In [55]:
```
import models
import os
import torch
import torch.nn as nn
from torch import optim, nn
from torchvision import transforms, datasets
import torchvision
import common
import matplotlib.pyplot as plt
from PIL import Image
x=[]
y=[]
```

In [2]:
```
#DIRECTORY SETTINGS
os.chdir("..")#Go up two directories
SAVE_DIR = 'models'
MODEL_SAVE_PATH = os.path.join(SAVE_DIR, '/home/prawat/homework/base.pt')


#HYPERPARAMETERS
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

EPOCHS=100
BATCH_SIZE = 8
criterion = nn.CrossEntropyLoss()
ADAM_OPTIMISER=True
LEARNING_RATE=0.001
```

In [3]:
```
train_transforms = transforms.Compose([# Data Transforms
                            transforms.Resize(256),#Resize
                            transforms.RandomHorizontalFlip(30),#Flip
                            transforms.RandomRotation(10),#Roatate
                            transforms.RandomCrop(256),#Crop
                            transforms.ToTensor(),#Convert to Tensor
                            transforms.Normalize((0.49139968, 0.48215827, 0.44653124), (
                        ])

test_transforms = transforms.Compose([
                            transforms.ToTensor(),
                            transforms.Normalize((0.49139968, 0.48215827, 0.44653124), (
                        ])
```

In [4]:
```
train_data = torchvision.datasets.CIFAR10(root='/home/prawat/homework/data/', train=Tru
train_data, valid_data = torch.utils.data.random_split(train_data, [int(len(train_data)
test_data = torchvision.datasets.CIFAR10(root='/home/prawat/homework/data/', train=Fals


print(f'Number of training examples: {len(train_data)}')
print(f'Number of validation examples: {len(valid_data)}')
print(f'Number of testing examples: {len(test_data)}')


train_iterator = torch.utils.data.DataLoader(train_data, shuffle=True, batch_size=BATCH
valid_iterator = torch.utils.data.DataLoader(valid_data, batch_size=BATCH_SIZE)
test_iterator = torch.utils.data.DataLoader(test_data, batch_size=BATCH_SIZE)
```

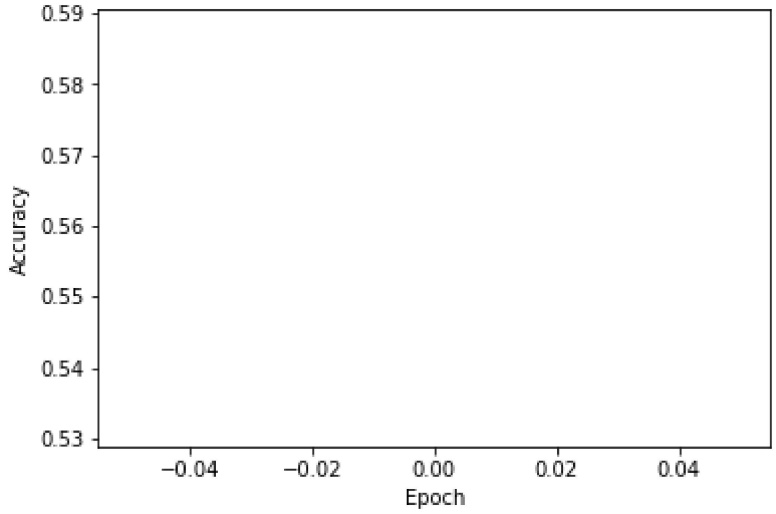
```
Files already downloaded and verified
Files already downloaded and verified
Number of training examples: 45000
```

```
Number of validation examples: 5000
Number of testing examples: 10000
```

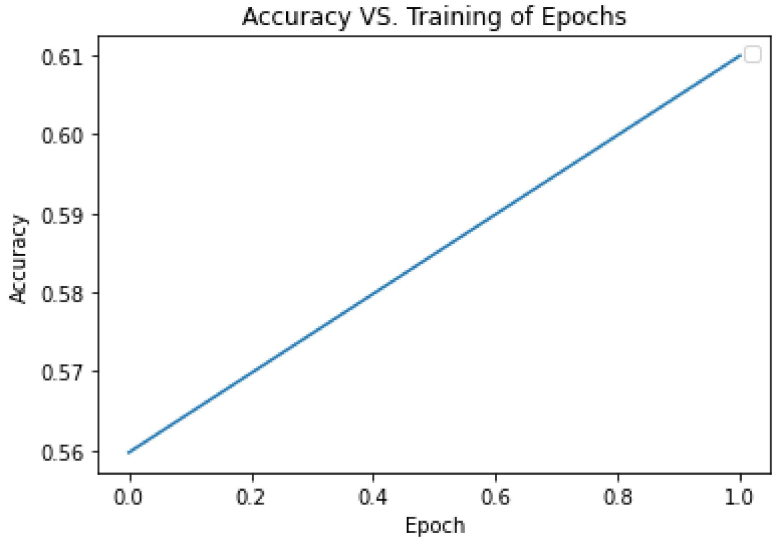
In [5]:
```
model = torchvision.models.resnet18(pretrained=True)#TorchVision

for param in model.parameters():
    param.requires_grad = False
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 10)
model = model.to(device)
model =nn.DataParallel(model).cuda()

#Hyperparameters
if(ADAM_OPTIMISER):
    optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)
else:
    optimizer = optim.SGD(model.classifier.parameters(), lr=0.001, momentum=0.5)
```

In [6]:
```
#Train
best_valid_loss = float('inf')
for epoch in range(EPOCHS):#Range of Epochs
    print(epoch)
    train_loss, train_acc = common.train(model, device, train_iterator, optimizer, crit
    valid_loss, valid_acc = common.evaluate(model, device, valid_iterator, criterion)#V

    if valid_loss < best_valid_loss:#Validation Loss - Is current lower than the saved
        best_valid_loss = valid_loss#Save the best loss (lowest)
        torch.save(model.state_dict(), MODEL_SAVE_PATH)#Save the model

    print(f'| Epoch: {epoch+1:02} | Train Loss: {train_loss:.3f} | Train Acc: {train_ac

    x.append(epoch)
    y.append(train_acc)
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.plot(x,y)
    plt.show()
    plt.legend(['train_acc'])
    plt.title('Accuracy VS. Training of Epochs')

#Test
```

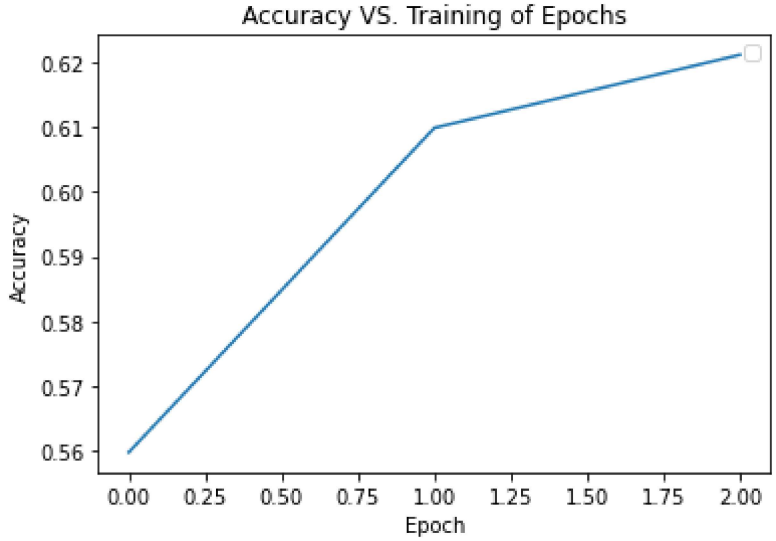
```
0
/home/prawat/.local/lib/python3.8/site-packages/torch/nn/functional.py:718: UserWarning:
Named tensors and all their associated APIs are an experimental feature and subject to c
hange. Please do not use them for anything important until they are released as stable.
(Triggered internally at  /pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
| Epoch: 01 | Train Loss: 1.289 | Train Acc: 55.97% | Val. Loss: 0.792| Val. Acc: 73.38%
```



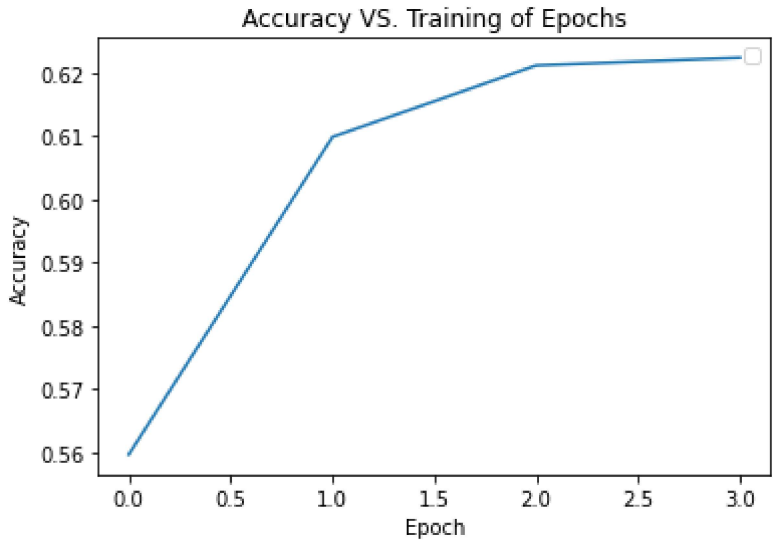
1
| Epoch: 02 | Train Loss: 1.161 | Train Acc: 60.99% | Val. Loss: 0.856| Val. Acc: 70.84%
|



2
| Epoch: 03 | Train Loss: 1.130 | Train Acc: 62.12% | Val. Loss: 0.775| Val. Acc: 73.98%
|

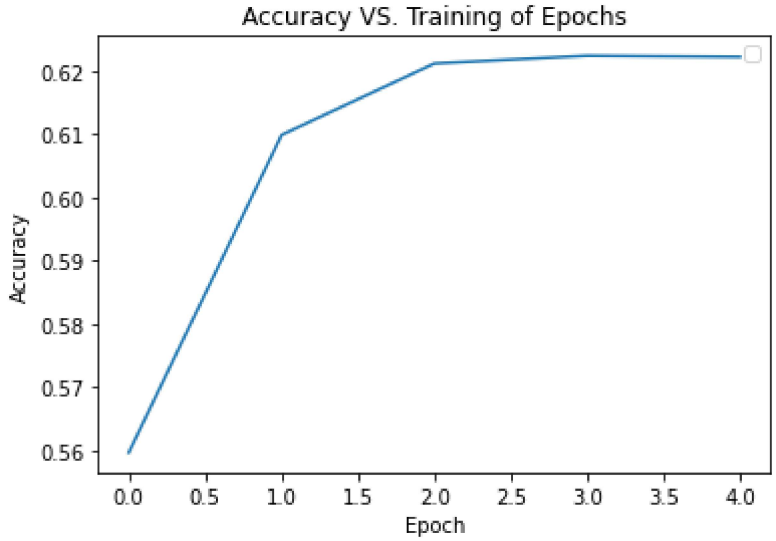


3
| Epoch: 04 | Train Loss: 1.133 | Train Acc: 62.25% | Val. Loss: 0.823| Val. Acc: 72.38%
|



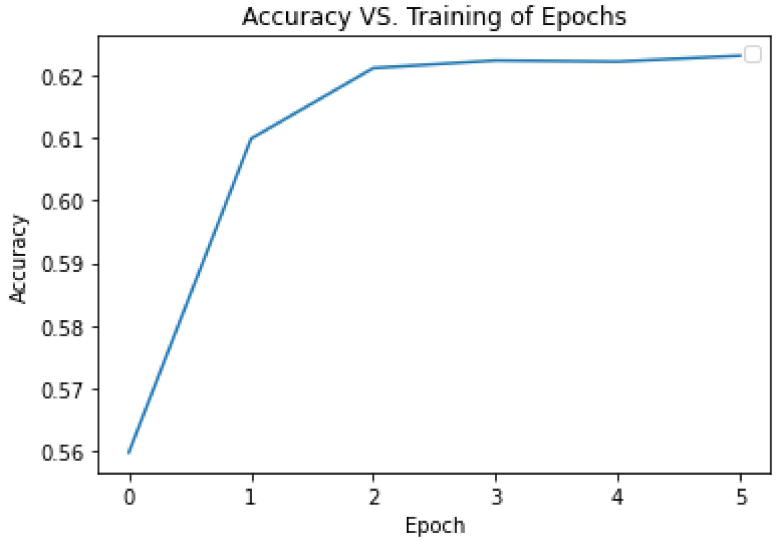
4

```
| Epoch: 05 | Train Loss: 1.131 | Train Acc: 62.23% | Val. Loss: 0.775| Val. Acc: 73.90%
```



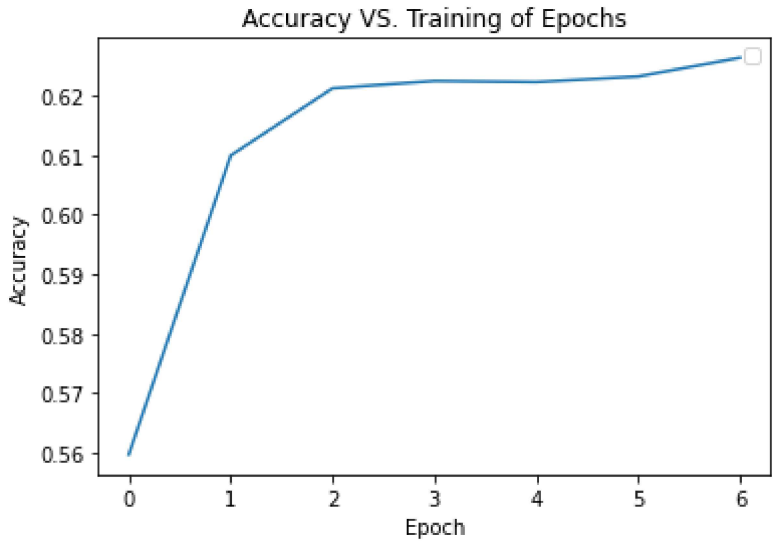
5

```
| Epoch: 06 | Train Loss: 1.124 | Train Acc: 62.32% | Val. Loss: 0.799| Val. Acc: 72.62%
```



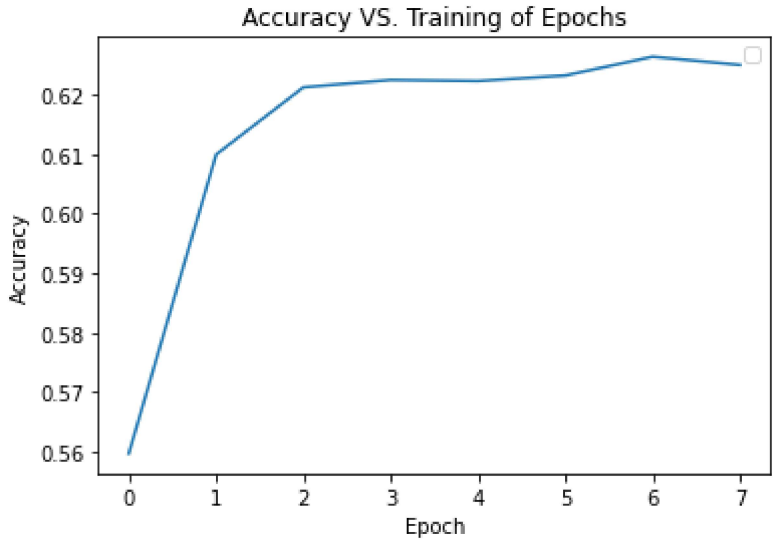
6

```
| Epoch: 07 | Train Loss: 1.121 | Train Acc: 62.64% | Val. Loss: 0.791| Val. Acc: 73.16%
```



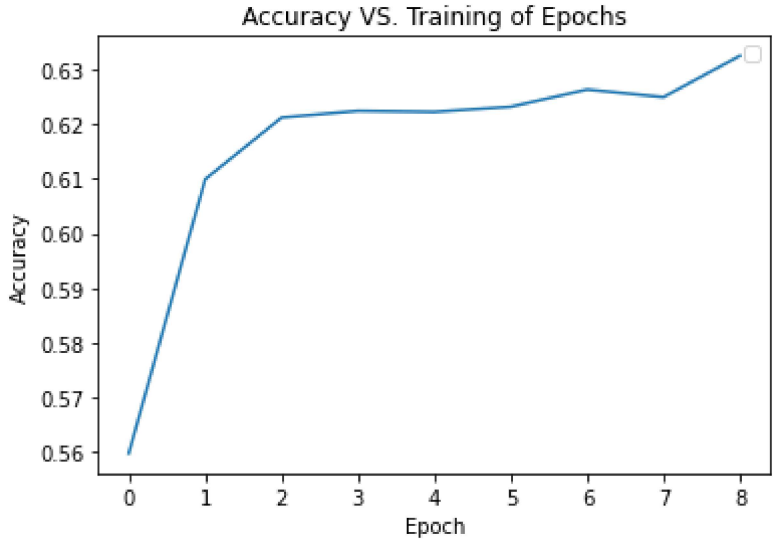
7

```
| Epoch: 08 | Train Loss: 1.119 | Train Acc: 62.50% | Val. Loss: 0.759| Val. Acc: 75.18%
```



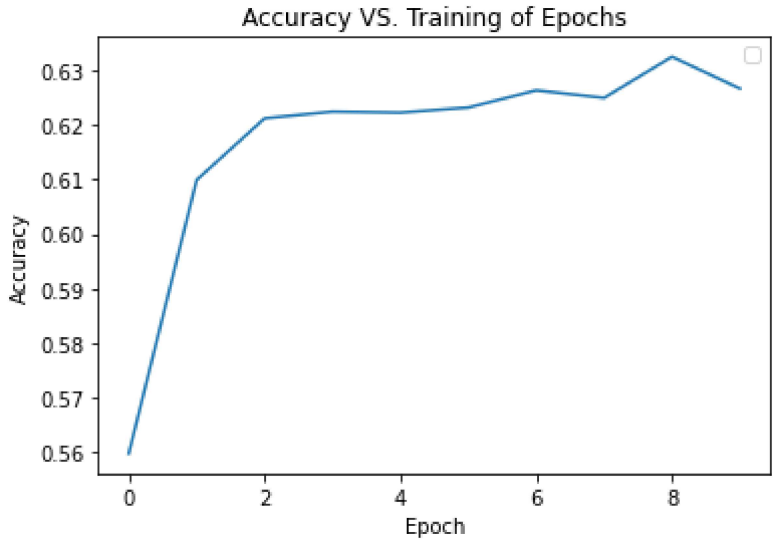
8

```
| Epoch: 09 | Train Loss: 1.113 | Train Acc: 63.26% | Val. Loss: 0.784| Val. Acc: 73.78%
```

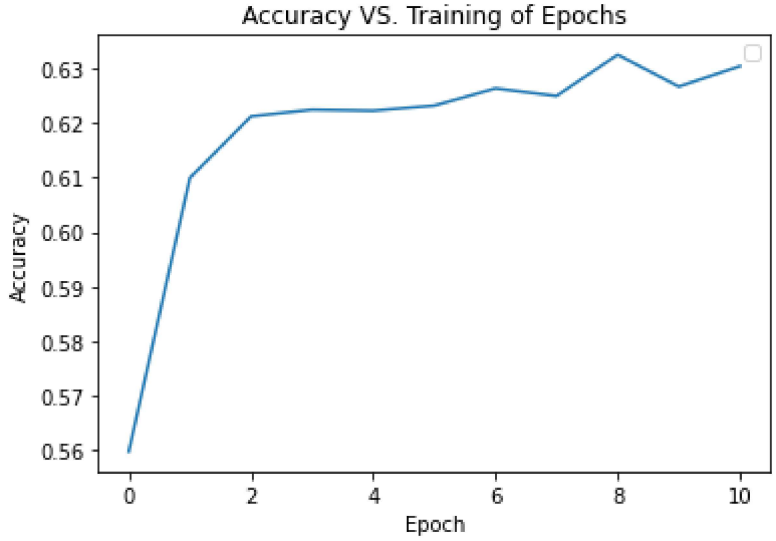


9

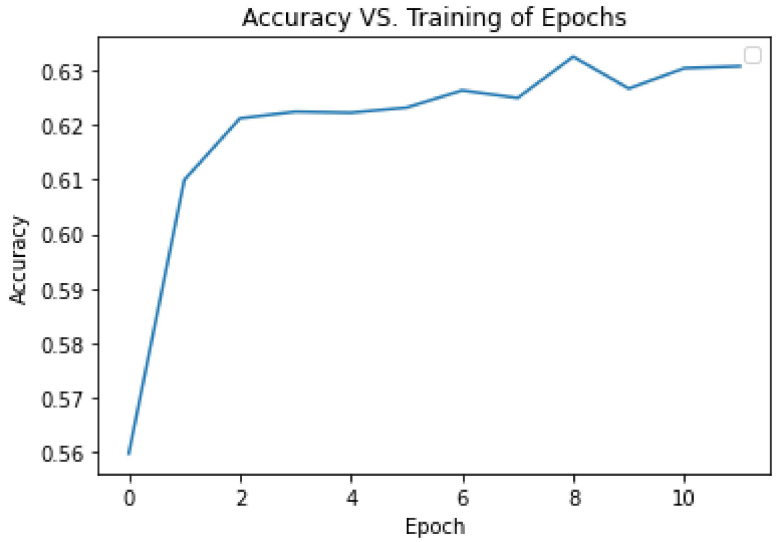
```
| Epoch: 10 | Train Loss: 1.123 | Train Acc: 62.67% | Val. Loss: 0.784| Val. Acc: 73.58%
```




```
10
| Epoch: 11 | Train Loss: 1.111 | Train Acc: 63.04% | Val. Loss: 0.798| Val. Acc: 72.52%
```

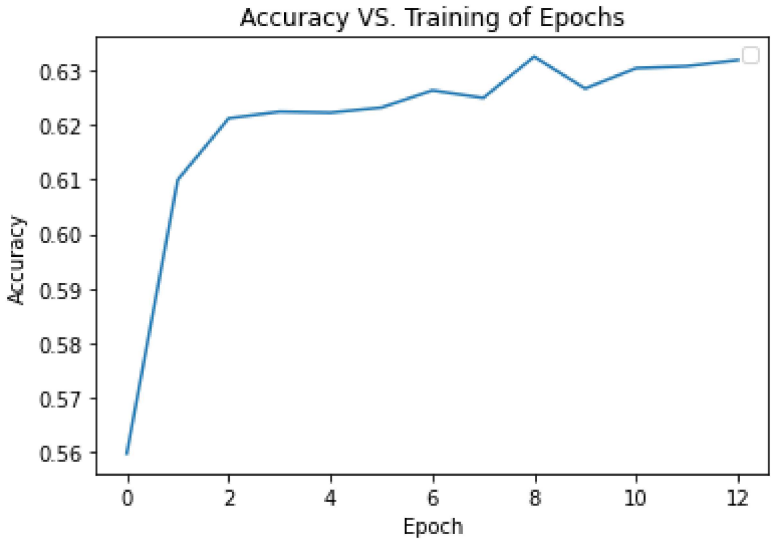



```
11
| Epoch: 12 | Train Loss: 1.111 | Train Acc: 63.08% | Val. Loss: 0.761| Val. Acc: 74.30%
```



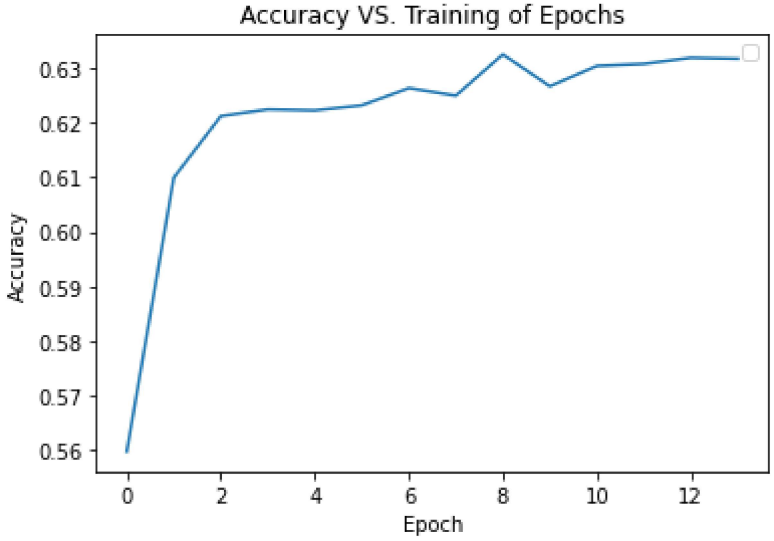
12

```
| Epoch: 13 | Train Loss: 1.107 | Train Acc: 63.19% | Val. Loss: 0.799| Val. Acc: 74.12%
```



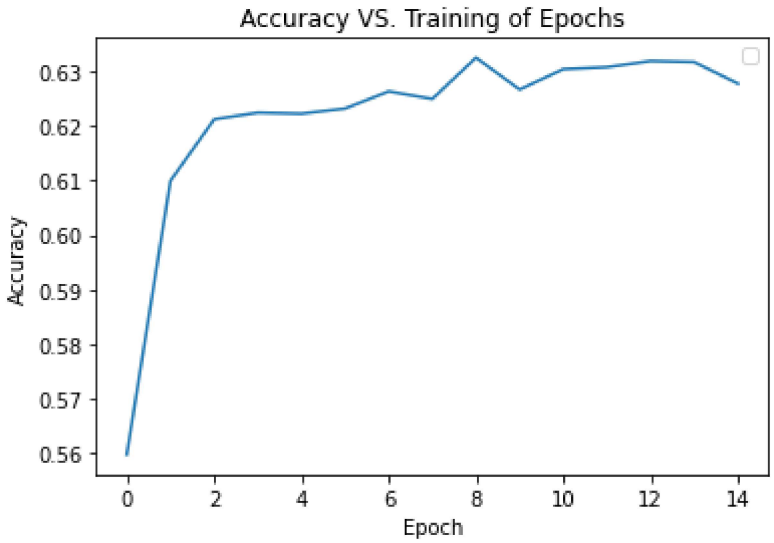
13

```
| Epoch: 14 | Train Loss: 1.116 | Train Acc: 63.18% | Val. Loss: 0.808| Val. Acc: 73.76%
```

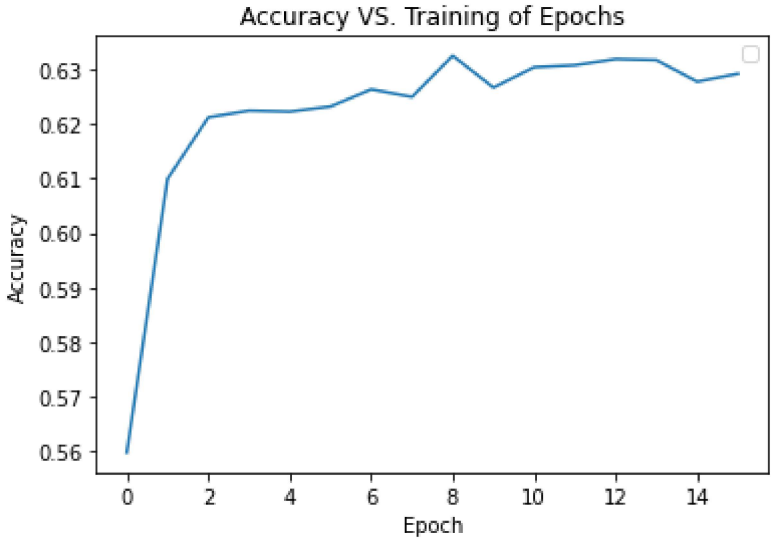


14

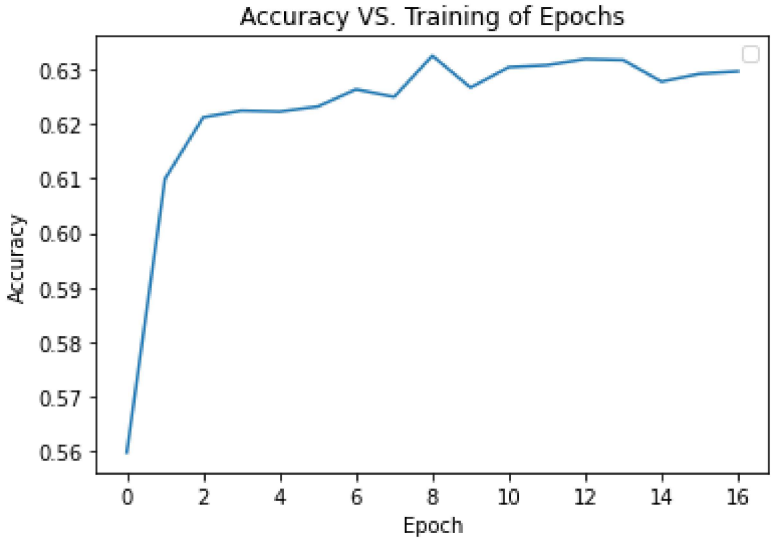
```
| Epoch: 15 | Train Loss: 1.116 | Train Acc: 62.78% | Val. Loss: 0.776| Val. Acc: 74.68%
```



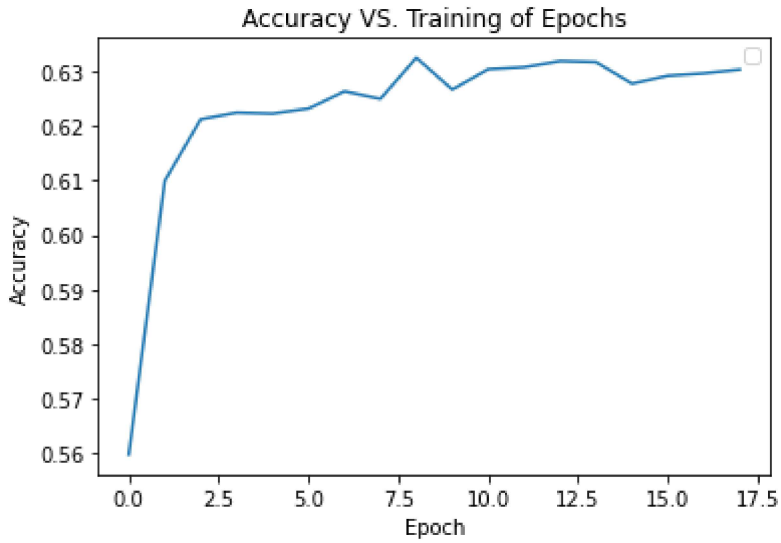
15
| Epoch: 16 | Train Loss: 1.110 | Train Acc: 62.92% | Val. Loss: 0.877| Val. Acc: 70.78%



16
| Epoch: 17 | Train Loss: 1.110 | Train Acc: 62.97% | Val. Loss: 0.784| Val. Acc: 74.26%

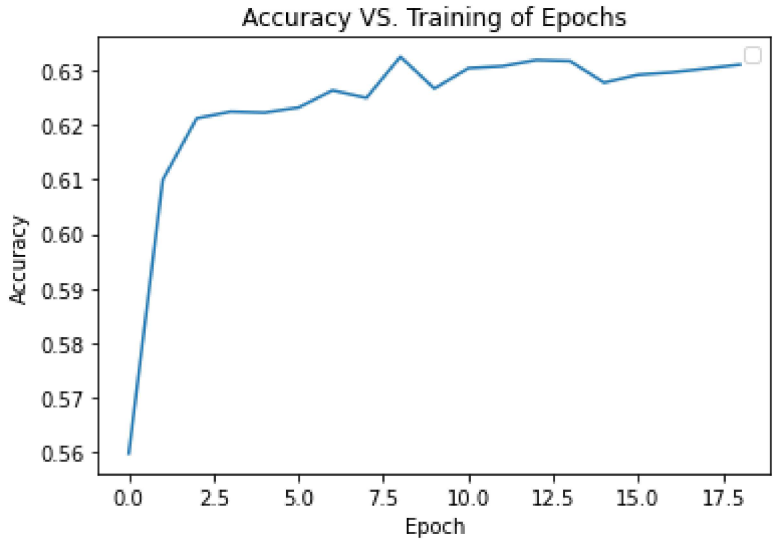


17
| Epoch: 18 | Train Loss: 1.107 | Train Acc: 63.04% | Val. Loss: 0.814| Val. Acc: 72.38%



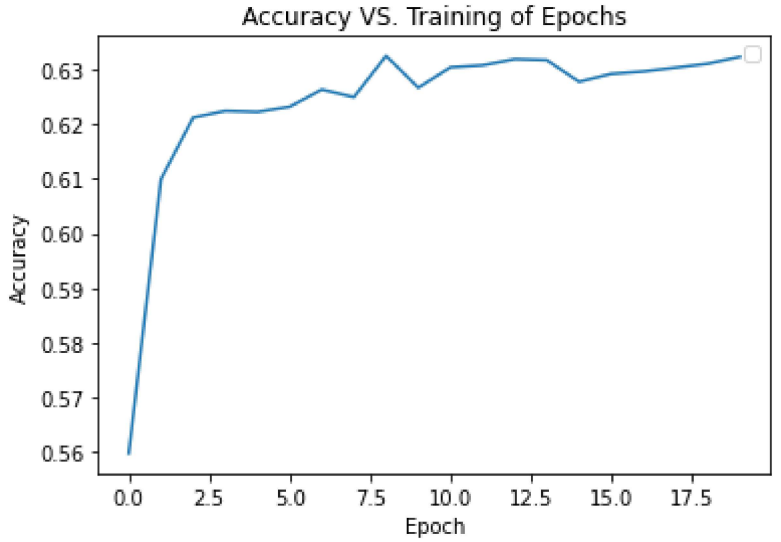
18

```
| Epoch: 19 | Train Loss: 1.117 | Train Acc: 63.11% | Val. Loss: 0.841| Val. Acc: 72.12%
```



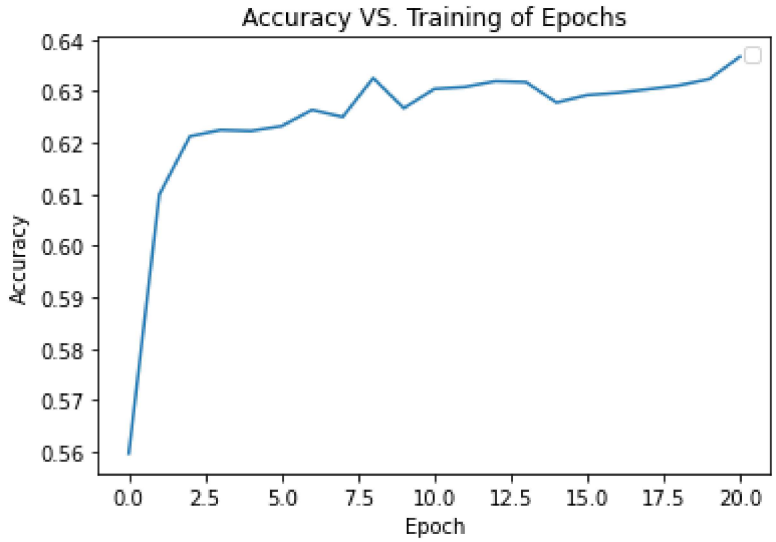
19

```
| Epoch: 20 | Train Loss: 1.112 | Train Acc: 63.24% | Val. Loss: 0.765| Val. Acc: 73.90%
```



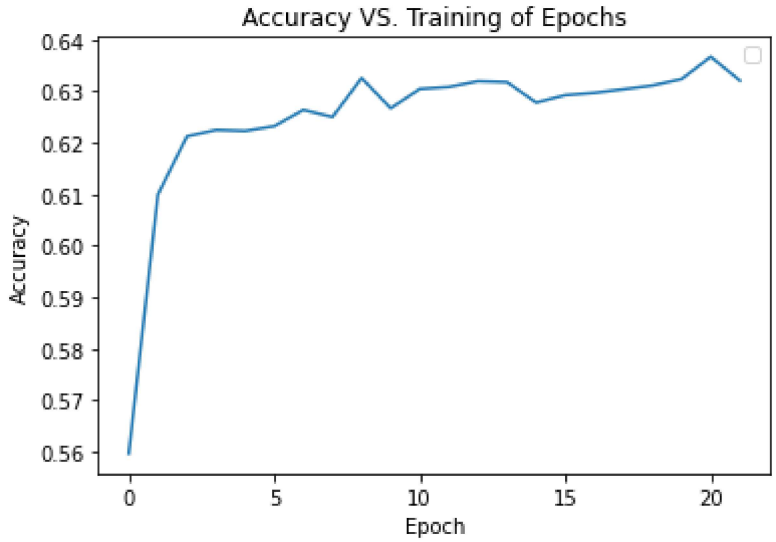
20

```
| Epoch: 21 | Train Loss: 1.097 | Train Acc: 63.67% | Val. Loss: 0.773| Val. Acc: 74.36%
```



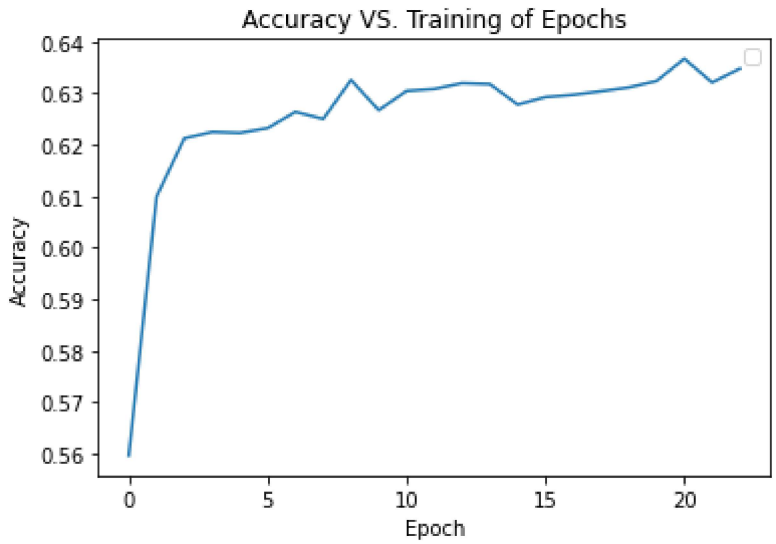
21

```
| Epoch: 22 | Train Loss: 1.115 | Train Acc: 63.21% | Val. Loss: 0.777| Val. Acc: 74.36%
```



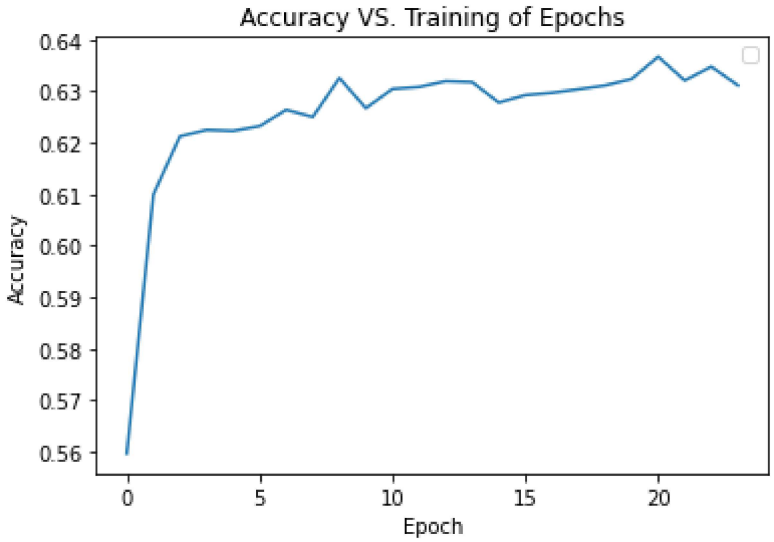
22

```
| Epoch: 23 | Train Loss: 1.103 | Train Acc: 63.48% | Val. Loss: 0.844| Val. Acc: 72.36%
```



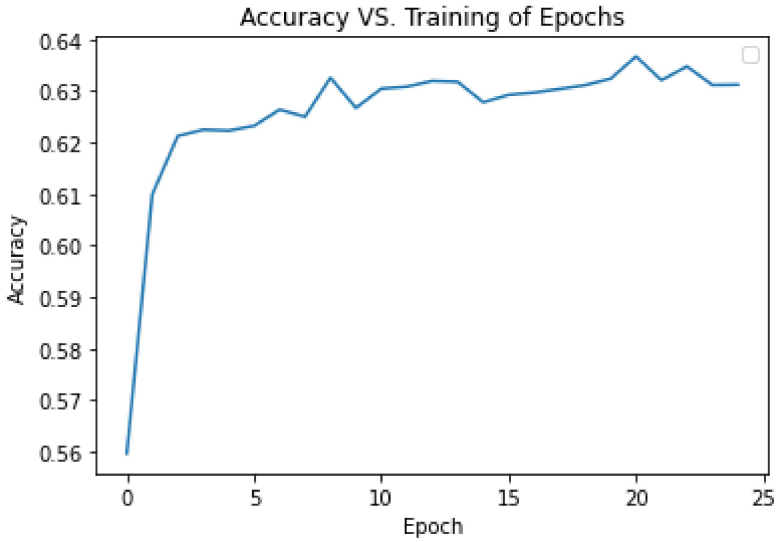
23

```
| Epoch: 24 | Train Loss: 1.114 | Train Acc: 63.11% | Val. Loss: 0.769| Val. Acc: 74.30%
```



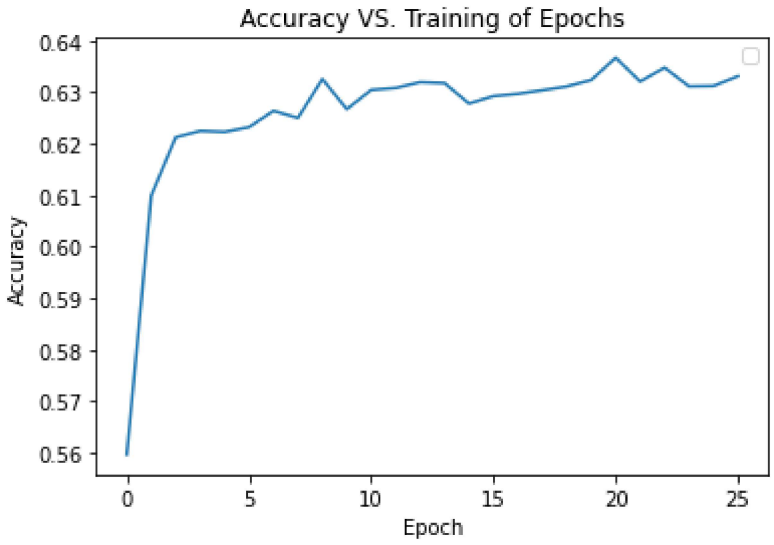
24

```
| Epoch: 25 | Train Loss: 1.116 | Train Acc: 63.12% | Val. Loss: 0.824| Val. Acc: 72.22%
```

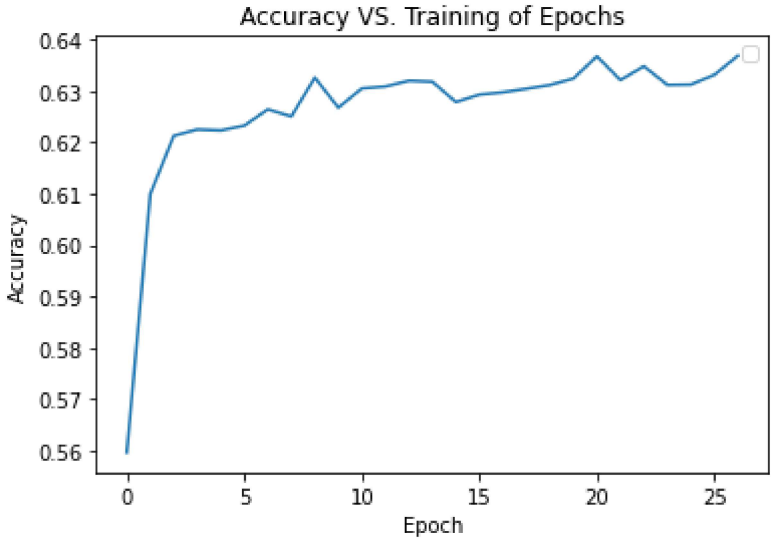


25

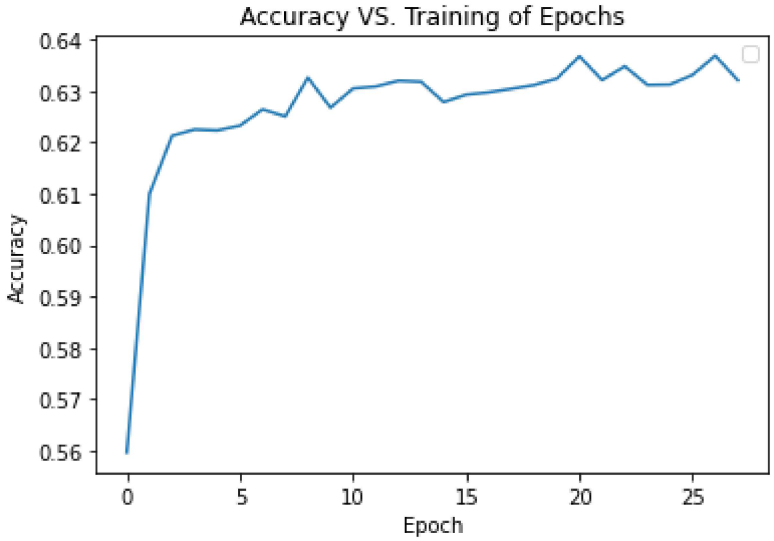
```
| Epoch: 26 | Train Loss: 1.108 | Train Acc: 63.31% | Val. Loss: 0.802| Val. Acc: 73.18%
```



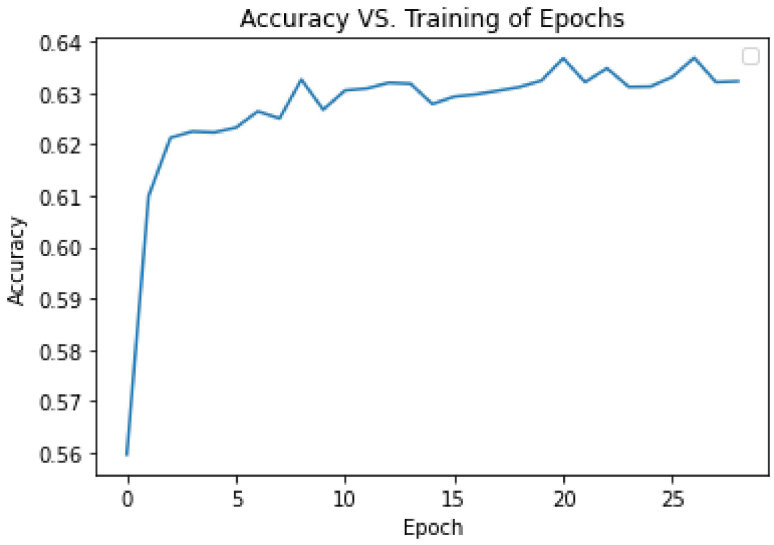
26
```
| Epoch: 27 | Train Loss: 1.102 | Train Acc: 63.68% | Val. Loss: 0.877| Val. Acc: 71.34%
```



27
```
| Epoch: 28 | Train Loss: 1.106 | Train Acc: 63.20% | Val. Loss: 0.768| Val. Acc: 74.62%
```

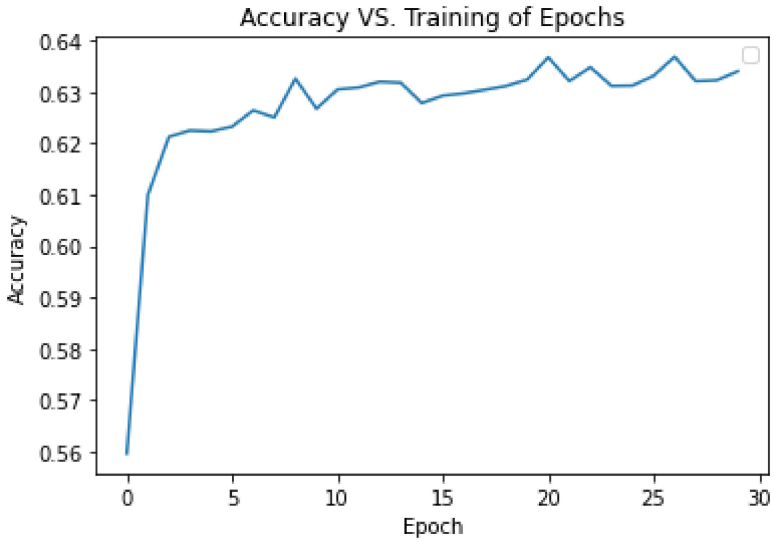


28
```
| Epoch: 29 | Train Loss: 1.107 | Train Acc: 63.22% | Val. Loss: 0.795| Val. Acc: 72.96%
```



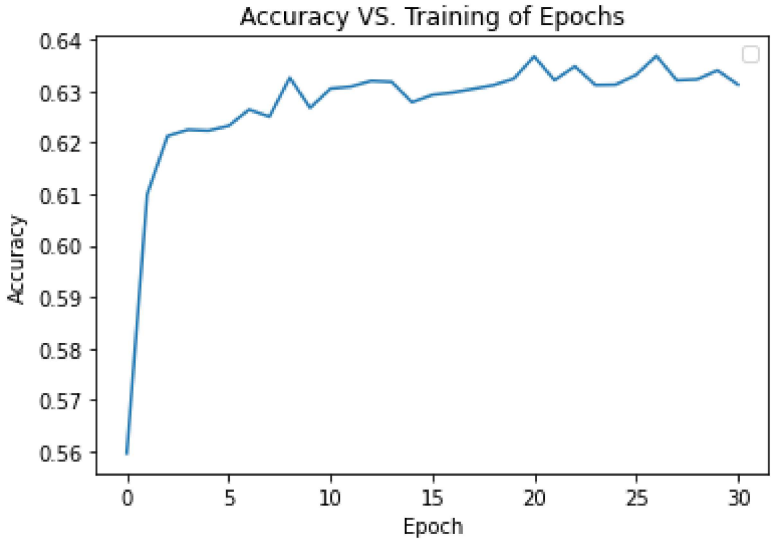
29

| Epoch: 30 | Train Loss: 1.105 | Train Acc: 63.40% | Val. Loss: 0.779| Val. Acc: 73.36%



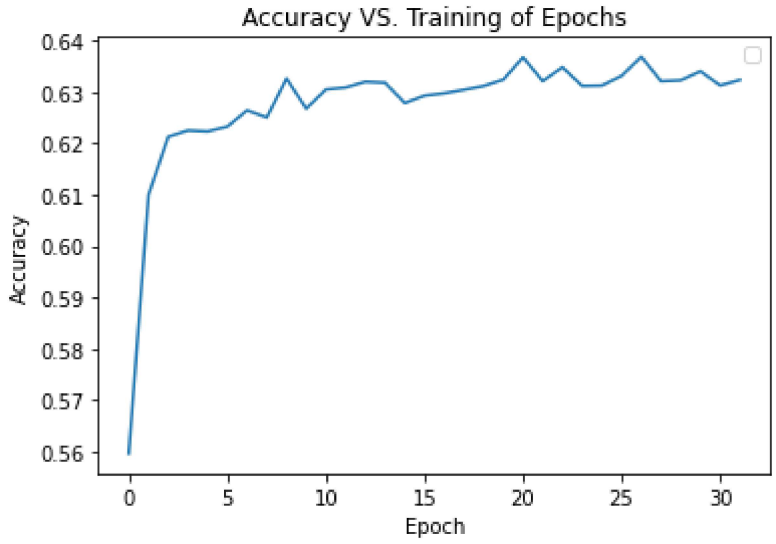
30

| Epoch: 31 | Train Loss: 1.112 | Train Acc: 63.12% | Val. Loss: 0.774| Val. Acc: 74.82%

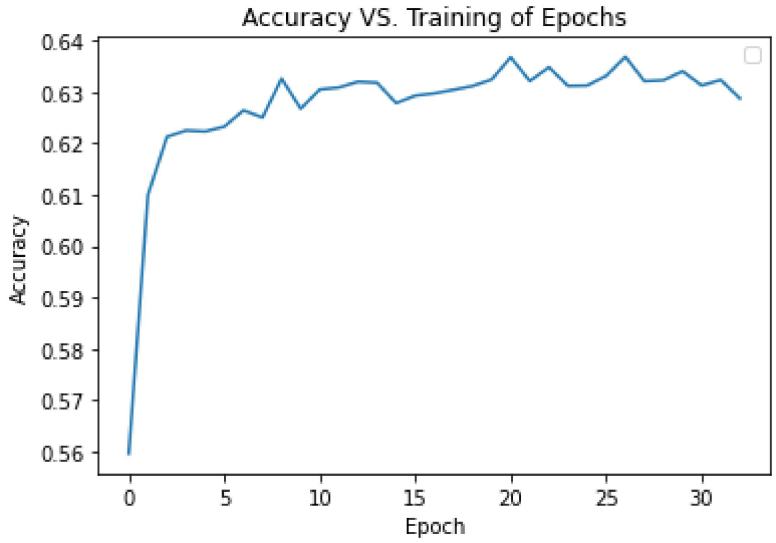


31

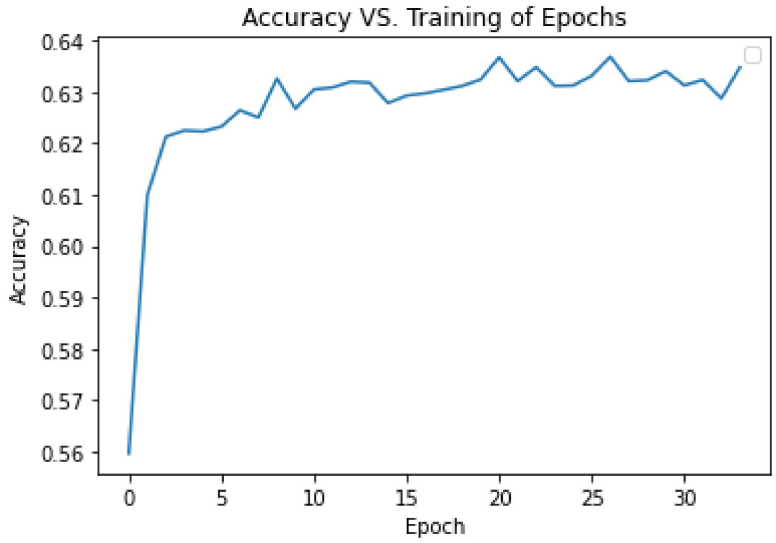
```
| Epoch: 32 | Train Loss: 1.113 | Train Acc: 63.23% | Val. Loss: 0.829| Val. Acc: 72.86%
```




```
32
| Epoch: 33 | Train Loss: 1.115 | Train Acc: 62.87% | Val. Loss: 0.768| Val. Acc: 74.40%
```

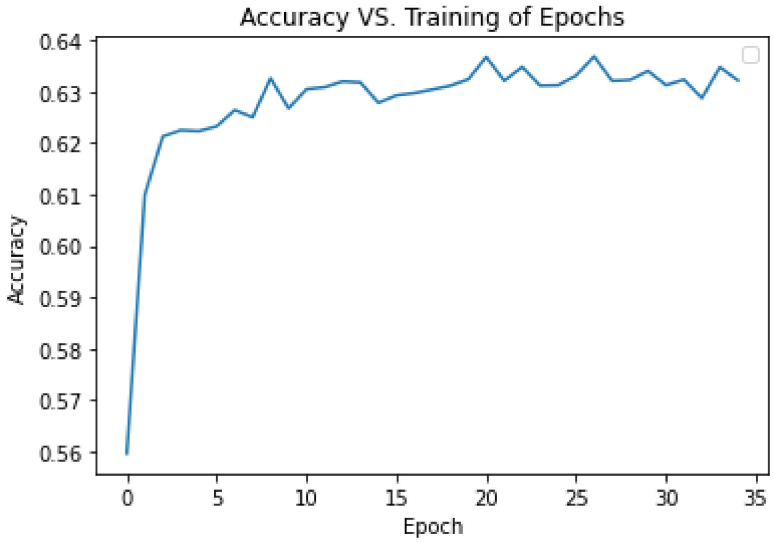



```
33
| Epoch: 34 | Train Loss: 1.110 | Train Acc: 63.47% | Val. Loss: 0.808| Val. Acc: 73.40%
```



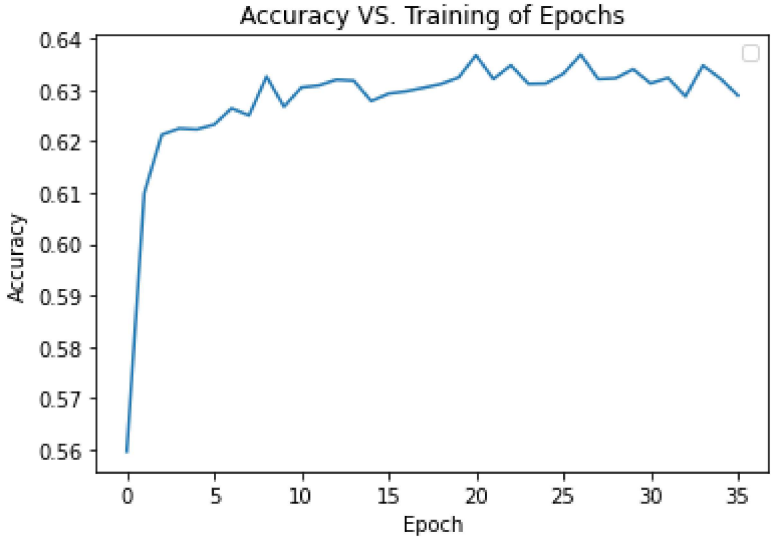
34

```
| Epoch: 35 | Train Loss: 1.109 | Train Acc: 63.22% | Val. Loss: 0.803| Val. Acc: 73.70%
```



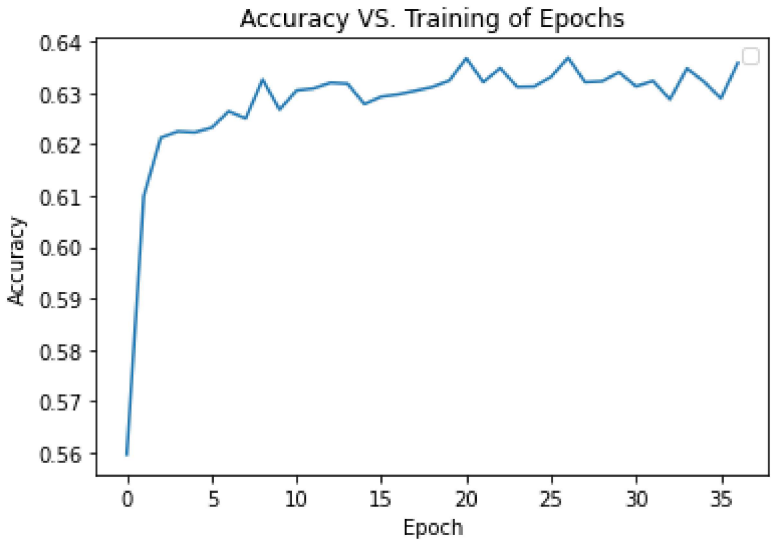
35

```
| Epoch: 36 | Train Loss: 1.114 | Train Acc: 62.89% | Val. Loss: 0.751| Val. Acc: 74.86%
```

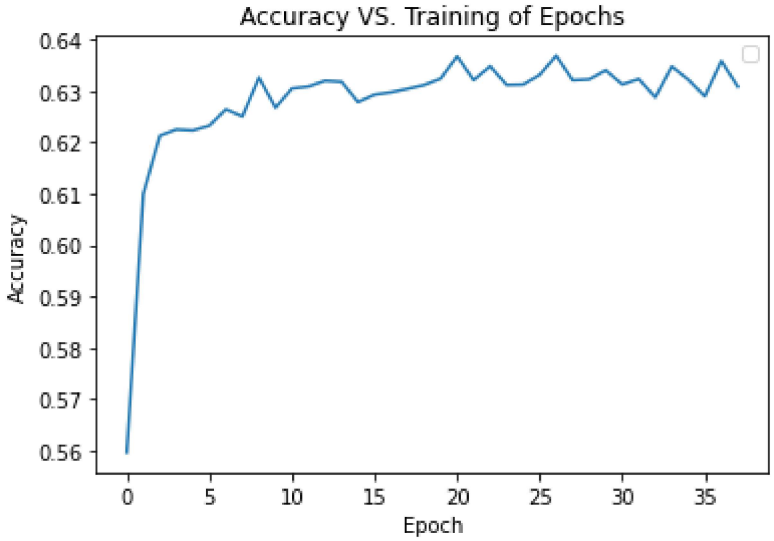


36

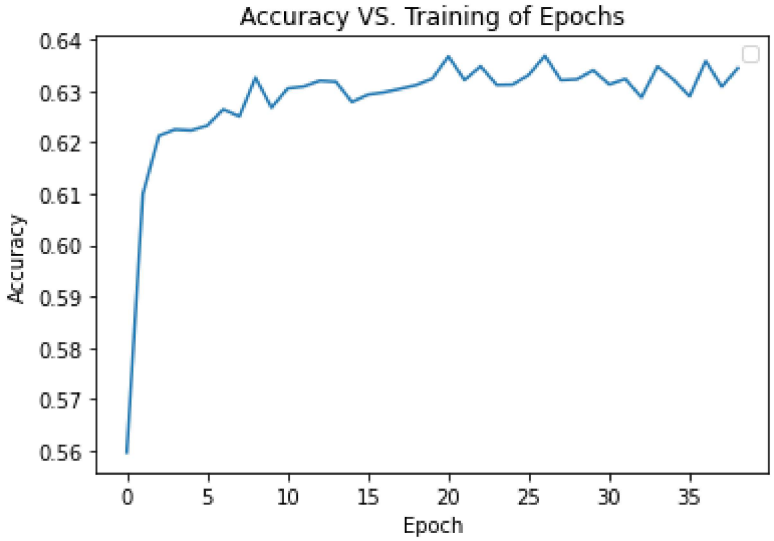
```
| Epoch: 37 | Train Loss: 1.101 | Train Acc: 63.58% | Val. Loss: 0.786| Val. Acc: 73.56%
```



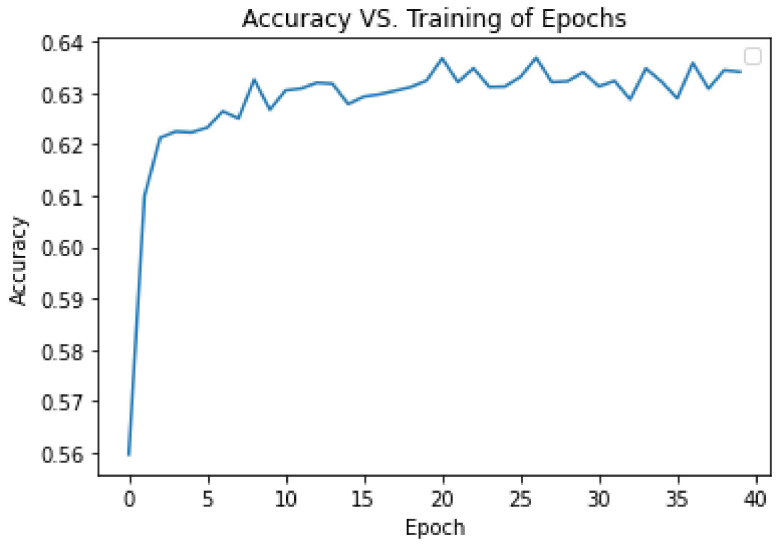
37
| Epoch: 38 | Train Loss: 1.117 | Train Acc: 63.08% | Val. Loss: 0.754| Val. Acc: 74.36%



38
| Epoch: 39 | Train Loss: 1.107 | Train Acc: 63.44% | Val. Loss: 0.781| Val. Acc: 74.34%

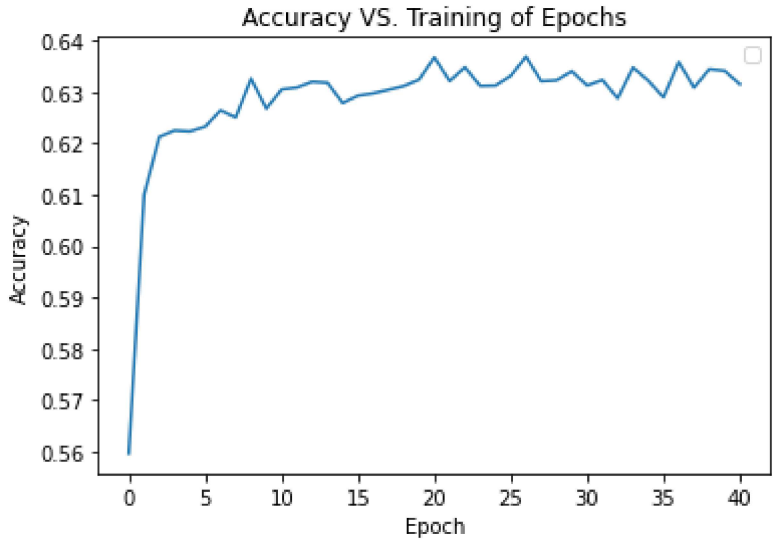


39
| Epoch: 40 | Train Loss: 1.108 | Train Acc: 63.41% | Val. Loss: 0.833| Val. Acc: 72.36%



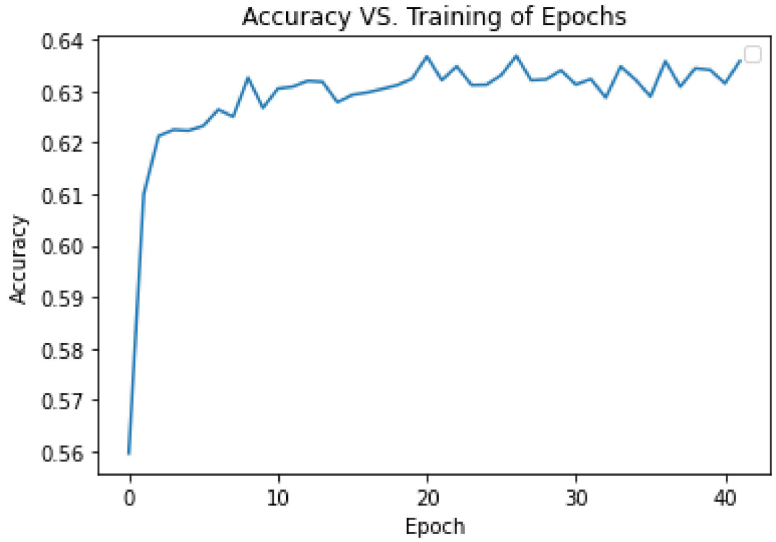
40

| Epoch: 41 | Train Loss: 1.113 | Train Acc: 63.15% | Val. Loss: 0.774| Val. Acc: 74.06%



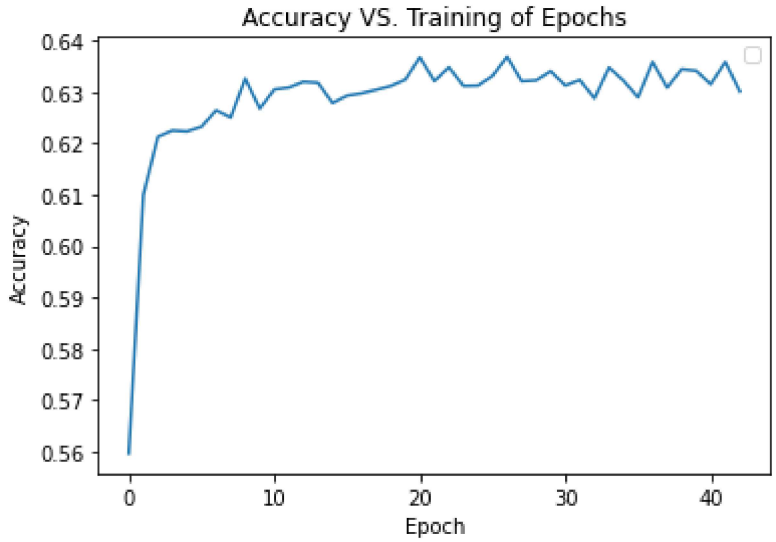
41

| Epoch: 42 | Train Loss: 1.100 | Train Acc: 63.58% | Val. Loss: 0.857| Val. Acc: 71.38%



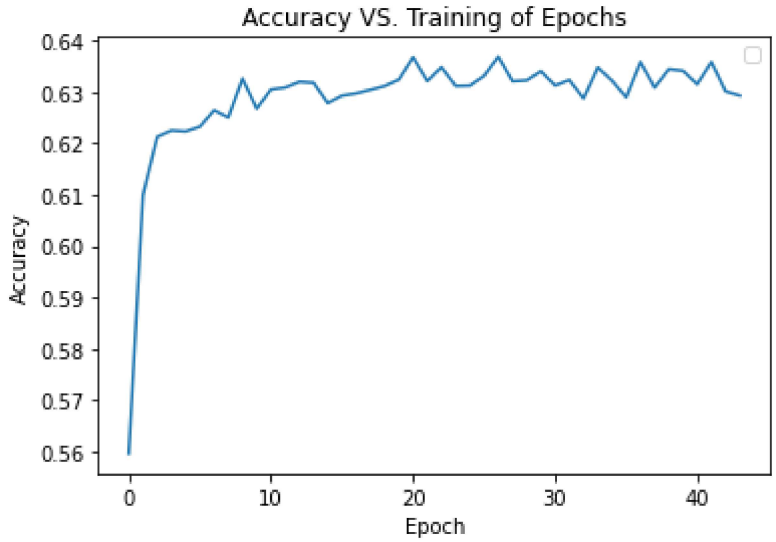
42

```
| Epoch: 43 | Train Loss: 1.116 | Train Acc: 63.01% | Val. Loss: 0.828| Val. Acc: 72.38%
```



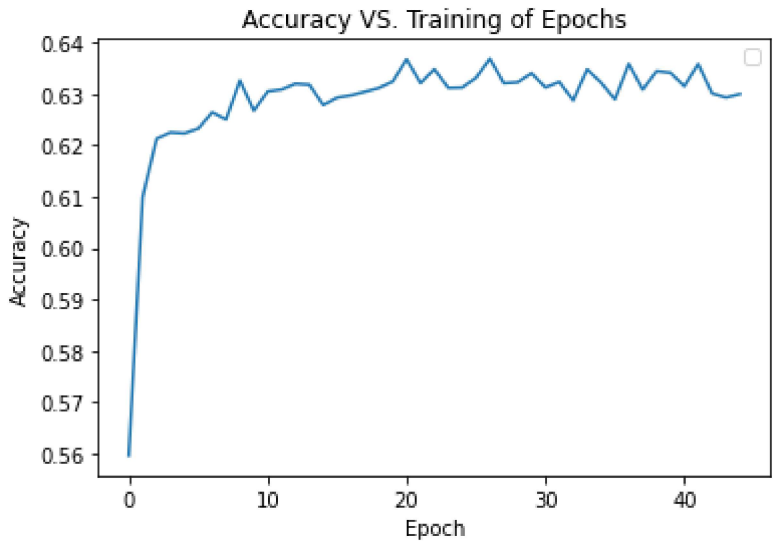
43

```
| Epoch: 44 | Train Loss: 1.119 | Train Acc: 62.93% | Val. Loss: 0.773| Val. Acc: 74.46%
```



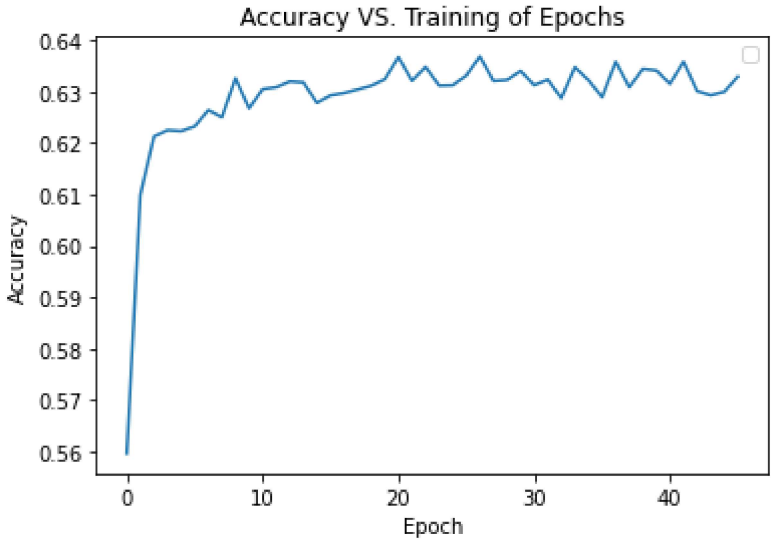
44

```
| Epoch: 45 | Train Loss: 1.114 | Train Acc: 62.99% | Val. Loss: 0.789| Val. Acc: 74.10%
```



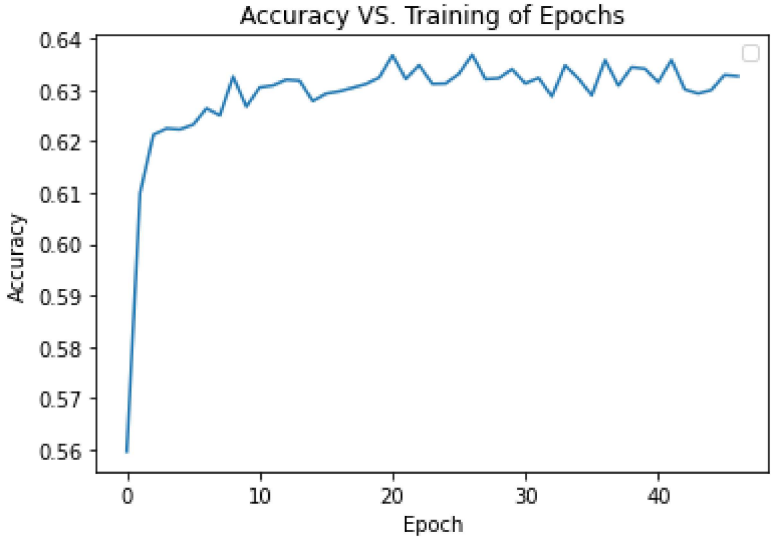
45

```
| Epoch: 46 | Train Loss: 1.110 | Train Acc: 63.29% | Val. Loss: 0.742| Val. Acc: 75.24%
```



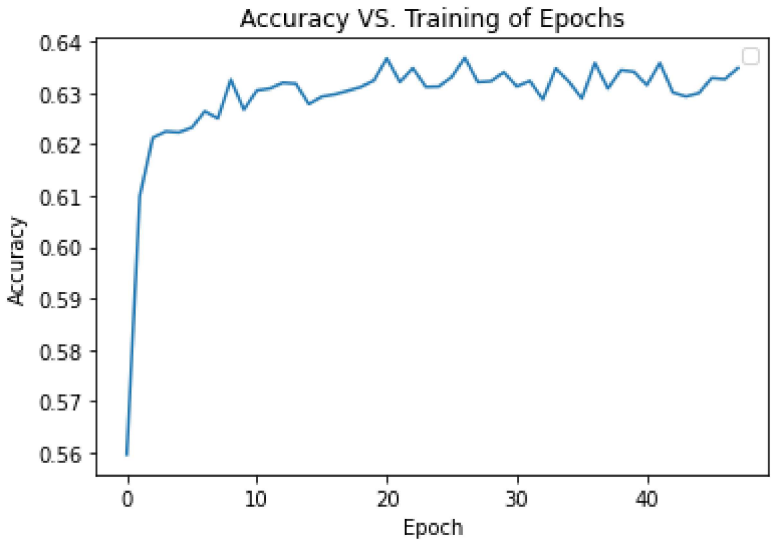
46

```
| Epoch: 47 | Train Loss: 1.116 | Train Acc: 63.26% | Val. Loss: 0.757| Val. Acc: 74.66%
```



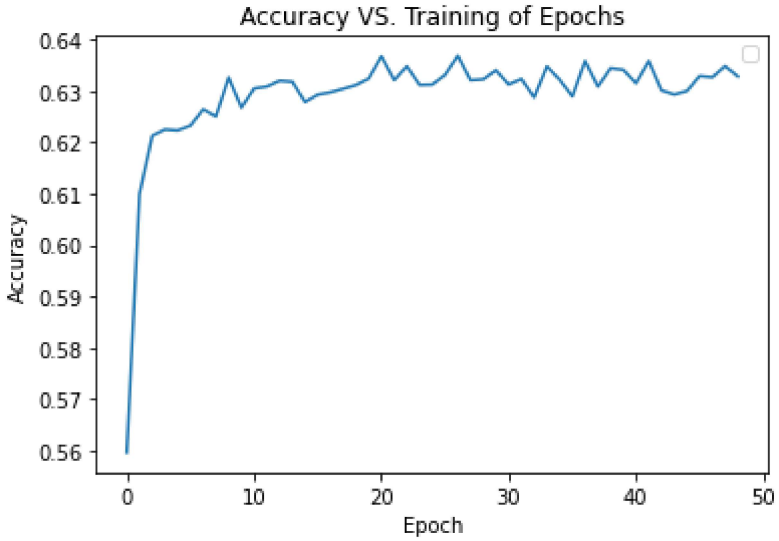
47

```
| Epoch: 48 | Train Loss: 1.104 | Train Acc: 63.48% | Val. Loss: 0.743| Val. Acc: 75.68%
```



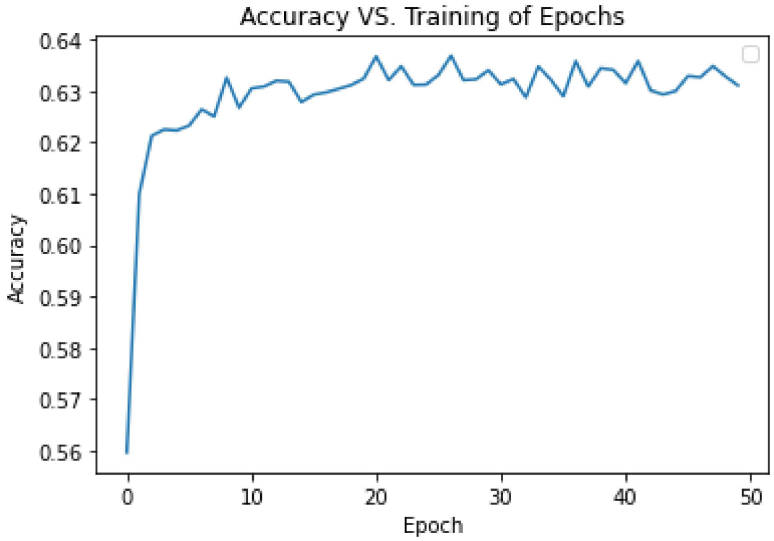
48

```
| Epoch: 49 | Train Loss: 1.109 | Train Acc: 63.28% | Val. Loss: 0.768| Val. Acc: 74.32%
```



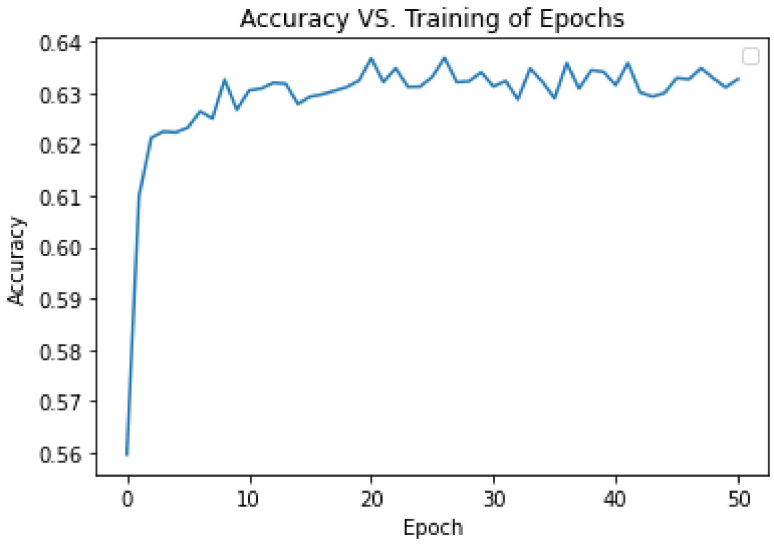
49

```
| Epoch: 50 | Train Loss: 1.110 | Train Acc: 63.10% | Val. Loss: 0.768| Val. Acc: 74.22%
```



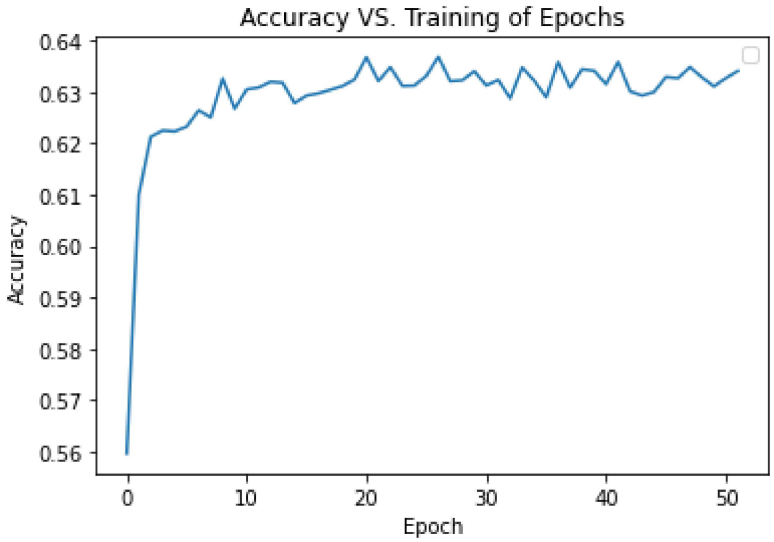
50

```
| Epoch: 51 | Train Loss: 1.104 | Train Acc: 63.26% | Val. Loss: 0.753| Val. Acc: 74.14%
```



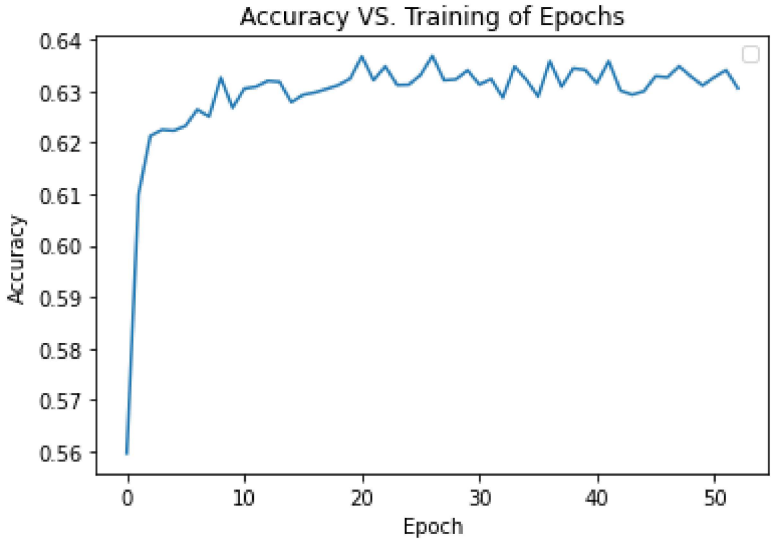
51

| Epoch: 52 | Train Loss: 1.109 | Train Acc: 63.40% | Val. Loss: 0.781| Val. Acc: 74.20%



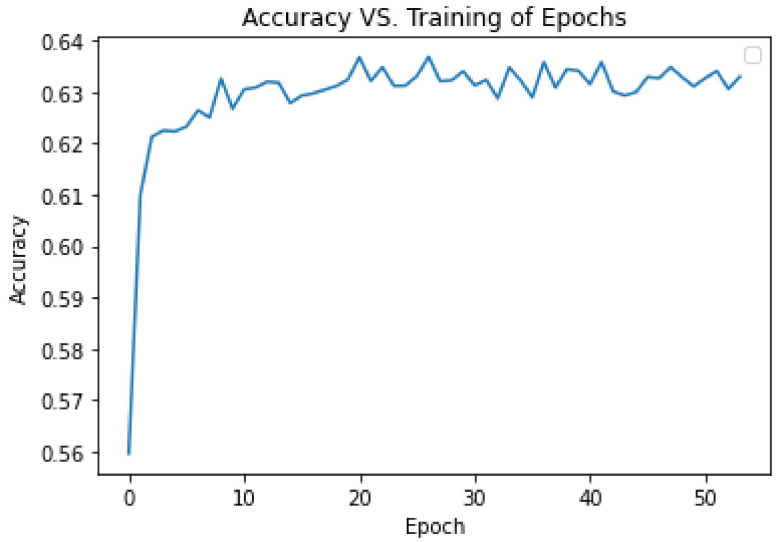
52

| Epoch: 53 | Train Loss: 1.114 | Train Acc: 63.05% | Val. Loss: 0.736| Val. Acc: 75.76%



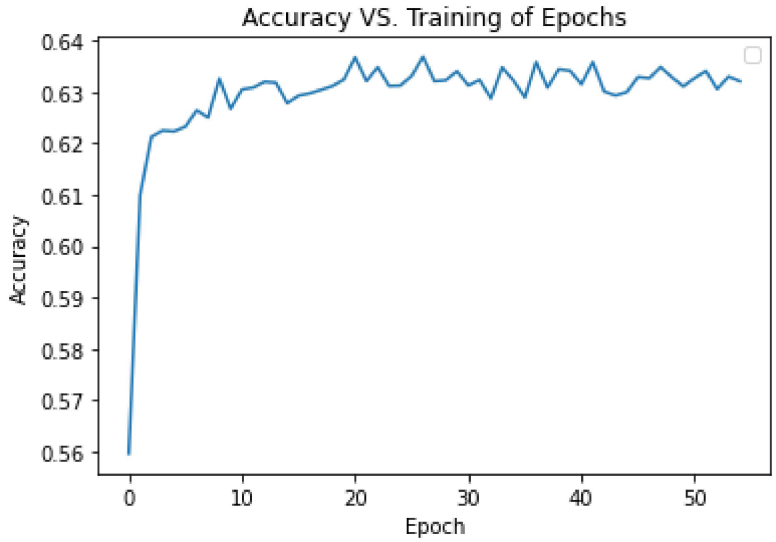
53

```
| Epoch: 54 | Train Loss: 1.115 | Train Acc: 63.29% | Val. Loss: 0.780| Val. Acc: 74.60%
```



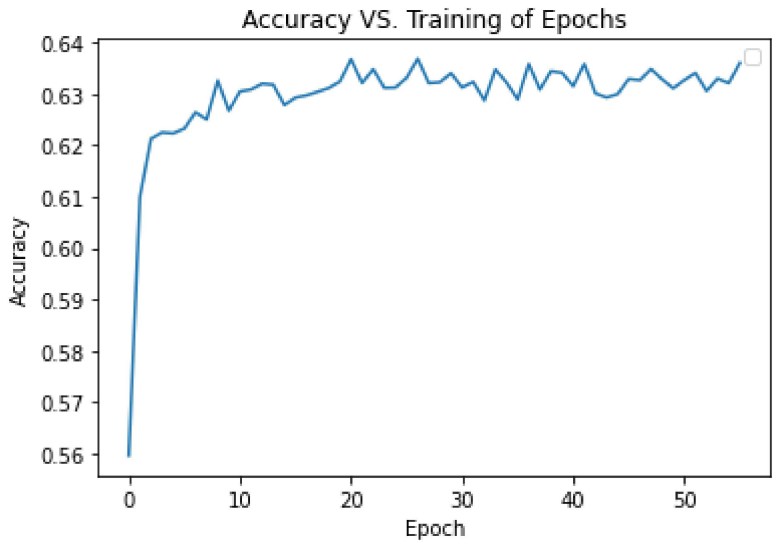
54

```
| Epoch: 55 | Train Loss: 1.108 | Train Acc: 63.20% | Val. Loss: 0.880| Val. Acc: 71.10%
```

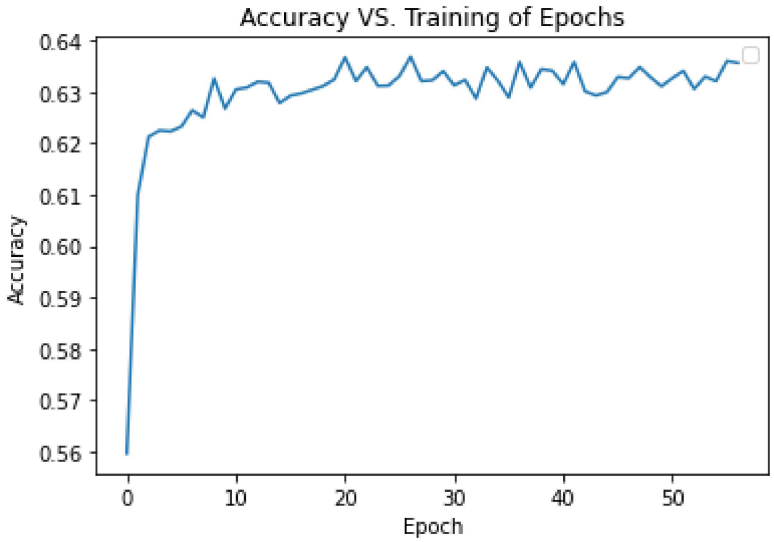


55

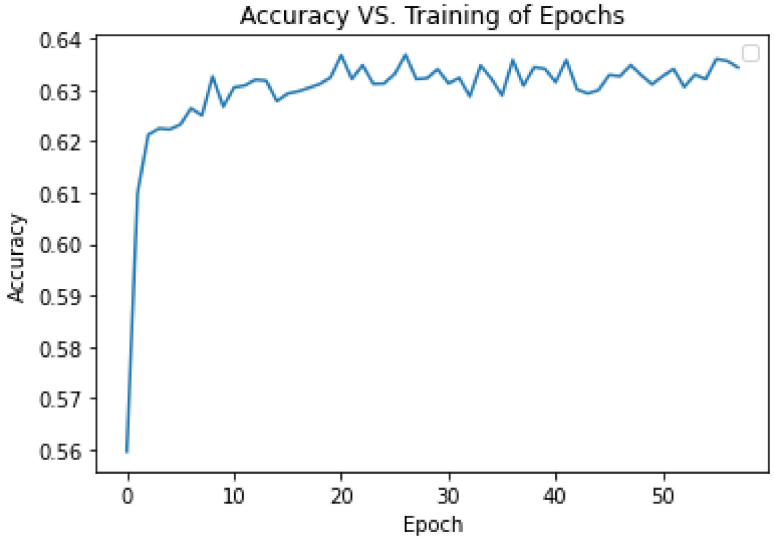
```
| Epoch: 56 | Train Loss: 1.103 | Train Acc: 63.60% | Val. Loss: 0.829| Val. Acc: 72.14%
```



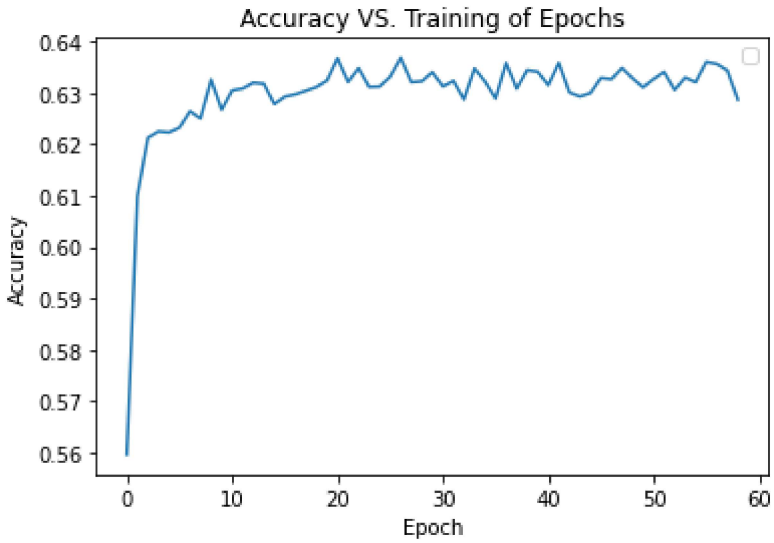
```
56
| Epoch: 57 | Train Loss: 1.098 | Train Acc: 63.56% | Val. Loss: 0.806| Val. Acc: 72.74%
```



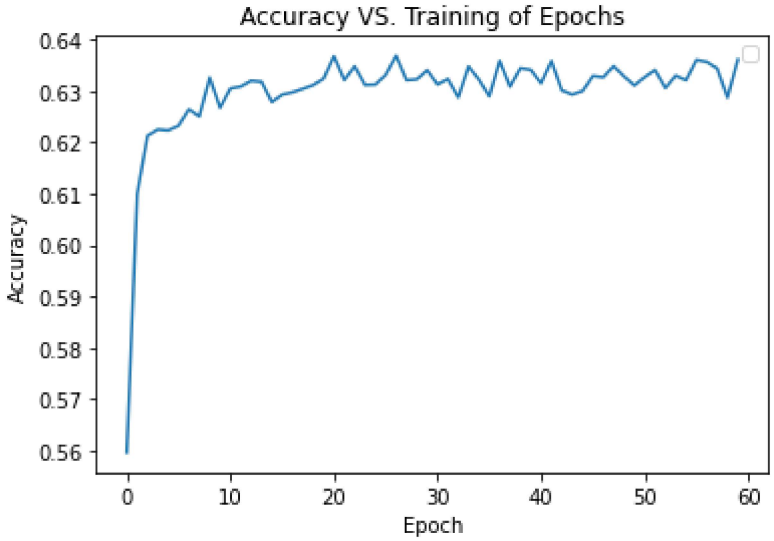

```
57
| Epoch: 58 | Train Loss: 1.111 | Train Acc: 63.43% | Val. Loss: 0.779| Val. Acc: 73.56%
```



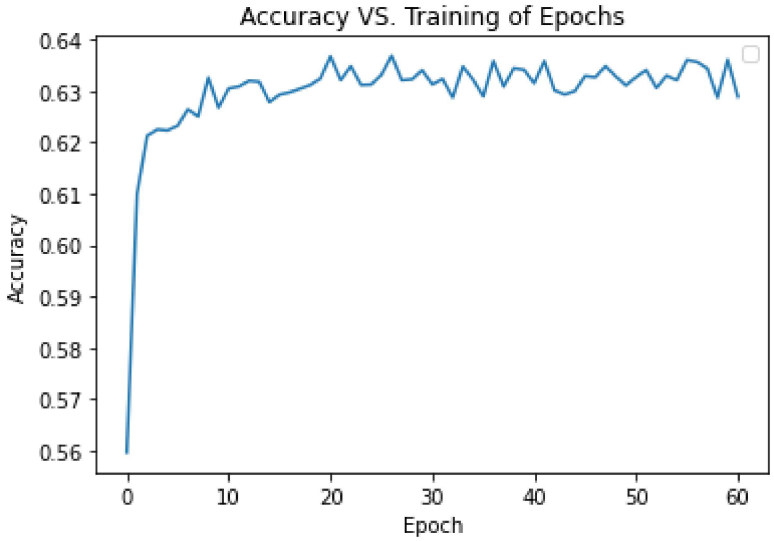

```
58
| Epoch: 59 | Train Loss: 1.114 | Train Acc: 62.87% | Val. Loss: 0.761| Val. Acc: 74.60%
```



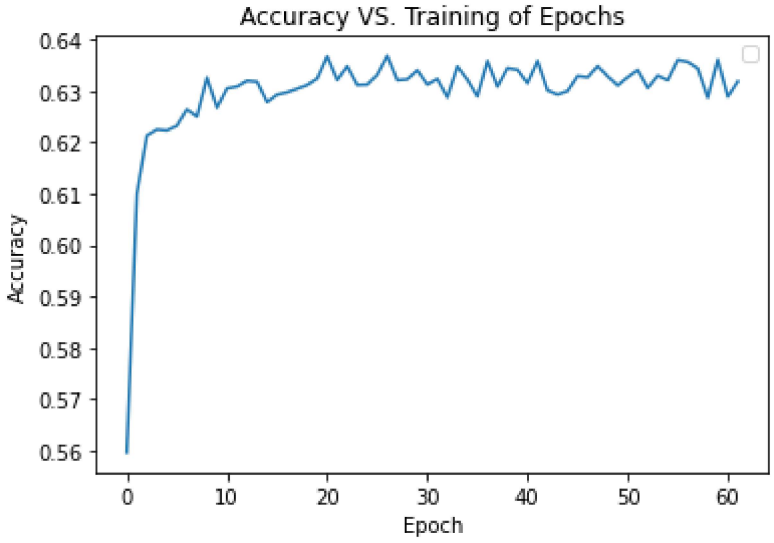
59
| Epoch: 60 | Train Loss: 1.102 | Train Acc: 63.60% | Val. Loss: 0.830| Val. Acc: 72.22%



60
| Epoch: 61 | Train Loss: 1.114 | Train Acc: 62.88% | Val. Loss: 0.866| Val. Acc: 71.74%

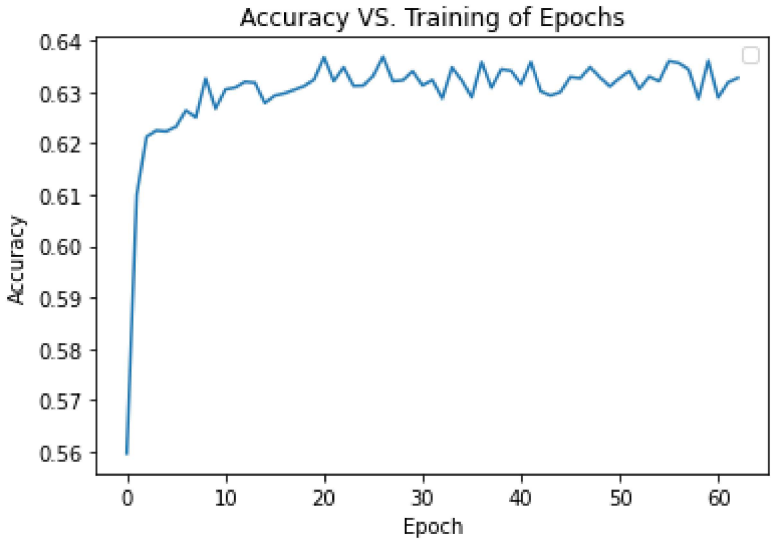


61
| Epoch: 62 | Train Loss: 1.114 | Train Acc: 63.18% | Val. Loss: 0.817| Val. Acc: 72.04%



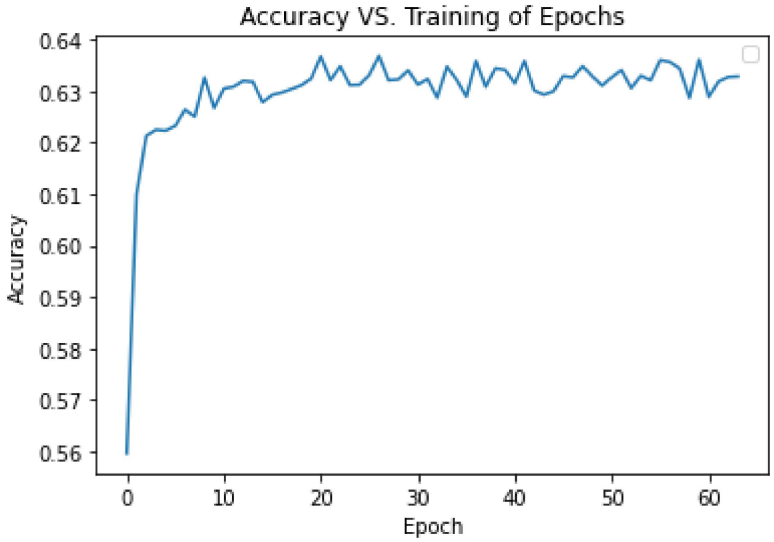
62

```
| Epoch: 63 | Train Loss: 1.112 | Train Acc: 63.27% | Val. Loss: 0.827| Val. Acc: 72.68%
```



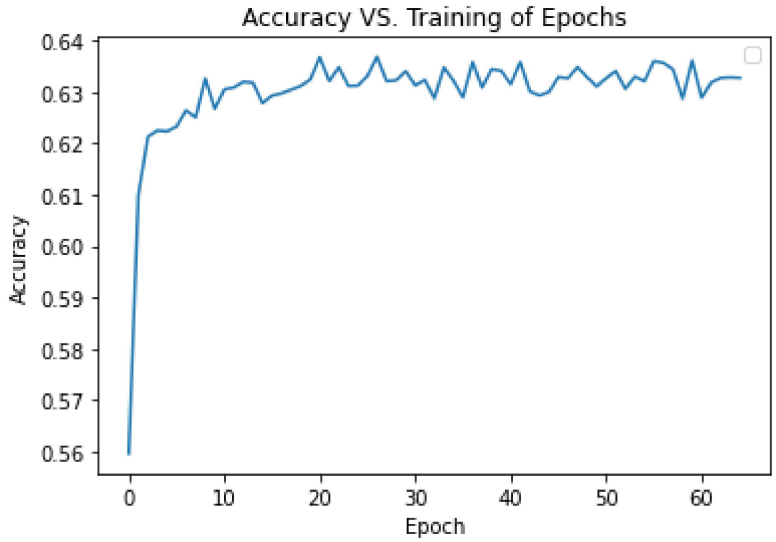
63

```
| Epoch: 64 | Train Loss: 1.109 | Train Acc: 63.28% | Val. Loss: 0.756| Val. Acc: 74.26%
```



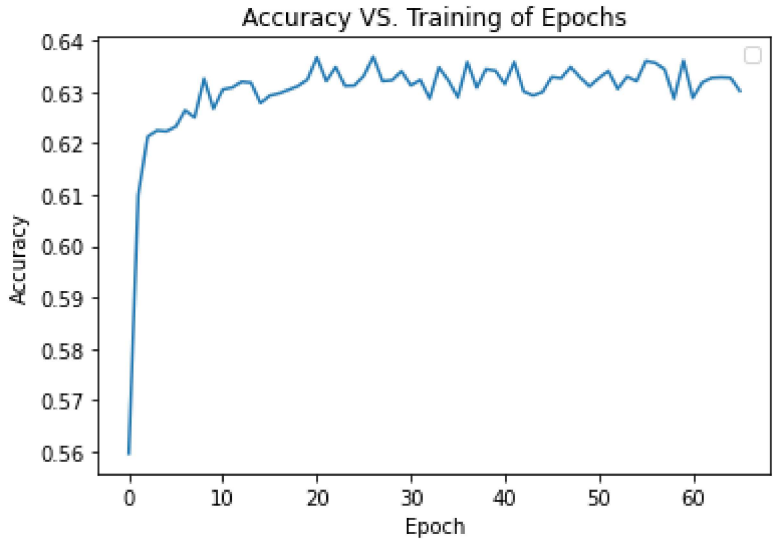
64

```
| Epoch: 65 | Train Loss: 1.112 | Train Acc: 63.27% | Val. Loss: 0.830| Val. Acc: 72.56%
```



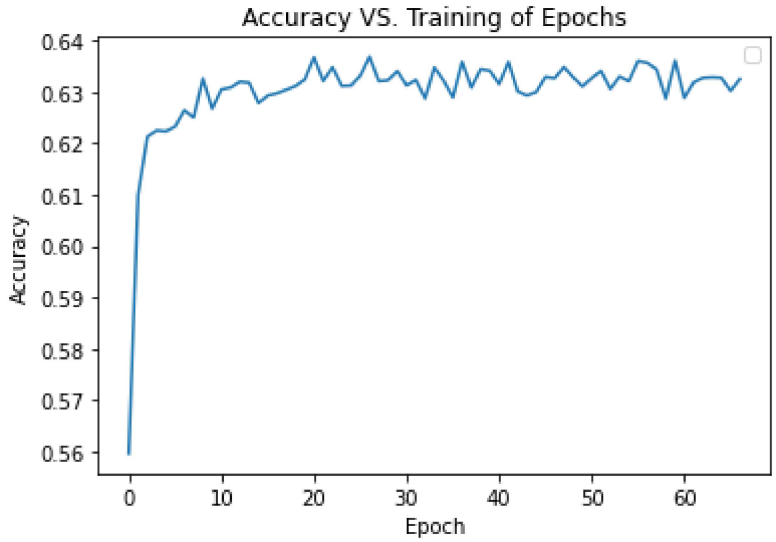
65

```
| Epoch: 66 | Train Loss: 1.116 | Train Acc: 63.02% | Val. Loss: 0.773| Val. Acc: 74.68%
```



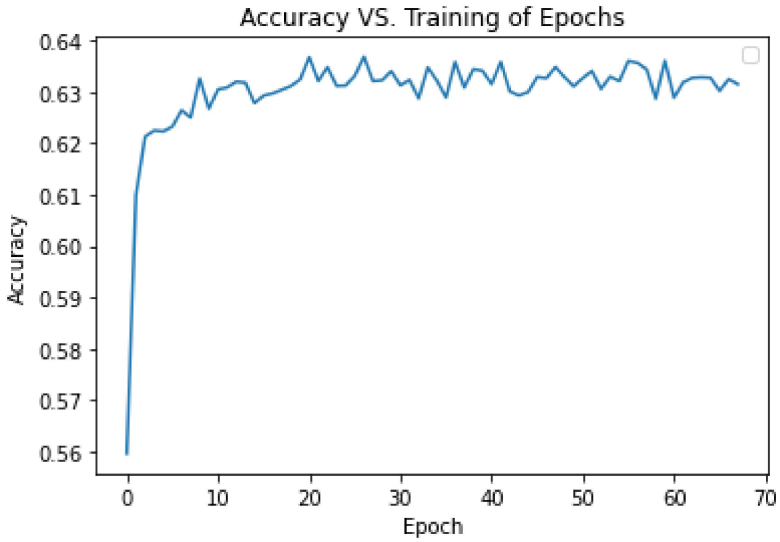
66

```
| Epoch: 67 | Train Loss: 1.111 | Train Acc: 63.24% | Val. Loss: 0.786| Val. Acc: 74.16%
```



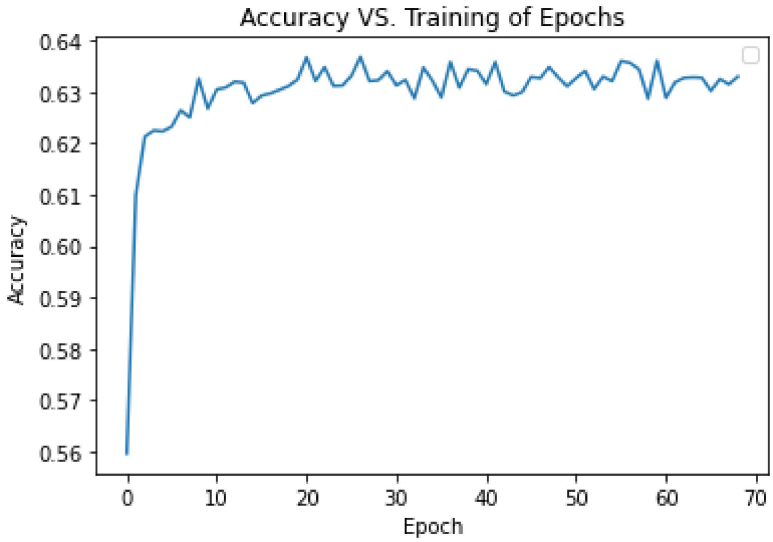
67

```
| Epoch: 68 | Train Loss: 1.115 | Train Acc: 63.14% | Val. Loss: 0.759| Val. Acc: 74.74%
```



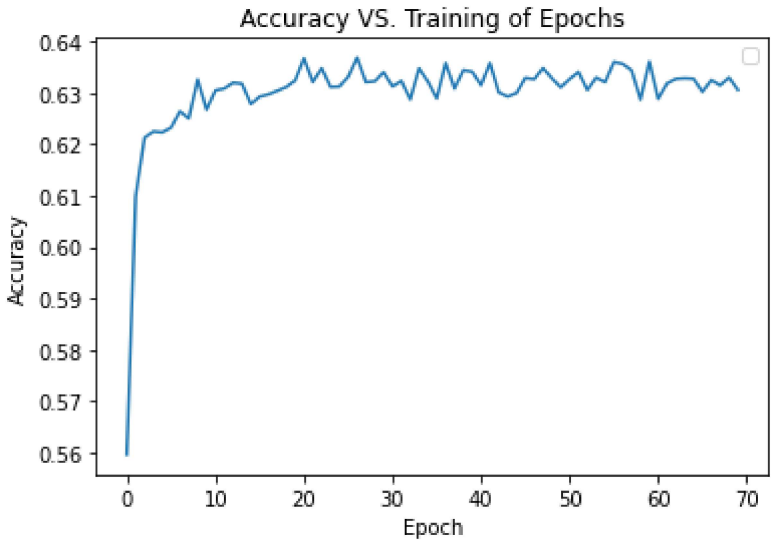
68

```
| Epoch: 69 | Train Loss: 1.110 | Train Acc: 63.29% | Val. Loss: 0.798| Val. Acc: 73.80%
```

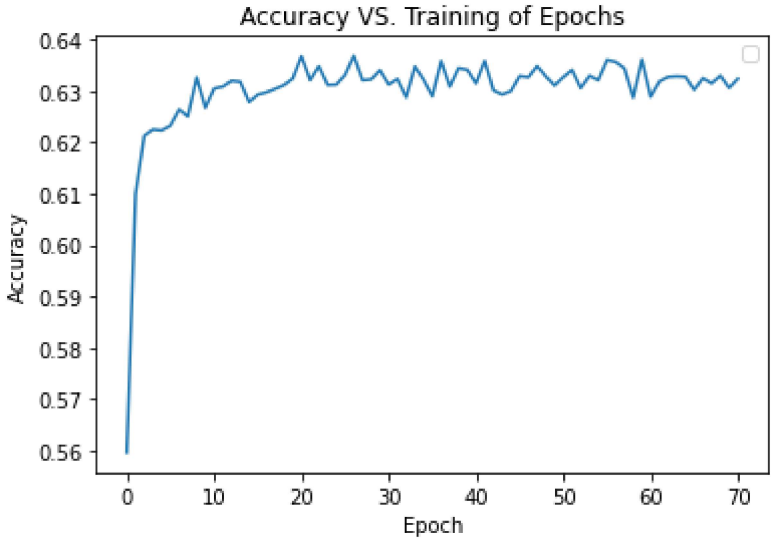


69

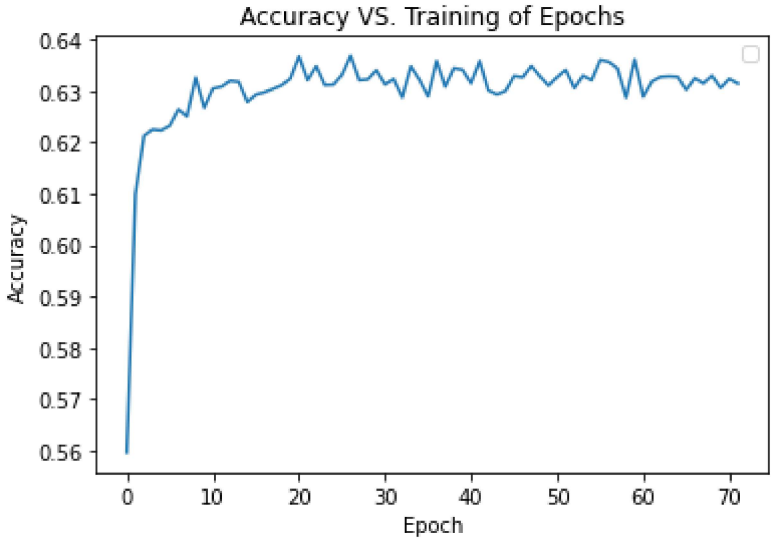
```
| Epoch: 70 | Train Loss: 1.112 | Train Acc: 63.06% | Val. Loss: 0.800| Val. Acc: 72.86%
```



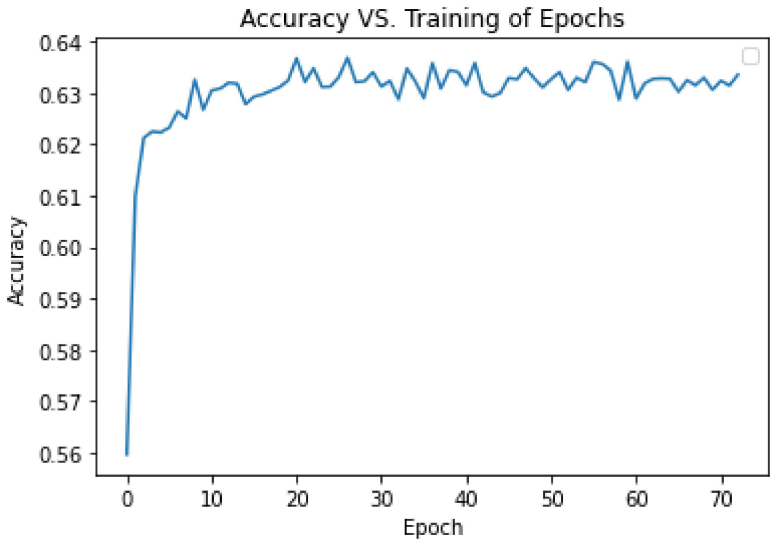
70
| Epoch: 71 | Train Loss: 1.106 | Train Acc: 63.23% | Val. Loss: 0.737| Val. Acc: 75.50%



71
| Epoch: 72 | Train Loss: 1.110 | Train Acc: 63.14% | Val. Loss: 0.764| Val. Acc: 73.98%

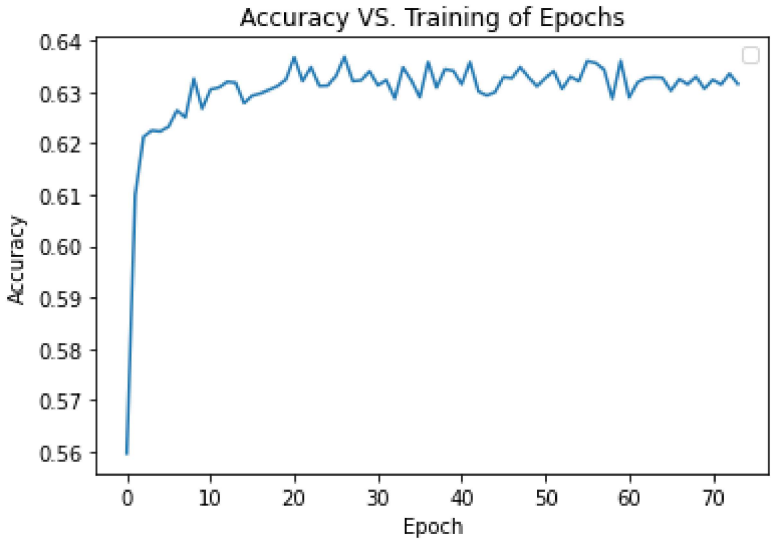


72
| Epoch: 73 | Train Loss: 1.109 | Train Acc: 63.35% | Val. Loss: 0.792| Val. Acc: 73.12%



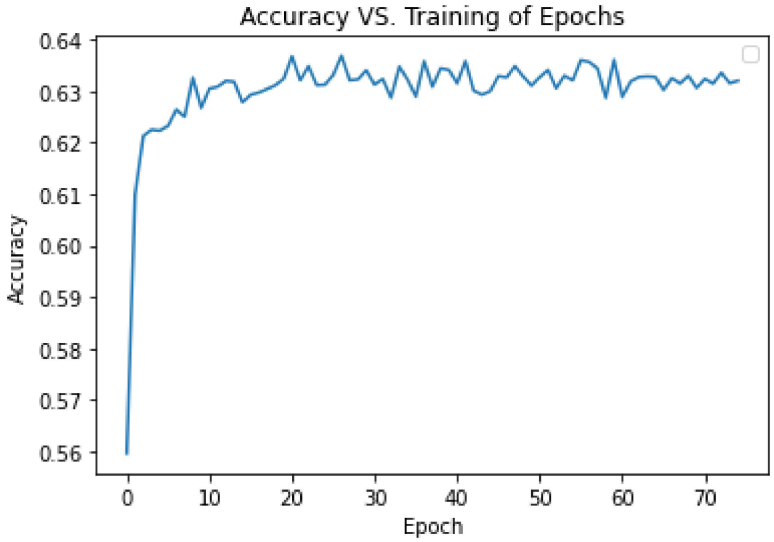
73

| Epoch: 74 | Train Loss: 1.109 | Train Acc: 63.15% | Val. Loss: 0.762| Val. Acc: 74.32%



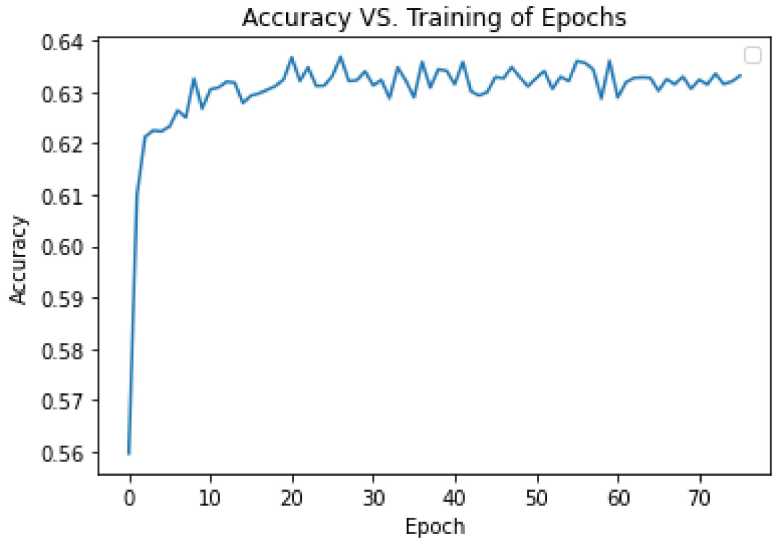
74

| Epoch: 75 | Train Loss: 1.108 | Train Acc: 63.20% | Val. Loss: 0.784| Val. Acc: 73.70%



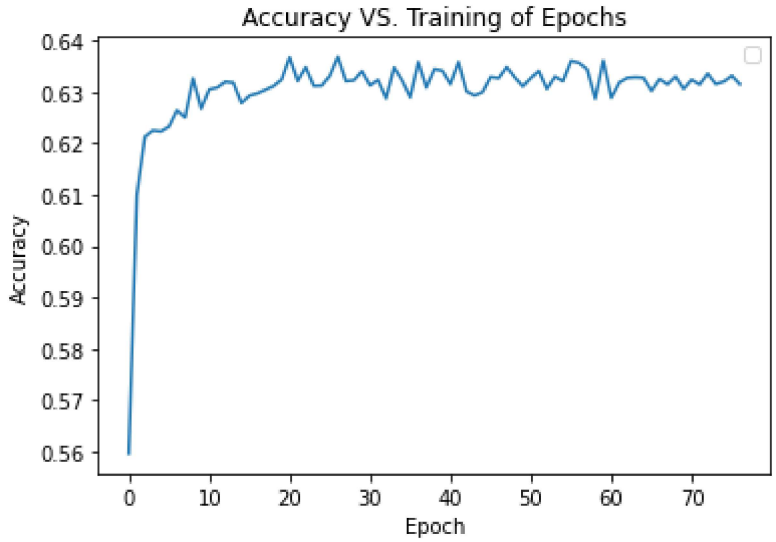
75

```
| Epoch: 76 | Train Loss: 1.107 | Train Acc: 63.31% | Val. Loss: 0.766| Val. Acc: 74.68%
```



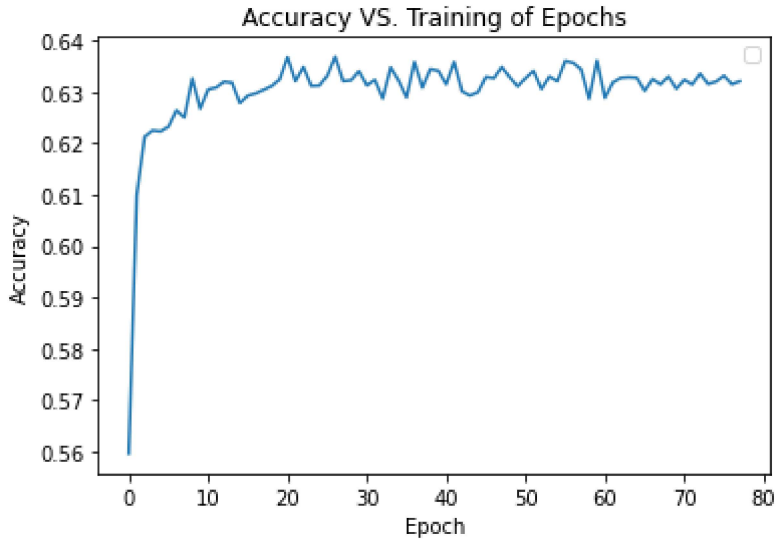
76

```
| Epoch: 77 | Train Loss: 1.119 | Train Acc: 63.15% | Val. Loss: 0.809| Val. Acc: 72.94%
```

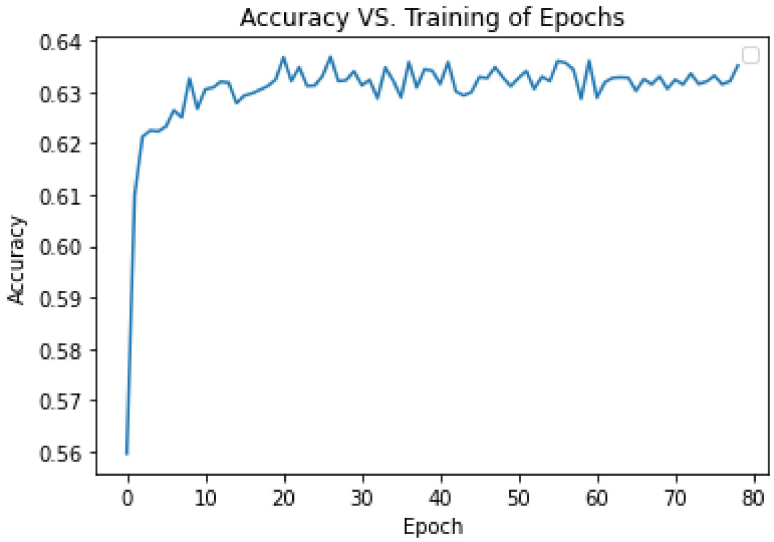


77

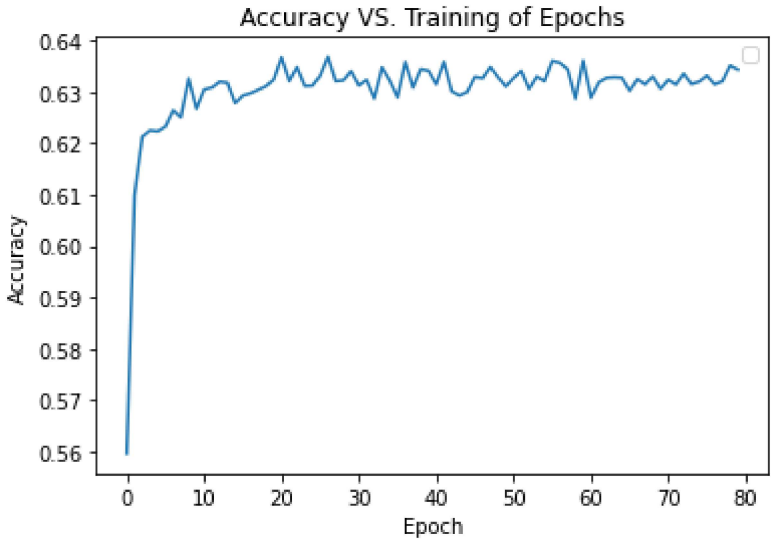
```
| Epoch: 78 | Train Loss: 1.111 | Train Acc: 63.20% | Val. Loss: 0.787| Val. Acc: 73.14%
```



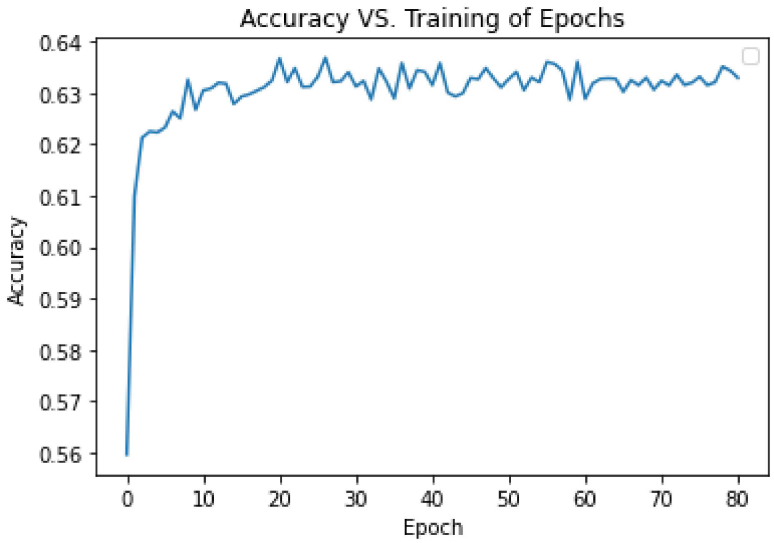
```
78
| Epoch: 79 | Train Loss: 1.104 | Train Acc: 63.51% | Val. Loss: 0.804| Val. Acc: 72.98%
|
```



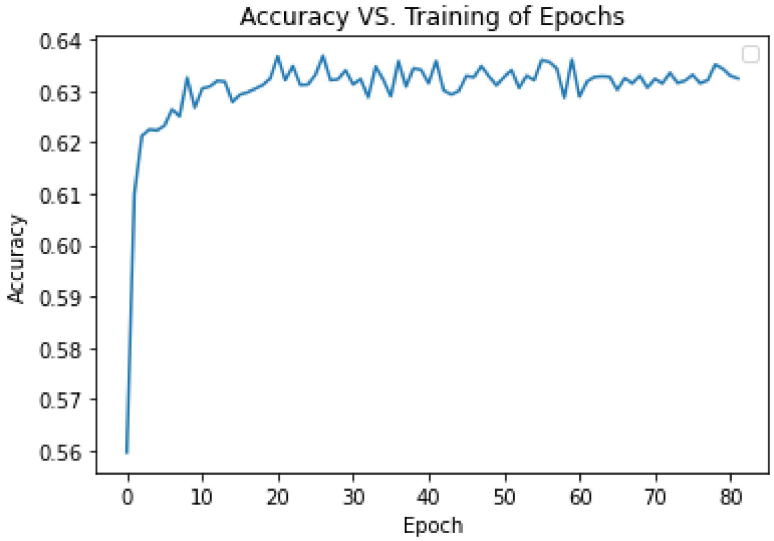

```
79
| Epoch: 80 | Train Loss: 1.101 | Train Acc: 63.43% | Val. Loss: 0.800| Val. Acc: 73.54%
|
```



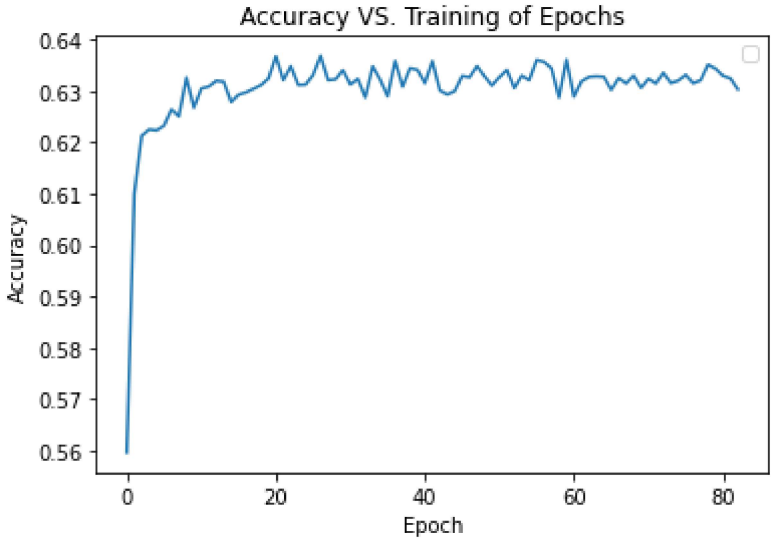

```
80
| Epoch: 81 | Train Loss: 1.115 | Train Acc: 63.29% | Val. Loss: 0.790| Val. Acc: 73.66%
|
```



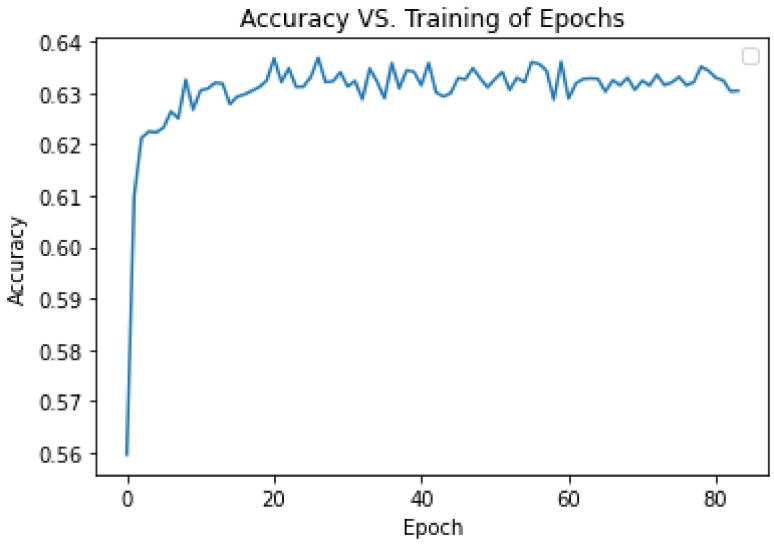
81
| Epoch: 82 | Train Loss: 1.113 | Train Acc: 63.24% | Val. Loss: 0.850| Val. Acc: 71.34%



82
| Epoch: 83 | Train Loss: 1.114 | Train Acc: 63.03% | Val. Loss: 0.777| Val. Acc: 73.94%

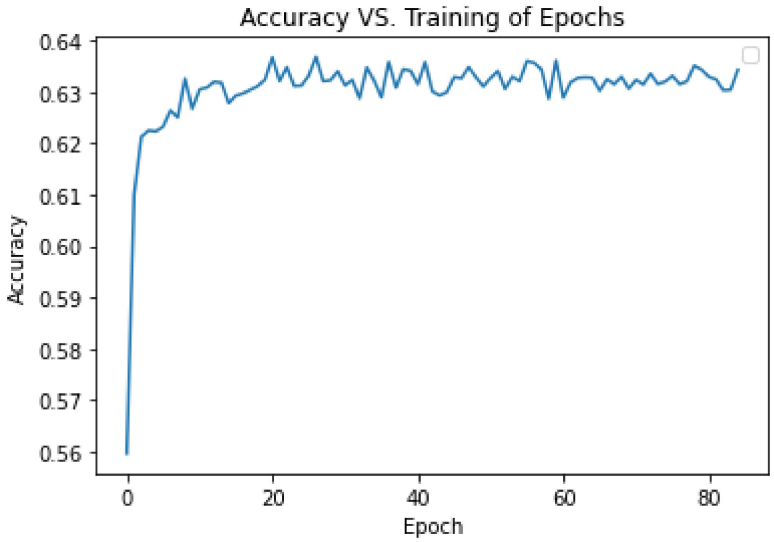


83
| Epoch: 84 | Train Loss: 1.115 | Train Acc: 63.04% | Val. Loss: 0.830| Val. Acc: 72.46%



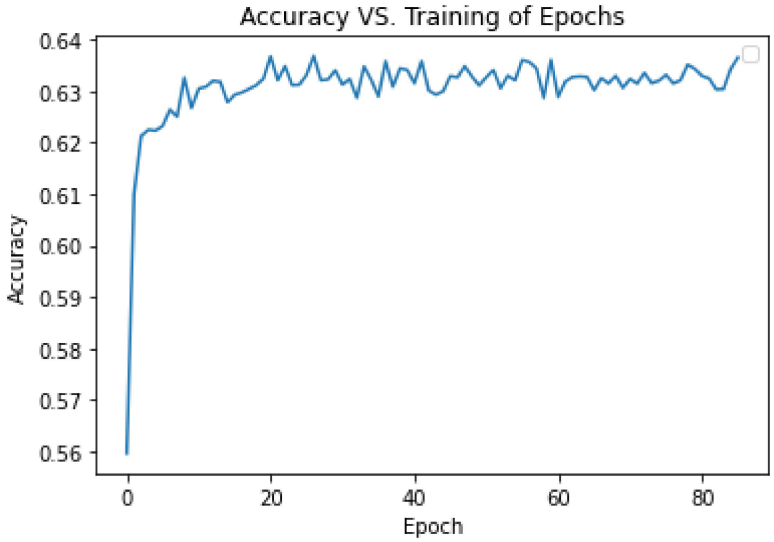
84

```
| Epoch: 85 | Train Loss: 1.106 | Train Acc: 63.42% | Val. Loss: 0.737| Val. Acc: 75.84%
```



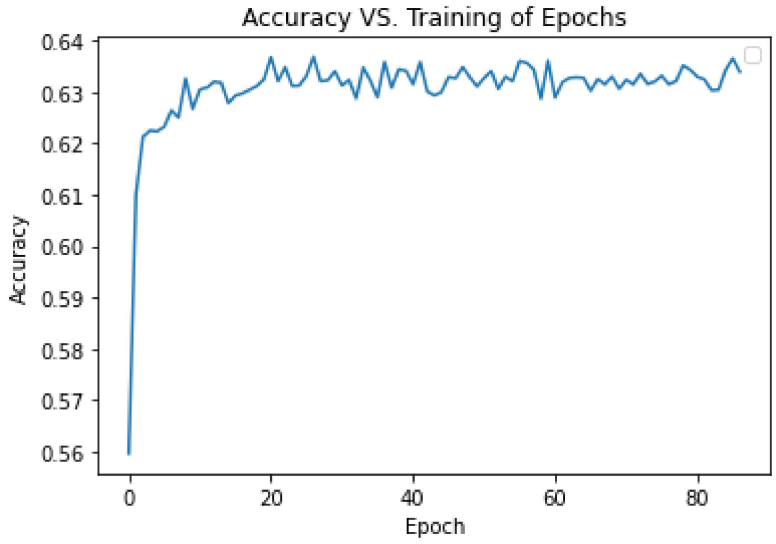
85

```
| Epoch: 86 | Train Loss: 1.100 | Train Acc: 63.64% | Val. Loss: 0.803| Val. Acc: 72.96%
```



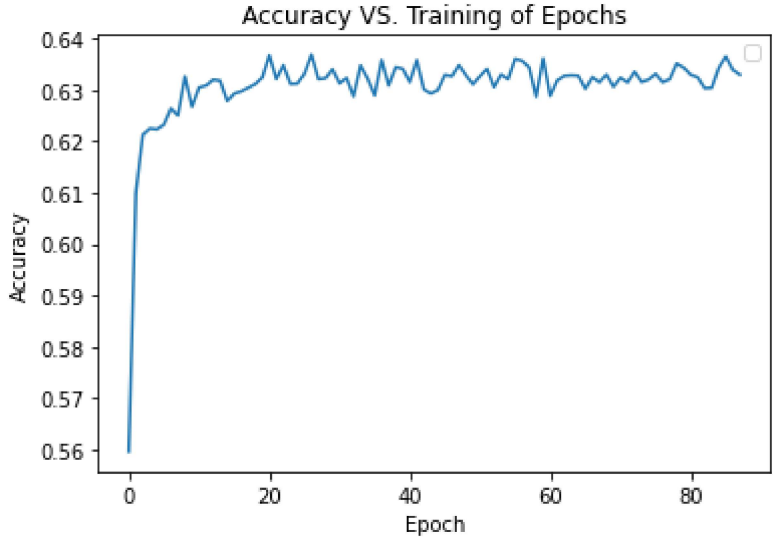
86

```
| Epoch: 87 | Train Loss: 1.103 | Train Acc: 63.39% | Val. Loss: 0.843| Val. Acc: 71.76%
```



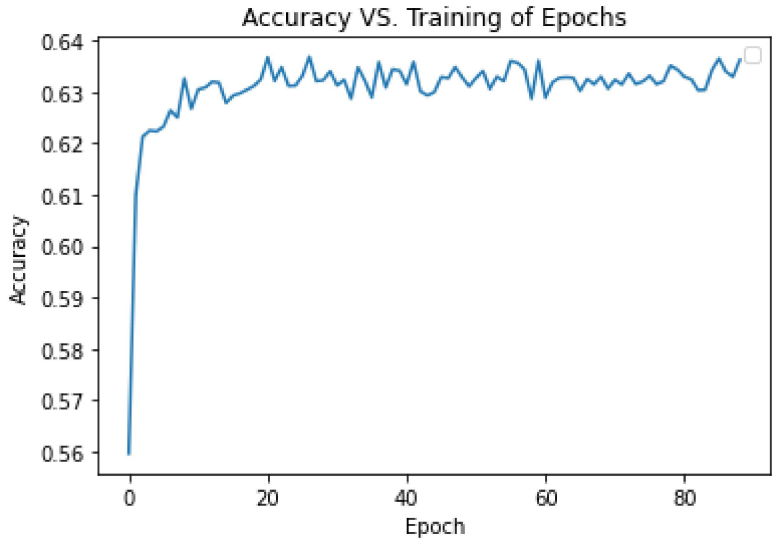
87

```
| Epoch: 88 | Train Loss: 1.109 | Train Acc: 63.29% | Val. Loss: 0.768| Val. Acc: 74.34%
```

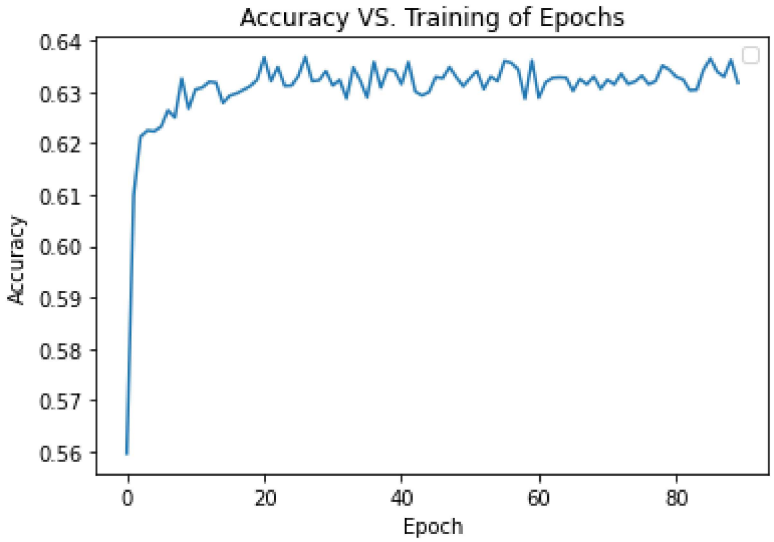


88

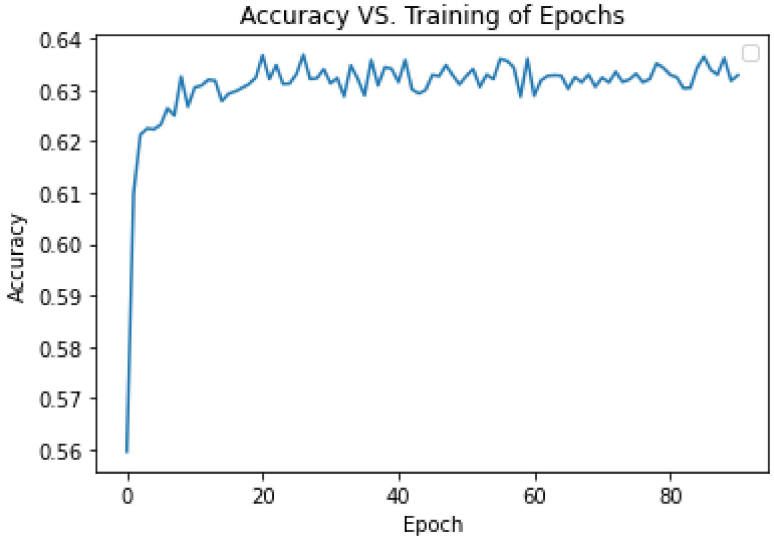
```
| Epoch: 89 | Train Loss: 1.099 | Train Acc: 63.62% | Val. Loss: 0.753| Val. Acc: 74.18%
```



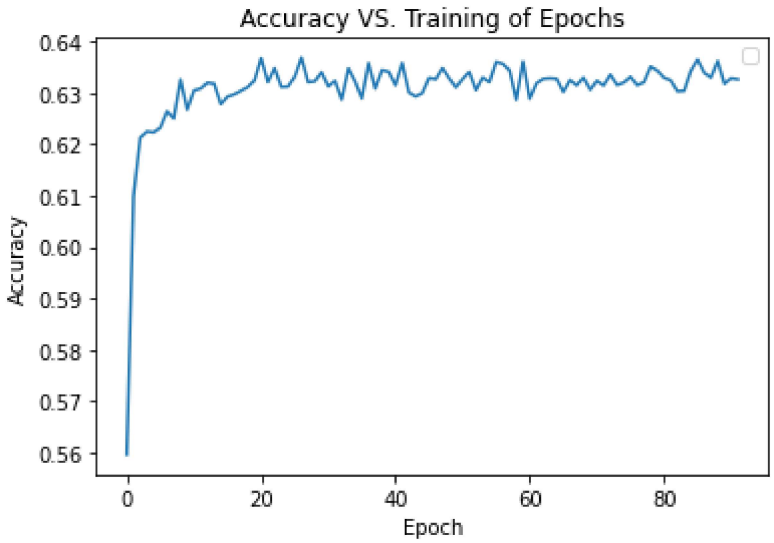
```
89
| Epoch: 90 | Train Loss: 1.112 | Train Acc: 63.17% | Val. Loss: 0.748| Val. Acc: 75.32%
```



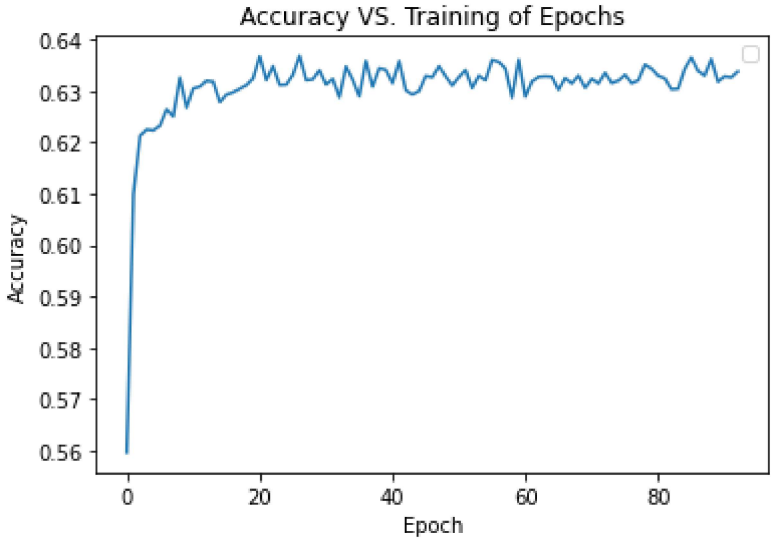

```
90
| Epoch: 91 | Train Loss: 1.114 | Train Acc: 63.28% | Val. Loss: 0.828| Val. Acc: 73.42%
```



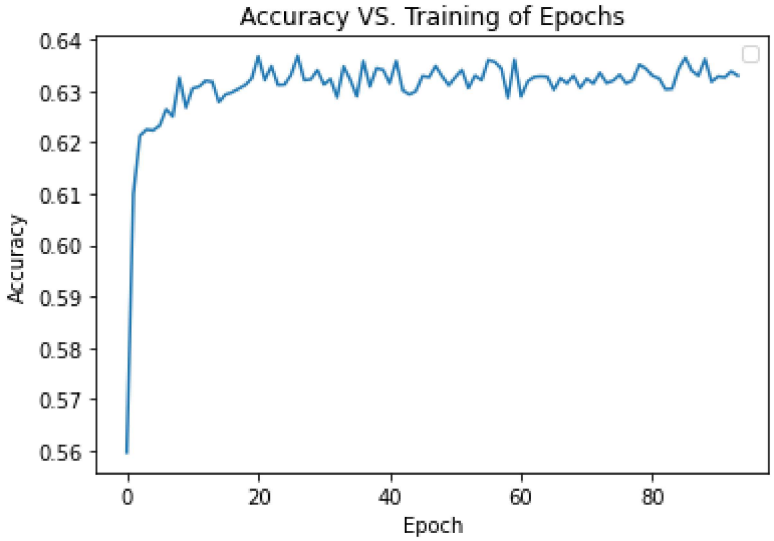

```
91
| Epoch: 92 | Train Loss: 1.110 | Train Acc: 63.26% | Val. Loss: 0.833| Val. Acc: 72.60%
```



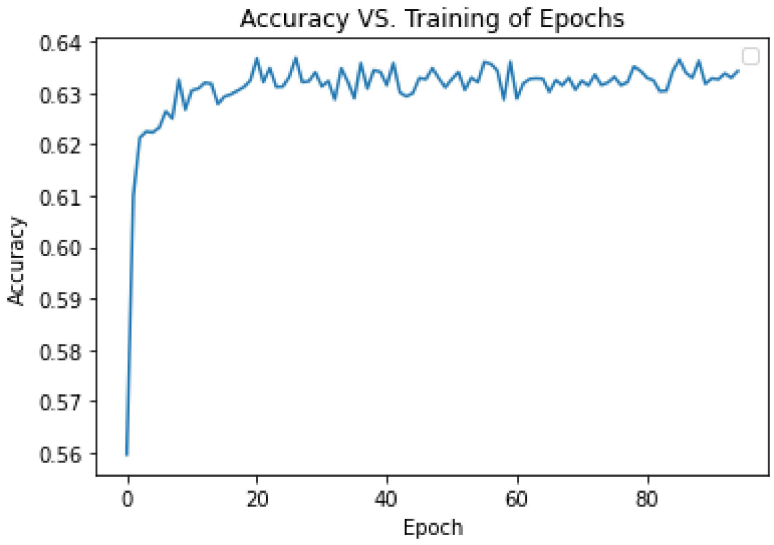
92
| Epoch: 93 | Train Loss: 1.103 | Train Acc: 63.37% | Val. Loss: 0.808| Val. Acc: 72.94%



93
| Epoch: 94 | Train Loss: 1.112 | Train Acc: 63.29% | Val. Loss: 0.768| Val. Acc: 74.26%

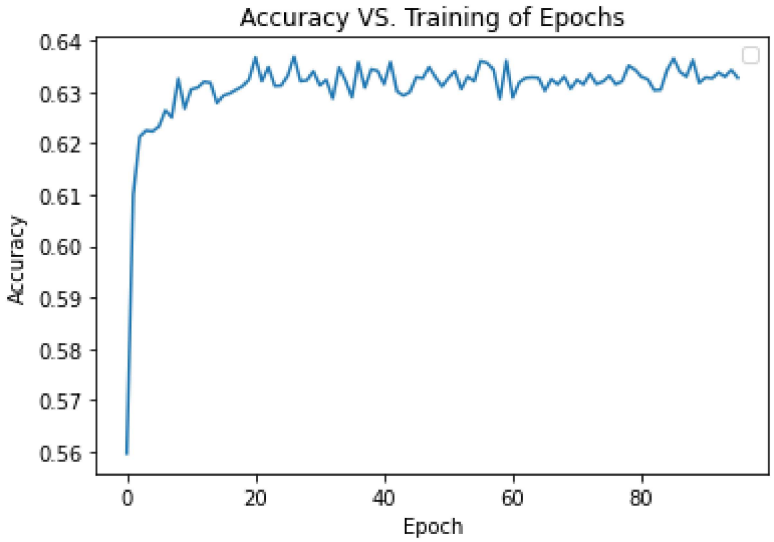


94
| Epoch: 95 | Train Loss: 1.101 | Train Acc: 63.42% | Val. Loss: 0.763| Val. Acc: 74.80%



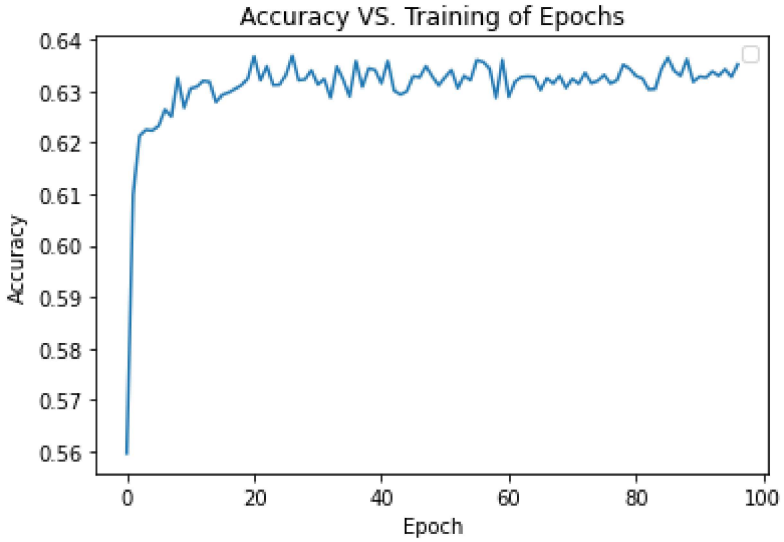
95

| Epoch: 96 | Train Loss: 1.112 | Train Acc: 63.27% | Val. Loss: 0.776| Val. Acc: 74.44%



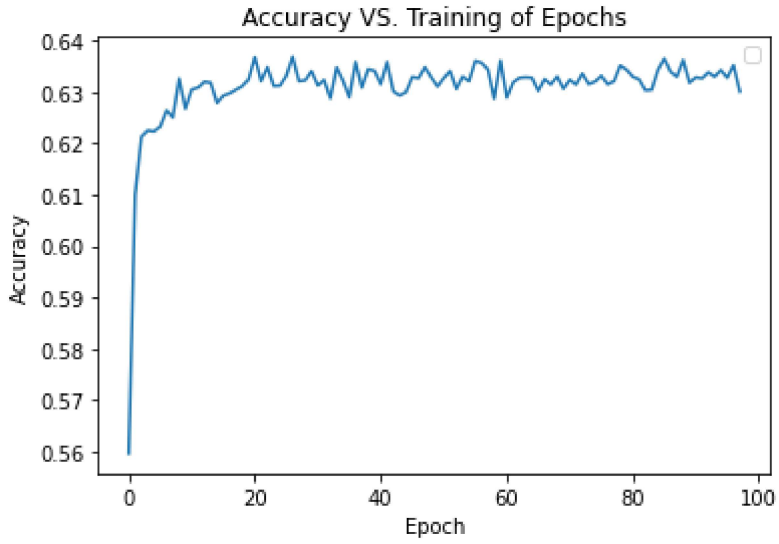
96

| Epoch: 97 | Train Loss: 1.106 | Train Acc: 63.51% | Val. Loss: 0.800| Val. Acc: 73.44%

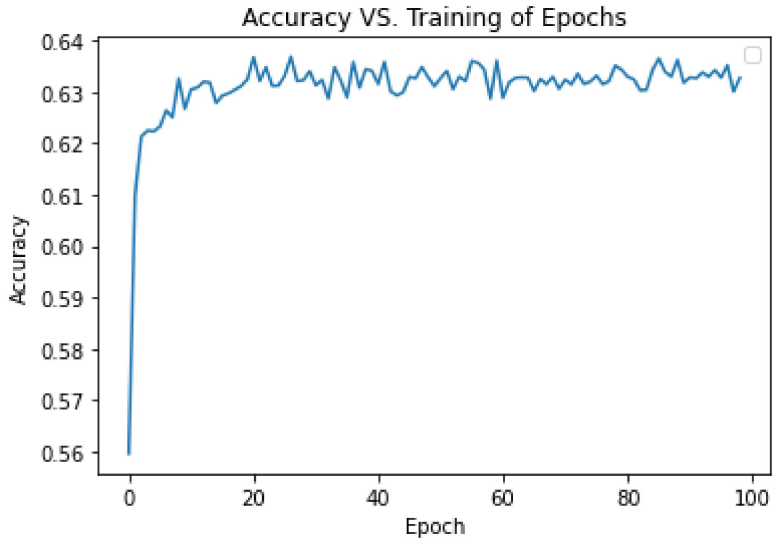


97

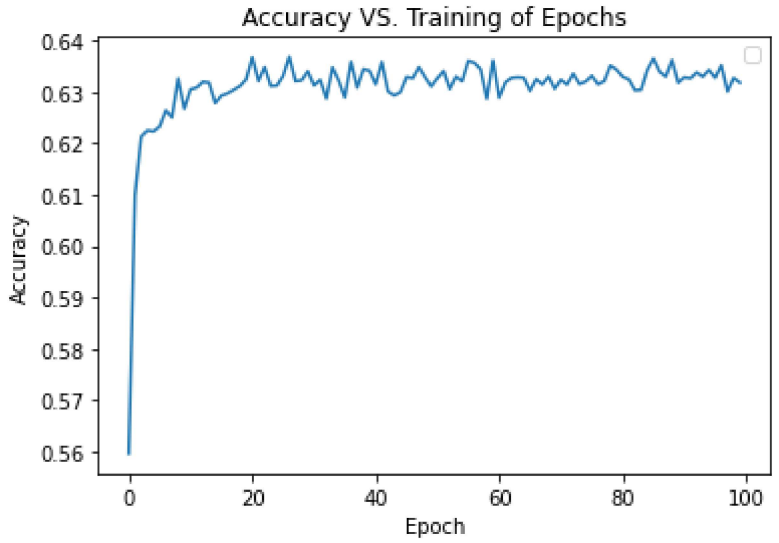
```
| Epoch: 98 | Train Loss: 1.114 | Train Acc: 63.00% | Val. Loss: 0.802| Val. Acc: 73.00%
|
```

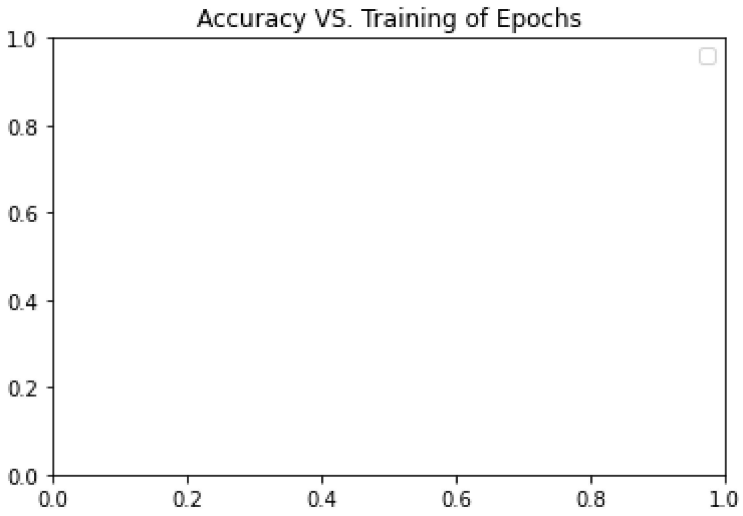



```
98
| Epoch: 99 | Train Loss: 1.104 | Train Acc: 63.27% | Val. Loss: 0.885| Val. Acc: 70.64%
|
```




```
99
| Epoch: 100 | Train Loss: 1.114 | Train Acc: 63.18% | Val. Loss: 0.799| Val. Acc: 73.5
6% |
```



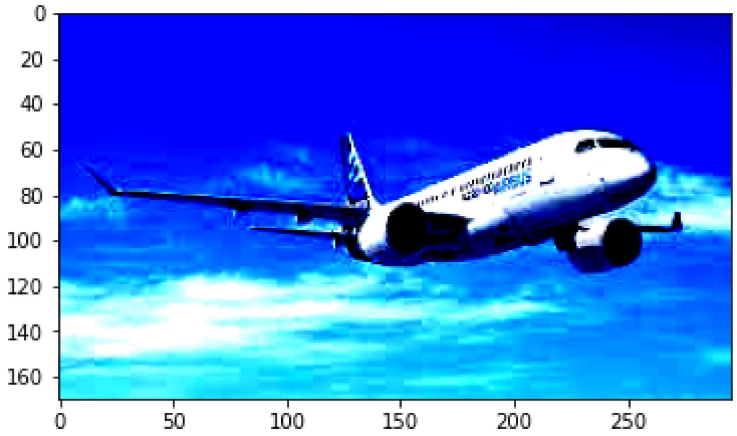


In [118…
```
#3. OUTPUT

model.load_state_dict(torch.load(MODEL_SAVE_PATH)) #Load best weights from file
test_loss, test_acc = common.evaluate(model, device, valid_iterator, criterion) #Test L
print(f'| Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:05.2f}%')
```

```
| Test Loss: 0.740 | Test Acc: 75.80%
```

In [132…
```
test_image = '/home/prawat/test_image_2.jpg'

def image_loader(loader, test_image):
    image = Image.open(test_image)
    image = loader(image).float()
    image = torch.tensor(image, requires_grad=True)
    image = image.unsqueeze(0)
    return image

infer_transforms = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.4

image = image_loader(infer_transforms, test_image)

plt.imshow(image[0].permute(1,2,0).clone().detach().numpy())

output=model(image)
index = output.data.cpu().numpy().argmax()
predict_value,predict_idx = torch.max(output,1)
classes = ('Airplane', 'Automobile', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Tr
print(index)
print('This is ' + classes[index] + '!!!')
```

```
/local_scratch/pbs.3501593.pbs02/ipykernel_3237046/3869487171.py:6: UserWarning: To copy
construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourc
eTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
  image = torch.tensor(image, requires_grad=True)
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
0
This is Airplane!!!
```



In [ ]: