In [1]:
```
import torchvision.models
```

In [2]:
```
#alexNet = torchvision.models.alexnet(pretrained=True)
#googleNet = torchvision.models.inception.inception_v3(pretrained=True)
vgg16 = torchvision.models.vgg.vgg16(pretrained=True)
#resnet152 = torchvision.models.resnet.resnet152(pretrained=True)
```

In [3]:
```
vgg16
```

Out[3]:
```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

In [4]:
```
import time
import os
import numpy as np
import torch

import torchvision
```

```
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt

import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim #for gradient descent


data_dir = "C:/Users/panji/OneDrive/Desktop/images"  # it is richard's drive location o
train_dir = os.path.join(data_dir, 'TRAIN/') # in the train folder
val_dir = os.path.join(data_dir, 'VALIDATION/') # in the validation folder
test_dir = os.path.join(data_dir,'TEST/')
# classes are folders in each directory with these names
classes = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']

#mnist_new = datasets.MNIST('data', train=True, download=True,
                           #transform=transforms.RandomRotation(25))

data_transform = transforms.Compose([transforms.ToTensor(),transforms.Resize([224,224])
         transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),transforms.RandomRotati


train_data = datasets.ImageFolder(train_dir, transform=data_transform)
val_data = datasets.ImageFolder(val_dir, transform=data_transform)
test_data = datasets.ImageFolder(test_dir, transform=data_transform)

print('Num training images: ', len(train_data))
print('Num validation images: ', len(val_data))
print('Num test images: ', len(test_data))
```

```
Num training images:  7392
Num validation images:  2565
Num test images:  2487
```

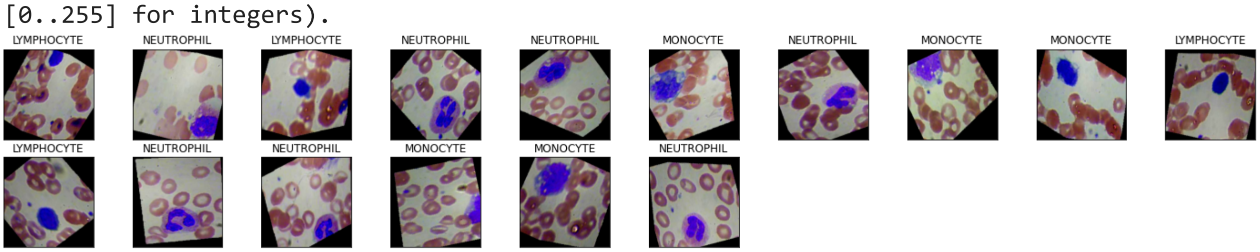
In [5]:
```
torch.cuda.is_available()
```

Out[5]:
```
True
```

In [6]:
```
batch_size  = 36


# prepare data loaders
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,  shuffle=
val_loader = torch.utils.data.DataLoader(val_data, batch_size=batch_size,  shuffle=True
test_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size,  shuffle=Tr
```

In [7]:
```
# Visualize some sample data

# obtain one batch of training images
dataiter = iter(train_loader)   # get the image from train_loader
images, labels = dataiter.next()
images = images.numpy() # convert images to numpy for display

# plot the images in the batch, along with the corresponding labels
fig = plt.figure(figsize=(25, 4))
for idx in np.arange(16):
    ax = fig.add_subplot(2, 20/2, idx+1, xticks=[], yticks=[])
    plt.imshow(np.transpose(images[idx], (1, 2, 0)))
    ax.set_title(classes[labels[idx]])
```

```
<ipython-input-7-82b2ddc7ad6d>:11: MatplotlibDeprecationWarning: Passing non-integers as three-element position specification is deprecated since 3.3 and will be removed two min or releases later.
  ax = fig.add_subplot(2, 20/2, idx+1, xticks=[], yticks=[])
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



In [8]:

```python
torch.manual_seed(1) # set the random seed

# define dataloader parameters
batch_size  = 1 # process 1 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
                                           num_workers=num_workers, shuffle=True)
val_loader = torch.utils.data.DataLoader(val_data, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size,
                                          num_workers=num_workers, shuffle=True)

# img = ... a PyTorch tensor with shape [N,3,224,224] containing hand images ...
#features = alexnet.features(img)

vgg16 = torchvision.models.vgg16(pretrained=True)
# location on Google Drive
```

```
train_data_path = 'C:/Users/panji/OneDrive/Desktop/vgg16_feature_augmentation/train/'
valid_data_path = 'C:/Users/panji/OneDrive/Desktop/vgg16_feature_augmentation/val/'
test_data_path = 'C:/Users/panji/OneDrive/Desktop/vgg16_feature_augmentation/test/'

#save the features of the trainning data
n = 1
for images, labels in train_loader:
    features = vgg16.features(images)
    features_train = torch.from_numpy(features.detach().numpy())

    folder_train = train_data_path + str(classes[labels])
    if not os.path.isdir(folder_train):
      os.mkdir(folder_train)
    torch.save(features_train.squeeze(0), folder_train + '/' + str(n) + '_aug' + '.tens
    n += 1


#save the features of the validation data
n = 1
for images, labels in val_loader:
    features = vgg16.features(images)
    features_val = torch.from_numpy(features.detach().numpy())

    folder_val = valid_data_path + str(classes[labels])
    if not os.path.isdir(folder_val):
      os.mkdir(folder_val)
    torch.save(features_val.squeeze(0), folder_val + '/' + str(n) + '_aug' + '.tensor')
    n += 1


#save the features of the test data
n = 1
for images, labels in test_loader:
    features = vgg16.features(images)
    features_test = torch.from_numpy(features.detach().numpy())

    folder_test = test_data_path + str(classes[labels])
    if not os.path.isdir(folder_test):
      os.mkdir(folder_test)
    torch.save(features_test.squeeze(0), folder_test + '/' + str(n) + '_aug' + '.tensor
    n += 1
```

In [9]:

```
#Artifical Neural Network Architecture
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.fc1 = nn.Linear(512*7*7, 256) # the ouput image size is 256*6*6, batch siz
        self.fc2 = nn.Linear(256, 128) # Hidden units = 128
        self.fc3 = nn.Linear(128, 32) # Hidden units = 32
        self.fc4 = nn.Linear(32, 4)

    def forward(self, x):
        x = x.view(-1, 512*7*7) #flatten feature data
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

In [10]:
```
def get_accuracy(model, train_loader,val_loader, train=False):
    if train:
        data_loader = train_loader
    else:
        data_loader = val_loader

    correct = 0
    total = 0
    for imgs, labels in data_loader:


        ###############################################
        #To Enable GPU Usage
        if use_cuda and torch.cuda.is_available():
          imgs = imgs.cuda()
          labels = labels.cuda()
        ###############################################

        output = model(imgs)

        #select index with maximum prediction score
        pred = output.max(1, keepdim=True)[1]
        correct += pred.eq(labels.view_as(pred)).sum().item()
        total += imgs.shape[0]
    return correct / total
```

In [11]:
```
def train(model, train_loader,val_loader, batch_size=32, num_epochs=1,lr = 0.001):
    #train_loader = torch.utils.data.DataLoader(data, batch_size=batch_size)
    # train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
                                            #num_workers=num_workers, shuffle=True)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)

    n_epochs, iters, losses, train_acc, val_acc = [], [], [], [],[]

    # training
    n = 0 # the number of iterations
    start_time=time.time()
    nepochs = 0

    for epoch in range(num_epochs):
        for features, labels in iter(train_loader):

            ###############################################
            #To Enable GPU Usage
            if use_cuda and torch.cuda.is_available():
              features = features.cuda()
              labels = labels.cuda()
            ###############################################

          #### ALNC is alexNet.features (AlexNet without classifier) ####

            out = model(features)             # forward pass
            loss = criterion(out, labels) # compute the total loss
            loss.backward()               # backward pass (compute parameter updates)
            optimizer.step()              # make the updates for each parameter
            optimizer.zero_grad()         # a clean up step for PyTorch
            n += 1
```

```
            # save the current training information
              iters.append(n)
              losses.append(float(loss)/batch_size)             # compute *average* loss
          nepochs+=1
          n_epochs.append(nepochs)
          val_acc.append(get_accuracy(model, train_loader,val_loader, train=False))  # co
          train_acc.append(get_accuracy(model, train_loader,val_loader, train=True))  # c
          model_path = "model_{0}_bs{1}_lr{2}_epoch{3}".format(model.name, batch_size, lr
          torch.save(model.state_dict(), model_path)


          print("Iteration: ",n,'Progress: % 6.2f ' % ((epoch * len(train_loader)) / (num


      print ("Epoch %d Finished. " % epoch ,"Time per Epoch: % 6.2f s "% ((time.time()-st


      end_time= time.time()
      # plotting
      plt.title("Training Curve")
      plt.plot(iters, losses, label="Train")
      plt.xlabel("Epochs")
      plt.ylabel("Loss")
      plt.show()

      plt.title("Training Curve")
      plt.plot(n_epochs, train_acc, label="Training")
      plt.plot(n_epochs, val_acc, label="Validation")
      plt.xlabel("Epochs")
      plt.ylabel("Validation Accuracy")
      plt.legend(loc='best')
      plt.show()

      print("Final Training Accuracy: {}".format(train_acc[-1]))
      print("Final Validation Accuracy: {}".format(val_acc[-1]))
      print ("Total time:  % 6.2f s  Time per Epoch: % 6.2f s " % ( (end_time-start_time)
```

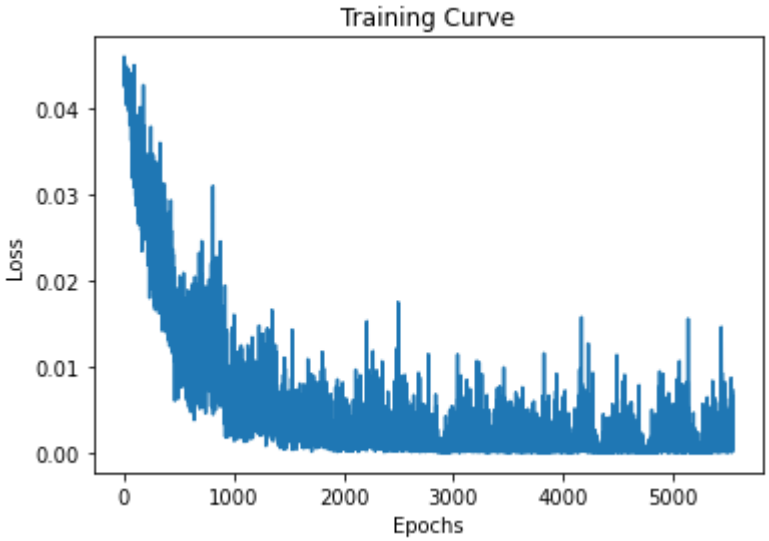
In [13]:
```
train_data_path = 'C:/Users/panji/OneDrive/Desktop/vgg16_feature_augmentation/train/'
val_data_path = 'C:/Users/panji/OneDrive/Desktop/vgg16_feature_augmentation/val/'
test_data_path = 'C:/Users/panji/OneDrive/Desktop/vgg16_feature_augmentation/test/'

# Load data from Google Drive using DatasetFolder
train_dataset_new = torchvision.datasets.DatasetFolder(train_data_path, loader=torch.lo
val_dataset_new = torchvision.datasets.DatasetFolder(val_data_path, loader=torch.load,
test_dataset_new = torchvision.datasets.DatasetFolder(test_data_path, loader=torch.load
```

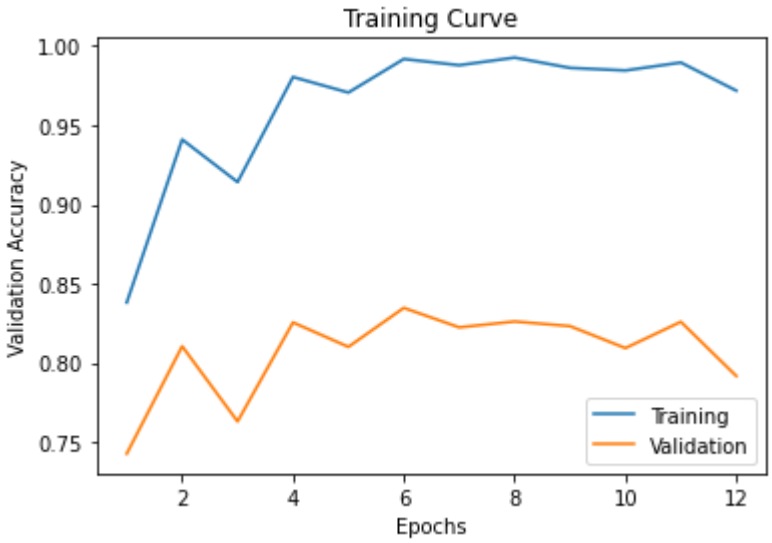
Trial 1 batch_size = 32, lr = 0.0005, #epochs = 12

In [14]:
```
# define dataloader parameters
batch_size  = 32 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
```

```
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=32, num_epochs=12,lr=0.0005)
print('vgg16 model:','batch_size=32, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  462 Progress:   0.00  % Time Elapsed:  134.95 s
Iteration:  924 Progress:   8.33  % Time Elapsed:  146.28 s
Iteration:  1386 Progress:  16.67  % Time Elapsed:  158.20 s
Iteration:  1848 Progress:  25.00  % Time Elapsed:  171.67 s
Iteration:  2310 Progress:  33.33  % Time Elapsed:  183.82 s
Iteration:  2772 Progress:  41.67  % Time Elapsed:  195.03 s
Iteration:  3234 Progress:  50.00  % Time Elapsed:  206.27 s
Iteration:  3696 Progress:  58.33  % Time Elapsed:  217.42 s
Iteration:  4158 Progress:  66.67  % Time Elapsed:  228.70 s
Iteration:  4620 Progress:  75.00  % Time Elapsed:  240.24 s
Iteration:  5082 Progress:  83.33  % Time Elapsed:  251.32 s
Iteration:  5544 Progress:  91.67  % Time Elapsed:  262.58 s
Epoch 11 Finished.  Time per Epoch:  21.88 s
```





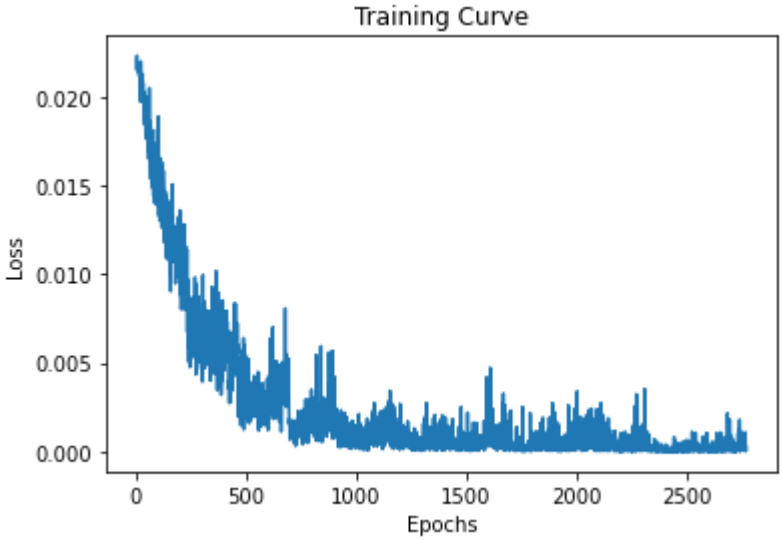

```
Final Training Accuracy: 0.9718614718614719
Final Validation Accuracy: 0.791796875
Total time:   262.58 s  Time per Epoch:  21.88 s
vgg16 model: batch_size=32, num_epochs=12,lr=0.0005
```

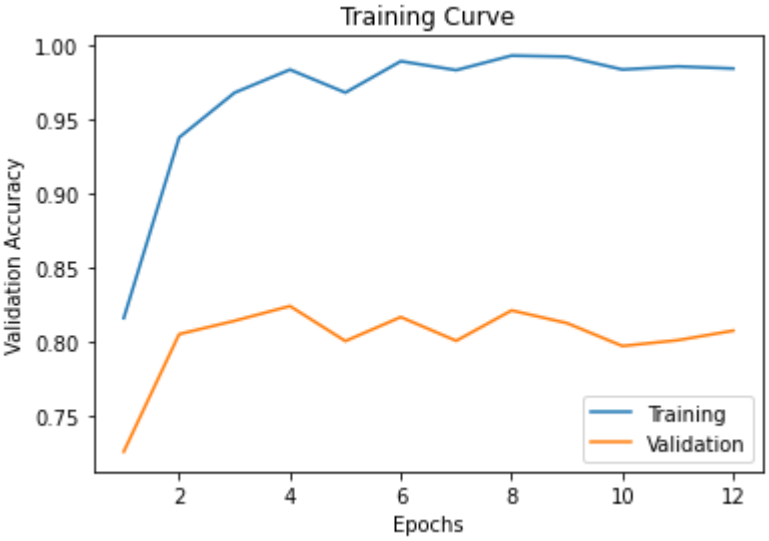
Trial 2 batch_size = 64, lr = 0.0005, #epochs = 12

In [15]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.13 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  20.26 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  30.37 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  40.39 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  50.45 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  60.71 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  70.76 s
```

```
Iteration:  1848 Progress:  58.33  % Time Elapsed:  80.83 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  90.92 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  101.03 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  111.08 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  121.11 s
Epoch 11 Finished.  Time per Epoch:  10.09 s
```





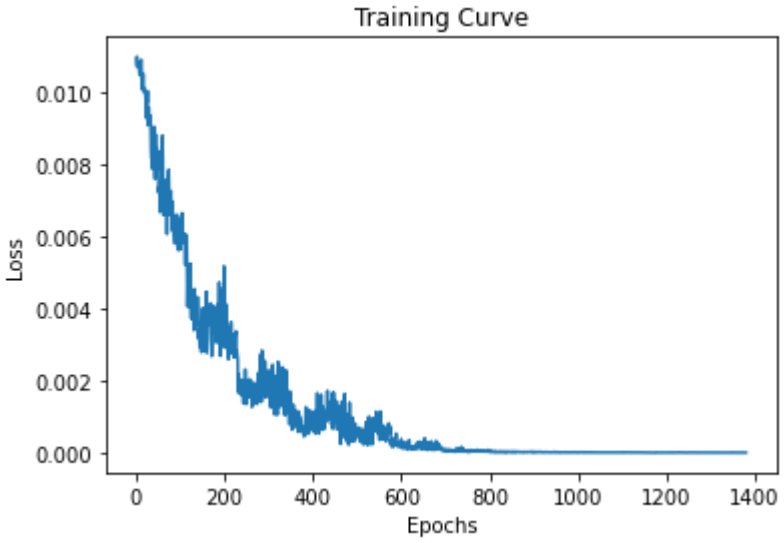

```
Final Training Accuracy: 0.9845102813852814
Final Validation Accuracy: 0.8076171875
Total time:   121.11 s  Time per Epoch:  10.09 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

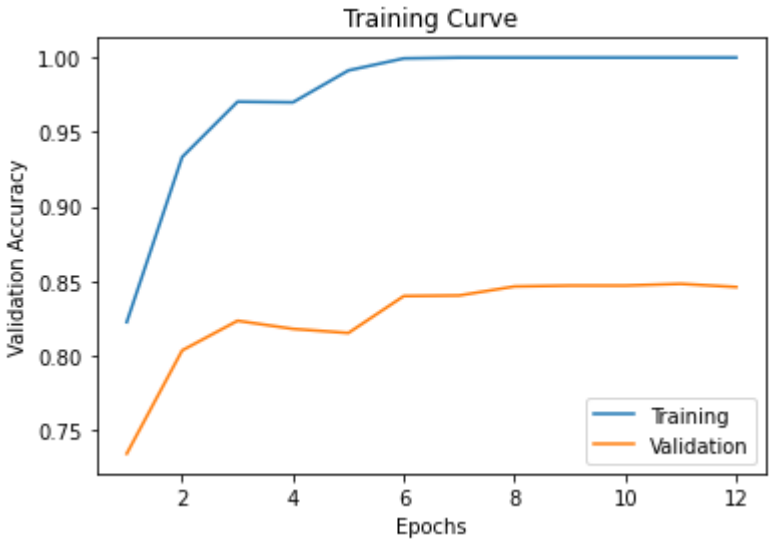
Trial 3 batch_size = 128, lr = 0.0005, #epochs = 12

In [16]:
```
# define dataloader parameters
batch_size  = 128 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
```

```
                                                num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  115 Progress:   0.00  % Time Elapsed:   9.64 s
Iteration:  230 Progress:   8.33  % Time Elapsed:  19.14 s
Iteration:  345 Progress:  16.67  % Time Elapsed:  28.66 s
Iteration:  460 Progress:  25.00  % Time Elapsed:  38.29 s
Iteration:  575 Progress:  33.33  % Time Elapsed:  47.84 s
Iteration:  690 Progress:  41.67  % Time Elapsed:  57.35 s
Iteration:  805 Progress:  50.00  % Time Elapsed:  66.89 s
Iteration:  920 Progress:  58.33  % Time Elapsed:  76.43 s
Iteration:  1035 Progress:  66.67  % Time Elapsed:  85.97 s
Iteration:  1150 Progress:  75.00  % Time Elapsed:  95.54 s
Iteration:  1265 Progress:  83.33  % Time Elapsed:  105.10 s
Iteration:  1380 Progress:  91.67  % Time Elapsed:  114.69 s
Epoch 11 Finished.  Time per Epoch:   9.56 s
```






```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.8458984375
Total time:   114.69 s  Time per Epoch:   9.56 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

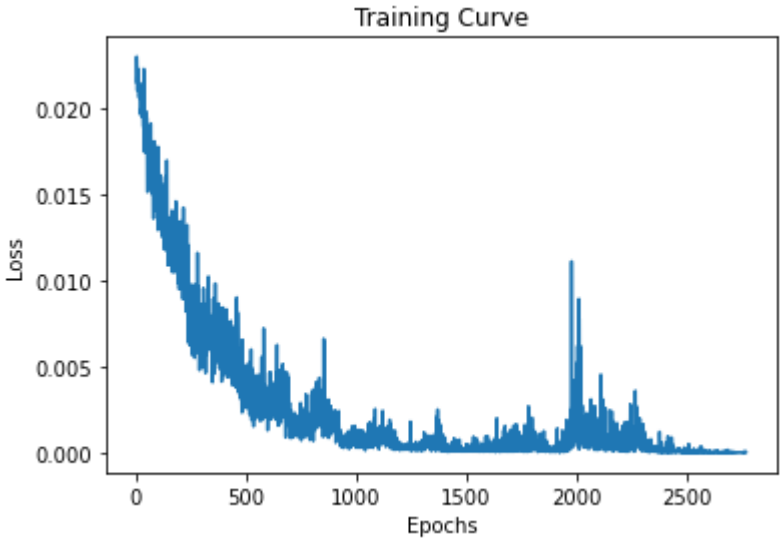
Decrease Hidden Layer

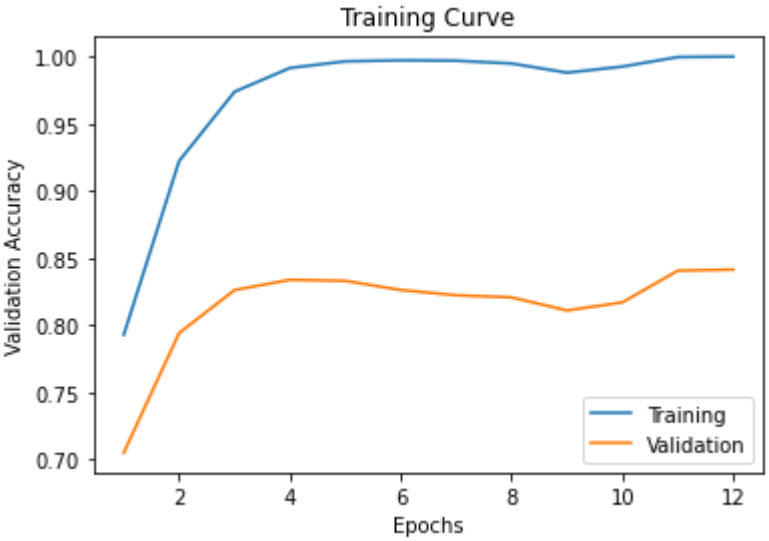
In [17]:
```
#Artifical Neural Network Architecture
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.fc1 = nn.Linear(512*7*7, 256) # the ouput image size is 256*6*6, batch siz
        self.fc2 = nn.Linear(256, 64) # Hidden units = 128
        self.fc3 = nn.Linear(64, 4)

    def forward(self, x):
        x = x.view(-1, 512*7*7) #flatten feature data
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [18]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
```

```
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.17 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  21.91 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  30.76 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  40.96 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  51.60 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  62.15 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  73.07 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  83.39 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  94.31 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  105.01 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  116.09 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  126.93 s
Epoch 11 Finished.  Time per Epoch:  10.58 s
```





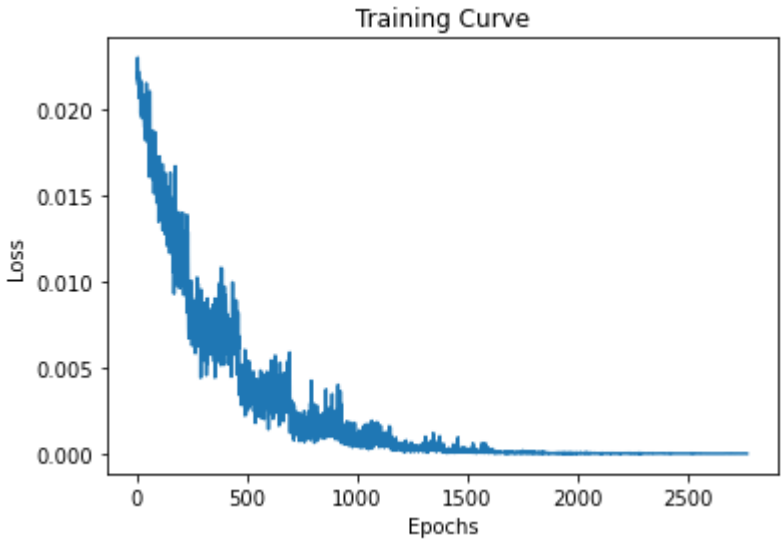

```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.84140625
Total time:   126.93 s  Time per Epoch:  10.58 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

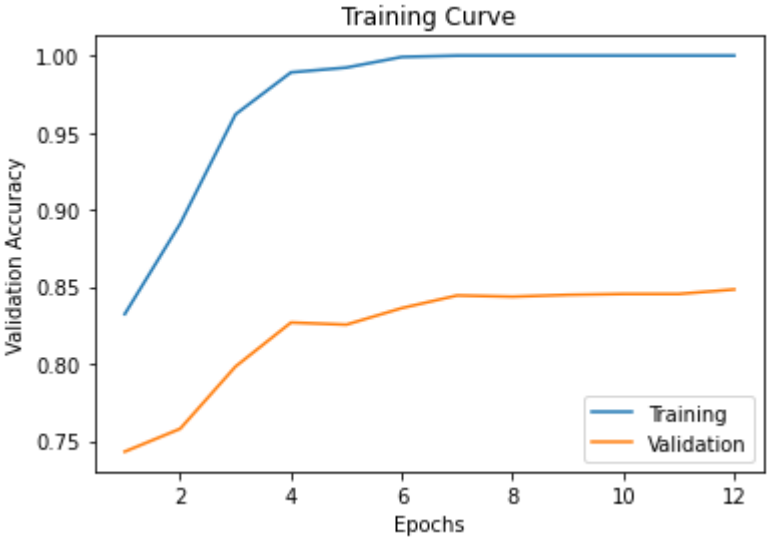
Also decrease numebr of hidden units

In [19]:
```
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.fc1 = nn.Linear(512*7*7, 128) # the ouput image size is 256*6*6, batch siz
        self.fc2 = nn.Linear(128, 32) # Hidden units = 128
        self.fc3 = nn.Linear(32, 4)

    def forward(self, x):
        x = x.view(-1, 512*7*7) #flatten feature data
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

In [20]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.60 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  21.14 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  31.54 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  41.94 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  52.50 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  62.82 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  73.24 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  83.77 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  94.22 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  104.68 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  114.96 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  125.27 s
Epoch 11 Finished.  Time per Epoch:  10.44 s
```






```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.8482421875
Total time:   125.27 s  Time per Epoch:  10.44 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

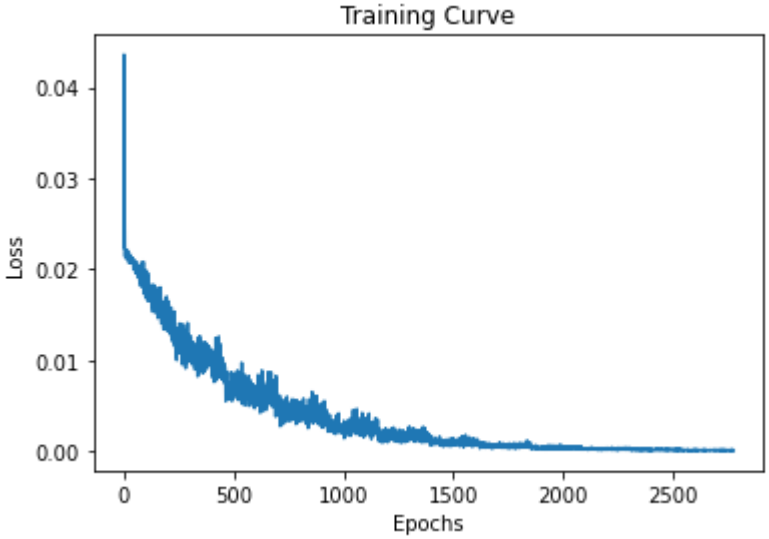
Decrease Hidden Layer

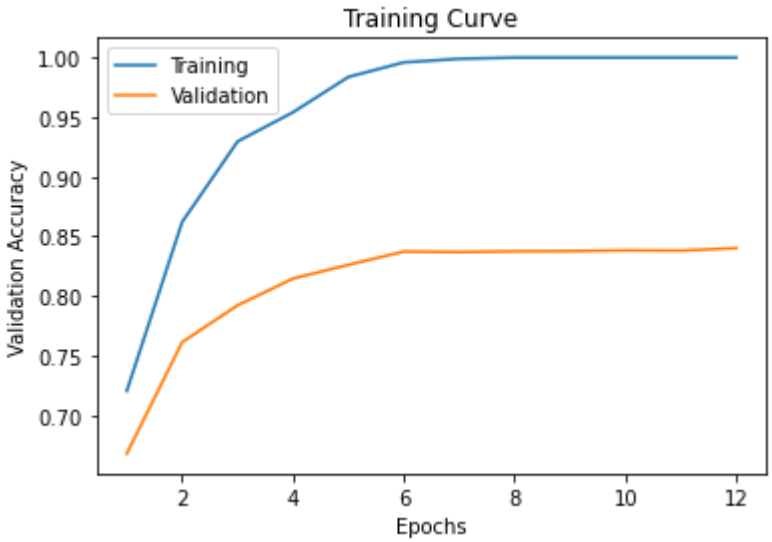
In [21]:
```
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.fc1 = nn.Linear(512*7*7, 128) # the ouput image size is 256*6*6, batch siz
        self.fc2 = nn.Linear(128, 4)

    def forward(self, x):
        x = x.view(-1, 512*7*7) #flatten feature data
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

In [22]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here
```

```
# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.30 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  20.56 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  30.78 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  41.00 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  51.25 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  61.71 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  71.97 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  82.40 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  92.96 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  103.33 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  113.83 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  124.24 s
Epoch 11 Finished.  Time per Epoch:  10.35 s
```






```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.840234375
Total time:   124.24 s  Time per Epoch:  10.35 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

Added Weight Decay

In [23]:
```
def train(model, train_loader,val_loader, batch_size=32, num_epochs=1,lr = 0.001,weight
    #train_loader = torch.utils.data.DataLoader(data, batch_size=batch_size)
    # train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
                                          #num_workers=num_workers, shuffle=True)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=lr,weight_decay=weight_decay)

    n_epochs, iters, losses, train_acc, val_acc = [], [], [], [],[]

    # training
    n = 0 # the number of iterations
    start_time=time.time()
    nepochs = 0

    for epoch in range(num_epochs):
        for features, labels in iter(train_loader):

            ##############################################
            #To Enable GPU Usage
            if use_cuda and torch.cuda.is_available():
              features = features.cuda()
              labels = labels.cuda()
            ##############################################

          #### ALNC is alexNet.features (AlexNet without classifier) ####

            out = model(features)             # forward pass
            loss = criterion(out, labels) # compute the total loss
            loss.backward()               # backward pass (compute parameter updates)
            optimizer.step()              # make the updates for each parameter
            optimizer.zero_grad()         # a clean up step for PyTorch
            n += 1


          # save the current training information
```

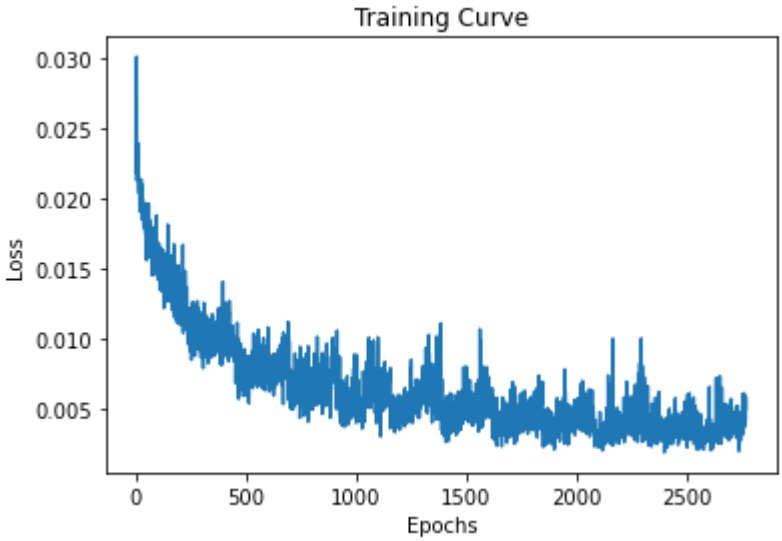
```
            iters.append(n)
            losses.append(float(loss)/batch_size)             # compute *average* loss
        nepochs+=1
        n_epochs.append(nepochs)
        val_acc.append(get_accuracy(model, train_loader,val_loader, train=False))  # co
        train_acc.append(get_accuracy(model, train_loader,val_loader, train=True))  # c
        model_path = "model_{0}_bs{1}_lr{2}_epoch{3}".format(model.name, batch_size, lr
        torch.save(model.state_dict(), model_path)


        print("Iteration: ",n,'Progress: % 6.2f ' % ((epoch * len(train_loader)) / (num


    print ("Epoch %d Finished. " % epoch ,"Time per Epoch: % 6.2f s "% ((time.time()-st


    end_time= time.time()
    # plotting
    plt.title("Training Curve")
    plt.plot(iters, losses, label="Train")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.show()

    plt.title("Training Curve")
    plt.plot(n_epochs, train_acc, label="Training")
    plt.plot(n_epochs, val_acc, label="Validation")
    plt.xlabel("Epochs")
    plt.ylabel("Validation Accuracy")
    plt.legend(loc='best')
    plt.show()

    print("Final Training Accuracy: {}".format(train_acc[-1]))
    print("Final Validation Accuracy: {}".format(val_acc[-1]))
    print ("Total time:  % 6.2f s  Time per Epoch: % 6.2f s " % ( (end_time-start_time)
```

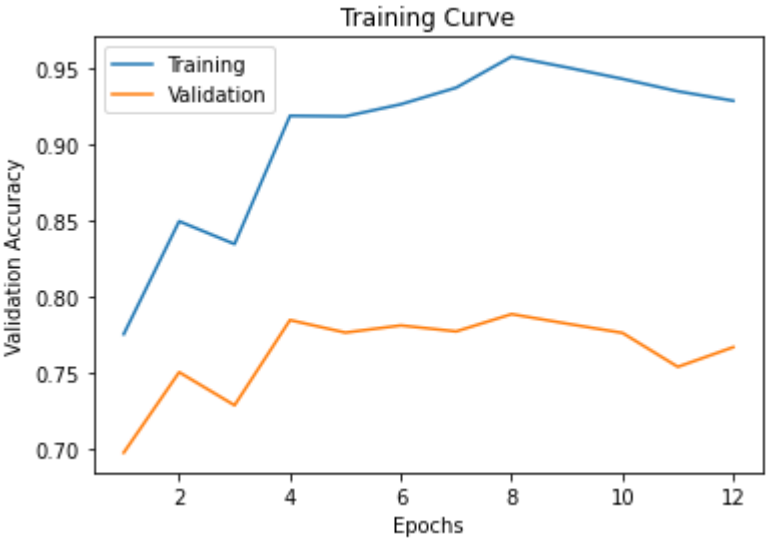
In [24]:

```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
```

```
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.54 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  20.54 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  30.62 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  40.43 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  50.80 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  61.51 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  72.18 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  82.87 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  93.47 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  104.08 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  114.46 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  124.73 s
Epoch 11 Finished.  Time per Epoch:  10.39 s
```






```
Final Training Accuracy: 0.9283685064935064
Final Validation Accuracy: 0.766796875
Total time:   124.73 s  Time per Epoch:  10.39 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

Added Dropout

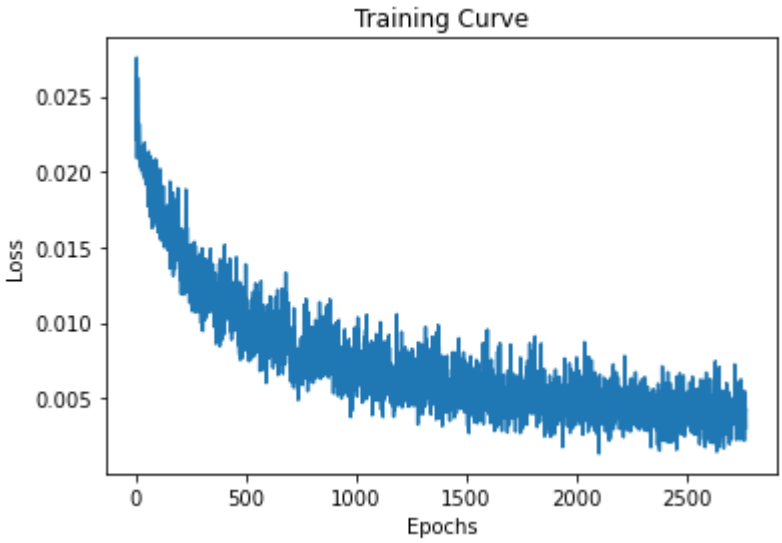
In [25]:
```
class ANNClassifier(nn.Module):
    def __init__(self):
```

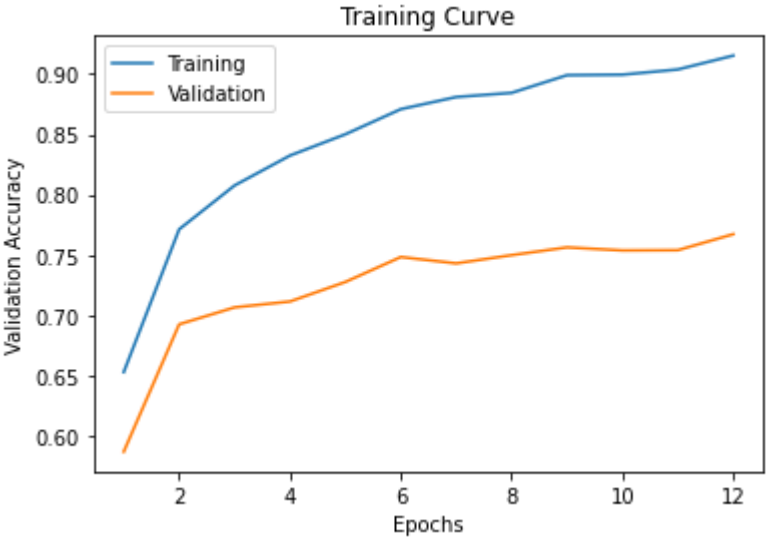
```
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.fc1 = nn.Linear(512*7*7, 128) # the ouput image size is 256*6*6, batch siz
        self.fc2 = nn.Linear(128, 4)
        self.dropout1 = nn.Dropout(0.4)
        self.dropout2 = nn.Dropout(0.4)

    def forward(self, x):
        x = x.view(-1, 512*7*7) #flatten feature data
        x = F.relu(self.fc1(self.dropout1(x)))
        x = self.fc2(self.dropout2(x))
        return x
```

In [26]:

```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.25 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  20.43 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  30.70 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  40.96 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  51.22 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  61.51 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  71.82 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  82.20 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  92.59 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  102.92 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  113.38 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  123.69 s
Epoch 11 Finished.  Time per Epoch:  10.31 s
```






```
Final Training Accuracy: 0.9153814935064936
Final Validation Accuracy: 0.7671875
Total time:   123.69 s  Time per Epoch:  10.31 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

batch_size = 64, lr = 0.0005, #epochs = 50, singlelayer

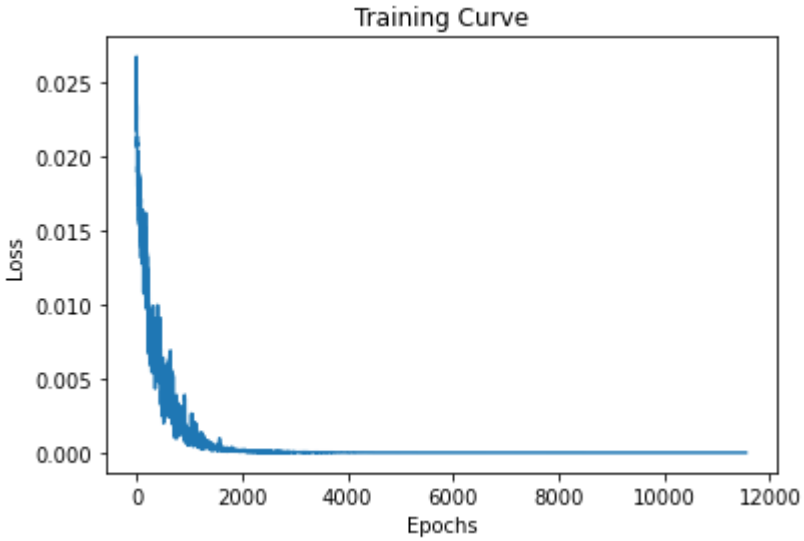
In [27]:
```
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.fc1 = nn.Linear(512*7*7, 128) # the ouput image size is 256*6*6, batch siz
        self.fc2 = nn.Linear(128, 4)

    def forward(self, x):
        x = x.view(-1, 512*7*7) #flatten feature data
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

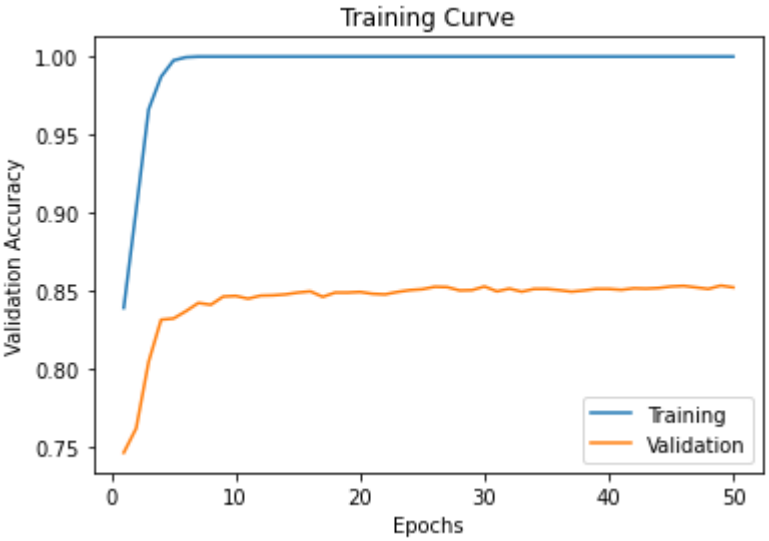
In [28]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
```

```
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=50,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.29 s
Iteration:  462 Progress:   2.00  % Time Elapsed:  20.58 s
Iteration:  693 Progress:   4.00  % Time Elapsed:  30.81 s
Iteration:  924 Progress:   6.00  % Time Elapsed:  41.02 s
Iteration:  1155 Progress:   8.00  % Time Elapsed:  51.23 s
Iteration:  1386 Progress:  10.00  % Time Elapsed:  61.41 s
Iteration:  1617 Progress:  12.00  % Time Elapsed:  71.66 s
Iteration:  1848 Progress:  14.00  % Time Elapsed:  81.86 s
Iteration:  2079 Progress:  16.00  % Time Elapsed:  92.34 s
Iteration:  2310 Progress:  18.00  % Time Elapsed:  103.20 s
Iteration:  2541 Progress:  20.00  % Time Elapsed:  114.05 s
Iteration:  2772 Progress:  22.00  % Time Elapsed:  125.08 s
Iteration:  3003 Progress:  24.00  % Time Elapsed:  135.88 s
Iteration:  3234 Progress:  26.00  % Time Elapsed:  146.80 s
Iteration:  3465 Progress:  28.00  % Time Elapsed:  157.67 s
Iteration:  3696 Progress:  30.00  % Time Elapsed:  168.60 s
Iteration:  3927 Progress:  32.00  % Time Elapsed:  179.42 s
Iteration:  4158 Progress:  34.00  % Time Elapsed:  190.32 s
Iteration:  4389 Progress:  36.00  % Time Elapsed:  201.26 s
Iteration:  4620 Progress:  38.00  % Time Elapsed:  212.33 s
Iteration:  4851 Progress:  40.00  % Time Elapsed:  223.26 s
Iteration:  5082 Progress:  42.00  % Time Elapsed:  234.11 s
Iteration:  5313 Progress:  44.00  % Time Elapsed:  244.90 s
Iteration:  5544 Progress:  46.00  % Time Elapsed:  255.68 s
Iteration:  5775 Progress:  48.00  % Time Elapsed:  266.55 s
Iteration:  6006 Progress:  50.00  % Time Elapsed:  277.46 s
Iteration:  6237 Progress:  52.00  % Time Elapsed:  288.40 s
Iteration:  6468 Progress:  54.00  % Time Elapsed:  299.28 s
Iteration:  6699 Progress:  56.00  % Time Elapsed:  310.22 s
Iteration:  6930 Progress:  58.00  % Time Elapsed:  321.03 s
Iteration:  7161 Progress:  60.00  % Time Elapsed:  331.96 s
Iteration:  7392 Progress:  62.00  % Time Elapsed:  342.68 s
Iteration:  7623 Progress:  64.00  % Time Elapsed:  353.52 s
Iteration:  7854 Progress:  66.00  % Time Elapsed:  364.47 s
Iteration:  8085 Progress:  68.00  % Time Elapsed:  375.39 s
Iteration:  8316 Progress:  70.00  % Time Elapsed:  386.23 s
Iteration:  8547 Progress:  72.00  % Time Elapsed:  397.03 s
Iteration:  8778 Progress:  74.00  % Time Elapsed:  407.92 s
Iteration:  9009 Progress:  76.00  % Time Elapsed:  418.83 s
```

```
Iteration:  9240 Progress:  78.00  % Time Elapsed:  429.70 s
Iteration:  9471 Progress:  80.00  % Time Elapsed:  440.52 s
Iteration:  9702 Progress:  82.00  % Time Elapsed:  451.71 s
Iteration:  9933 Progress:  84.00  % Time Elapsed:  462.17 s
Iteration:  10164 Progress:  86.00  % Time Elapsed:  472.55 s
Iteration:  10395 Progress:  88.00  % Time Elapsed:  482.87 s
Iteration:  10626 Progress:  90.00  % Time Elapsed:  493.22 s
Iteration:  10857 Progress:  92.00  % Time Elapsed:  503.66 s
Iteration:  11088 Progress:  94.00  % Time Elapsed:  514.06 s
Iteration:  11319 Progress:  96.00  % Time Elapsed:  524.59 s
Iteration:  11550 Progress:  98.00  % Time Elapsed:  535.18 s
Epoch 49 Finished.  Time per Epoch:  10.70 s
```





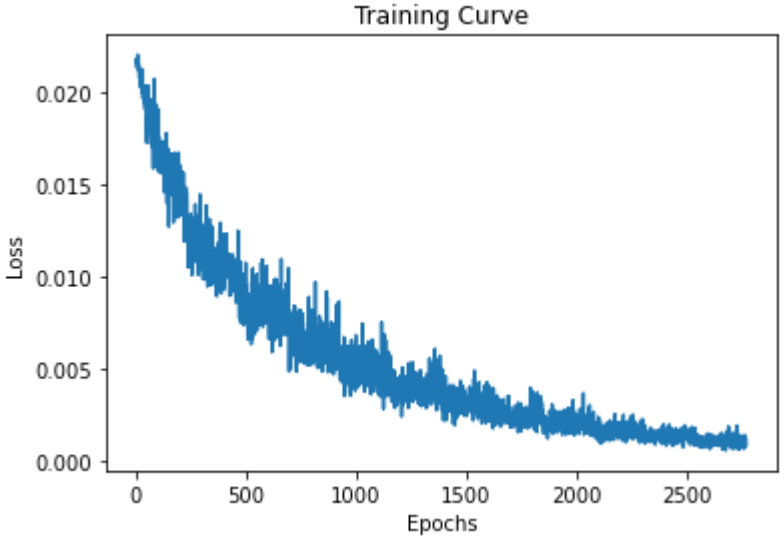

```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.8521484375
Total time:   535.18 s  Time per Epoch:  10.70 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

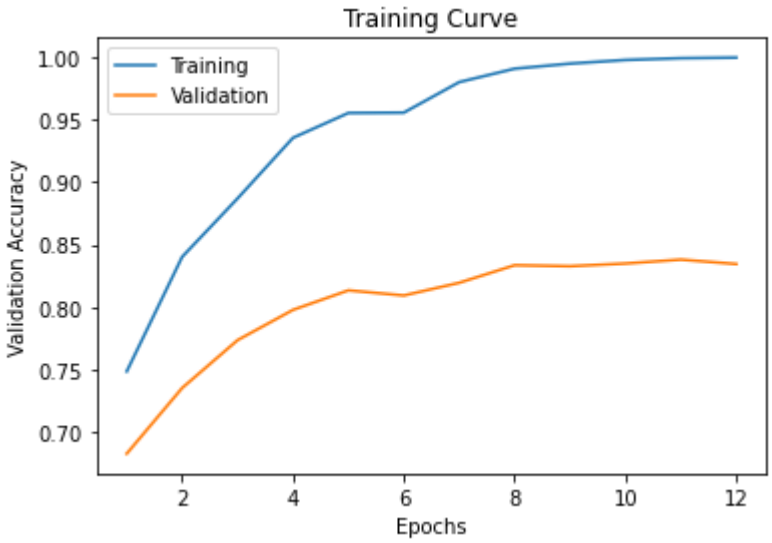
batch_size = 64, lr = 0.00005, #epochs = 12, singlelayer

In [30]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here
```

```
# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:   9.79 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  20.47 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  31.09 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  41.69 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  52.30 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  63.30 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  74.26 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  85.17 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  96.13 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  107.14 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  118.17 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  129.04 s
Epoch 11 Finished.  Time per Epoch:  10.75 s
```





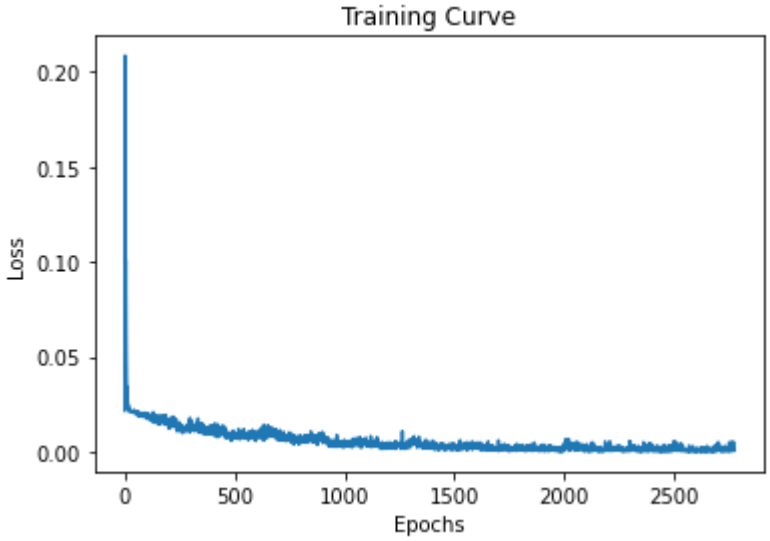

```
Final Training Accuracy: 0.9995265151515151
Final Validation Accuracy: 0.834375
Total time:   129.04 s  Time per Epoch:  10.75 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

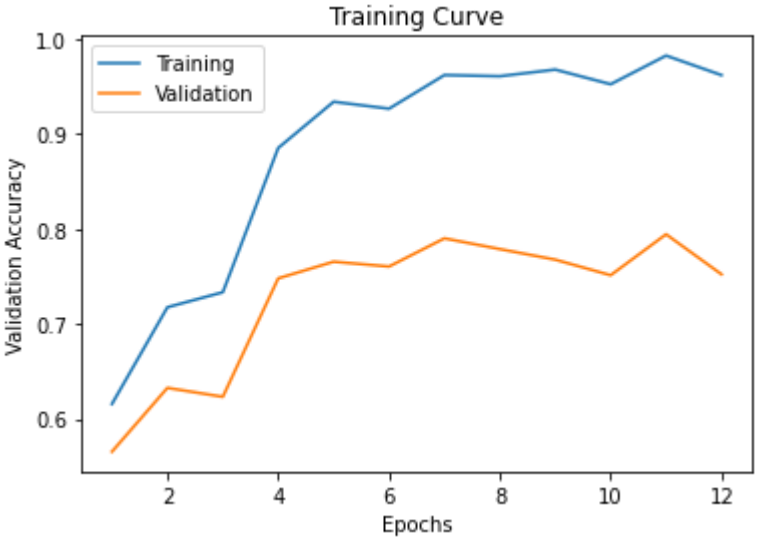
batch_size = 64, lr = 0.005, #epochs = 12, singlelayer

In [31]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.32 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  20.80 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  31.92 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  42.87 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  53.85 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  64.82 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  75.81 s
```

```
Iteration:  1848 Progress:  58.33  % Time Elapsed:  86.82 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  97.69 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  108.54 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  119.48 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  130.36 s
Epoch 11 Finished.  Time per Epoch:  10.86 s
```






```
Final Training Accuracy: 0.9622564935064936
Final Validation Accuracy: 0.75234375
Total time:   130.36 s  Time per Epoch:  10.86 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

Decrease hidden unit add one cnn

In [32]:
```
train_dataset_new[1][0].size()
```

Out[32]:
```
torch.Size([512, 7, 7])
```

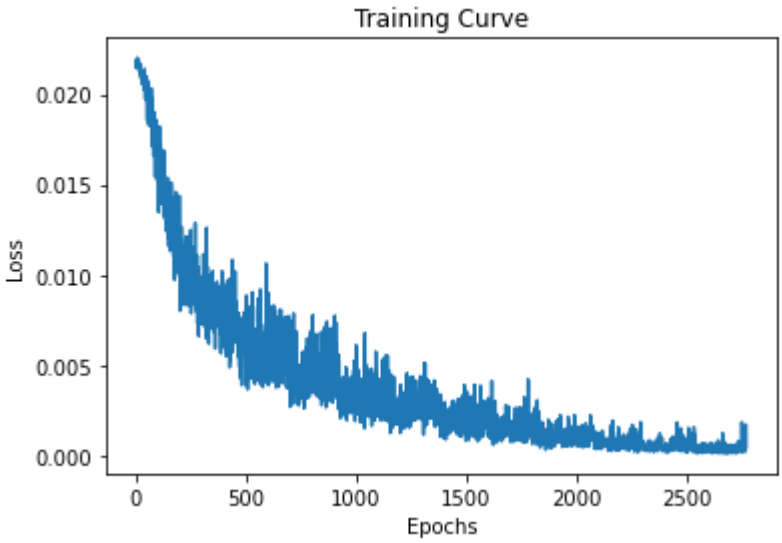
In [33]:
```
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.conv1 = nn.Conv2d(512,50,1)
        self.pool = nn.MaxPool2d(3,2)
        #conv1: 7-1+1 = 7
```

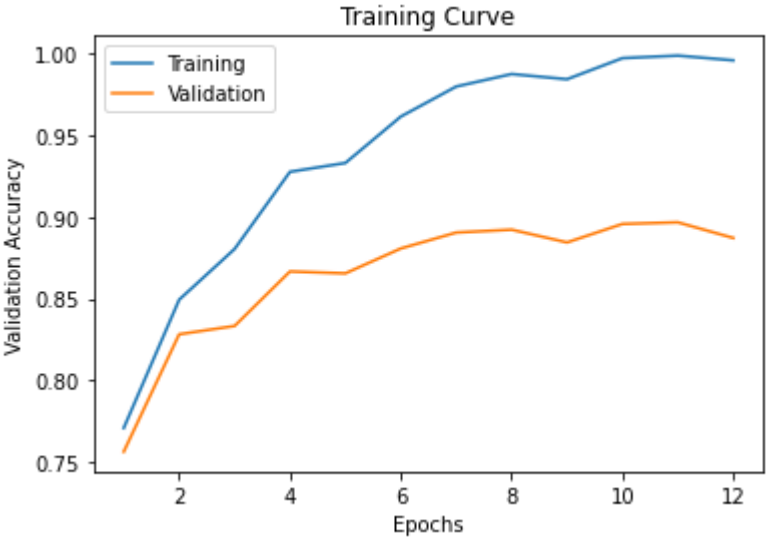
```
        #pool: (7-3)/2 + 1 = 3
        self.fc1 = nn.Linear(50*3*3, 128) # the ouput image size is 256*6*6, batch size
        self.fc2 = nn.Linear(128, 4)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = x.view(-1, 50*3*3) #flatten feature data
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

In [35]:

```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=12,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.24 s
Iteration:  462 Progress:   8.33  % Time Elapsed:  21.04 s
Iteration:  693 Progress:  16.67  % Time Elapsed:  31.79 s
Iteration:  924 Progress:  25.00  % Time Elapsed:  42.55 s
Iteration:  1155 Progress:  33.33  % Time Elapsed:  53.33 s
Iteration:  1386 Progress:  41.67  % Time Elapsed:  64.12 s
Iteration:  1617 Progress:  50.00  % Time Elapsed:  74.96 s
Iteration:  1848 Progress:  58.33  % Time Elapsed:  85.77 s
Iteration:  2079 Progress:  66.67  % Time Elapsed:  96.57 s
Iteration:  2310 Progress:  75.00  % Time Elapsed:  107.42 s
Iteration:  2541 Progress:  83.33  % Time Elapsed:  118.41 s
Iteration:  2772 Progress:  91.67  % Time Elapsed:  129.27 s
Epoch 11 Finished.  Time per Epoch:  10.77 s
```





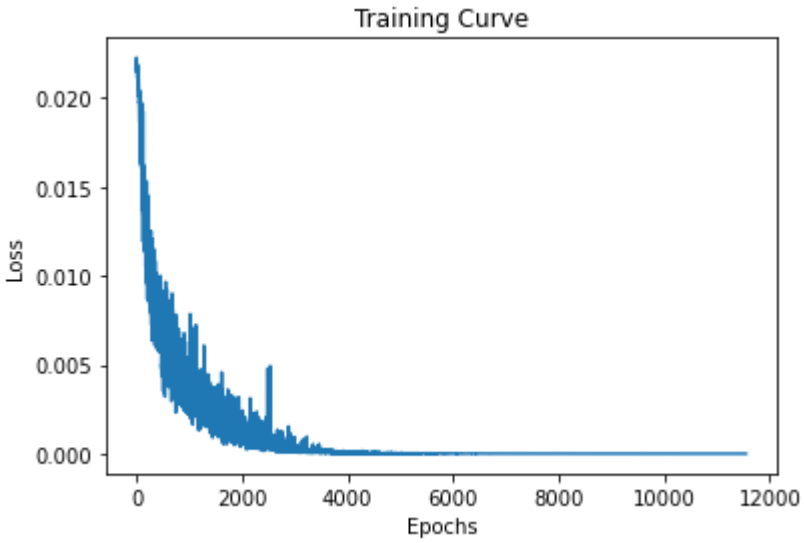

```
Final Training Accuracy: 0.9958739177489178
Final Validation Accuracy: 0.887109375
Total time:   129.28 s  Time per Epoch:  10.77 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

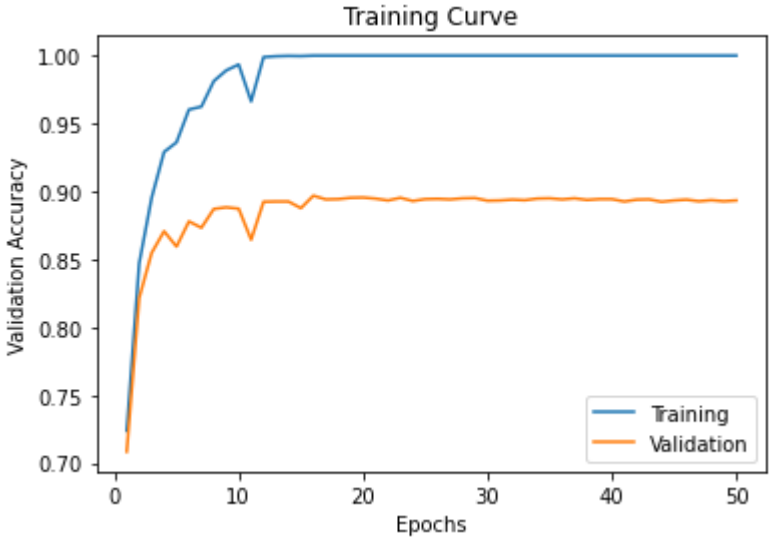
batch_size = 64, lr = 0.0005, #epochs = 50, singlelayer+cnn

In [36]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()
```

```
if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=50,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.93 s
Iteration:  462 Progress:   2.00  % Time Elapsed:  21.97 s
Iteration:  693 Progress:   4.00  % Time Elapsed:  32.82 s
Iteration:  924 Progress:   6.00  % Time Elapsed:  43.72 s
Iteration:  1155 Progress:   8.00  % Time Elapsed:  54.29 s
Iteration:  1386 Progress:  10.00  % Time Elapsed:  65.14 s
Iteration:  1617 Progress:  12.00  % Time Elapsed:  75.82 s
Iteration:  1848 Progress:  14.00  % Time Elapsed:  85.82 s
Iteration:  2079 Progress:  16.00  % Time Elapsed:  96.60 s
Iteration:  2310 Progress:  18.00  % Time Elapsed:  106.56 s
Iteration:  2541 Progress:  20.00  % Time Elapsed:  116.69 s
Iteration:  2772 Progress:  22.00  % Time Elapsed:  126.84 s
Iteration:  3003 Progress:  24.00  % Time Elapsed:  137.65 s
Iteration:  3234 Progress:  26.00  % Time Elapsed:  148.43 s
Iteration:  3465 Progress:  28.00  % Time Elapsed:  159.30 s
Iteration:  3696 Progress:  30.00  % Time Elapsed:  170.13 s
Iteration:  3927 Progress:  32.00  % Time Elapsed:  180.95 s
Iteration:  4158 Progress:  34.00  % Time Elapsed:  191.77 s
Iteration:  4389 Progress:  36.00  % Time Elapsed:  202.51 s
Iteration:  4620 Progress:  38.00  % Time Elapsed:  213.25 s
Iteration:  4851 Progress:  40.00  % Time Elapsed:  224.03 s
Iteration:  5082 Progress:  42.00  % Time Elapsed:  234.87 s
Iteration:  5313 Progress:  44.00  % Time Elapsed:  245.68 s
Iteration:  5544 Progress:  46.00  % Time Elapsed:  256.50 s
Iteration:  5775 Progress:  48.00  % Time Elapsed:  267.31 s
Iteration:  6006 Progress:  50.00  % Time Elapsed:  278.12 s
Iteration:  6237 Progress:  52.00  % Time Elapsed:  288.95 s
Iteration:  6468 Progress:  54.00  % Time Elapsed:  299.59 s
Iteration:  6699 Progress:  56.00  % Time Elapsed:  310.22 s
Iteration:  6930 Progress:  58.00  % Time Elapsed:  320.71 s
Iteration:  7161 Progress:  60.00  % Time Elapsed:  331.44 s
Iteration:  7392 Progress:  62.00  % Time Elapsed:  342.16 s
Iteration:  7623 Progress:  64.00  % Time Elapsed:  352.99 s
Iteration:  7854 Progress:  66.00  % Time Elapsed:  363.82 s
Iteration:  8085 Progress:  68.00  % Time Elapsed:  374.60 s
Iteration:  8316 Progress:  70.00  % Time Elapsed:  385.42 s
Iteration:  8547 Progress:  72.00  % Time Elapsed:  396.32 s
Iteration:  8778 Progress:  74.00  % Time Elapsed:  407.38 s
Iteration:  9009 Progress:  76.00  % Time Elapsed:  418.44 s
Iteration:  9240 Progress:  78.00  % Time Elapsed:  428.83 s
Iteration:  9471 Progress:  80.00  % Time Elapsed:  439.45 s
Iteration:  9702 Progress:  82.00  % Time Elapsed:  450.27 s
Iteration:  9933 Progress:  84.00  % Time Elapsed:  461.08 s
Iteration:  10164 Progress:  86.00  % Time Elapsed:  471.78 s
Iteration:  10395 Progress:  88.00  % Time Elapsed:  482.56 s
Iteration:  10626 Progress:  90.00  % Time Elapsed:  493.22 s
Iteration:  10857 Progress:  92.00  % Time Elapsed:  503.84 s
Iteration:  11088 Progress:  94.00  % Time Elapsed:  514.55 s
Iteration:  11319 Progress:  96.00  % Time Elapsed:  525.22 s
Iteration:  11550 Progress:  98.00  % Time Elapsed:  535.98 s
Epoch 49 Finished.  Time per Epoch:  10.72 s
```






```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.893359375
Total time:   535.98 s  Time per Epoch:  10.72 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

Decrease hidden unit add two cnns

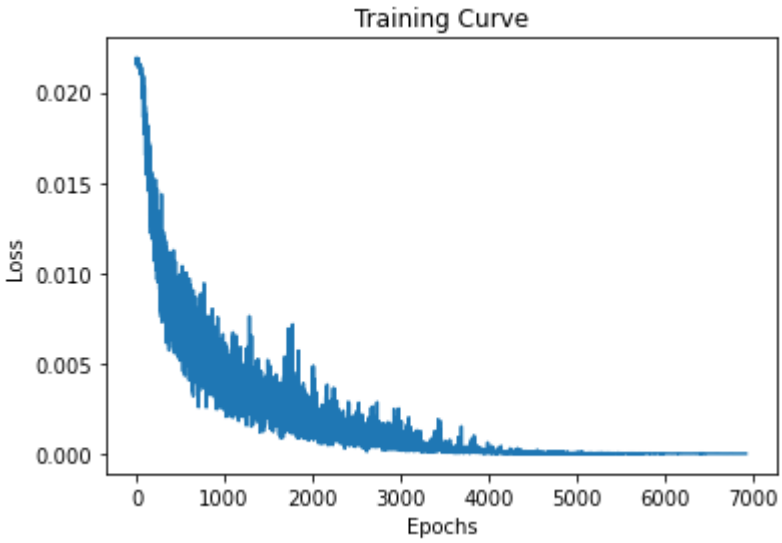
In [37]:
```
class ANNClassifier(nn.Module):
    def __init__(self):
        super(ANNClassifier, self).__init__()  # Fully connector layers with 3 hidden l
        self.name = 'ANNClassifier'
        self.conv1 = nn.Conv2d(512,50,1)
        self.conv2 = nn.Conv2d(50,25,1)
        self.pool = nn.MaxPool2d(3,2)
        #conv1: 7-1+1 = 7
        #pool: (7-3)/2 + 1 = 3
        #conv2: 3-1+1 = 3
        self.fc1 = nn.Linear(25*3*3, 64) # the ouput image size is 256*6*6, batch size
        self.fc2 = nn.Linear(64, 4)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = F.relu(self.conv2(x))
        x = x.view(-1, 25*3*3) #flatten feature data
```

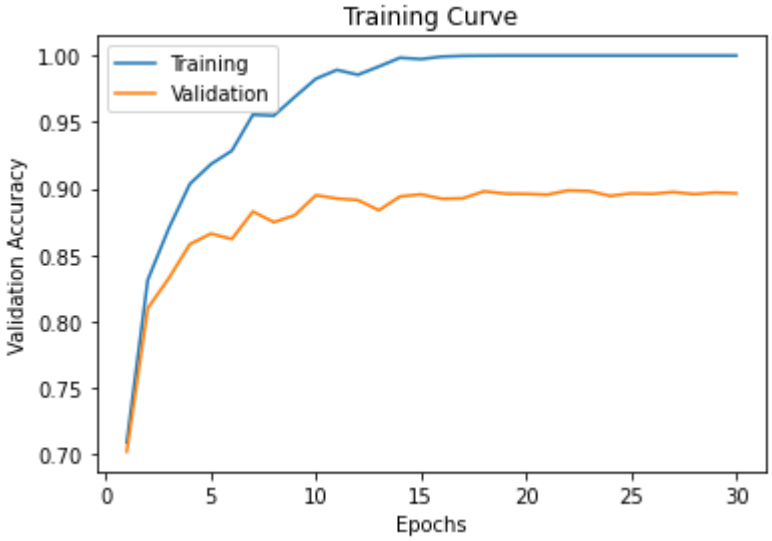
```
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

In [38]:

```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()

if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=30,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.90 s
Iteration:  462 Progress:   3.33  % Time Elapsed:  21.77 s
Iteration:  693 Progress:   6.67  % Time Elapsed:  32.69 s
Iteration:  924 Progress:  10.00  % Time Elapsed:  43.67 s
Iteration:  1155 Progress:  13.33  % Time Elapsed:  54.54 s
Iteration:  1386 Progress:  16.67  % Time Elapsed:  65.47 s
Iteration:  1617 Progress:  20.00  % Time Elapsed:  76.37 s
Iteration:  1848 Progress:  23.33  % Time Elapsed:  87.27 s
Iteration:  2079 Progress:  26.67  % Time Elapsed:  98.13 s
Iteration:  2310 Progress:  30.00  % Time Elapsed:  108.93 s
Iteration:  2541 Progress:  33.33  % Time Elapsed:  119.78 s
Iteration:  2772 Progress:  36.67  % Time Elapsed:  130.66 s
Iteration:  3003 Progress:  40.00  % Time Elapsed:  141.54 s
Iteration:  3234 Progress:  43.33  % Time Elapsed:  152.45 s
Iteration:  3465 Progress:  46.67  % Time Elapsed:  163.22 s
Iteration:  3696 Progress:  50.00  % Time Elapsed:  173.92 s
Iteration:  3927 Progress:  53.33  % Time Elapsed:  184.74 s
Iteration:  4158 Progress:  56.67  % Time Elapsed:  195.56 s
Iteration:  4389 Progress:  60.00  % Time Elapsed:  206.39 s
Iteration:  4620 Progress:  63.33  % Time Elapsed:  217.29 s
Iteration:  4851 Progress:  66.67  % Time Elapsed:  228.21 s
Iteration:  5082 Progress:  70.00  % Time Elapsed:  238.92 s
Iteration:  5313 Progress:  73.33  % Time Elapsed:  249.68 s
Iteration:  5544 Progress:  76.67  % Time Elapsed:  260.55 s
Iteration:  5775 Progress:  80.00  % Time Elapsed:  271.37 s
Iteration:  6006 Progress:  83.33  % Time Elapsed:  282.27 s
Iteration:  6237 Progress:  86.67  % Time Elapsed:  293.12 s
Iteration:  6468 Progress:  90.00  % Time Elapsed:  303.99 s
Iteration:  6699 Progress:  93.33  % Time Elapsed:  314.90 s
Iteration:  6930 Progress:  96.67  % Time Elapsed:  325.78 s
Epoch 29 Finished.  Time per Epoch:  10.86 s
```





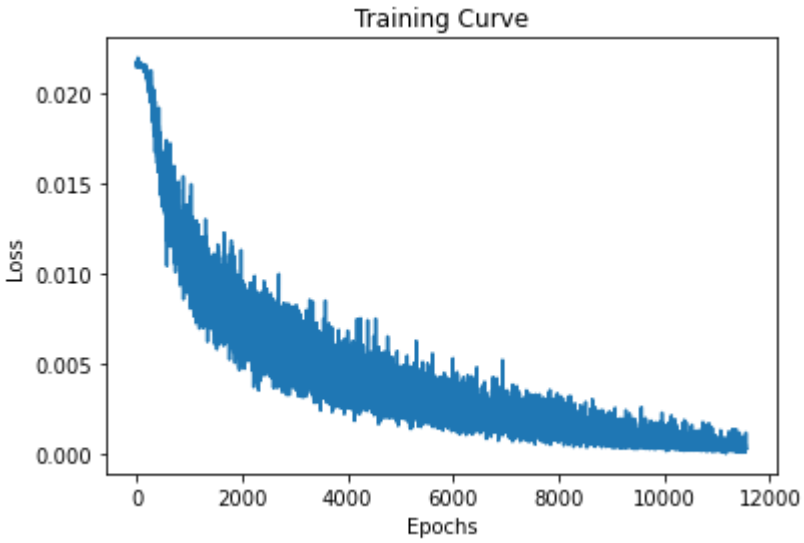

```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.896484375
Total time:   325.78 s  Time per Epoch:  10.86 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

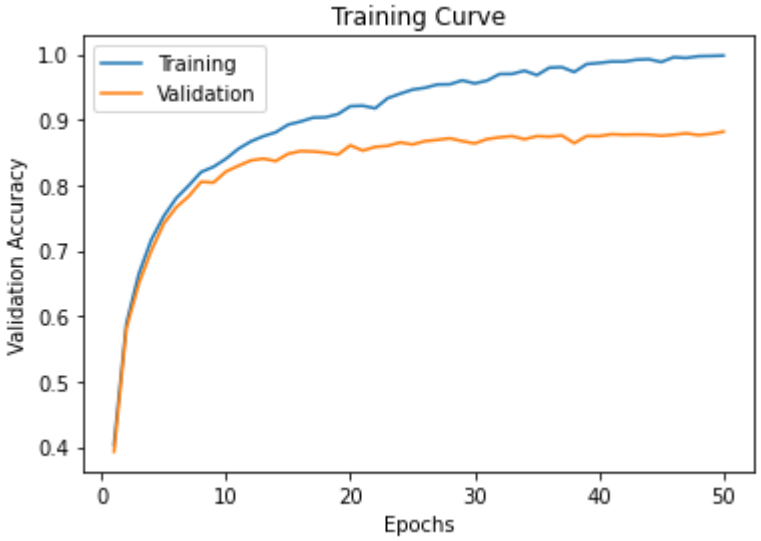
batch_size = 64, lr = 0.0001, #epochs = 50, singlelayer+2xcnn

In [39]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()
```

```
if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=50,lr
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.89 s
Iteration:  462 Progress:   2.00  % Time Elapsed:  21.87 s
Iteration:  693 Progress:   4.00  % Time Elapsed:  32.70 s
Iteration:  924 Progress:   6.00  % Time Elapsed:  43.65 s
Iteration:  1155 Progress:   8.00  % Time Elapsed:  54.42 s
Iteration:  1386 Progress:  10.00  % Time Elapsed:  65.37 s
Iteration:  1617 Progress:  12.00  % Time Elapsed:  76.29 s
Iteration:  1848 Progress:  14.00  % Time Elapsed:  87.21 s
Iteration:  2079 Progress:  16.00  % Time Elapsed:  97.93 s
Iteration:  2310 Progress:  18.00  % Time Elapsed:  108.70 s
Iteration:  2541 Progress:  20.00  % Time Elapsed:  119.87 s
Iteration:  2772 Progress:  22.00  % Time Elapsed:  130.93 s
Iteration:  3003 Progress:  24.00  % Time Elapsed:  142.17 s
Iteration:  3234 Progress:  26.00  % Time Elapsed:  153.38 s
Iteration:  3465 Progress:  28.00  % Time Elapsed:  164.54 s
Iteration:  3696 Progress:  30.00  % Time Elapsed:  175.54 s
Iteration:  3927 Progress:  32.00  % Time Elapsed:  186.65 s
Iteration:  4158 Progress:  34.00  % Time Elapsed:  197.88 s
Iteration:  4389 Progress:  36.00  % Time Elapsed:  208.88 s
Iteration:  4620 Progress:  38.00  % Time Elapsed:  219.74 s
Iteration:  4851 Progress:  40.00  % Time Elapsed:  230.64 s
Iteration:  5082 Progress:  42.00  % Time Elapsed:  241.53 s
Iteration:  5313 Progress:  44.00  % Time Elapsed:  252.25 s
Iteration:  5544 Progress:  46.00  % Time Elapsed:  263.11 s
Iteration:  5775 Progress:  48.00  % Time Elapsed:  274.06 s
Iteration:  6006 Progress:  50.00  % Time Elapsed:  284.77 s
Iteration:  6237 Progress:  52.00  % Time Elapsed:  295.58 s
Iteration:  6468 Progress:  54.00  % Time Elapsed:  306.48 s
Iteration:  6699 Progress:  56.00  % Time Elapsed:  317.41 s
Iteration:  6930 Progress:  58.00  % Time Elapsed:  328.29 s
Iteration:  7161 Progress:  60.00  % Time Elapsed:  339.22 s
Iteration:  7392 Progress:  62.00  % Time Elapsed:  350.12 s
Iteration:  7623 Progress:  64.00  % Time Elapsed:  361.10 s
Iteration:  7854 Progress:  66.00  % Time Elapsed:  372.03 s
Iteration:  8085 Progress:  68.00  % Time Elapsed:  382.94 s
Iteration:  8316 Progress:  70.00  % Time Elapsed:  393.28 s
Iteration:  8547 Progress:  72.00  % Time Elapsed:  403.38 s
Iteration:  8778 Progress:  74.00  % Time Elapsed:  414.27 s
Iteration:  9009 Progress:  76.00  % Time Elapsed:  425.19 s
Iteration:  9240 Progress:  78.00  % Time Elapsed:  436.11 s
Iteration:  9471 Progress:  80.00  % Time Elapsed:  446.96 s
Iteration:  9702 Progress:  82.00  % Time Elapsed:  457.87 s
Iteration:  9933 Progress:  84.00  % Time Elapsed:  468.73 s
Iteration:  10164 Progress:  86.00  % Time Elapsed:  479.60 s
Iteration:  10395 Progress:  88.00  % Time Elapsed:  490.51 s
Iteration:  10626 Progress:  90.00  % Time Elapsed:  501.42 s
Iteration:  10857 Progress:  92.00  % Time Elapsed:  512.11 s
Iteration:  11088 Progress:  94.00  % Time Elapsed:  522.37 s
Iteration:  11319 Progress:  96.00  % Time Elapsed:  532.30 s
Iteration:  11550 Progress:  98.00  % Time Elapsed:  542.61 s
Epoch 49 Finished.  Time per Epoch:  10.85 s
```






```
Final Training Accuracy: 0.9978354978354979
Final Validation Accuracy: 0.881640625
Total time:   542.61 s  Time per Epoch:  10.85 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

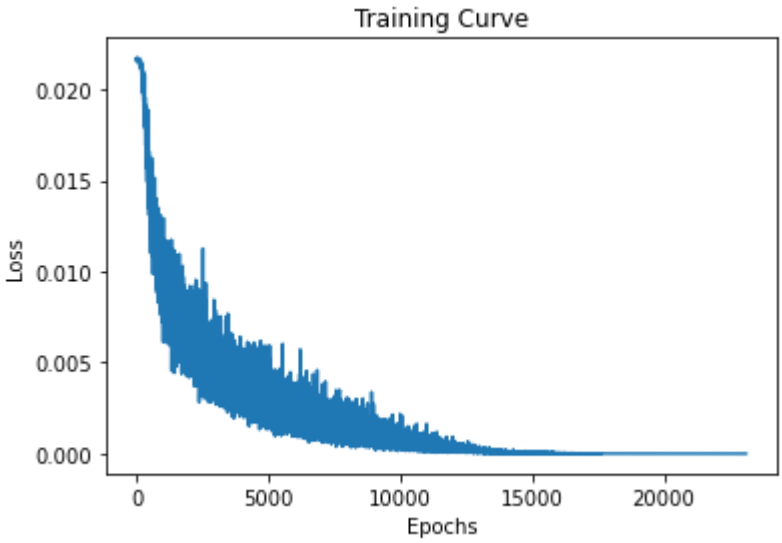
batch_size = 64, lr = 0.0001, #epochs = 100, singlelayer+2xcnn

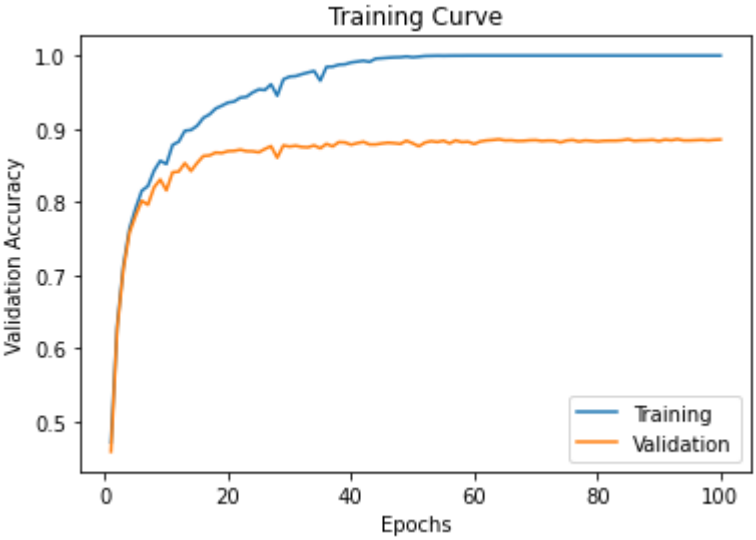
In [40]:
```
# define dataloader parameters
batch_size  = 64 # process 32 images at a time
num_workers = 0 # we only need 1 worker here

# prepare data loaders
train_data_loader = torch.utils.data.DataLoader(train_dataset_new, batch_size=batch_siz
                                                num_workers=num_workers, shuffle=True, drop_
val_data_loader = torch.utils.data.DataLoader(val_dataset_new, batch_size=batch_size,
                                              num_workers=num_workers, shuffle=True, drop_l
test_data_loader = torch.utils.data.DataLoader(test_dataset_new, batch_size=batch_size,
                                               num_workers=num_workers, shuffle=True, drop_l

use_cuda = True

model = ANNClassifier()
```

```
if use_cuda and torch.cuda.is_available():
  model.cuda()
  print('CUDA is available!  Training on GPU ...')
else:
  print('CUDA is not available.  Training on CPU ...')

#proper model
train(model, train_data_loader,val_data_loader, batch_size=batch_size, num_epochs=100,l
print('vgg16 model:','batch_size=64, num_epochs=12,lr=0.0005')
```

```
CUDA is available!  Training on GPU ...
Iteration:  231 Progress:   0.00  % Time Elapsed:  10.87 s
Iteration:  462 Progress:   1.00  % Time Elapsed:  21.58 s
Iteration:  693 Progress:   2.00  % Time Elapsed:  32.40 s
Iteration:  924 Progress:   3.00  % Time Elapsed:  42.97 s
Iteration:  1155 Progress:   4.00  % Time Elapsed:  53.62 s
Iteration:  1386 Progress:   5.00  % Time Elapsed:  64.51 s
Iteration:  1617 Progress:   6.00  % Time Elapsed:  75.39 s
Iteration:  1848 Progress:   7.00  % Time Elapsed:  86.26 s
Iteration:  2079 Progress:   8.00  % Time Elapsed:  97.13 s
Iteration:  2310 Progress:   9.00  % Time Elapsed:  108.03 s
Iteration:  2541 Progress:  10.00  % Time Elapsed:  118.97 s
Iteration:  2772 Progress:  11.00  % Time Elapsed:  129.87 s
Iteration:  3003 Progress:  12.00  % Time Elapsed:  140.37 s
Iteration:  3234 Progress:  13.00  % Time Elapsed:  150.28 s
Iteration:  3465 Progress:  14.00  % Time Elapsed:  160.06 s
Iteration:  3696 Progress:  15.00  % Time Elapsed:  170.06 s
Iteration:  3927 Progress:  16.00  % Time Elapsed:  180.90 s
Iteration:  4158 Progress:  17.00  % Time Elapsed:  191.85 s
Iteration:  4389 Progress:  18.00  % Time Elapsed:  202.70 s
Iteration:  4620 Progress:  19.00  % Time Elapsed:  213.57 s
Iteration:  4851 Progress:  20.00  % Time Elapsed:  224.43 s
Iteration:  5082 Progress:  21.00  % Time Elapsed:  235.32 s
Iteration:  5313 Progress:  22.00  % Time Elapsed:  246.19 s
Iteration:  5544 Progress:  23.00  % Time Elapsed:  257.13 s
Iteration:  5775 Progress:  24.00  % Time Elapsed:  267.93 s
Iteration:  6006 Progress:  25.00  % Time Elapsed:  278.78 s
Iteration:  6237 Progress:  26.00  % Time Elapsed:  289.67 s
Iteration:  6468 Progress:  27.00  % Time Elapsed:  300.57 s
Iteration:  6699 Progress:  28.00  % Time Elapsed:  311.36 s
Iteration:  6930 Progress:  29.00  % Time Elapsed:  322.09 s
Iteration:  7161 Progress:  30.00  % Time Elapsed:  332.76 s
Iteration:  7392 Progress:  31.00  % Time Elapsed:  343.57 s
Iteration:  7623 Progress:  32.00  % Time Elapsed:  354.43 s
Iteration:  7854 Progress:  33.00  % Time Elapsed:  364.77 s
Iteration:  8085 Progress:  34.00  % Time Elapsed:  374.96 s
Iteration:  8316 Progress:  35.00  % Time Elapsed:  385.30 s
Iteration:  8547 Progress:  36.00  % Time Elapsed:  395.45 s
Iteration:  8778 Progress:  37.00  % Time Elapsed:  405.69 s
Iteration:  9009 Progress:  38.00  % Time Elapsed:  415.57 s
Iteration:  9240 Progress:  39.00  % Time Elapsed:  425.31 s
Iteration:  9471 Progress:  40.00  % Time Elapsed:  435.06 s
Iteration:  9702 Progress:  41.00  % Time Elapsed:  444.79 s
Iteration:  9933 Progress:  42.00  % Time Elapsed:  454.49 s
Iteration:  10164 Progress:  43.00  % Time Elapsed:  464.22 s
Iteration:  10395 Progress:  44.00  % Time Elapsed:  473.97 s
Iteration:  10626 Progress:  45.00  % Time Elapsed:  483.73 s
Iteration:  10857 Progress:  46.00  % Time Elapsed:  493.63 s
Iteration:  11088 Progress:  47.00  % Time Elapsed:  503.38 s
Iteration:  11319 Progress:  48.00  % Time Elapsed:  513.11 s
Iteration:  11550 Progress:  49.00  % Time Elapsed:  522.86 s
Iteration:  11781 Progress:  50.00  % Time Elapsed:  532.65 s
Iteration:  12012 Progress:  51.00  % Time Elapsed:  542.41 s
```

```
Iteration:  12243 Progress:  52.00  % Time Elapsed:  552.20 s
Iteration:  12474 Progress:  53.00  % Time Elapsed:  561.97 s
Iteration:  12705 Progress:  54.00  % Time Elapsed:  571.70 s
Iteration:  12936 Progress:  55.00  % Time Elapsed:  581.45 s
Iteration:  13167 Progress:  56.00  % Time Elapsed:  591.19 s
Iteration:  13398 Progress:  57.00  % Time Elapsed:  600.94 s
Iteration:  13629 Progress:  58.00  % Time Elapsed:  610.95 s
Iteration:  13860 Progress:  59.00  % Time Elapsed:  620.93 s
Iteration:  14091 Progress:  60.00  % Time Elapsed:  631.16 s
Iteration:  14322 Progress:  61.00  % Time Elapsed:  641.35 s
Iteration:  14553 Progress:  62.00  % Time Elapsed:  651.36 s
Iteration:  14784 Progress:  63.00  % Time Elapsed:  661.31 s
Iteration:  15015 Progress:  64.00  % Time Elapsed:  671.31 s
Iteration:  15246 Progress:  65.00  % Time Elapsed:  681.31 s
Iteration:  15477 Progress:  66.00  % Time Elapsed:  691.29 s
Iteration:  15708 Progress:  67.00  % Time Elapsed:  701.27 s
Iteration:  15939 Progress:  68.00  % Time Elapsed:  711.26 s
Iteration:  16170 Progress:  69.00  % Time Elapsed:  721.20 s
Iteration:  16401 Progress:  70.00  % Time Elapsed:  731.26 s
Iteration:  16632 Progress:  71.00  % Time Elapsed:  741.81 s
Iteration:  16863 Progress:  72.00  % Time Elapsed:  752.23 s
Iteration:  17094 Progress:  73.00  % Time Elapsed:  762.25 s
Iteration:  17325 Progress:  74.00  % Time Elapsed:  772.29 s
Iteration:  17556 Progress:  75.00  % Time Elapsed:  782.30 s
Iteration:  17787 Progress:  76.00  % Time Elapsed:  792.38 s
Iteration:  18018 Progress:  77.00  % Time Elapsed:  802.43 s
Iteration:  18249 Progress:  78.00  % Time Elapsed:  812.49 s
Iteration:  18480 Progress:  79.00  % Time Elapsed:  822.60 s
Iteration:  18711 Progress:  80.00  % Time Elapsed:  832.73 s
Iteration:  18942 Progress:  81.00  % Time Elapsed:  842.80 s
Iteration:  19173 Progress:  82.00  % Time Elapsed:  852.98 s
Iteration:  19404 Progress:  83.00  % Time Elapsed:  862.96 s
Iteration:  19635 Progress:  84.00  % Time Elapsed:  872.95 s
Iteration:  19866 Progress:  85.00  % Time Elapsed:  882.94 s
Iteration:  20097 Progress:  86.00  % Time Elapsed:  892.97 s
Iteration:  20328 Progress:  87.00  % Time Elapsed:  902.97 s
Iteration:  20559 Progress:  88.00  % Time Elapsed:  913.02 s
Iteration:  20790 Progress:  89.00  % Time Elapsed:  923.02 s
Iteration:  21021 Progress:  90.00  % Time Elapsed:  933.06 s
Iteration:  21252 Progress:  91.00  % Time Elapsed:  943.12 s
Iteration:  21483 Progress:  92.00  % Time Elapsed:  953.65 s
Iteration:  21714 Progress:  93.00  % Time Elapsed:  964.21 s
Iteration:  21945 Progress:  94.00  % Time Elapsed:  974.29 s
Iteration:  22176 Progress:  95.00  % Time Elapsed:  984.33 s
Iteration:  22407 Progress:  96.00  % Time Elapsed:  994.34 s
Iteration:  22638 Progress:  97.00  % Time Elapsed:  1004.39 s
Iteration:  22869 Progress:  98.00  % Time Elapsed:  1014.48 s
Iteration:  23100 Progress:  99.00  % Time Elapsed:  1024.61 s
Epoch 99 Finished.  Time per Epoch:  10.25 s
```






```
Final Training Accuracy: 1.0
Final Validation Accuracy: 0.8853515625
Total time:   1024.61 s  Time per Epoch:  10.25 s
vgg16 model: batch_size=64, num_epochs=12,lr=0.0005
```

In [ ]: