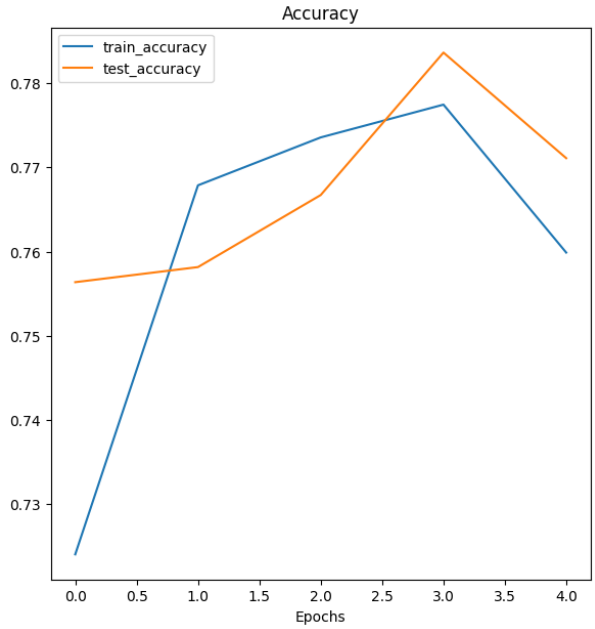
- Saving the trained model, loading, and re-evaluating the training.
- Testing the Impact of Attacks on the models
- Defining attacks effnet PGD:
  Purtubed Test accuracy: 0.0864
- Defining attacks resnet PGD:
  Purtubed Test accuracy: 0.0183
- Defining attacks effnet Jitter:
  Purtubed Test accuracy: 0.0721
- Defining attacks resnet Jitter:
  Purtubed Test accuracy: 0.0362

Procedure For task 3 (detect adversarial attacks):

- Import required packages. (PyTorch, NumPy, Matplotlib, torchmetrics …etc.)
- Setting up device-agnostic code for faster training.
- Downloading the Dataset using PyTorch vision datasets.
- Modify the dataset to make 2 classes, attacked or non-attacked data.
- Modification is done with help of the previous saved model.
- Train the model for 2 class classification for the above dataset.
- Train Results:

Epoch: 1 | train_loss: 0.5410 | train_acc: 0.7240 | test_loss: 0.5188 | test_acc: 0.7564
Epoch: 2 | train_loss: 0.4856 | train_acc: 0.7679 | test_loss: 0.5127 | test_acc: 0.7582
Epoch: 3 | train_loss: 0.4766 | train_acc: 0.7736 | test_loss: 0.4988 | test_acc: 0.7667
Epoch: 4 | train_loss: 0.4792 | train_acc: 0.7775 | test_loss: 0.4698 | test_acc: 0.7836
Epoch: 5 | train_loss: 0.4928 | train_acc: 0.7599 | test_loss: 0.4978 | test_acc: 0.7711
total training time: 75.495 sec.





- Checking FGSM with different epsilons.

Purtubed Test accuracy: 0.7739 with epsilion: 0.0
Purtubed Test accuracy: 0.7751 with epsilion: 0.0625
Purtubed Test accuracy: 0.7705 with epsilion: 0.125
Purtubed Test accuracy: 0.7643 with epsilion: 0.1875
Purtubed Test accuracy: 0.7665 with epsilion: 0.25
Purtubed Test accuracy: 0.7759 with epsilion: 0.3125
Purtubed Test accuracy: 0.7691 with epsilion: 0.375
Purtubed Test accuracy: 0.7598 with epsilion: 0.4375
Purtubed Test accuracy: 0.7785 with epsilion: 0.5
Purtubed Test accuracy: 0.7657 with epsilion: 0.5625
Purtubed Test accuracy: 0.7665 with epsilion: 0.625
Purtubed Test accuracy: 0.7669 with epsilion: 0.6875
Purtubed Test accuracy: 0.7729 with epsilion: 0.75
Purtubed Test accuracy: 0.7757 with epsilion: 0.8125
Purtubed Test accuracy: 0.7701 with epsilion: 0.875
Purtubed Test accuracy: 0.7703 with epsilion: 0.9375

- Doing same with saving and loading model for test.

Purtubed Test accuracy: 0.7739 with epsilion: 0.0
Purtubed Test accuracy: 0.7751 with epsilion: 0.0625
Purtubed Test accuracy: 0.7705 with epsilion: 0.125
Purtubed Test accuracy: 0.7643 with epsilion: 0.1875
Purtubed Test accuracy: 0.7665 with epsilion: 0.25
Purtubed Test accuracy: 0.7759 with epsilion: 0.3125
Purtubed Test accuracy: 0.7691 with epsilion: 0.375
Purtubed Test accuracy: 0.7598 with epsilion: 0.4375
Purtubed Test accuracy: 0.7785 with epsilion: 0.5
Purtubed Test accuracy: 0.7657 with epsilion: 0.5625
Purtubed Test accuracy: 0.7665 with epsilion: 0.625
Purtubed Test accuracy: 0.7669 with epsilion: 0.6875
Purtubed Test accuracy: 0.7729 with epsilion: 0.75
Purtubed Test accuracy: 0.7757 with epsilion: 0.8125
Purtubed Test accuracy: 0.7701 with epsilion: 0.875
Purtubed Test accuracy: 0.7703 with epsilion: 0.9375

Task: Deepfake Detection

Objectives:

- **Create 100 deepfakes/faceswap from the existing tools of your choice using your own face images.**
- **Split this data into fine-tuned and test sets (50-50).**
- **Finetune your model in Q1(iii) and test on the remaining 50 test samples, and report the performance.**

Procedure:

- Import required packages. (PyTorch, NumPy, Matplotlib, torchmetrics …etc.)
- Setting up device-agnostic code for faster training.
- Loading Saved model from question 1, part 3.
- Model Summary:

```
==========================================================================
==============================================
Layer (type (var_name))                  Input Shape          Output Shape         Param #
Trainable
==========================================================================
==============================================
ResNet (ResNet)                          [32, 3, 224, 224]    [32, 2]              --
Partial
├─Conv2d (conv1)                        [32, 3, 224, 224]    [32, 64, 112, 112]   (9,408)
False
├─BatchNorm2d (bn1)                     [32, 64, 112, 112]   [32, 64, 112, 112]   (128)
False
├─ReLU (relu)                           [32, 64, 112, 112]   [32, 64, 112, 112]   --
--
├─MaxPool2d (maxpool)                   [32, 64, 112, 112]   [32, 64, 56, 56]     --
--
├─Sequential (layer1)                   [32, 64, 56, 56]     [32, 64, 56, 56]     --
False
│    └─BasicBlock (0)                   [32, 64, 56, 56]     [32, 64, 56, 56]     --
False
│    │    └─Conv2d (conv1)              [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)
False
│    │    └─BatchNorm2d (bn1)           [32, 64, 56, 56]     [32, 64, 56, 56]     (128)
False
│    │    └─ReLU (relu)                 [32, 64, 56, 56]     [32, 64, 56, 56]     --
--
│    │    └─Conv2d (conv2)              [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)
False
│    │    └─BatchNorm2d (bn2)           [32, 64, 56, 56]     [32, 64, 56, 56]     (128)
False
│    │    └─ReLU (relu)                 [32, 64, 56, 56]     [32, 64, 56, 56]     --
--
│    └─BasicBlock (1)                   [32, 64, 56, 56]     [32, 64, 56, 56]     --
False
│    │    └─Conv2d (conv1)              [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)
False
│    │    └─BatchNorm2d (bn1)           [32, 64, 56, 56]     [32, 64, 56, 56]     (128)
False
│    │    └─ReLU (relu)                 [32, 64, 56, 56]     [32, 64, 56, 56]     --
--
│    │    └─Conv2d (conv2)              [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)
False
│    │    └─BatchNorm2d (bn2)           [32, 64, 56, 56]     [32, 64, 56, 56]     (128)
False
│    │    └─ReLU (relu)                 [32, 64, 56, 56]     [32, 64, 56, 56]     --
--
│    └─BasicBlock (2)                   [32, 64, 56, 56]     [32, 64, 56, 56]     --
False
│    │    └─Conv2d (conv1)              [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)
False
│    │    └─BatchNorm2d (bn1)           [32, 64, 56, 56]     [32, 64, 56, 56]     (128)
False
│    │    └─ReLU (relu)                 [32, 64, 56, 56]     [32, 64, 56, 56]     --
--
│    │    └─Conv2d (conv2)              [32, 64, 56, 56]     [32, 64, 56, 56]     (36,864)
False
```

```
│    │    └─BatchNorm2d (bn2)          [32, 64, 56, 56]    [32, 64, 56, 56]    (128)             False
│    │    └─ReLU (relu)                [32, 64, 56, 56]    [32, 64, 56, 56]    --                --
├─Sequential (layer2)                  [32, 64, 56, 56]    [32, 128, 28, 28]   --                False
│    └─BasicBlock (0)                  [32, 64, 56, 56]    [32, 128, 28, 28]   --                False
│    │    └─Conv2d (conv1)             [32, 64, 56, 56]    [32, 128, 28, 28]   (73,728)          False
│    │    └─BatchNorm2d (bn1)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
│    │    └─Conv2d (conv2)             [32, 128, 28, 28]   [32, 128, 28, 28]   (147,456)
False
│    │    └─BatchNorm2d (bn2)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─Sequential (downsample)    [32, 64, 56, 56]    [32, 128, 28, 28]   (8,448)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
│    └─BasicBlock (1)                  [32, 128, 28, 28]   [32, 128, 28, 28]   --                False
│    │    └─Conv2d (conv1)             [32, 128, 28, 28]   [32, 128, 28, 28]   (147,456)
False
│    │    └─BatchNorm2d (bn1)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
│    │    └─Conv2d (conv2)             [32, 128, 28, 28]   [32, 128, 28, 28]   (147,456)
False
│    │    └─BatchNorm2d (bn2)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
│    └─BasicBlock (2)                  [32, 128, 28, 28]   [32, 128, 28, 28]   --                False
│    │    └─Conv2d (conv1)             [32, 128, 28, 28]   [32, 128, 28, 28]   (147,456)
False
│    │    └─BatchNorm2d (bn1)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
│    │    └─Conv2d (conv2)             [32, 128, 28, 28]   [32, 128, 28, 28]   (147,456)
False
│    │    └─BatchNorm2d (bn2)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
│    └─BasicBlock (3)                  [32, 128, 28, 28]   [32, 128, 28, 28]   --                False
│    │    └─Conv2d (conv1)             [32, 128, 28, 28]   [32, 128, 28, 28]   (147,456)
False
│    │    └─BatchNorm2d (bn1)          [32, 128, 28, 28]   [32, 128, 28, 28]   (256)
False
│    │    └─ReLU (relu)                [32, 128, 28, 28]   [32, 128, 28, 28]   --                --
```

```
│    │    └─Conv2d (conv2)                [32, 128, 28, 28]    [32, 128, 28, 28]    (147,456)
False
│    │    └─BatchNorm2d (bn2)             [32, 128, 28, 28]    [32, 128, 28, 28]    (256)
False
│    └─ReLU (relu)                        [32, 128, 28, 28]    [32, 128, 28, 28]    --                --
├─Sequential (layer3)                     [32, 128, 28, 28]    [32, 256, 14, 14]    --                False
│    └─BasicBlock (0)                     [32, 128, 28, 28]    [32, 256, 14, 14]    --                False
│    │    └─Conv2d (conv1)                [32, 128, 28, 28]    [32, 256, 14, 14]    (294,912)
False
│    │    └─BatchNorm2d (bn1)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    │    └─Conv2d (conv2)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
│    │    └─BatchNorm2d (bn2)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    │    └─Sequential (downsample)       [32, 128, 28, 28]    [32, 256, 14, 14]    (33,280)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    └─BasicBlock (1)                     [32, 256, 14, 14]    [32, 256, 14, 14]    --                False
│    │    └─Conv2d (conv1)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
│    │    └─BatchNorm2d (bn1)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    │    └─Conv2d (conv2)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
│    │    └─BatchNorm2d (bn2)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    └─BasicBlock (2)                     [32, 256, 14, 14]    [32, 256, 14, 14]    --                False
│    │    └─Conv2d (conv1)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
│    │    └─BatchNorm2d (bn1)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    │    └─Conv2d (conv2)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
│    │    └─BatchNorm2d (bn2)             [32, 256, 14, 14]    [32, 256, 14, 14]    (512)
False
│    └─ReLU (relu)                        [32, 256, 14, 14]    [32, 256, 14, 14]    --                --
│    └─BasicBlock (3)                     [32, 256, 14, 14]    [32, 256, 14, 14]    --                False
│    │    └─Conv2d (conv1)                [32, 256, 14, 14]    [32, 256, 14, 14]    (589,824)
False
```

```
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)               [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    └─BasicBlock (4)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             False
│    │    └─Conv2d (conv1)               [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)               [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    └─BasicBlock (5)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             False
│    │    └─Conv2d (conv1)               [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn1)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
│    │    └─Conv2d (conv2)               [32, 256, 14, 14]   [32, 256, 14, 14]   (589,824)
False
│    │    └─BatchNorm2d (bn2)              [32, 256, 14, 14]   [32, 256, 14, 14]   (512)
False
│    │  └─ReLU (relu)             [32, 256, 14, 14]   [32, 256, 14, 14]   --             --
├─Sequential (layer4)              [32, 256, 14, 14]   [32, 512, 7, 7]     --             False
│    └─BasicBlock (0)             [32, 256, 14, 14]   [32, 512, 7, 7]     --             False
│    │    └─Conv2d (conv1)               [32, 256, 14, 14]   [32, 512, 7, 7]     (1,179,648)
False
│    │  └─BatchNorm2d (bn1)          [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)          False
│    │  └─ReLU (relu)             [32, 512, 7, 7]     [32, 512, 7, 7]     --             --
│    │  └─Conv2d (conv2)           [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)      False
│    │  └─BatchNorm2d (bn2)          [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)          False
│    │    └─Sequential (downsample)        [32, 256, 14, 14]   [32, 512, 7, 7]     (132,096)
False
│    │  └─ReLU (relu)             [32, 512, 7, 7]     [32, 512, 7, 7]     --             --
│    └─BasicBlock (1)             [32, 512, 7, 7]     [32, 512, 7, 7]     --             False
│    │  └─Conv2d (conv1)           [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)      False
│    │  └─BatchNorm2d (bn1)          [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)          False
```

```
│    │    └─ReLU (relu)              [32, 512, 7, 7]     [32, 512, 7, 7]     --                  --
│    │    └─Conv2d (conv2)           [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)         False
│    │    └─BatchNorm2d (bn2)        [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)             False
│    │    └─ReLU (relu)              [32, 512, 7, 7]     [32, 512, 7, 7]     --                  --
│    └─BasicBlock (2)                [32, 512, 7, 7]     [32, 512, 7, 7]     --                  False
│    │    └─Conv2d (conv1)           [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)         False
│    │    └─BatchNorm2d (bn1)        [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)             False
│    │    └─ReLU (relu)              [32, 512, 7, 7]     [32, 512, 7, 7]     --                  --
│    │    └─Conv2d (conv2)           [32, 512, 7, 7]     [32, 512, 7, 7]     (2,359,296)         False
│    │    └─BatchNorm2d (bn2)        [32, 512, 7, 7]     [32, 512, 7, 7]     (1,024)             False
│    │    └─ReLU (relu)              [32, 512, 7, 7]     [32, 512, 7, 7]     --                  --
├─AdaptiveAvgPool2d (avgpool)        [32, 512, 7, 7]     [32, 512, 1, 1]     --                  --
├─Linear (fc)                        [32, 512]           [32, 2]             1,026               True
==========================================================================
==============================================
Total params: 21,285,698
Trainable params: 1,026
Non-trainable params: 21,284,672
Total mult-adds (G): 117.22
==========================================================================
==============================================
Input size (MB): 19.27
Forward/backward pass size (MB): 1913.91
Params size (MB): 85.14
Estimated Total Size (MB): 2018.32
==========================================================================
==============================================
```

- Loading Created Dataset with the help of torchvision.datasets.ImageFolder
- Dataset Info:

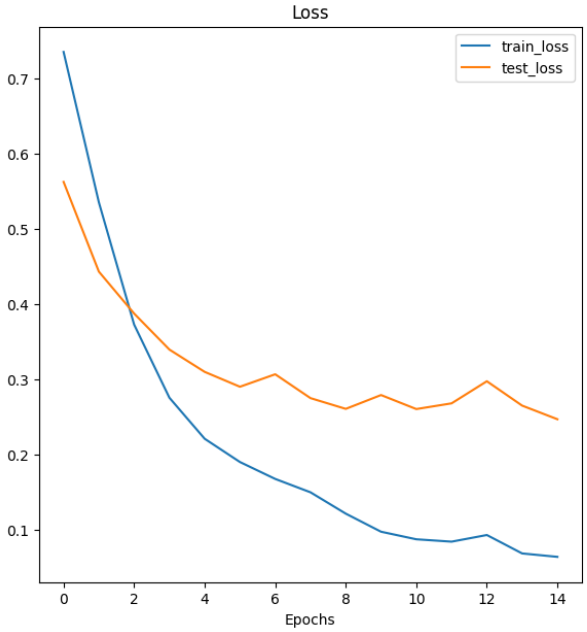
```
(Dataset ImageFolder
    Number of datapoints: 108
    Root location: data/gan_custom_data/train/
    StandardTransform
 Transform: ImageClassification(
               crop_size=[224]
               resize_size=[256]
               mean=[0.485, 0.456, 0.406]
               std=[0.229, 0.224, 0.225]
               interpolation=InterpolationMode.BILINEAR
           ),
 Dataset ImageFolder
    Number of datapoints: 108
```

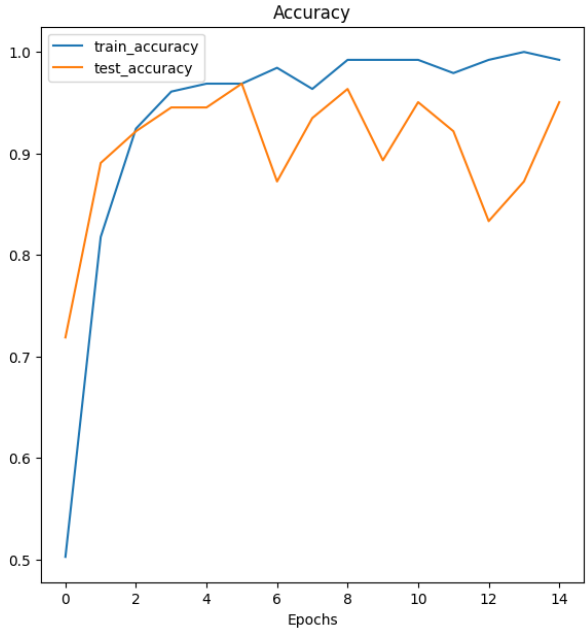
Root location: data/gan_custom_data/test/
StandardTransform
Transform: ImageClassification(
crop_size=[224]
resize_size=[256]
mean=[0.485, 0.456, 0.406]
std=[0.229, 0.224, 0.225]
interpolation=InterpolationMode.BILINEAR
))

- Visulazing random datasamples

- Here Label 0 means an original image with applied augmentation, while 1 Represents modified/ Gan Generated.
- Converting the dataset to data loader.
- Training the model for the same
- Training Results:

Epoch: 1 | train_loss: 0.7352 | train_acc: 0.5026 | test_loss: 0.5626 | test_acc: 0.7188
Epoch: 2 | train_loss: 0.5355 | train_acc: 0.8177 | test_loss: 0.4434 | test_acc: 0.8906
Epoch: 3 | train_loss: 0.3730 | train_acc: 0.9245 | test_loss: 0.3877 | test_acc: 0.9219
Epoch: 4 | train_loss: 0.2758 | train_acc: 0.9609 | test_loss: 0.3398 | test_acc: 0.9453
Epoch: 5 | train_loss: 0.2213 | train_acc: 0.9688 | test_loss: 0.3102 | test_acc: 0.9453
Epoch: 6 | train_loss: 0.1904 | train_acc: 0.9688 | test_loss: 0.2904 | test_acc: 0.9688
Epoch: 7 | train_loss: 0.1680 | train_acc: 0.9844 | test_loss: 0.3070 | test_acc: 0.8724
Epoch: 8 | train_loss: 0.1502 | train_acc: 0.9635 | test_loss: 0.2753 | test_acc: 0.9349
Epoch: 9 | train_loss: 0.1219 | train_acc: 0.9922 | test_loss: 0.2611 | test_acc: 0.9635
Epoch: 10 | train_loss: 0.0978 | train_acc: 0.9922 | test_loss: 0.2794 | test_acc: 0.8932
Epoch: 11 | train_loss: 0.0878 | train_acc: 0.9922 | test_loss: 0.2608 | test_acc: 0.9505
Epoch: 12 | train_loss: 0.0846 | train_acc: 0.9792 | test_loss: 0.2684 | test_acc: 0.9219
Epoch: 13 | train_loss: 0.0935 | train_acc: 0.9922 | test_loss: 0.2978 | test_acc: 0.8333
Epoch: 14 | train_loss: 0.0690 | train_acc: 1.0000 | test_loss: 0.2654 | test_acc: 0.8724
Epoch: 15 | train_loss: 0.0645 | train_acc: 0.9922 | test_loss: 0.2473 | test_acc: 0.9505
total training time: 43.407 sec.





- Doing the same with saving and loading models for testing purposes.

test_loss: 0.2473 | test_acc: 0.9505

Task: Audio Deepfake Detection

Objectives:

- **Record 1000 hindi and 1000 english sentences in your voice sampled from the given text files**
- **generated audios for spoof detection i.e real vs fake classification.**
- **Share the recorded and generated audios within separate folders respectively, using the drive link.**

Procedure:

- Record using voice:
  - https://www.dictate.app/annotation/record/selcted_lines_eng_20230221_171133#
  - https://www.dictate.app/annotation/record/selcted_lines_hin_20230224_072858#
- Generated Voice Using Google Text To Speech Engine:

Python

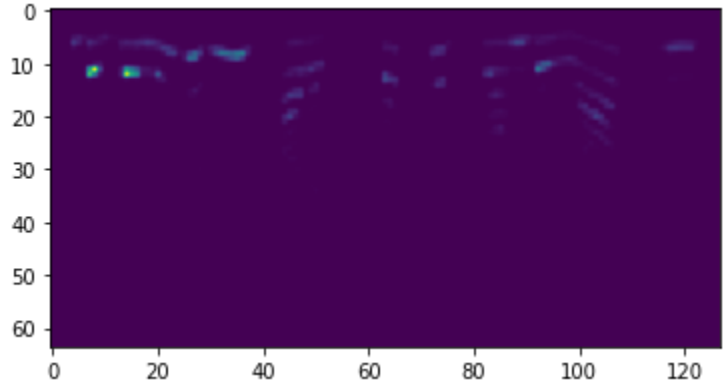
```
from gtts import gTTS
from pathlib import Path

language = 'en'
# language = 'hi'

# mytext = 'this is an example in english!'
# mytext = 'यह हिंदी में एक उदाहरण है'
Path("data/english").mkdir(parents=True, exist_ok=True)
with open("selcted_lines_eng.txt") as fp:
    lines = fp.readlines()
    i=0
    for mytext in lines[:5]:
        myobj = gTTS(text=mytext, lang=language,
tld='co.in', slow=False)
        myobj.save(f"data/english/{i}_female_eng.mp3")
        print(f"{i}/{len(lines)} done...")
        i += 1
```

- Restructuring the dataset.
- Dataset Link:- https://drive.google.com/file/d/1T0BpspJHZ9aEDztdvVc9T5bsbIZITnz_/view?usp=share_link
- Generating CSV File for the dataset will be used in Custom DATASET Class.
- Defining Classes for the Dataset - Spectrogram

- One sample from the dataset.



(torch.Size([1, 64, 127]), 0)

- Creating Data loaders with 80-20 Split of train and test data loaders.
- Defining Training and Testing Steps function for Training the Model.
- Defining models with pre-trained weights.
- Model Architecture:

```
==========================================================================
==============================================
Layer (type (var_name))                  Input Shape          Output Shape         Param #
Trainable
==========================================================================
==============================================
VGG (VGG)                                [32, 1, 64, 173]     [32, 2]              --                   Partial
├─Sequential (features)                  [32, 1, 64, 173]     [32, 512, 2, 5]      --                   False
│    └─Conv2d (0)                        [32, 1, 64, 173]     [32, 64, 64, 173]    (640)                False
│    └─ReLU (1)                          [32, 64, 64, 173]    [32, 64, 64, 173]    --                   --
│    └─MaxPool2d (2)                     [32, 64, 64, 173]    [32, 64, 32, 86]     --                   --
│    └─Conv2d (3)                        [32, 64, 32, 86]     [32, 128, 32, 86]    (73,856)             False
│    └─ReLU (4)                          [32, 128, 32, 86]    [32, 128, 32, 86]    --                   --
│    └─MaxPool2d (5)                     [32, 128, 32, 86]    [32, 128, 16, 43]    --                   --
│    └─Conv2d (6)                        [32, 128, 16, 43]    [32, 256, 16, 43]    (295,168)            False
│    └─ReLU (7)                          [32, 256, 16, 43]    [32, 256, 16, 43]    --                   --
│    └─Conv2d (8)                        [32, 256, 16, 43]    [32, 256, 16, 43]    (590,080)            False
│    └─ReLU (9)                          [32, 256, 16, 43]    [32, 256, 16, 43]    --                   --
│    └─MaxPool2d (10)                    [32, 256, 16, 43]    [32, 256, 8, 21]     --                   --
│    └─Conv2d (11)                       [32, 256, 8, 21]     [32, 512, 8, 21]     (1,180,160)          False
│    └─ReLU (12)                         [32, 512, 8, 21]     [32, 512, 8, 21]     --                   --
│    └─Conv2d (13)                       [32, 512, 8, 21]     [32, 512, 8, 21]     (2,359,808)          False
│    └─ReLU (14)                         [32, 512, 8, 21]     [32, 512, 8, 21]     --                   --
│    └─MaxPool2d (15)                    [32, 512, 8, 21]     [32, 512, 4, 10]     --                   --
│    └─Conv2d (16)                       [32, 512, 4, 10]     [32, 512, 4, 10]     (2,359,808)          False
│    └─ReLU (17)                         [32, 512, 4, 10]     [32, 512, 4, 10]     --                   --
```

```
│    └─Conv2d (18)                [32, 512, 4, 10]    [32, 512, 4, 10]    (2,359,808)        False
│    └─ReLU (19)                  [32, 512, 4, 10]    [32, 512, 4, 10]    --                 --
│    └─MaxPool2d (20)             [32, 512, 4, 10]    [32, 512, 2, 5]     --                 --
├─AdaptiveAvgPool2d (avgpool)      [32, 512, 2, 5]     [32, 512, 7, 7]     --                 --
├─Sequential (classifier)          [32, 25088]         [32, 2]             --                 True
│    └─Linear (0)                 [32, 25088]         [32, 4096]          102,764,544        True
│    └─ReLU (1)                   [32, 4096]          [32, 4096]          --                 --
│    └─Dropout (2)                [32, 4096]          [32, 4096]          --                 --
│    └─Linear (3)                 [32, 4096]          [32, 4096]          16,781,312         True
│    └─ReLU (4)                   [32, 4096]          [32, 4096]          --                 --
│    └─Dropout (5)                [32, 4096]          [32, 4096]          --                 --
│    └─Linear (6)                 [32, 4096]          [32, 2]             8,194              True
==========================================================================
==============================================
Total params: 128,773,378
Trainable params: 119,554,050
Non-trainable params: 9,219,328
Total mult-adds (G): 55.12
==========================================================================
==============================================
Input size (MB): 1.42
Forward/backward pass size (MB): 418.38
Params size (MB): 515.09
Estimated Total Size (MB): 934.89
==========================================================================
==============================================
```

- Trained Model Results;

```
Epoch: 1 | train_loss: 53.9074 | train_acc: 0.9081 | test_loss: 42.5900 | test_acc: 0.9075
Epoch: 2 | train_loss: 20.1229 | train_acc: 0.9522 | test_loss: 4.0608 | test_acc: 0.9738
Epoch: 3 | train_loss: 3.9477 | train_acc: 0.9759 | test_loss: 0.9426 | test_acc: 0.9938
Epoch: 4 | train_loss: 25.0096 | train_acc: 0.9628 | test_loss: 50.8121 | test_acc: 0.9750
Epoch: 5 | train_loss: 86.8253 | train_acc: 0.9544 | test_loss: 16.2660 | test_acc: 0.9938
Epoch: 6 | train_loss: 27.7979 | train_acc: 0.9822 | test_loss: 1.1427 | test_acc: 0.9962
Epoch: 7 | train_loss: 10.8485 | train_acc: 0.9850 | test_loss: 3.0007 | test_acc: 0.9888
Epoch: 8 | train_loss: 5.0821 | train_acc: 0.9888 | test_loss: 19.8806 | test_acc: 0.9688
Epoch: 9 | train_loss: 3.6336 | train_acc: 0.9866 | test_loss: 1.4706 | test_acc: 0.9975
Epoch: 10 | train_loss: 1.4425 | train_acc: 0.9931 | test_loss: 1.7719 | test_acc: 0.9888
Trained feed forward net saved at model_pre_trained.pth
```

- Training Other CNN Arch. for the same

- Results:

Epoch: 1 | train_loss: 0.8077 | train_acc: 0.5056 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 2 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 3 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 4 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 5 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 6 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 7 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 8 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 9 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Epoch: 10 | train_loss: 0.8079 | train_acc: 0.5053 | test_loss: 0.8095 | test_acc: 0.5038
Trained feed forward net saved at model_custom_cnn_arch.pth

ALL CODE + DATA:- A1

Note:-

- **Pre Trained Acrh. performs very well because it is pre-trained, and the dataset is very small for training a network from scratch.**
- **Please use the IITJ Email id for accessing the data and the code.**

References:

https://arxiv.org/abs/2010.01950
https://arxiv.org/pdf/2110.01200.pdf
https://github.com/clovaai/aasist
https://github.com/Harry24k/adversarial-attacks-pytorch
https://adversarial-attacks-pytorch.readthedocs.io/en/latest/attacks.html
https://replicate.com/yoyo-nb/thin-plate-spline-motion-model
https://replicate.com/yoyo-nb/thin-plate-spline-motion-model/api
https://giphy.com/ **-> For deep fake samples references**
https://gtts.readthedocs.io/en/latest/
https://www.dictate.app/ **-> For Recoding Audio**
https://pytorch.org/docs/stable/index.html
https://numpy.org/doc/stable/reference/index.html
https://torchmetrics.readthedocs.io/en/stable/
https://stackoverflow.com/