In [1]:

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.autograd import Variable
import torchvision
from torchvision import datasets, models, transforms
from torch.utils.data import Dataset, DataLoader, ConcatDataset
from skimage import io, transform,data
from torchvision import transforms, utils
import numpy as np
import math
import glob
import matplotlib.pyplot as plt
import time
import os
import copy
import sklearn.svm
import cv2
from matplotlib import pyplot as plt
import numpy as np
from os.path import exists
import pandas as pd
import PIL
import random
from google.colab import drive
from sklearn.metrics.cluster import completeness_score
from sklearn.cluster import KMeans
from tqdm import tqdm, tqdm_notebook
from functools import partial
from torchsummary import summary
from torchvision.datasets import ImageFolder
from torch.utils.data.sampler import SubsetRandomSampler

#cuda_output = !ldconfig -p|grep cudart.so|sed -e 's/.*\.\([0-9]*\)\.\([0-9]*\)$/cu\1\2/'
#accelerator = cuda_output[0] if exists('/dev/nvidia0') else 'cpu'

#print("Accelerator type = ",accelerator)
#print("Pytorch verision: ", torch.__version__)
```

In [2]:

```
!pip install tifffile
```

```
Requirement already satisfied: tifffile in /usr/local/lib/python3.7/dist-packages (2021.4
.8)
Requirement already satisfied: numpy>=1.15.1 in /usr/local/lib/python3.7/dist-packages (f
rom tifffile) (1.19.5)
```

Import Drive

In [3]:

```
# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Sentinel-2 Land-Cover Subet Dataset with 5 Categories, having 100 images each

In [4]:

```
aerial_path = '/content/drive/MyDrive/sentinel-2_rgb'
```

In [5]:

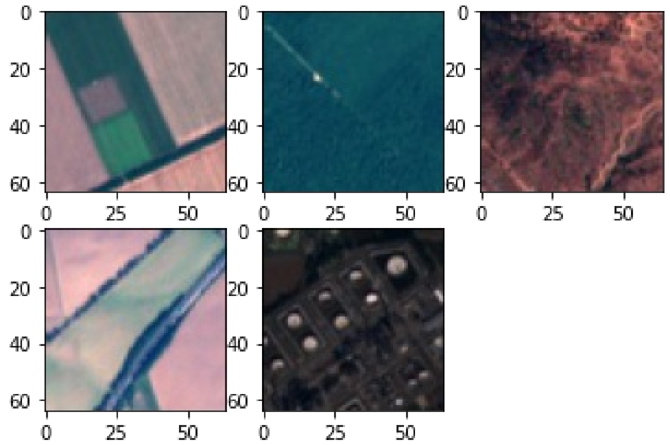
```
import tifffile
from PIL import Image
from torchvision.transforms import ToTensor

files_0 = os.listdir(aerial_path+'/0/')
files_1 = os.listdir(aerial_path+'/1/')
files_2 = os.listdir(aerial_path+'/2/')
files_3 = os.listdir(aerial_path+'/3/')
files_4 = os.listdir(aerial_path+'/4/')



img_rgb_0  = Image.open(aerial_path + '/0/' + files_0[5])
img_rgb_1  = Image.open(aerial_path + '/1/' + files_1[15])
img_rgb_2  = Image.open(aerial_path + '/2/' + files_2[45])
img_rgb_3  = Image.open(aerial_path + '/3/' + files_3[30])
img_rgb_4  = Image.open(aerial_path + '/4/' + files_4[70])
```

In [6]:

```
plt.subplot(231)
plt.imshow(img_rgb_0)
plt.subplot(232)
plt.imshow(img_rgb_1)
plt.subplot(233)
plt.imshow(img_rgb_2)
plt.subplot(234)
plt.imshow(img_rgb_3)
plt.subplot(235)
plt.imshow(img_rgb_4)
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x7f20805e2f10>
```



Path to Aerial Dataset

(A combination of samples from Small Village and Industrial Datasets from Sensefly)

Aerial Dataset Class

In [7]:

```
def get_sat_data(folder_path,transforms=None):
  if transforms:
    dataset_full = ImageFolder(folder_path,preprocessor)
  else:
    dataset_full = ImageFolder(folder_path,preprocessor)
  return dataset_full
```

In [8]:

```
'''
target = '/content/drive/MyDrive/sentinel-2_rgb/4/'
files = os.listdir(aerial_path+'/4/')
files.sort()
for file in tqdm(files):
  img_13  = tifffile.imread(aerial_path + '/4/' + file)
  img_bgr = img_13[:,:,1:4]
  #img = img_bgr[::-1]
  img = cv2.normalize(img_bgr, None, 0, 255, cv2.NORM_MINMAX, dtype=cv2.CV_8U)

  cv2.imwrite(target+file.split('.')[0]+'.jpg',img)
'''
```

Out[8]:

```
"\ntarget = '/content/drive/MyDrive/sentinel-2_rgb/4/'\nfiles = os.listdir(aerial_path+'/
4/')\nfiles.sort()\nfor file in tqdm(files):\n  img_13  = tifffile.imread(aerial_path + '
/4/' + file)\n  img_bgr = img_13[:,:,1:4]\n  #img = img_bgr[::-1]\n  img = cv2.normalize(
img_bgr, None, 0, 255, cv2.NORM_MINMAX, dtype=cv2.CV_8U)\n\n  cv2.imwrite(target+file.spl
it('.')[0]+'.jpg',img)\n"
```

In [9]:

```
'''
def find_classes(dir):
    classes = [d for d in os.listdir(dir) if os.path.isdir(os.path.join(dir, d))]
    classes.sort()
    class_to_idx = {classes[i]: i for i in range(len(classes))}
    return classes, class_to_idx

IMG_EXTENSIONS = [
    '.jpg', '.JPG', '.jpeg', '.JPEG',
    '.png', '.PNG', '.ppm', '.PPM', '.bmp', '.BMP','.tif'
]

def is_image_file(filename):
    return any(filename.endswith(extension) for extension in IMG_EXTENSIONS)


def make_dataset(dir, class_to_idx):
    images = []
    dir = os.path.expanduser(dir)
    for target in sorted(os.listdir(dir)):
        d = os.path.join(dir, target)
        if not os.path.isdir(d):
            continue

        for root, _, fnames in sorted(os.walk(d)):
            for fname in sorted(fnames):
                if is_image_file(fname):
                    path = os.path.join(root, fname)
                    item = (path, class_to_idx[target])
                    images.append(item)

    return images

class ImageFolder(torch.utils.data.Dataset):

  def __init__(self, root, transform=None, target_transform=None):
      classes, class_to_idx = find_classes(root)
      imgs = make_dataset(root, class_to_idx)
      if len(imgs) == 0:
          raise(RuntimeError("Found 0 images in subfolders of: " + root + "\n"
                              "Supported image extensions are: " + ",".join(IMG_EXTENSIO
NS)))

      self.root = root
      self.imgs = imgs
      self.classes = classes
      self.class_to_idx = class_to_idx
      self.transform = transform
      self.target_transform = target_transform

  def __getitem__(self, index):

      path, target = self.imgs[index]
      #print(ok)
      #img = self.loader(path)
      img_13  = tifffile.imread(path)
      img_bgr = img_13[:,:,1:4]
      img = img_bgr[::-1]
      img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX, dtype=cv2.CV_8U)
      if self.transform is not None:
          img = self.transform(img)
      if self.target_transform is not None:
          target = self.target_transform(target)
      return img, target

  def __len__(self):
      return len(self.imgs)
'''
```

Out[9]:

```
'\ndef find_classes(dir):\n    classes = [d for d in os.listdir(dir) if os.path.isdir(os.
path.join(dir, d))]\n    classes.sort()\n    class_to_idx = {classes[i]: i for i in range
(len(classes))}\n    return classes, class_to_idx\n\nIMG_EXTENSIONS = [\n    \'.jpg\', \'
.JPG\', \'.jpeg\', \'.JPEG\',\n    \'.png\', \'.PNG\', \'.ppm\', \'.PPM\', \'.bmp\', \'.B
MP\',\'.tif\'\n]\n\ndef is_image_file(filename):\n    return any(filename.endswith(extens
ion) for extension in IMG_EXTENSIONS)\n\n\ndef make_dataset(dir, class_to_idx):\n    imag
es = []\n    dir = os.path.expanduser(dir)\n    for target in sorted(os.listdir(dir)):\n
d = os.path.join(dir, target)\n        if not os.path.isdir(d):\n            continue\n\n
for root, _, fnames in sorted(os.walk(d)):\n            for fname in sorted(fnames):\n
if is_image_file(fname):\n                    path = os.path.join(root, fname)\n
item = (path, class_to_idx[target])\n                    images.append(item)\n\n    retu
rn images\n\nclass ImageFolder(torch.utils.data.Dataset):\n\n  def __init__(self, root, t
ransform=None, target_transform=None):\n      classes, class_to_idx = find_classes(root)\
n      imgs = make_dataset(root, class_to_idx)\n      if len(imgs) == 0:\n          raise
(RuntimeError("Found 0 images in subfolders of: " + root + "\n"\n
"Supported image extensions are: " + ",".join(IMG_EXTENSIONS)))\n\n      self.root = root
\n      self.imgs = imgs\n      self.classes = classes\n      self.class_to_idx = class_t
o_idx\n      self.transform = transform\n      self.target_transform = target_transform\n
\n  def __getitem__(self, index):\n\n      path, target = self.imgs[index]\n      #print(
ok)\n      #img = self.loader(path)\n      img_13  = tifffile.imread(path)\n      img_bgr
= img_13[:,:,1:4]\n      img = img_bgr[::-1]\n      img = cv2.normalize(img, None, 0, 255
, cv2.NORM_MINMAX, dtype=cv2.CV_8U)\n      if self.transform is not None:\n          img
= self.transform(img)\n      if self.target_transform is not None:\n          target = se
lf.target_transform(target)\n      return img, target\n\n  def __len__(self):\n      retu
rn len(self.imgs)\n'
```

Dataset and Dataloader initialization

In [10]:

```
#Pre processing the data
normalize = transforms.Normalize(mean = [0.485,0.456,0.406],
                                std = [0.229,0.224,0.225])
resize = transforms.Resize((224,224))



preprocessor = transforms.Compose([ resize, transforms.ToTensor(), normalize
                                   ])

aerial_dataset_full = get_sat_data(aerial_path,preprocessor)

# Creating data indices for training and validation splits:
dataset_size = len(aerial_dataset_full)
```

```
indices = list(range(dataset_size))
validation_split = 0.2
split = int(np.floor(validation_split * dataset_size))
shuffle_dataset = True
random_seed= 101

if shuffle_dataset :
    np.random.seed(random_seed)
    np.random.shuffle(indices)
train_indices, val_indices = indices[split:], indices[:split]

# Creating PT data samplers and loaders:
train_sampler = SubsetRandomSampler(train_indices)
valid_sampler = SubsetRandomSampler(val_indices)

aerial_train_loader = torch.utils.data.DataLoader(aerial_dataset_full, batch_size=16,
                                           sampler=train_sampler)
aerial_validation_loader = torch.utils.data.DataLoader(aerial_dataset_full, batch_size=16
,
                                                sampler=valid_sampler)
```

In [11]:

```
print(aerial_dataset_full[0])
```

```
(tensor([[[-1.6898, -1.6898, -1.6898,  ...,  0.3994,  0.3823,  0.3823],
         [-1.6898, -1.6898, -1.6898,  ...,  0.3994,  0.3823,  0.3823],
         [-1.6898, -1.6898, -1.6898,  ...,  0.3823,  0.3652,  0.3652],
         ...,
         [-2.0494, -2.0494, -2.0494,  ...,  1.0844,  1.1015,  1.1015],
         [-2.0494, -2.0494, -2.0494,  ...,  1.1015,  1.1187,  1.1187],
         [-2.0494, -2.0494, -2.0494,  ...,  1.1015,  1.1187,  1.1187]],

        [[-0.5651, -0.5651, -0.5651,  ..., -0.1800, -0.1975, -0.1975],
         [-0.5651, -0.5651, -0.5651,  ..., -0.1800, -0.1975, -0.1975],
         [-0.5651, -0.5651, -0.5651,  ..., -0.1975, -0.2150, -0.2150],
         ...,
         [-1.3004, -1.3004, -1.3004,  ...,  0.1702,  0.1877,  0.1877],
         [-1.2829, -1.2829, -1.2829,  ...,  0.1877,  0.2052,  0.2052],
         [-1.2829, -1.2829, -1.2829,  ...,  0.1877,  0.2052,  0.2052]],

        [[-0.6541, -0.6541, -0.6541,  ..., -0.2707, -0.2881, -0.2881],
         [-0.6541, -0.6541, -0.6541,  ..., -0.2707, -0.2881, -0.2881],
         [-0.6541, -0.6541, -0.6541,  ..., -0.2881, -0.3055, -0.3055],
         ...,
         [-0.9330, -0.9330, -0.9330,  ...,  0.0779,  0.0953,  0.0953],
         [-0.9330, -0.9330, -0.9330,  ...,  0.0953,  0.1128,  0.1128],
         [-0.9330, -0.9330, -0.9330,  ...,  0.0953,  0.1128,  0.1128]]]), 0)
```

In [12]:

```
num_classes = 5
```

Training and Validation/Test loop

In [13]:

```
def training_and_validation_loop(epochs,xp_lr_scheduler,model,optmizer,aerial_train_loade
r,aerial_validation_loader,best_acc,best_model_wts,saved_model_name):

  train_loss = []
  test_loss = []
  accuracy = []

  for e in range(epochs):
        step_lr_scheduler.step()

        #put model in training mode
        model.train()
        avg_loss = 0

        for i, (x,y) in enumerate(aerial_train_loader):
              optimizer.zero_grad()

              if gpu_flag:
                    img_var = Variable(x).cuda()
                    label_actual = Variable(y).cuda()
              else:
                    img_var = Variable(x)
                    label_actual = Variable(y)

              label_predicted = model.forward(img_var)
              loss = criterion(label_predicted,label_actual)
              loss.backward()

              if(i%10 == 0):
                    print(i, loss.item())
              avg_loss+=loss.item()
              optimizer.step()

        print("Done Training")
        train_loss.append(avg_loss*1.0/(i+1))

        #set model in evaluation mode
        model.eval()
        avg_loss = 0
        correct_pred = 0
        total_pred = 0

        for i, (x_test,y_test) in enumerate(aerial_validation_loader):

            if gpu_flag:
                img_test_var = Variable(x_test).cuda()
                label_test_var = Variable(y_test).cuda()
            else:
                img_test_var = Variable(x_test)
                label_test_var = Variable(y_test)

            label_predicted_test = model.forward(img_test_var)
            loss = criterion(label_predicted_test,label_test_var)
            avg_loss+=loss.item()
            vals, label_predicted = torch.max(label_predicted_test,1)

            correct_pred += (label_predicted.cpu().data.numpy()==label_test_var.cpu().da
ta.numpy()).sum()
            total_pred += len(label_predicted_test.cpu())

        test_loss.append(avg_loss*1.0/i)
        accuracy.append(correct_pred*100.0/total_pred)
        print("Epoch: ", e, "Train Loss: ", train_loss[-1], "Test Loss: ", test_loss[-1]
, "Accuracy: ", accuracy[-1])

        #replace model saved
        if accuracy[-1]>best_acc:
            best_acc = accuracy[-1]
            best_model_wts = copy.deepcopy(model.state_dict())
            model.load_state_dict(best_model_wts)
            torch.save(model,f'/content/drive/My Drive/sentinel-2_rgb/{saved_model_name}
.pt')
            print("Saved model with accuracy: ", best_acc)

  return train_loss,test_loss,accuracy
```

VGG-16 Model Initialization

In [44]:

```
# Initialize the model
model = models.vgg16(pretrained=True)
```

```
# Change the device to GPU
device = torch.device('cuda:0' if torch.cuda.is_available() else "cpu")
```

In [45]:

```
# Freeze training for all layers
for param in model.features.parameters():
    param.require_grad = False


num_features = model.classifier[6].in_features
# Remove last layer
features = list(model.classifier.children())[:-1]

# Add our layer with 10 outputs
features.extend([nn.Linear(num_features, num_classes)])

# Replace the model classifier
model.classifier = nn.Sequential(*features)

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)


#preprocessor = transforms.Compose([resize,transforms.ToTensor(),normalize])
```

In [46]:

```
gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()
    criterion = criterion.cuda()

print(model)
```

```
True
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=5, bias=True)
  )
)
```

Summary of how an example image (224,224,3) is processed through the model-pipleine

In [17]:

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

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 224, 224]           1,792
              ReLU-2         [-1, 64, 224, 224]               0
            Conv2d-3         [-1, 64, 224, 224]          36,928
              ReLU-4         [-1, 64, 224, 224]               0
         MaxPool2d-5         [-1, 64, 112, 112]               0
            Conv2d-6        [-1, 128, 112, 112]          73,856
              ReLU-7        [-1, 128, 112, 112]               0
            Conv2d-8        [-1, 128, 112, 112]         147,584
              ReLU-9        [-1, 128, 112, 112]               0
        MaxPool2d-10          [-1, 128, 56, 56]               0
           Conv2d-11          [-1, 256, 56, 56]         295,168
             ReLU-12          [-1, 256, 56, 56]               0
           Conv2d-13          [-1, 256, 56, 56]         590,080
             ReLU-14          [-1, 256, 56, 56]               0
           Conv2d-15          [-1, 256, 56, 56]         590,080
             ReLU-16          [-1, 256, 56, 56]               0
        MaxPool2d-17          [-1, 256, 28, 28]               0
           Conv2d-18          [-1, 512, 28, 28]       1,180,160
             ReLU-19          [-1, 512, 28, 28]               0
           Conv2d-20          [-1, 512, 28, 28]       2,359,808
             ReLU-21          [-1, 512, 28, 28]               0
           Conv2d-22          [-1, 512, 28, 28]       2,359,808
             ReLU-23          [-1, 512, 28, 28]               0
        MaxPool2d-24          [-1, 512, 14, 14]               0
           Conv2d-25          [-1, 512, 14, 14]       2,359,808
             ReLU-26          [-1, 512, 14, 14]               0
           Conv2d-27          [-1, 512, 14, 14]       2,359,808
             ReLU-28          [-1, 512, 14, 14]               0
           Conv2d-29          [-1, 512, 14, 14]       2,359,808
             ReLU-30          [-1, 512, 14, 14]               0
        MaxPool2d-31            [-1, 512, 7, 7]               0
AdaptiveAvgPool2d-32            [-1, 512, 7, 7]               0
           Linear-33                 [-1, 4096]     102,764,544
             ReLU-34                 [-1, 4096]               0
          Dropout-35                 [-1, 4096]               0
           Linear-36                 [-1, 4096]      16,781,312
             ReLU-37                 [-1, 4096]               0
          Dropout-38                 [-1, 4096]               0
           Linear-39                    [-1, 5]          20,485
================================================================
Total params: 134,281,029
Trainable params: 134,281,029
Non-trainable params: 0
```

```
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 218.77
Params size (MB): 512.24
Estimated Total Size (MB): 731.59
----------------------------------------------------------------
```

Feature-Extraction and Display of Feature Maps of Intermediate Layers for an example image tensor

In [47]:

```
class FeatureExtractor(nn.Module):
  def __init__(self, model,num):
    super(FeatureExtractor, self).__init__()
  # Extract VGG-16 Feature Layers
    self.features = list(model.features)
    self.features = nn.Sequential(*self.features)
    self.num = num
  # Extract VGG-16 Average Pooling Layer
    self.pooling = model.avgpool
  # Convert the image into one-dimensional vector
    #self.flatten = nn.Flatten()
  # Extract the first part of fully-connected layer from VGG16
    #self.fc = model.classifier[0]

  def forward(self, x):
  # It will take the input 'x' until it returns the feature vector called 'out'

    filters=[]
    layers = []
    all_out= []

    for layer in self.num:
      feat_op = self.features[:layer]
      layers.append(feat_op)
      out = feat_op(x)
      filters.append(out.shape[1])
      all_out.append(out)
    out = self.pooling(out)
    all_out.append(out)
    list2 = []
    list2.append(self.features)
    list2.append(self.pooling)
    layers.append(list2)

      #out = self.flatten(out)
      #out = self.fc(out)
    return all_out,layers,filters
```

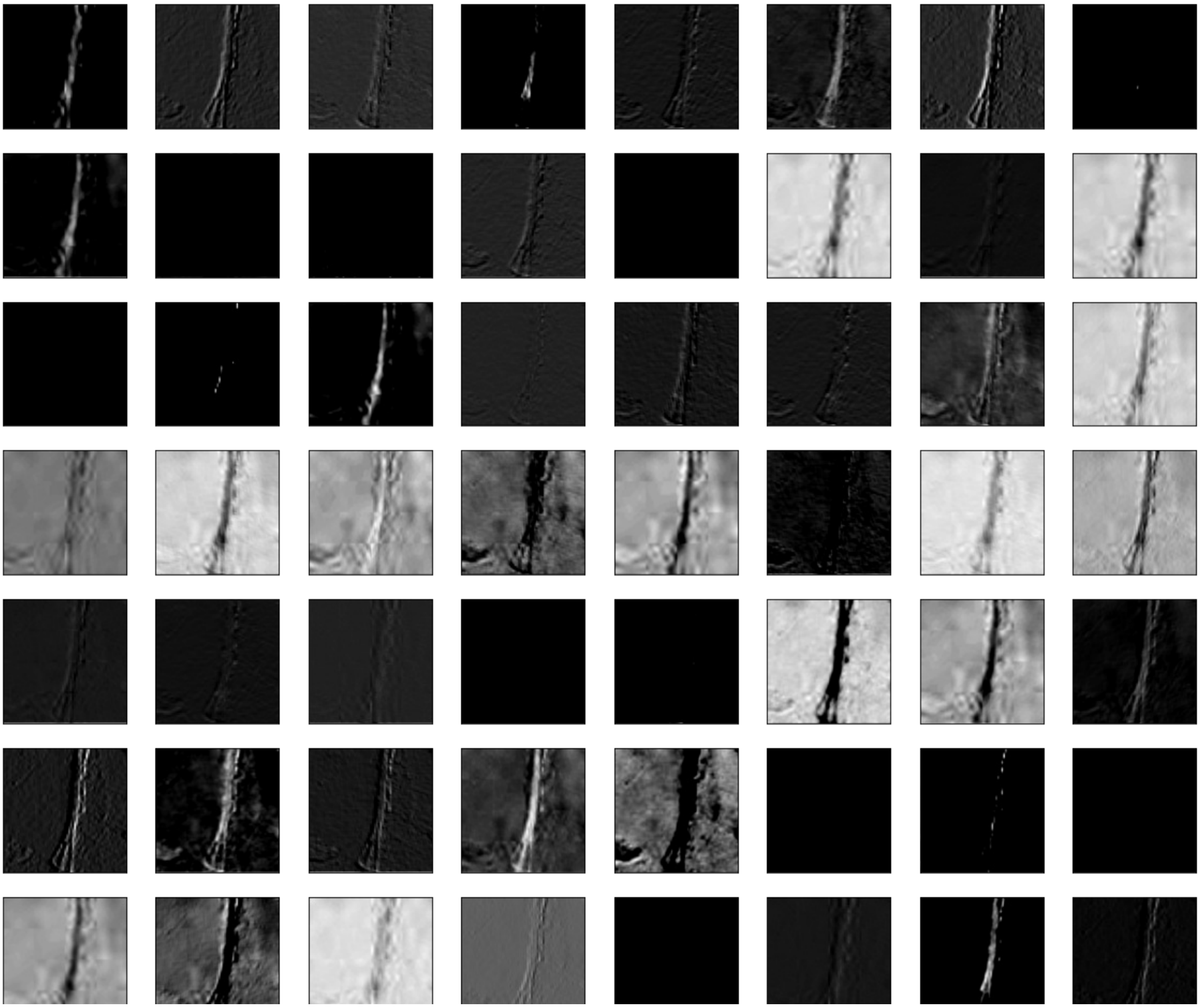
In [48]:

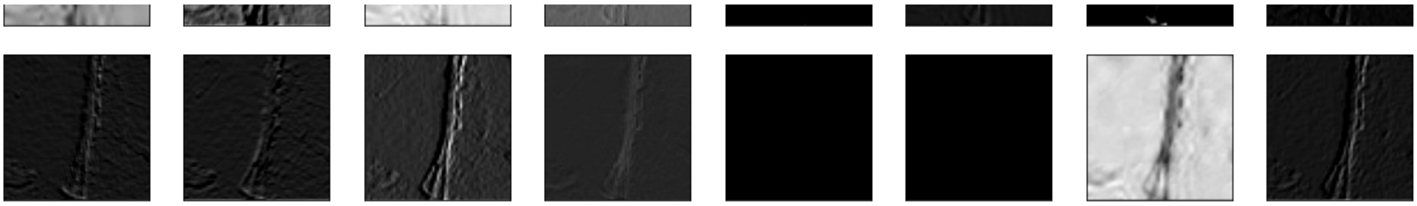
```
imp_layers = [2,5,9,13,17,24,31]
new_model = FeatureExtractor(model,imp_layers)
```

In [49]:

```
#plt.figure(figsize = (20,10))


for i, (x,y) in enumerate(aerial_train_loader):
  img_var = Variable(x).cuda()

  img_feat_all,layers,filters = new_model(img_var)


  #print(img_feat_1.shape)

  for j,img_feat_1 in enumerate(img_feat_all):
    plt.figure(figsize = (20,10))


    img_numpy = img_feat_1.cpu().data.numpy()

    print(layers[j])

    # plot all 64 maps in an 8x8 squares
    square = 8
    square1 = int(8)
    ind = 1
    plt.figure(figsize = (20,20))
    for _ in range(square1):
      for _ in range(square):
        # specify subplot and turn of axis
        ax = plt.subplot(square1, square, ind)
        ax.set_xticks([])
        ax.set_yticks([])
        # plot filter channel in grayscale
        plt.imshow(img_numpy[0, ind-1, :, :-1], cmap='gray')
        ind += 1
    # show the figure
    plt.show()

  print('Feature Extraction (minus the classifier layer) done for first image')
  break
```

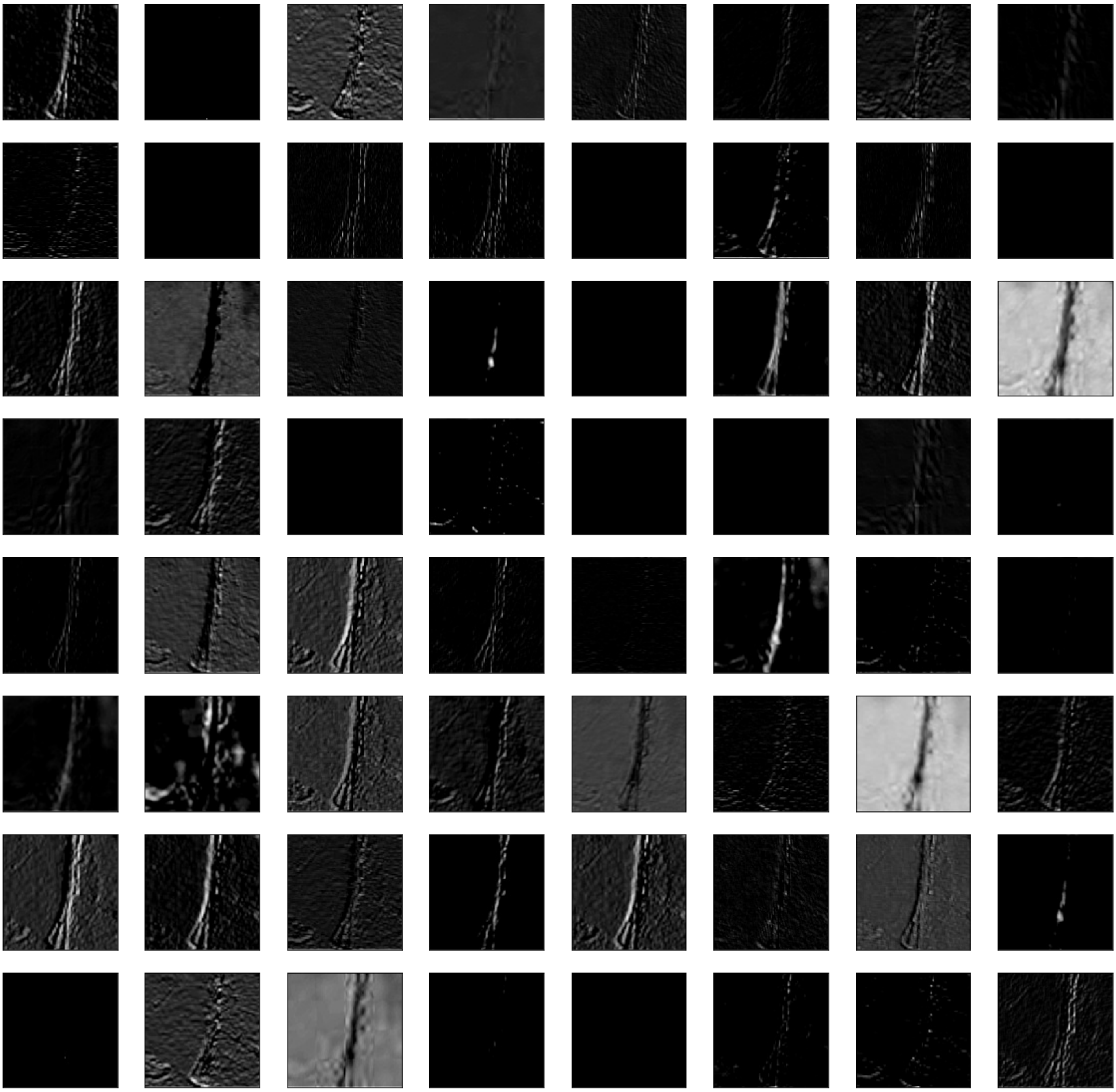
```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
)

<Figure size 1440x720 with 0 Axes>
```





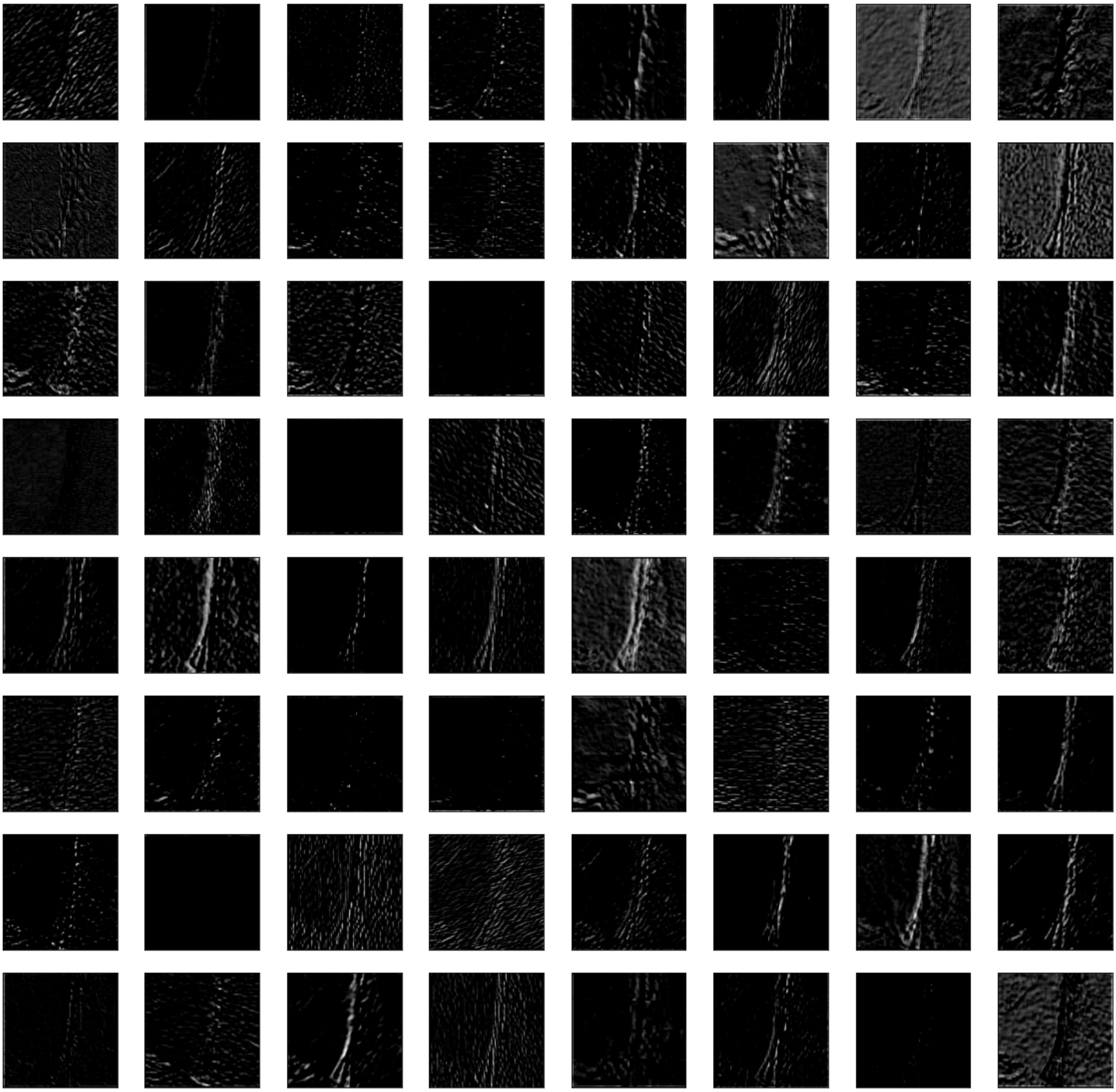
```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

```
<Figure size 1440x720 with 0 Axes>
```



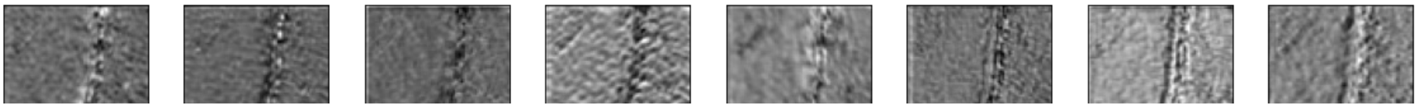
```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): 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)
)
```

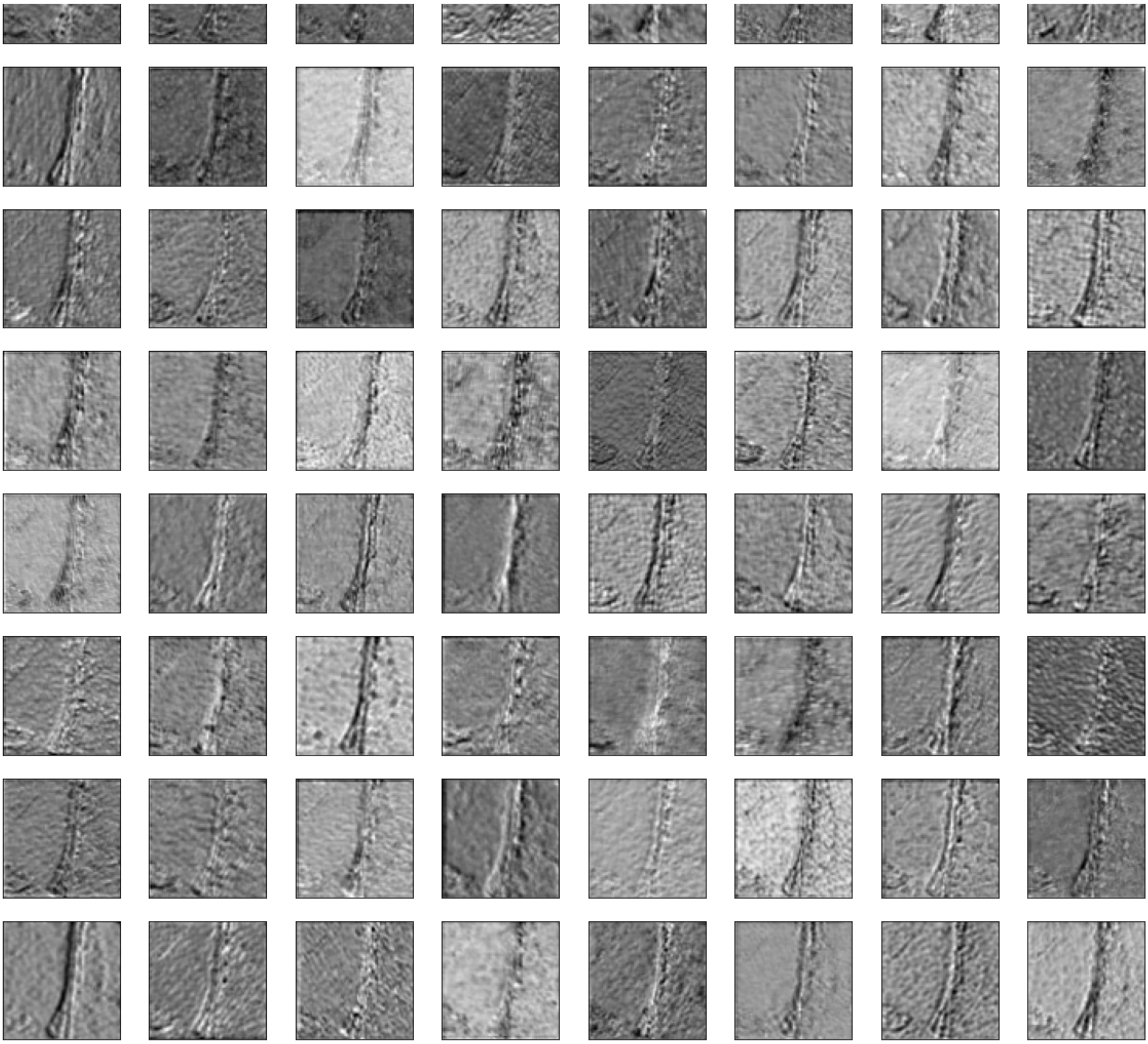
```
<Figure size 1440x720 with 0 Axes>
```



```
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))
)
```

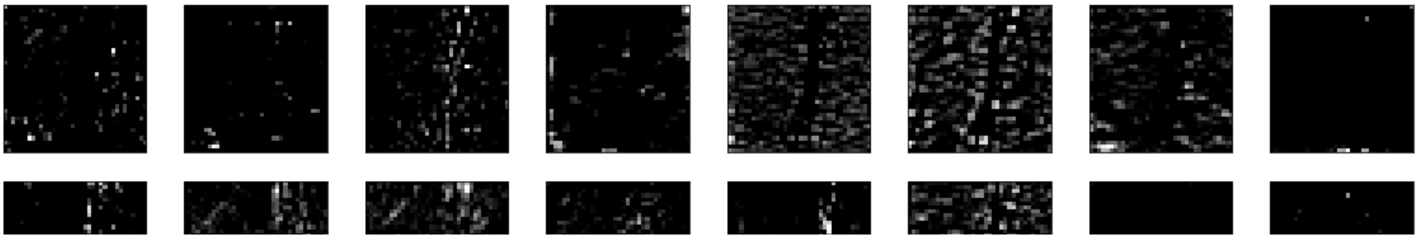
```
<Figure size 1440x720 with 0 Axes>
```

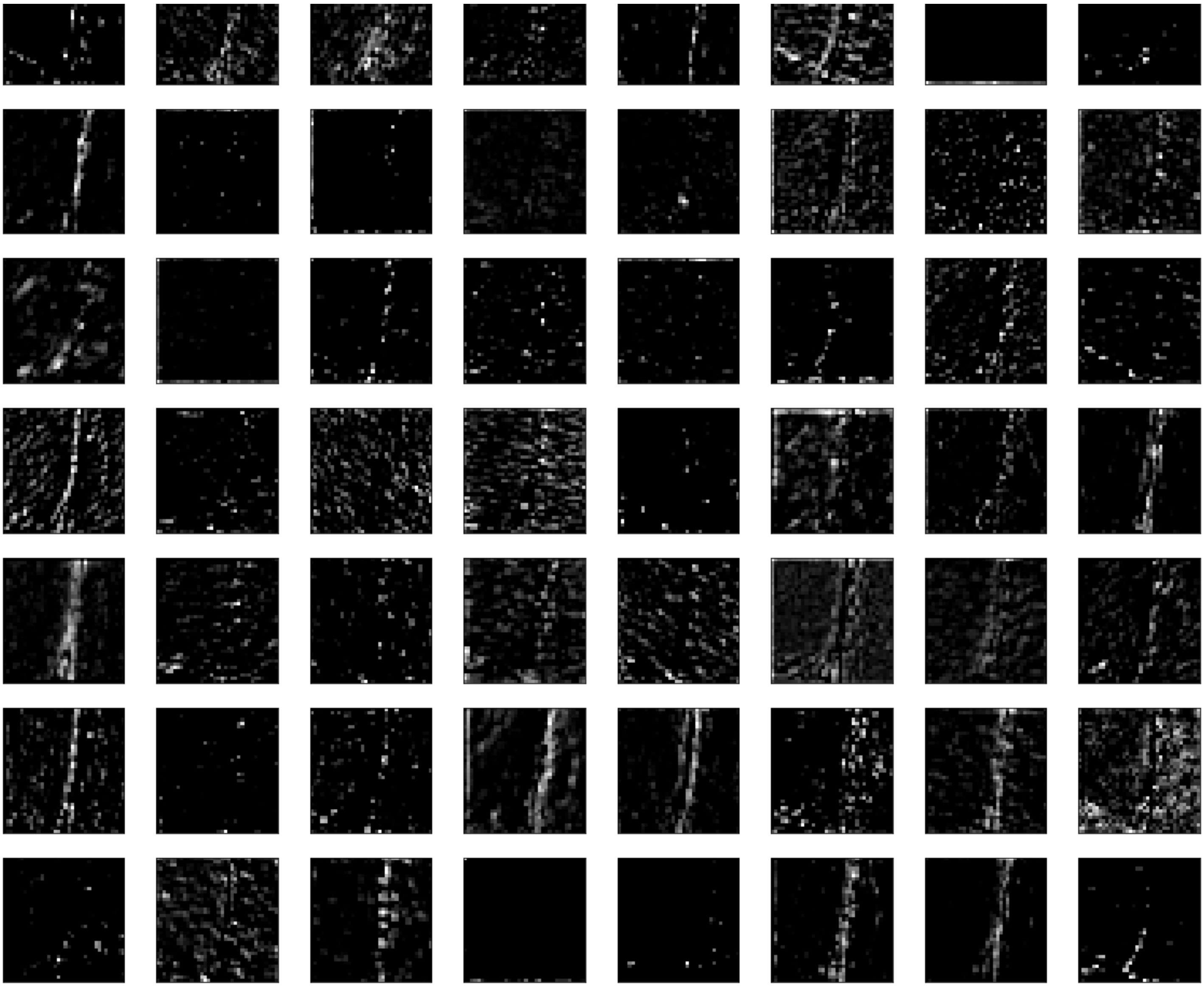




```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): 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)
)
```

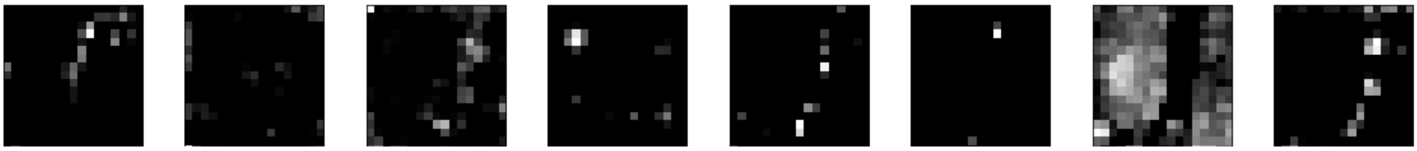
```
<Figure size 1440x720 with 0 Axes>
```

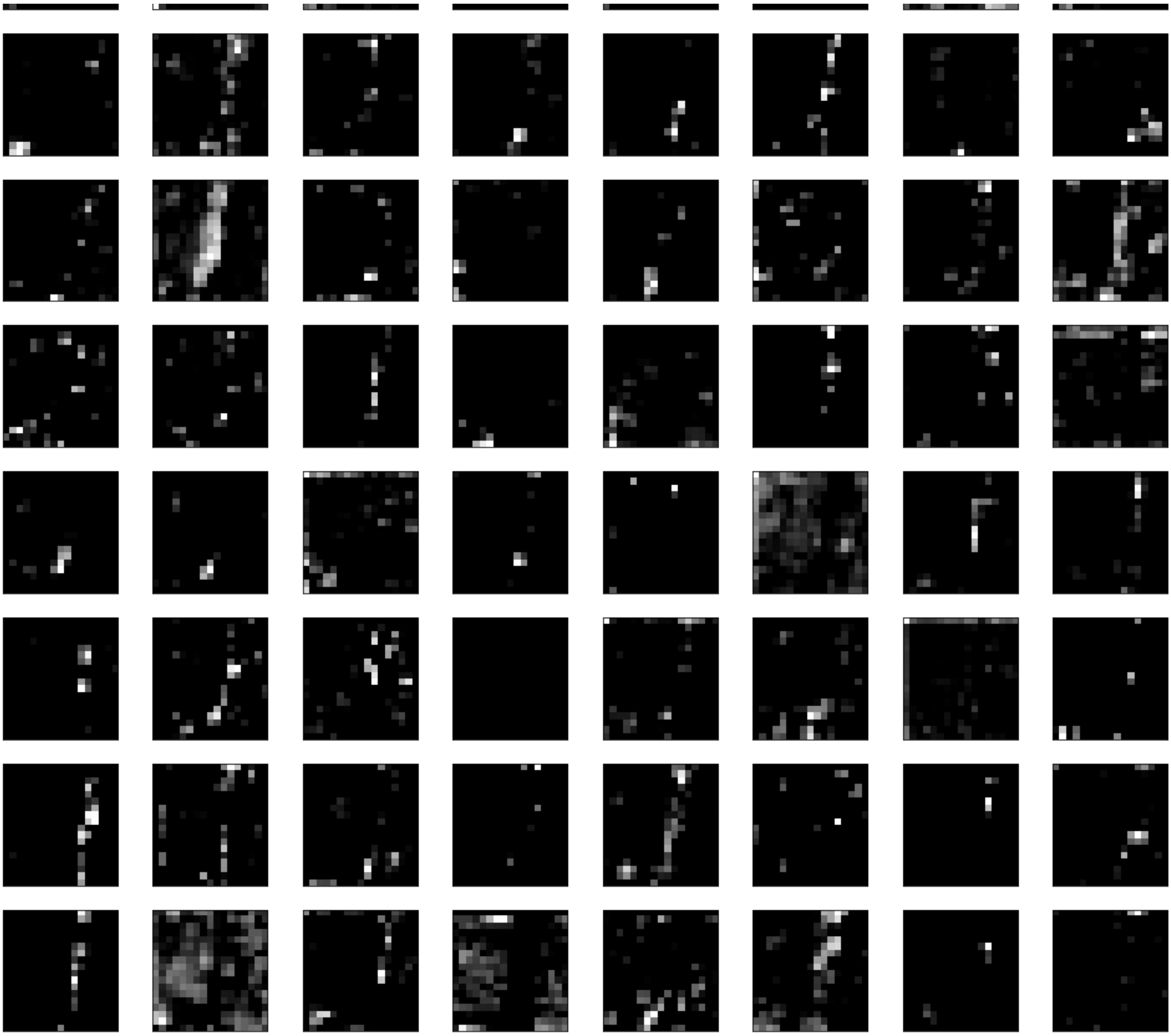




```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): 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)
)
```

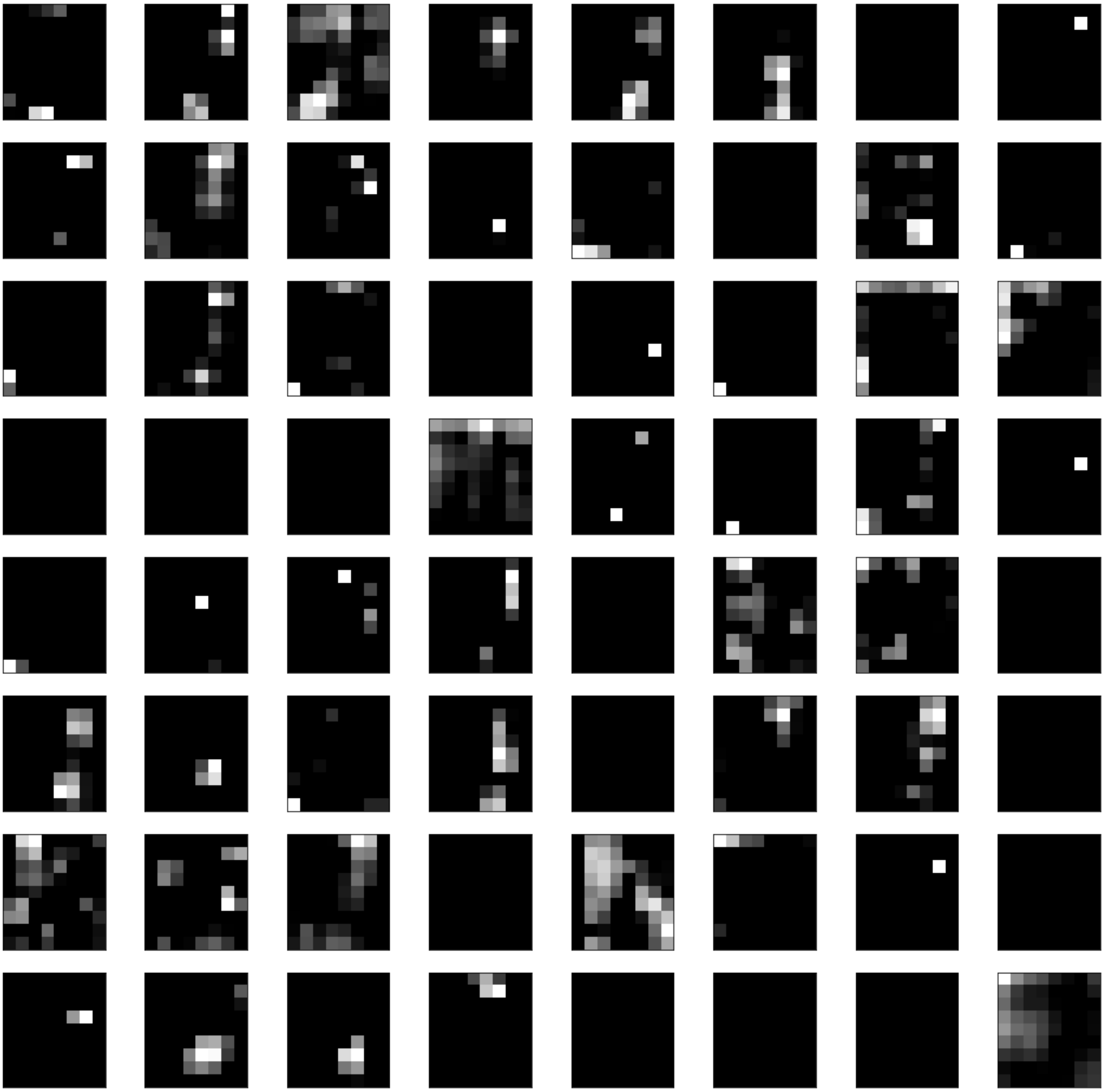
```
<Figure size 1440x720 with 0 Axes>
```





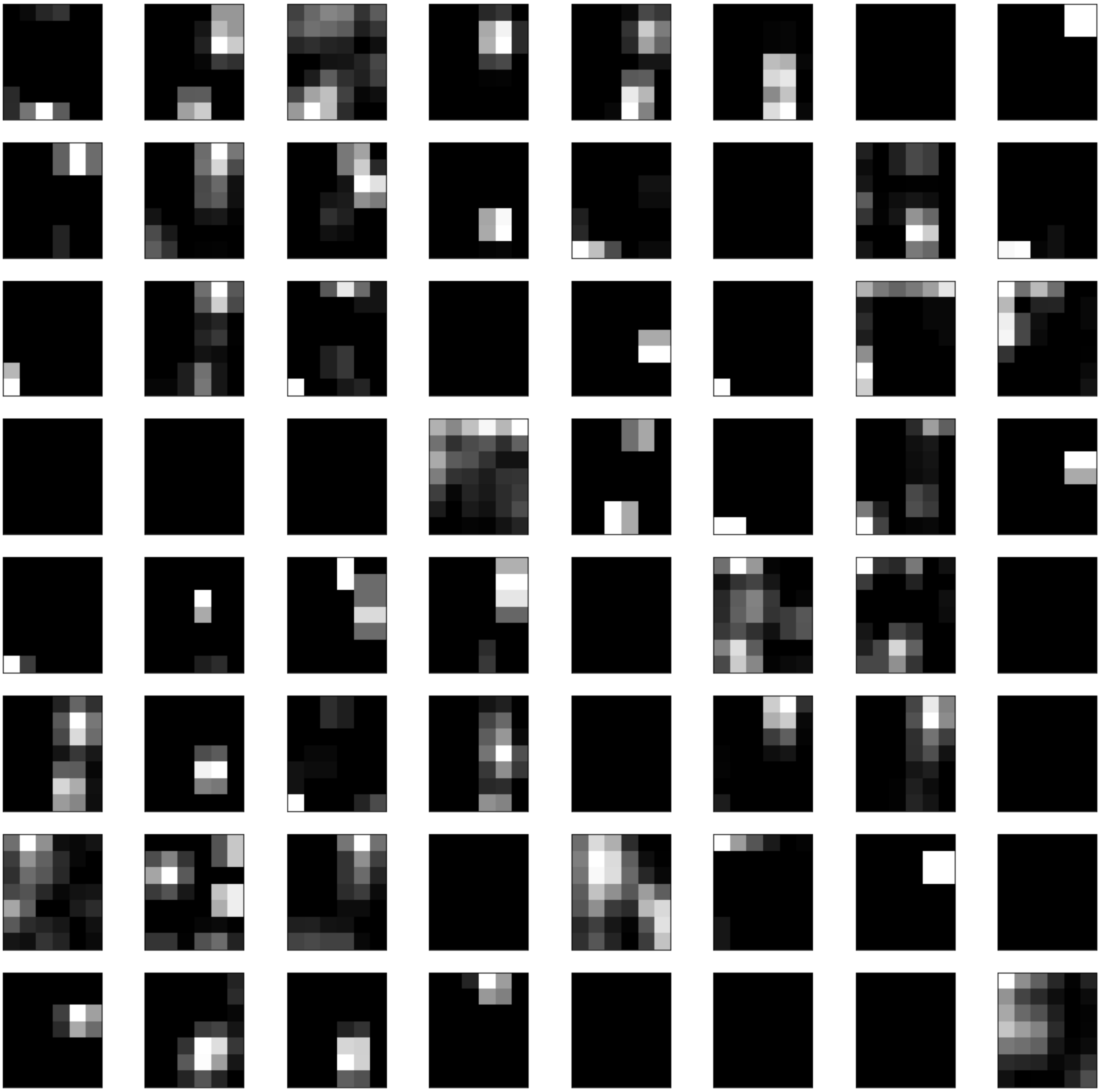
```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): 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)
  (29): ReLU(inplace=True)
  (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
```

```
<Figure size 1440x720 with 0 Axes>
```



```
[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)
), AdaptiveAvgPool2d(output_size=(7, 7))]
```

```
<Figure size 1440x720 with 0 Axes>
```



```
Feature Extraction (minus the classifier layer) done for first image
```

In [18]:

```
tqdm = partial(tqdm, position=0, leave=True)
```

VGG16 Model Training and Validation

In [19]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_vgg,test_loss_vgg,accuracy_vgg = training_and_validation_loop(epochs,step_lr_s
cheduler,model,optimizer,aerial_train_loader,aerial_validation_loader,best_acc,best_model
_wts,'vgg16')
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
0 1.562848687171936
10 0.8532980680465698
20 0.35322946310043335
Done Training
Epoch:  0 Train Loss:  0.8429346442795717 Test Loss:  0.18512837355956435 Accuracy:  93.0
6930693069307
Saved model with accuracy:  93.06930693069307
0 0.1873331367969513
10 0.02580554410815239
20 0.047210127115249634
Done Training
Epoch:  1 Train Loss:  0.18528383952350572 Test Loss:  0.2575116629401843 Accuracy:  95.0
4950495049505
Saved model with accuracy:  95.04950495049505
0 0.17666293680667877
10 0.020201733335852623
20 0.185530886054039
Done Training
Epoch:  2 Train Loss:  0.10453654078838344 Test Loss:  0.2417783897059659 Accuracy:  95.0
4950495049505
0 0.03812095522880554
10 0.20057663321495056
20 0.12721890211105347
Done Training
Epoch:  3 Train Loss:  0.09927637392852026 Test Loss:  0.17437662300653756 Accuracy:  95.
04950495049505
0 0.0521421879529953
10 0.0286464411765337
20 0.02009604126214981
Done Training
Epoch:  4 Train Loss:  0.0974644476082176 Test Loss:  0.16087727372845015 Accuracy:  97.0
2970297029702
Saved model with accuracy:  97.02970297029702
0 0.0482952706515789
10 0.013316628523170948
20 0.00370302377268672
Done Training
Epoch:  5 Train Loss:  0.05999359112376204 Test Loss:  0.1487810741721963 Accuracy:  97.0
2970297029702
0 0.005354869645088911
10 0.0005078981630504131
20 0.008100450970232487
Done Training
Epoch:  6 Train Loss:  0.00987188382826459 Test Loss:  0.15350441700623682 Accuracy:  97.
02970297029702
0 0.002125280676409602
10 0.0020256692077964544
20 0.0016977409832179546
Done Training
Epoch:  7 Train Loss:  0.005225961689855187 Test Loss:  0.12287178949918598 Accuracy:  97
.02970297029702
0 0.0013736592372879386
10 0.005083252675831318
20 0.006679120939224958
Done Training
```

```
Epoch:  8 Train Loss:  0.006373852133177794 Test Loss:  0.1216922367263275 Accuracy:  98.
01980198019803
Saved model with accuracy:  98.01980198019803
0 0.008967640809714794
10 0.0017910231836140156
20 0.0011385509278625250
Done Training
Epoch:  9 Train Loss:  0.005092114116101025 Test Loss:  0.12231693868913378 Accuracy:  98
.01980198019803
```

Resnet-50 Model Training and Validation

In [50]:

```python
gpu_flag = torch.cuda.is_available()

#preloading Resnet18
model = models.resnet50(pretrained = True)

#append a new last layer
model.fc = nn.Linear(2048,num_classes)

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
True
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
```

```
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fal
se)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
```

```
      (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (4): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
```

```
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=5, bias=True)
)
```

Summary of how an example image (224,224,3) is processed through the model pipeline

model-pipleine

In [21]:

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

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5           [-1, 64, 56, 56]           4,096
       BatchNorm2d-6           [-1, 64, 56, 56]             128
              ReLU-7           [-1, 64, 56, 56]               0
            Conv2d-8           [-1, 64, 56, 56]          36,864
       BatchNorm2d-9           [-1, 64, 56, 56]             128
             ReLU-10           [-1, 64, 56, 56]               0
           Conv2d-11          [-1, 256, 56, 56]          16,384
      BatchNorm2d-12          [-1, 256, 56, 56]             512
           Conv2d-13          [-1, 256, 56, 56]          16,384
      BatchNorm2d-14          [-1, 256, 56, 56]             512
             ReLU-15          [-1, 256, 56, 56]               0
       Bottleneck-16          [-1, 256, 56, 56]               0
           Conv2d-17           [-1, 64, 56, 56]          16,384
      BatchNorm2d-18           [-1, 64, 56, 56]             128
             ReLU-19           [-1, 64, 56, 56]               0
           Conv2d-20           [-1, 64, 56, 56]          36,864
      BatchNorm2d-21           [-1, 64, 56, 56]             128
             ReLU-22           [-1, 64, 56, 56]               0
           Conv2d-23          [-1, 256, 56, 56]          16,384
      BatchNorm2d-24          [-1, 256, 56, 56]             512
             ReLU-25          [-1, 256, 56, 56]               0
       Bottleneck-26          [-1, 256, 56, 56]               0
           Conv2d-27           [-1, 64, 56, 56]          16,384
      BatchNorm2d-28           [-1, 64, 56, 56]             128
             ReLU-29           [-1, 64, 56, 56]               0
           Conv2d-30           [-1, 64, 56, 56]          36,864
      BatchNorm2d-31           [-1, 64, 56, 56]             128
             ReLU-32           [-1, 64, 56, 56]               0
           Conv2d-33          [-1, 256, 56, 56]          16,384
      BatchNorm2d-34          [-1, 256, 56, 56]             512
             ReLU-35          [-1, 256, 56, 56]               0
       Bottleneck-36          [-1, 256, 56, 56]               0
           Conv2d-37          [-1, 128, 56, 56]          32,768
      BatchNorm2d-38          [-1, 128, 56, 56]             256
             ReLU-39          [-1, 128, 56, 56]               0
           Conv2d-40          [-1, 128, 28, 28]         147,456
      BatchNorm2d-41          [-1, 128, 28, 28]             256
             ReLU-42          [-1, 128, 28, 28]               0
           Conv2d-43          [-1, 512, 28, 28]          65,536
      BatchNorm2d-44          [-1, 512, 28, 28]           1,024
           Conv2d-45          [-1, 512, 28, 28]         131,072
      BatchNorm2d-46          [-1, 512, 28, 28]           1,024
             ReLU-47          [-1, 512, 28, 28]               0
       Bottleneck-48          [-1, 512, 28, 28]               0
           Conv2d-49          [-1, 128, 28, 28]          65,536
      BatchNorm2d-50          [-1, 128, 28, 28]             256
             ReLU-51          [-1, 128, 28, 28]               0
           Conv2d-52          [-1, 128, 28, 28]         147,456
      BatchNorm2d-53          [-1, 128, 28, 28]             256
             ReLU-54          [-1, 128, 28, 28]               0
           Conv2d-55          [-1, 512, 28, 28]          65,536
      BatchNorm2d-56          [-1, 512, 28, 28]           1,024
             ReLU-57          [-1, 512, 28, 28]               0
       Bottleneck-58          [-1, 512, 28, 28]               0
           Conv2d-59          [-1, 128, 28, 28]          65,536
      BatchNorm2d-60          [-1, 128, 28, 28]             256
             ReLU-61          [-1, 128, 28, 28]               0
           Conv2d-62          [-1, 128, 28, 28]         147,456
      BatchNorm2d-63          [-1, 128, 28, 28]             256
```

```
             ReLU-64          [-1, 128, 28, 28]               0
           Conv2d-65          [-1, 512, 28, 28]          65,536
      BatchNorm2d-66          [-1, 512, 28, 28]           1,024
             ReLU-67          [-1, 512, 28, 28]               0
       Bottleneck-68          [-1, 512, 28, 28]               0
           Conv2d-69          [-1, 128, 28, 28]          65,536
      BatchNorm2d-70          [-1, 128, 28, 28]             256
             ReLU-71          [-1, 128, 28, 28]               0
           Conv2d-72          [-1, 128, 28, 28]         147,456
      BatchNorm2d-73          [-1, 128, 28, 28]             256
             ReLU-74          [-1, 128, 28, 28]               0
           Conv2d-75          [-1, 512, 28, 28]          65,536
      BatchNorm2d-76          [-1, 512, 28, 28]           1,024
             ReLU-77          [-1, 512, 28, 28]               0
       Bottleneck-78          [-1, 512, 28, 28]               0
           Conv2d-79          [-1, 256, 28, 28]         131,072
      BatchNorm2d-80          [-1, 256, 28, 28]             512
             ReLU-81          [-1, 256, 28, 28]               0
           Conv2d-82          [-1, 256, 14, 14]         589,824
      BatchNorm2d-83          [-1, 256, 14, 14]             512
             ReLU-84          [-1, 256, 14, 14]               0
           Conv2d-85         [-1, 1024, 14, 14]         262,144
      BatchNorm2d-86         [-1, 1024, 14, 14]           2,048
           Conv2d-87         [-1, 1024, 14, 14]         524,288
      BatchNorm2d-88         [-1, 1024, 14, 14]           2,048
             ReLU-89         [-1, 1024, 14, 14]               0
       Bottleneck-90         [-1, 1024, 14, 14]               0
           Conv2d-91          [-1, 256, 14, 14]         262,144
      BatchNorm2d-92          [-1, 256, 14, 14]             512
             ReLU-93          [-1, 256, 14, 14]               0
           Conv2d-94          [-1, 256, 14, 14]         589,824
      BatchNorm2d-95          [-1, 256, 14, 14]             512
             ReLU-96          [-1, 256, 14, 14]               0
           Conv2d-97         [-1, 1024, 14, 14]         262,144
      BatchNorm2d-98         [-1, 1024, 14, 14]           2,048
             ReLU-99         [-1, 1024, 14, 14]               0
      Bottleneck-100         [-1, 1024, 14, 14]               0
          Conv2d-101          [-1, 256, 14, 14]         262,144
     BatchNorm2d-102          [-1, 256, 14, 14]             512
            ReLU-103          [-1, 256, 14, 14]               0
          Conv2d-104          [-1, 256, 14, 14]         589,824
     BatchNorm2d-105          [-1, 256, 14, 14]             512
            ReLU-106          [-1, 256, 14, 14]               0
          Conv2d-107         [-1, 1024, 14, 14]         262,144
     BatchNorm2d-108         [-1, 1024, 14, 14]           2,048
            ReLU-109         [-1, 1024, 14, 14]               0
      Bottleneck-110         [-1, 1024, 14, 14]               0
          Conv2d-111          [-1, 256, 14, 14]         262,144
     BatchNorm2d-112          [-1, 256, 14, 14]             512
            ReLU-113          [-1, 256, 14, 14]               0
          Conv2d-114          [-1, 256, 14, 14]         589,824
     BatchNorm2d-115          [-1, 256, 14, 14]             512
            ReLU-116          [-1, 256, 14, 14]               0
          Conv2d-117         [-1, 1024, 14, 14]         262,144
     BatchNorm2d-118         [-1, 1024, 14, 14]           2,048
            ReLU-119         [-1, 1024, 14, 14]               0
      Bottleneck-120         [-1, 1024, 14, 14]               0
          Conv2d-121          [-1, 256, 14, 14]         262,144
     BatchNorm2d-122          [-1, 256, 14, 14]             512
            ReLU-123          [-1, 256, 14, 14]               0
          Conv2d-124          [-1, 256, 14, 14]         589,824
     BatchNorm2d-125          [-1, 256, 14, 14]             512
            ReLU-126          [-1, 256, 14, 14]               0
          Conv2d-127         [-1, 1024, 14, 14]         262,144
     BatchNorm2d-128         [-1, 1024, 14, 14]           2,048
            ReLU-129         [-1, 1024, 14, 14]               0
      Bottleneck-130         [-1, 1024, 14, 14]               0
          Conv2d-131          [-1, 256, 14, 14]         262,144
     BatchNorm2d-132          [-1, 256, 14, 14]             512
            ReLU-133          [-1, 256, 14, 14]               0
          Conv2d-134          [-1, 256, 14, 14]         589,824
     BatchNorm2d-135          [-1, 256, 14, 14]             512
```

```
            ReLU-136          [-1, 256, 14, 14]               0
          Conv2d-137         [-1, 1024, 14, 14]         262,144
     BatchNorm2d-138         [-1, 1024, 14, 14]           2,048
            ReLU-139         [-1, 1024, 14, 14]               0
      Bottleneck-140         [-1, 1024, 14, 14]               0
          Conv2d-141          [-1, 512, 14, 14]         524,288
     BatchNorm2d-142          [-1, 512, 14, 14]           1,024
            ReLU-143          [-1, 512, 14, 14]               0
          Conv2d-144            [-1, 512, 7, 7]       2,359,296
     BatchNorm2d-145            [-1, 512, 7, 7]           1,024
            ReLU-146            [-1, 512, 7, 7]               0
          Conv2d-147           [-1, 2048, 7, 7]       1,048,576
     BatchNorm2d-148           [-1, 2048, 7, 7]           4,096
          Conv2d-149           [-1, 2048, 7, 7]       2,097,152
     BatchNorm2d-150           [-1, 2048, 7, 7]           4,096
            ReLU-151           [-1, 2048, 7, 7]               0
      Bottleneck-152           [-1, 2048, 7, 7]               0
          Conv2d-153            [-1, 512, 7, 7]       1,048,576
     BatchNorm2d-154            [-1, 512, 7, 7]           1,024
            ReLU-155            [-1, 512, 7, 7]               0
          Conv2d-156            [-1, 512, 7, 7]       2,359,296
     BatchNorm2d-157            [-1, 512, 7, 7]           1,024
            ReLU-158            [-1, 512, 7, 7]               0
          Conv2d-159           [-1, 2048, 7, 7]       1,048,576
     BatchNorm2d-160           [-1, 2048, 7, 7]           4,096
            ReLU-161           [-1, 2048, 7, 7]               0
      Bottleneck-162           [-1, 2048, 7, 7]               0
          Conv2d-163            [-1, 512, 7, 7]       1,048,576
     BatchNorm2d-164            [-1, 512, 7, 7]           1,024
            ReLU-165            [-1, 512, 7, 7]               0
          Conv2d-166            [-1, 512, 7, 7]       2,359,296
     BatchNorm2d-167            [-1, 512, 7, 7]           1,024
            ReLU-168            [-1, 512, 7, 7]               0
          Conv2d-169           [-1, 2048, 7, 7]       1,048,576
     BatchNorm2d-170           [-1, 2048, 7, 7]           4,096
            ReLU-171           [-1, 2048, 7, 7]               0
      Bottleneck-172           [-1, 2048, 7, 7]               0
AdaptiveAvgPool2d-173           [-1, 2048, 1, 1]               0
          Linear-174                    [-1, 5]          10,245
================================================================
Total params: 23,518,277
Trainable params: 23,518,277
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 286.55
Params size (MB): 89.72
Estimated Total Size (MB): 376.84
----------------------------------------------------------------
```

In [22]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_resnet,test_loss_resnet,accuracy_resnet = training_and_validation_loop(epochs,
step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_loader,best_acc,b
est_model_wts,'resnet50')
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
0 1.5865631103515625
10 1.4026012420654297
20 0.24139684438705444
Done Training
Epoch:  0 Train Loss:  0.9044700637459755 Test Loss:  0.9265706514318784 Accuracy:  70.29
702970297029
```

```
702970297029
Saved model with accuracy:  70.29702970297029
0 0.728845477104187
10 0.13556507229804993
20 0.6039993762969971
Done Training
Epoch:  1 Train Loss:  0.4641397901667425 Test Loss:  0.48482512434323627 Accuracy:  86.1
3861386138613
Saved model with accuracy:  86.13861386138613
0 0.1928005963563919
10 0.37228062748908997
20 0.28592565655708313
Done Training
Epoch:  2 Train Loss:  0.40839955081733376 Test Loss:  0.2526685843137481 Accuracy:  91.0
8910891089108
Saved model with accuracy:  91.08910891089108
0 0.11323396116495132
10 0.46861761808395386
20 0.12560990452766418
Done Training
Epoch:  3 Train Loss:  0.4709880407899618 Test Loss:  0.40851516493906576 Accuracy:  91.0
8910891089108
0 0.5878835916519165
10 0.05155441164970398
20 0.3235367238521576
Done Training
Epoch:  4 Train Loss:  0.2708692201055013 Test Loss:  0.24884696785981456 Accuracy:  94.0
5940594059406
Saved model with accuracy:  94.05940594059406
0 0.0858665257692337
10 0.007001958787441254
20 0.002611076459288597
Done Training
Epoch:  5 Train Loss:  0.06206498868190325 Test Loss:  0.1191311440585802 Accuracy:  98.0
1980198019803
Saved model with accuracy:  98.01980198019803
0 0.020420847460627556
10 0.009400611743330956
20 0.03128553181886673
Done Training
Epoch:  6 Train Loss:  0.06428567153544953 Test Loss:  0.12623501926039657 Accuracy:  96.
03960396039604
0 0.0031262750271707773
10 0.004796699155122042
20 0.2517113983631134
Done Training
Epoch:  7 Train Loss:  0.05369600997969078 Test Loss:  0.1328670463602369 Accuracy:  97.0
2970297029702
0 0.004860566463321447
10 0.0026574586518108845
20 0.018294580280780792
Done Training
Epoch:  8 Train Loss:  0.034289274096059114 Test Loss:  0.12322042199472587 Accuracy:  97
.02970297029702
0 0.005496365949511528
10 0.025362126529216766
20 0.0020803108345717119
Done Training
Epoch:  9 Train Loss:  0.051317634022920035 Test Loss:  0.08581632406761248 Accuracy:  98
.01980198019803
```

Densenet (121) Model Training and Validation

In [23]:

```
gpu_flag = torch.cuda.is_available()

#preloading Resnet18
model = models.densenet121(pretrained = True)
```

```
#append a new last layer
model.fc = nn.Linear(1024,num_classes)

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
Downloading: "https://download.pytorch.org/models/densenet121-a639ec97.pth" to /root/.cac
he/torch/hub/checkpoints/densenet121-a639ec97.pth
```

```
True
DenseNet(
  (features): Sequential(
    (conv0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (norm0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    (relu0): ReLU(inplace=True)
    (pool0): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (denseblock1): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer2): _DenseLayer(
        (norm1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(96, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer3): _DenseLayer(
        (norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer4): _DenseLayer(
        (norm1): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(160, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer5): _DenseLayer(
        (norm1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer6): _DenseLayer(
        (norm1): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(224, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
    )
    (transition1): _Transition(
      (norm): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
      (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )
    (denseblock2): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer2): _DenseLayer(
        (norm1): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(160, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer3): _DenseLayer(
        (norm1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer4): _DenseLayer(
        (norm1): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(224, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer5): _DenseLayer(
        (norm1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer6): _DenseLayer(
        (norm1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(288, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer7): _DenseLayer(
        (norm1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(320, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer8): _DenseLayer(
        (norm1): BatchNorm2d(352, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(352, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer9): _DenseLayer(
        (norm1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer10): _DenseLayer(
        (norm1): BatchNorm2d(416, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(416, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer11): _DenseLayer(
        (norm1): BatchNorm2d(448, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(448, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer12): _DenseLayer(
        (norm1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(480, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
    )
    (transition2): _Transition(
      (norm): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
      (conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )
    (denseblock3): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer2): _DenseLayer(
        (norm1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(288, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer3): _DenseLayer(
        (norm1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(320, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer4): _DenseLayer(
        (norm1): BatchNorm2d(352, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(352, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer5): _DenseLayer(
        (norm1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer6): _DenseLayer(
        (norm1): BatchNorm2d(416, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(416, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer7): _DenseLayer(
        (norm1): BatchNorm2d(448, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(448, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer8): _DenseLayer(
        (norm1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(480, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer9): _DenseLayer(
        (norm1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer10): _DenseLayer(
        (norm1): BatchNorm2d(544, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(544, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer11): _DenseLayer(
        (norm1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(576, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer12): _DenseLayer(
        (norm1): BatchNorm2d(608, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(608, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer13): _DenseLayer(
        (norm1): BatchNorm2d(640, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(640, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer14): _DenseLayer(
        (norm1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(672, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer15): _DenseLayer(
        (norm1): BatchNorm2d(704, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(704, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer16): _DenseLayer(
        (norm1): BatchNorm2d(736, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(736, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer17): _DenseLayer(
        (norm1): BatchNorm2d(768, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
```

```
      )
      (denselayer18): _DenseLayer(
        (norm1): BatchNorm2d(800, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(800, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer19): _DenseLayer(
        (norm1): BatchNorm2d(832, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(832, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer20): _DenseLayer(
        (norm1): BatchNorm2d(864, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(864, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer21): _DenseLayer(
        (norm1): BatchNorm2d(896, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(896, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer22): _DenseLayer(
        (norm1): BatchNorm2d(928, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(928, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer23): _DenseLayer(
        (norm1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(960, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer24): _DenseLayer(
        (norm1): BatchNorm2d(992, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(992, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
    )
    (transition3): _Transition(
      (norm): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats
=True)
      (relu): ReLU(inplace=True)
      (conv): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )
    (denseblock4): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer2): _DenseLayer(
        (norm1): BatchNorm2d(544, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(544, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer3): _DenseLayer(
        (norm1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(576, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer4): _DenseLayer(
        (norm1): BatchNorm2d(608, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(608, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer5): _DenseLayer(
        (norm1): BatchNorm2d(640, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(640, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer6): _DenseLayer(
        (norm1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(672, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer7): _DenseLayer(
        (norm1): BatchNorm2d(704, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(704, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer8): _DenseLayer(
        (norm1): BatchNorm2d(736, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(736, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer9): _DenseLayer(
        (norm1): BatchNorm2d(768, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer10): _DenseLayer(
        (norm1): BatchNorm2d(800, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(800, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer11): _DenseLayer(
        (norm1): BatchNorm2d(832, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(832, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer12): _DenseLayer(
        (norm1): BatchNorm2d(864, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(864, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer13): _DenseLayer(
        (norm1): BatchNorm2d(896, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(896, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer14): _DenseLayer(
        (norm1): BatchNorm2d(928, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(928, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer15): _DenseLayer(
        (norm1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(960, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
      (denselayer16): _DenseLayer(
        (norm1): BatchNorm2d(992, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(992, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_sta
ts=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=
False)
      )
    )
    (norm5): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
  )
  (classifier): Linear(in_features=1024, out_features=1000, bias=True)
  (fc): Linear(in_features=1024, out_features=5, bias=True)
)
```

Summary of how an example image (224,224,3) is processed through the model-pipleine

In [29]:

```
#!pip install torch-summary==1.4.4
```

```
Requirement already satisfied: torch-summary==1.4.4 in /usr/local/lib/python3.7/dist-pack
ages (1.4.4)
```

In [30]:

```
#from torchsummary import summary
summary(model,(3,224,224) )
```

```
=========================================================================================
=
Layer (type:depth-idx)                   Output Shape              Param #
=========================================================================================
=
├─Sequential: 1-1                        [-1, 1024, 7, 7]          --
|    └─Conv2d: 2-1                       [-1, 64, 112, 112]        9,408
|    └─BatchNorm2d: 2-2                  [-1, 64, 112, 112]        128
|    └─ReLU: 2-3                         [-1, 64, 112, 112]        --
|    └─MaxPool2d: 2-4                    [-1, 64, 56, 56]          --
|    └─_DenseBlock: 2-5                  [-1, 256, 56, 56]         --
|    |    └─_DenseLayer: 3-1             [-1, 32, 56, 56]          45,440
|    |    └─_DenseLayer: 3-2             [-1, 32, 56, 56]          49,600
|    |    └─_DenseLayer: 3-3             [-1, 32, 56, 56]          53,760
|    |    └─_DenseLayer: 3-4             [-1, 32, 56, 56]          57,920
|    |    └─_DenseLayer: 3-5             [-1, 32, 56, 56]          62,080
|    |    └─_DenseLayer: 3-6             [-1, 32, 56, 56]          66,240
|    └─_Transition: 2-6                  [-1, 128, 28, 28]         --
|    |    └─BatchNorm2d: 3-7             [-1, 256, 56, 56]         512
|    |    └─ReLU: 3-8                    [-1, 256, 56, 56]         --
|    |    └─Conv2d: 3-9                  [-1, 128, 56, 56]         32,768
|    |    └─AvgPool2d: 3-10              [-1, 128, 28, 28]         --
|    └─_DenseBlock: 2-7                  [-1, 512, 28, 28]         --
|    |    └─_DenseLayer: 3-11            [-1, 32, 28, 28]          53,760
|    |    └─_DenseLayer: 3-12            [-1, 32, 28, 28]          57,920
|    |    └─_DenseLayer: 3-13            [-1, 32, 28, 28]          62,080
|    |    └─_DenseLayer: 3-14            [-1, 32, 28, 28]          66,240
|    |    └─_DenseLayer: 3-15            [-1, 32, 28, 28]          70,400
|    |    └─_DenseLayer: 3-16            [-1, 32, 28, 28]          74,560
|    |    └─_DenseLayer: 3-17            [-1, 32, 28, 28]          78,720
|    |    └─_DenseLayer: 3-18            [-1, 32, 28, 28]          82,880
|    |    └─_DenseLayer: 3-19            [-1, 32, 28, 28]          87,040
|    |    └─_DenseLayer: 3-20            [-1, 32, 28, 28]          91,200
|    |    └─_DenseLayer: 3-21            [-1, 32, 28, 28]          95,360
|    |    └─_DenseLayer: 3-22            [-1, 32, 28, 28]          99,520
|    └─_Transition: 2-8                  [-1, 256, 14, 14]         --
|    |    └─BatchNorm2d: 3-23            [-1, 512, 28, 28]         1,024
|    |    └─ReLU: 3-24                   [-1, 512, 28, 28]         --
|    |    └─Conv2d: 3-25                 [-1, 256, 28, 28]         131,072
|    |    └─AvgPool2d: 3-26              [-1, 256, 14, 14]         --
|    └─_DenseBlock: 2-9                  [-1, 1024, 14, 14]        --
|    |    └─_DenseLayer: 3-27            [-1, 32, 14, 14]          70,400
|    |    └─_DenseLayer: 3-28            [-1, 32, 14, 14]          74,560
|    |    └─_DenseLayer: 3-29            [-1, 32, 14, 14]          78,720
|    |    └─_DenseLayer: 3-30            [-1, 32, 14, 14]          82,880
|    |    └─_DenseLayer: 3-31            [-1, 32, 14, 14]          87,040
|    |    └─_DenseLayer: 3-32            [-1, 32, 14, 14]          91,200
|    |    └─_DenseLayer: 3-33            [-1, 32, 14, 14]          95,360
|    |    └─_DenseLayer: 3-34            [-1, 32, 14, 14]          99,520
|    |    └─_DenseLayer: 3-35            [-1, 32, 14, 14]          103,680
|    |    └─_DenseLayer: 3-36            [-1, 32, 14, 14]          107,840
|    |    └─_DenseLayer: 3-37            [-1, 32, 14, 14]          112,000
|    |    └─_DenseLayer: 3-38            [-1, 32, 14, 14]          116,160
|    |    └─_DenseLayer: 3-39            [-1, 32, 14, 14]          120,320
|    |    └─_DenseLayer: 3-40            [-1, 32, 14, 14]          124,480
|    |    └─_DenseLayer: 3-41            [-1, 32, 14, 14]          128,640
|    |    └─_DenseLayer: 3-42            [-1, 32, 14, 14]          132,800
|    |    └─_DenseLayer: 3-43            [-1, 32, 14, 14]          136,960
|    |    └─_DenseLayer: 3-44            [-1, 32, 14, 14]          141,120
|    |    └─_DenseLayer: 3-45            [-1, 32, 14, 14]          145,280
|    |    └─_DenseLayer: 3-46            [-1, 32, 14, 14]          149,440
|    |    └─_DenseLayer: 3-47            [-1, 32, 14, 14]          153,600
|    |    └─_DenseLayer: 3-48            [-1, 32, 14, 14]          157,760
|    |    └─_DenseLayer: 3-49            [-1, 32, 14, 14]          161,920
|    |    └─_DenseLayer: 3-50            [-1, 32, 14, 14]          166,080
|    └─_Transition: 2-10                 [-1, 512, 7, 7]           --
|    |    └─BatchNorm2d: 3-51            [-1, 1024, 14, 14]        2,048
|    |    └─ReLU: 3-52                   [-1, 1024, 14, 14]        --
|    |    └─Conv2d: 3-53                 [-1, 512, 14, 14]         524,288
|    |    └─AvgPool2d: 3-54              [-1, 512, 7, 7]           --
|    └─_DenseBlock: 2-11                 [-1, 1024, 7, 7]          --
|    |    └─_DenseLayer: 3-55            [-1, 32, 7, 7]            103,680
|    |    └─_DenseLayer: 3-56            [-1, 32, 7, 7]            107,840
|    |    └─_DenseLayer: 3-57            [-1, 32, 7, 7]            112,000
|    |    └─_DenseLayer: 3-58            [-1, 32, 7, 7]            116,160
|    |    └─_DenseLayer: 3-59            [-1, 32, 7, 7]            120,320
|    |    └─_DenseLayer: 3-60            [-1, 32, 7, 7]            124,480
|    |    └─_DenseLayer: 3-61            [-1, 32, 7, 7]            128,640
|    |    └─_DenseLayer: 3-62            [-1, 32, 7, 7]            132,800
|    |    └─_DenseLayer: 3-63            [-1, 32, 7, 7]            136,960
|    |    └─_DenseLayer: 3-64            [-1, 32, 7, 7]            141,120
|    |    └─_DenseLayer: 3-65            [-1, 32, 7, 7]            145,280
|    |    └─_DenseLayer: 3-66            [-1, 32, 7, 7]            149,440
|    |    └─_DenseLayer: 3-67            [-1, 32, 7, 7]            153,600
|    |    └─_DenseLayer: 3-68            [-1, 32, 7, 7]            157,760
|    |    └─_DenseLayer: 3-69            [-1, 32, 7, 7]            161,920
|    |    └─_DenseLayer: 3-70            [-1, 32, 7, 7]            166,080
|    └─BatchNorm2d: 2-12                 [-1, 1024, 7, 7]          2,048
├─Linear: 1-2                            [-1, 1000]                1,025,000
=========================================================================================
=
Total params: 7,978,856
Trainable params: 7,978,856
Non-trainable params: 0
Total mult-adds (G): 2.85
=========================================================================================
=
Input size (MB): 0.57
Forward/backward pass size (MB): 172.18
Params size (MB): 30.44
Estimated Total Size (MB): 203.19
=========================================================================================
=
```

Out[30]:

```
=========================================================================================
=
Layer (type:depth-idx)                   Output Shape              Param #
=========================================================================================
=
├─Sequential: 1-1                        [-1, 1024, 7, 7]          --
|    └─Conv2d: 2-1                       [-1, 64, 112, 112]        9,408
|    └─BatchNorm2d: 2-2                  [-1, 64, 112, 112]        128
|    └─ReLU: 2-3                         [-1, 64, 112, 112]        --
|    └─MaxPool2d: 2-4                    [-1, 64, 56, 56]          --
|    └─_DenseBlock: 2-5                  [-1, 256, 56, 56]         --
|    |    └─_DenseLayer: 3-1             [-1, 32, 56, 56]          45,440
|    |    └─_DenseLayer: 3-2             [-1, 32, 56, 56]          49,600
|    |    └─_DenseLayer: 3-3             [-1, 32, 56, 56]          53,760
|    |    └─_DenseLayer: 3-4             [-1, 32, 56, 56]          57,920
|    |    └─_DenseLayer: 3-5             [-1, 32, 56, 56]          62,080
|    |    └─_DenseLayer: 3-6             [-1, 32, 56, 56]          66,240
|    └─_Transition: 2-6                  [-1, 128, 28, 28]         --
|    |    └─BatchNorm2d: 3-7             [-1, 256, 56, 56]         512
|    |    └─ReLU: 3-8                    [-1, 256, 56, 56]         --
|    |    └─Conv2d: 3-9                  [-1, 128, 56, 56]         32,768
|    |    └─AvgPool2d: 3-10              [-1, 128, 28, 28]         --
|    └─_DenseBlock: 2-7                  [-1, 512, 28, 28]         --
|    |    └─_DenseLayer: 3-11            [-1, 32, 28, 28]          53,760
|    |    └─_DenseLayer: 3-12            [-1, 32, 28, 28]          57,920
|    |    └─_DenseLayer: 3-13            [-1, 32, 28, 28]          62,080
|    |    └─_DenseLayer: 3-14            [-1, 32, 28, 28]          66,240
|    |    └─_DenseLayer: 3-15            [-1, 32, 28, 28]          70,400
|    |    └─_DenseLayer: 3-16            [-1, 32, 28, 28]          74,560
|    |    └─_DenseLayer: 3-17            [-1, 32, 28, 28]          78,720
|    |    └─_DenseLayer: 3-18            [-1, 32, 28, 28]          82,880
|    |    └─_DenseLayer: 3-19            [-1, 32, 28, 28]          87,040
|    |    └─_DenseLayer: 3-20            [-1, 32, 28, 28]          91,200
|    |    └─_DenseLayer: 3-21            [-1, 32, 28, 28]          95,360
|    |    └─_DenseLayer: 3-22            [-1, 32, 28, 28]          99,520
|    └─_Transition: 2-8                  [-1, 256, 14, 14]         --
|    |    └─BatchNorm2d: 3-23            [-1, 512, 28, 28]         1,024
|    |    └─ReLU: 3-24                   [-1, 512, 28, 28]         --
|    |    └─Conv2d: 3-25                 [-1, 256, 28, 28]         131,072
|    |    └─AvgPool2d: 3-26              [-1, 256, 14, 14]         --
|    └─_DenseBlock: 2-9                  [-1, 1024, 14, 14]        --
|    |    └─_DenseLayer: 3-27            [-1, 32, 14, 14]          70,400
|    |    └─_DenseLayer: 3-28            [-1, 32, 14, 14]          74,560
|    |    └─_DenseLayer: 3-29            [-1, 32, 14, 14]          78,720
|    |    └─_DenseLayer: 3-30            [-1, 32, 14, 14]          82,880
|    |    └─_DenseLayer: 3-31            [-1, 32, 14, 14]          87,040
|    |    └─_DenseLayer: 3-32            [-1, 32, 14, 14]          91,200
|    |    └─_DenseLayer: 3-33            [-1, 32, 14, 14]          95,360
|    |    └─_DenseLayer: 3-34            [-1, 32, 14, 14]          99,520
|    |    └─_DenseLayer: 3-35            [-1, 32, 14, 14]          103,680
|    |    └─_DenseLayer: 3-36            [-1, 32, 14, 14]          107,840
|    |    └─_DenseLayer: 3-37            [-1, 32, 14, 14]          112,000
|    |    └─_DenseLayer: 3-38            [-1, 32, 14, 14]          116,160
|    |    └─_DenseLayer: 3-39            [-1, 32, 14, 14]          120,320
|    |    └─_DenseLayer: 3-40            [-1, 32, 14, 14]          124,480
|    |    └─_DenseLayer: 3-41            [-1, 32, 14, 14]          128,640
|    |    └─_DenseLayer: 3-42            [-1, 32, 14, 14]          132,800
|    |    └─_DenseLayer: 3-43            [-1, 32, 14, 14]          136,960
|    |    └─_DenseLayer: 3-44            [-1, 32, 14, 14]          141,120
|    |    └─_DenseLayer: 3-45            [-1, 32, 14, 14]          145,280
|    |    └─_DenseLayer: 3-46            [-1, 32, 14, 14]          149,440
|    |    └─_DenseLayer: 3-47            [-1, 32, 14, 14]          153,600
|    |    └─_DenseLayer: 3-48            [-1, 32, 14, 14]          157,760
|    |    └─_DenseLayer: 3-49            [-1, 32, 14, 14]          161,920
|    |    └─_DenseLayer: 3-50            [-1, 32, 14, 14]          166,080
|    └─_Transition: 2-10                 [-1, 512, 7, 7]           --
|    |    └─BatchNorm2d: 3-51            [-1, 1024, 14, 14]        2,048
|    |    └─ReLU: 3-52                   [-1, 1024, 14, 14]        --
|    |    └─Conv2d: 3-53                 [-1, 512, 14, 14]         524,288
|    |    └─AvgPool2d: 3-54              [-1, 512, 7, 7]           --
|    └─_DenseBlock: 2-11                 [-1, 1024, 7, 7]          --
|    |    └─_DenseLayer: 3-55            [-1, 32, 7, 7]            103,680
|    |    └─_DenseLayer: 3-56            [-1, 32, 7, 7]            107,840
|    |    └─_DenseLayer: 3-57            [-1, 32, 7, 7]            112,000
|    |    └─_DenseLayer: 3-58            [-1, 32, 7, 7]            116,160
|    |    └─_DenseLayer: 3-59            [-1, 32, 7, 7]            120,320
|    |    └─_DenseLayer: 3-60            [-1, 32, 7, 7]            124,480
|    |    └─_DenseLayer: 3-61            [-1, 32, 7, 7]            128,640
|    |    └─_DenseLayer: 3-62            [-1, 32, 7, 7]            132,800
|    |    └─_DenseLayer: 3-63            [-1, 32, 7, 7]            136,960
|    |    └─_DenseLayer: 3-64            [-1, 32, 7, 7]            141,120
|    |    └─_DenseLayer: 3-65            [-1, 32, 7, 7]            145,280
|    |    └─_DenseLayer: 3-66            [-1, 32, 7, 7]            149,440
|    |    └─_DenseLayer: 3-67            [-1, 32, 7, 7]            153,600
|    |    └─_DenseLayer: 3-68            [-1, 32, 7, 7]            157,760
|    |    └─_DenseLayer: 3-69            [-1, 32, 7, 7]            161,920
|    |    └─_DenseLayer: 3-70            [-1, 32, 7, 7]            166,080
|    └─BatchNorm2d: 2-12                 [-1, 1024, 7, 7]          2,048
├─Linear: 1-2                            [-1, 1000]                1,025,000
=========================================================================================
=
Total params: 7,978,856
Trainable params: 7,978,856
Non-trainable params: 0
Total mult-adds (G): 2.85
=========================================================================================
=
Input size (MB): 0.57
Forward/backward pass size (MB): 172.18
Params size (MB): 30.44
Estimated Total Size (MB): 203.19
=========================================================================================
=
```

In [24]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_densenet,test_loss_densenet,accuracy_densenet = training_and_validation_loop(e
pochs,step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_loader,best
_acc,best_model_wts,'densenet121')
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
0 9.138383865356445
10 0.9650443196296692
20 0.06078092008829117
Done Training
Epoch:  0 Train Loss:  2.2466538224655848 Test Loss:  3.3180753191312156 Accuracy:  55.44
5544554455445
Saved model with accuracy:  55.445544554455445
0 0.29421496391296387
10 0.4615996181964874
20 0.7444113492965698
Done Training
Epoch:  1 Train Loss:  0.8245844027170768 Test Loss:  1.306866079568863 Accuracy:  83.168
31683168317
Saved model with accuracy:  83.16831683168317
0 0.5446934103965759
10 0.42699798941612244
20 0.8005310893058777
Done Training
Epoch:  2 Train Loss:  0.6438356200949504 Test Loss:  1.1363329117496808 Accuracy:  77.22
772277227723
0 1.1019139289855957
10 0.17914268374443054
20 0.1420760601758957
Done Training
Epoch:  3 Train Loss:  0.5895436649712232 Test Loss:  0.28949844096011174 Accuracy:  91.0
8910891089108
Saved model with accuracy:  91.08910891089108
0 0.22750571370124817
10 0.31196632981300354
20 0.028994735330343246
Done Training
Epoch:  4 Train Loss:  0.3227866735452643 Test Loss:  0.21168913235790873 Accuracy:  97.0
2970297029702
Saved model with accuracy:  97.02970297029702
0 0.1526409089565277
10 0.027456073090434074
20 0.007800552528351545
Done Training
Epoch:  5 Train Loss:  0.0681473980210005 Test Loss:  0.2579683663789183 Accuracy:  94.05
940594059406
0 0.4365171194076538
10 0.008106556721031666
20 0.0019801075104624033
Done Training
Epoch:  6 Train Loss:  0.21490904450631484 Test Loss:  0.16929724862954268 Accuracy:  97.
02970297029702
0 0.015598079189658165
10 0.04921852424740791
20 0.004458426032215357
Done Training
Epoch:  7 Train Loss:  0.3877580885936578 Test Loss:  0.24885696970159188 Accuracy:  95.0
4950495049505
0 0.0017346213571727276
10 0.09415960311889648
20 0.3547869324684143
Done Training
Epoch:  8 Train Loss:  0.07521977331131123 Test Loss:  0.24324786166350046 Accuracy:  97.
02970297029702
0 0.054559849202632904
10 0.059700291603803635
20 0.01803106814622879
Done Training
Done Training
Epoch:  9 Train Loss:  0.0667133486399857 Test Loss:  0.22480724577811392 Accuracy:  98.0
1980198019803
Saved model with accuracy:  98.01980198019803
```

ShuffleNet Model Training and Validation

In [25]:

```
gpu_flag = torch.cuda.is_available()

#preloading Resnet18
model = models.shufflenet_v2_x1_0(pretrained = True)
#append a new last layer
model.fc = nn.Linear(1024,num_classes)

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
Downloading: "https://download.pytorch.org/models/shufflenetv2_x1-5666bf0f80.pth" to /roo
t/.cache/torch/hub/checkpoints/shufflenetv2_x1-5666bf0f80.pth
```

```
True
ShuffleNetV2(
  (conv1): Sequential(
    (0): Conv2d(3, 24, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (stage2): Sequential(
    (0): InvertedResidual(
      (branch1): Sequential(
        (0): Conv2d(24, 24, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=24,
bias=False)
        (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (2): Conv2d(24, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (4): ReLU(inplace=True)
      )
      (branch2): Sequential(
        (0): Conv2d(24, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (2): ReLU(inplace=True)
        (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=58,
bias=False)
        (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (7): ReLU(inplace=True)
      )
    )
    (1): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (2): ReLU(inplace=True)
        (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=58,
bias=False)
        (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (7): ReLU(inplace=True)
      )
    )
    (2): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (2): ReLU(inplace=True)
        (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=58,
bias=False)
        (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (7): ReLU(inplace=True)
      )
    )
    (3): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (2): ReLU(inplace=True)
        (3): Conv2d(58, 58, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=58,
bias=False)
        (4): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (5): Conv2d(58, 58, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(58, eps=1e-05, momentum=0.1, affine=True, track_running_stats=Tr
ue)
        (7): ReLU(inplace=True)
      )
    )
  )
  (stage3): Sequential(
    (0): InvertedResidual(
      (branch1): Sequential(
        (0): Conv2d(116, 116, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=1
16, bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (4): ReLU(inplace=True)
      )
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
```

```
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (1): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (2): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (3): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (4): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (5): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (6): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (7): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(116, 116, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
16, bias=False)
        (4): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(116, 116, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(116, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
  )
  (stage4): Sequential(
    (0): InvertedResidual(
      (branch1): Sequential(
        (0): Conv2d(232, 232, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=2
32, bias=False)
        (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (3): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (4): ReLU(inplace=True)
      )
      (branch2): Sequential(
        (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=2
32, bias=False)
        (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (1): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=2
32, bias=False)
        (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (2): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=2
32, bias=False)
        (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
    (3): InvertedResidual(
      (branch1): Sequential()
      (branch2): Sequential(
        (0): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (2): ReLU(inplace=True)
        (3): Conv2d(232, 232, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=2
32, bias=False)
        (4): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (5): Conv2d(232, 232, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (6): BatchNorm2d(232, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
        (7): ReLU(inplace=True)
      )
    )
  )
  (conv5): Sequential(
    (0): Conv2d(464, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
)
    (2): ReLU(inplace=True)
  )
  (fc): Linear(in_features=1024, out_features=5, bias=True)
)
```

Summary of how an example image (224,224,3) is processed through the

Summary of how an example image (224,224,3) is processed through the model-pipleine

In [26]:

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

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 24, 112, 112]             648
       BatchNorm2d-2         [-1, 24, 112, 112]              48
              ReLU-3         [-1, 24, 112, 112]               0
         MaxPool2d-4           [-1, 24, 56, 56]               0
            Conv2d-5           [-1, 24, 28, 28]             216
       BatchNorm2d-6           [-1, 24, 28, 28]              48
            Conv2d-7           [-1, 58, 28, 28]           1,392
       BatchNorm2d-8           [-1, 58, 28, 28]             116
              ReLU-9           [-1, 58, 28, 28]               0
           Conv2d-10           [-1, 58, 56, 56]           1,392
      BatchNorm2d-11           [-1, 58, 56, 56]             116
             ReLU-12           [-1, 58, 56, 56]               0
           Conv2d-13           [-1, 58, 28, 28]             522
      BatchNorm2d-14           [-1, 58, 28, 28]             116
           Conv2d-15           [-1, 58, 28, 28]           3,364
      BatchNorm2d-16           [-1, 58, 28, 28]             116
             ReLU-17           [-1, 58, 28, 28]               0
 InvertedResidual-18          [-1, 116, 28, 28]               0
           Conv2d-19           [-1, 58, 28, 28]           3,364
      BatchNorm2d-20           [-1, 58, 28, 28]             116
             ReLU-21           [-1, 58, 28, 28]               0
           Conv2d-22           [-1, 58, 28, 28]             522
      BatchNorm2d-23           [-1, 58, 28, 28]             116
           Conv2d-24           [-1, 58, 28, 28]           3,364
      BatchNorm2d-25           [-1, 58, 28, 28]             116
             ReLU-26           [-1, 58, 28, 28]               0
 InvertedResidual-27          [-1, 116, 28, 28]               0
           Conv2d-28           [-1, 58, 28, 28]           3,364
      BatchNorm2d-29           [-1, 58, 28, 28]             116
             ReLU-30           [-1, 58, 28, 28]               0
           Conv2d-31           [-1, 58, 28, 28]             522
      BatchNorm2d-32           [-1, 58, 28, 28]             116
           Conv2d-33           [-1, 58, 28, 28]           3,364
      BatchNorm2d-34           [-1, 58, 28, 28]             116
             ReLU-35           [-1, 58, 28, 28]               0
 InvertedResidual-36          [-1, 116, 28, 28]               0
           Conv2d-37           [-1, 58, 28, 28]           3,364
      BatchNorm2d-38           [-1, 58, 28, 28]             116
             ReLU-39           [-1, 58, 28, 28]               0
           Conv2d-40           [-1, 58, 28, 28]             522
      BatchNorm2d-41           [-1, 58, 28, 28]             116
           Conv2d-42           [-1, 58, 28, 28]           3,364
      BatchNorm2d-43           [-1, 58, 28, 28]             116
             ReLU-44           [-1, 58, 28, 28]               0
 InvertedResidual-45          [-1, 116, 28, 28]               0
           Conv2d-46          [-1, 116, 14, 14]           1,044
      BatchNorm2d-47          [-1, 116, 14, 14]             232
           Conv2d-48          [-1, 116, 14, 14]          13,456
      BatchNorm2d-49          [-1, 116, 14, 14]             232
             ReLU-50          [-1, 116, 14, 14]               0
           Conv2d-51          [-1, 116, 28, 28]          13,456
      BatchNorm2d-52          [-1, 116, 28, 28]             232
             ReLU-53          [-1, 116, 28, 28]               0
           Conv2d-54          [-1, 116, 14, 14]           1,044
      BatchNorm2d-55          [-1, 116, 14, 14]             232
           Conv2d-56          [-1, 116, 14, 14]          13,456
      BatchNorm2d-57          [-1, 116, 14, 14]             232
             ReLU-58          [-1, 116, 14, 14]               0
 InvertedResidual-59          [-1, 232, 14, 14]               0
           Conv2d-60          [-1, 116, 14, 14]          13,456
      BatchNorm2d-61          [-1, 116, 14, 14]             232
             ReLU-62          [-1, 116, 14, 14]               0
           Conv2d-63          [-1, 116, 14, 14]           1,044
      BatchNorm2d-64          [-1, 116, 14, 14]             232
           Conv2d-65          [-1, 116, 14, 14]          13,456
      BatchNorm2d-66          [-1, 116, 14, 14]             232
             ReLU-67          [-1, 116, 14, 14]               0
 InvertedResidual-68          [-1, 232, 14, 14]               0
           Conv2d-69          [-1, 116, 14, 14]          13,456
      BatchNorm2d-70          [-1, 116, 14, 14]             232
             ReLU-71          [-1, 116, 14, 14]               0
           Conv2d-72          [-1, 116, 14, 14]           1,044
      BatchNorm2d-73          [-1, 116, 14, 14]             232
           Conv2d-74          [-1, 116, 14, 14]          13,456
      BatchNorm2d-75          [-1, 116, 14, 14]             232
             ReLU-76          [-1, 116, 14, 14]               0
 InvertedResidual-77          [-1, 232, 14, 14]               0
           Conv2d-78          [-1, 116, 14, 14]          13,456
      BatchNorm2d-79          [-1, 116, 14, 14]             232
             ReLU-80          [-1, 116, 14, 14]               0
           Conv2d-81          [-1, 116, 14, 14]           1,044
      BatchNorm2d-82          [-1, 116, 14, 14]             232
           Conv2d-83          [-1, 116, 14, 14]          13,456
      BatchNorm2d-84          [-1, 116, 14, 14]             232
             ReLU-85          [-1, 116, 14, 14]               0
 InvertedResidual-86          [-1, 232, 14, 14]               0
           Conv2d-87          [-1, 116, 14, 14]          13,456
      BatchNorm2d-88          [-1, 116, 14, 14]             232
             ReLU-89          [-1, 116, 14, 14]               0
           Conv2d-90          [-1, 116, 14, 14]           1,044
      BatchNorm2d-91          [-1, 116, 14, 14]             232
           Conv2d-92          [-1, 116, 14, 14]          13,456
      BatchNorm2d-93          [-1, 116, 14, 14]             232
             ReLU-94          [-1, 116, 14, 14]               0
 InvertedResidual-95          [-1, 232, 14, 14]               0
           Conv2d-96          [-1, 116, 14, 14]          13,456
      BatchNorm2d-97          [-1, 116, 14, 14]             232
             ReLU-98          [-1, 116, 14, 14]               0
           Conv2d-99          [-1, 116, 14, 14]           1,044
     BatchNorm2d-100          [-1, 116, 14, 14]             232
          Conv2d-101          [-1, 116, 14, 14]          13,456
     BatchNorm2d-102          [-1, 116, 14, 14]             232
            ReLU-103          [-1, 116, 14, 14]               0
InvertedResidual-104          [-1, 232, 14, 14]               0
          Conv2d-105          [-1, 116, 14, 14]          13,456
     BatchNorm2d-106          [-1, 116, 14, 14]             232
            ReLU-107          [-1, 116, 14, 14]               0
          Conv2d-108          [-1, 116, 14, 14]           1,044
     BatchNorm2d-109          [-1, 116, 14, 14]             232
          Conv2d-110          [-1, 116, 14, 14]          13,456
     BatchNorm2d-111          [-1, 116, 14, 14]             232
            ReLU-112          [-1, 116, 14, 14]               0
InvertedResidual-113          [-1, 232, 14, 14]               0
          Conv2d-114          [-1, 116, 14, 14]          13,456
     BatchNorm2d-115          [-1, 116, 14, 14]             232
            ReLU-116          [-1, 116, 14, 14]               0
          Conv2d-117          [-1, 116, 14, 14]           1,044
     BatchNorm2d-118          [-1, 116, 14, 14]             232
          Conv2d-119          [-1, 116, 14, 14]          13,456
     BatchNorm2d-120          [-1, 116, 14, 14]             232
            ReLU-121          [-1, 116, 14, 14]               0
InvertedResidual-122          [-1, 232, 14, 14]               0
          Conv2d-123            [-1, 232, 7, 7]           2,088
     BatchNorm2d-124            [-1, 232, 7, 7]             464
          Conv2d-125            [-1, 232, 7, 7]          53,824
     BatchNorm2d-126            [-1, 232, 7, 7]             464
            ReLU-127            [-1, 232, 7, 7]               0
          Conv2d-128          [-1, 232, 14, 14]          53,824
     BatchNorm2d-129          [-1, 232, 14, 14]             464
            ReLU-130          [-1, 232, 14, 14]               0
          Conv2d-131            [-1, 232, 7, 7]           2,088
     BatchNorm2d-132            [-1, 232, 7, 7]             464
          Conv2d-133            [-1, 232, 7, 7]          53,824
     BatchNorm2d-134            [-1, 232, 7, 7]             464
```

```
            ReLU-135            [-1, 232, 7, 7]               0
InvertedResidual-136            [-1, 464, 7, 7]               0
          Conv2d-137            [-1, 232, 7, 7]          53,824
     BatchNorm2d-138            [-1, 232, 7, 7]             464
            ReLU-139            [-1, 232, 7, 7]               0
          Conv2d-140            [-1, 232, 7, 7]           2,088
     BatchNorm2d-141            [-1, 232, 7, 7]             464
          Conv2d-142            [-1, 232, 7, 7]          53,824
     BatchNorm2d-143            [-1, 232, 7, 7]             464
            ReLU-144            [-1, 232, 7, 7]               0
InvertedResidual-145            [-1, 464, 7, 7]               0
          Conv2d-146            [-1, 232, 7, 7]          53,824
     BatchNorm2d-147            [-1, 232, 7, 7]             464
            ReLU-148            [-1, 232, 7, 7]               0
          Conv2d-149            [-1, 232, 7, 7]           2,088
     BatchNorm2d-150            [-1, 232, 7, 7]             464
          Conv2d-151            [-1, 232, 7, 7]          53,824
     BatchNorm2d-152            [-1, 232, 7, 7]             464
            ReLU-153            [-1, 232, 7, 7]               0
InvertedResidual-154            [-1, 464, 7, 7]               0
          Conv2d-155            [-1, 232, 7, 7]          53,824
     BatchNorm2d-156            [-1, 232, 7, 7]             464
            ReLU-157            [-1, 232, 7, 7]               0
          Conv2d-158            [-1, 232, 7, 7]           2,088
     BatchNorm2d-159            [-1, 232, 7, 7]             464
          Conv2d-160            [-1, 232, 7, 7]          53,824
     BatchNorm2d-161            [-1, 232, 7, 7]             464
            ReLU-162            [-1, 232, 7, 7]               0
InvertedResidual-163            [-1, 464, 7, 7]               0
          Conv2d-164           [-1, 1024, 7, 7]         475,136
     BatchNorm2d-165           [-1, 1024, 7, 7]           2,048
            ReLU-166           [-1, 1024, 7, 7]               0
          Linear-167                    [-1, 5]           5,125
================================================================
Total params: 1,258,729
Trainable params: 1,258,729
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 47.93
Params size (MB): 4.80
Estimated Total Size (MB): 53.31
----------------------------------------------------------------
```

In [27]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_shufflenet,test_loss_shufflenet,accuracy_shufflenet = training_and_validation_
loop(epochs,step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_loade
r,best_acc,best_model_wts,'shufflenet')
```

```
0 1.6090871095657349
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
10 1.6052305698394775
20 1.6109883785247803
Done Training
Epoch:  0 Train Loss:  1.6053229570388794 Test Loss:  1.8703609704971313 Accuracy:  12.87
1287128712872
Saved model with accuracy:  12.871287128712872
0 1.5985110998153687
10 1.5939191579818726
20 1.584482192993164
Done Training
Done Training
Epoch:  1 Train Loss:  1.5943793608592107 Test Loss:  1.8570719559987385 Accuracy:  15.84
1584158415841
Saved model with accuracy:  15.841584158415841
0 1.5969932079315186
10 1.577415943145752
20 1.5746530294418335
Done Training
Epoch:  2 Train Loss:  1.5820250465319707 Test Loss:  1.8494968016942341 Accuracy:  12.87
1287128712872
0 1.5703006982803345
10 1.5678825378417969
20 1.5601204633712769
Done Training
Epoch:  3 Train Loss:  1.5708449758016145 Test Loss:  1.8308024605115254 Accuracy:  13.86
1386138613861
0 1.5614498853683472
10 1.548805832862854
20 1.5916839838027954
Done Training
Epoch:  4 Train Loss:  1.5606456903310924 Test Loss:  1.8245122631390889 Accuracy:  15.84
1584158415841
0 1.5563945770263672
10 1.5644454956054688
20 1.557642936706543
Done Training
Epoch:  5 Train Loss:  1.5527420273193946 Test Loss:  1.82076096534729 Accuracy:  16.8316
83168316832
Saved model with accuracy:  16.831683168316832
0 1.5463088750839233
10 1.5625419616699219
20 1.546794056892395
Done Training
Epoch:  6 Train Loss:  1.5485718708771925 Test Loss:  1.8110153675079346 Accuracy:  26.73
267326732673
Saved model with accuracy:  26.73267326732673
0 1.5512956380844116
10 1.5675294399261475
20 1.538069725036621
Done Training
Epoch:  7 Train Loss:  1.5431613463621874 Test Loss:  1.8043713768323262 Accuracy:  34.65
346534653465
Saved model with accuracy:  34.65346534653465
0 1.5200252532958984
10 1.5225118398666382
20 1.5307083129882812
Done Training
Epoch:  8 Train Loss:  1.5359941124916077 Test Loss:  1.8010853131612141 Accuracy:  48.51
485148514851
Saved model with accuracy:  48.51485148514851
0 1.5498642921447754
10 1.5239590406417847
20 1.5012123584747314
Done Training
Epoch:  9 Train Loss:  1.5304675010534434 Test Loss:  1.7933701872825623 Accuracy:  51.48
514851485149
Saved model with accuracy:  51.48514851485149
```

SqueezeNet Model Training and Validation

In [28]:

```
gpu_flag = torch.cuda.is_available()

#preloading Resnet18
model = models.squeezenet1_1(pretrained = True)
#append a new last layer
#model.fc = nn.Linear(1024,num_classes)

# Freeze training for all layers
```

```
for param in model.features.parameters():
    param.require_grad = False


# num_features = model.classifier[6].in_features
model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1), stride=(1,1))
model.num_classes = num_classes

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
Downloading: "https://download.pytorch.org/models/squeezenet1_1-f364aa15.pth" to /root/.c
ache/torch/hub/checkpoints/squeezenet1_1-f364aa15.pth
```

```
True
SqueezeNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
    (3): Fire(
      (squeeze): Conv2d(64, 16, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(16, 64, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(16, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (4): Fire(
      (squeeze): Conv2d(128, 16, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(16, 64, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(16, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
    (6): Fire(
      (squeeze): Conv2d(128, 32, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(32, 128, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(32, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (7): Fire(
      (squeeze): Conv2d(256, 32, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(32, 128, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(32, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (8): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
    (9): Fire(
      (squeeze): Conv2d(256, 48, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(48, 192, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(48, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3): Conv2d(48, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (10): Fire(
      (squeeze): Conv2d(384, 48, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(48, 192, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(48, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (11): Fire(
      (squeeze): Conv2d(384, 64, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(64, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
    (12): Fire(
      (squeeze): Conv2d(512, 64, kernel_size=(1, 1), stride=(1, 1))
      (squeeze_activation): ReLU(inplace=True)
      (expand1x1): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1))
      (expand1x1_activation): ReLU(inplace=True)
      (expand3x3): Conv2d(64, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (expand3x3_activation): ReLU(inplace=True)
    )
  )
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Conv2d(512, 5, kernel_size=(1, 1), stride=(1, 1))
    (2): ReLU(inplace=True)
    (3): AdaptiveAvgPool2d(output_size=(1, 1))
  )
)
```

Summary of how an example image (224,224,3) is processed through the model-pipleine

In [29]:

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

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 111, 111]           1,792
              ReLU-2         [-1, 64, 111, 111]               0
         MaxPool2d-3           [-1, 64, 55, 55]               0
            Conv2d-4           [-1, 16, 55, 55]           1,040
              ReLU-5           [-1, 16, 55, 55]               0
            Conv2d-6           [-1, 64, 55, 55]           1,088
              ReLU-7           [-1, 64, 55, 55]               0
            Conv2d-8           [-1, 64, 55, 55]           9,280
              ReLU-9           [-1, 64, 55, 55]               0
             Fire-10          [-1, 128, 55, 55]               0
           Conv2d-11           [-1, 16, 55, 55]           2,064
             ReLU-12           [-1, 16, 55, 55]               0
           Conv2d-13           [-1, 64, 55, 55]           1,088
             ReLU-14           [-1, 64, 55, 55]               0
           Conv2d-15           [-1, 64, 55, 55]           9,280
             ReLU-16           [-1, 64, 55, 55]               0
             Fire-17          [-1, 128, 55, 55]               0
        MaxPool2d-18          [-1, 128, 27, 27]               0
           Conv2d-19           [-1, 32, 27, 27]           4,128
             ReLU-20           [-1, 32, 27, 27]               0
           Conv2d-21          [-1, 128, 27, 27]           4,224
             ReLU-22          [-1, 128, 27, 27]               0
           Conv2d-23          [-1, 128, 27, 27]          36,992
             ReLU-24          [-1, 128, 27, 27]               0
             Fire-25          [-1, 256, 27, 27]               0
           Conv2d-26          [-1, 32, 27, 27]           8,224
             ReLU-27          [-1, 32, 27, 27]               0
           Conv2d-28         [-1, 128, 27, 27]           4,224
             ReLU-29         [-1, 128, 27, 27]               0
           Conv2d-30         [-1, 128, 27, 27]          36,992
             ReLU-31         [-1, 128, 27, 27]               0
             Fire-32         [-1, 256, 27, 27]               0
        MaxPool2d-33         [-1, 256, 13, 13]               0
           Conv2d-34          [-1, 48, 13, 13]          12,336
             ReLU-35          [-1, 48, 13, 13]               0
           Conv2d-36         [-1, 192, 13, 13]           9,408
             ReLU-37         [-1, 192, 13, 13]               0
           Conv2d-38         [-1, 192, 13, 13]          83,136
             ReLU-39         [-1, 192, 13, 13]               0
             Fire-40         [-1, 384, 13, 13]               0
           Conv2d-41          [-1, 48, 13, 13]          18,480
             ReLU-42          [-1, 48, 13, 13]               0
           Conv2d-43         [-1, 192, 13, 13]           9,408
             ReLU-44         [-1, 192, 13, 13]               0
           Conv2d-45         [-1, 192, 13, 13]          83,136
             ReLU-46         [-1, 192, 13, 13]               0
             Fire-47         [-1, 384, 13, 13]               0
           Conv2d-48          [-1, 64, 13, 13]          24,640
             ReLU-49          [-1, 64, 13, 13]               0
           Conv2d-50         [-1, 256, 13, 13]          16,640
             ReLU-51         [-1, 256, 13, 13]               0
           Conv2d-52         [-1, 256, 13, 13]         147,712
             ReLU-53         [-1, 256, 13, 13]               0
             Fire-54         [-1, 512, 13, 13]               0
           Conv2d-55          [-1, 64, 13, 13]          32,832
             ReLU-56          [-1, 64, 13, 13]               0
           Conv2d-57         [-1, 256, 13, 13]          16,640
             ReLU-58         [-1, 256, 13, 13]               0
           Conv2d-59         [-1, 256, 13, 13]         147,712
             ReLU-60         [-1, 256, 13, 13]               0
             Fire-61         [-1, 512, 13, 13]               0
          Dropout-62         [-1, 512, 13, 13]               0
           Conv2d-63           [-1, 5, 13, 13]           2,565
             ReLU-64           [-1, 5, 13, 13]               0
AdaptiveAvgPool2d-65             [-1, 5, 1, 1]               0
================================================================
Total params: 725,061
Trainable params: 725,061
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 51.19
Params size (MB): 2.77
Estimated Total Size (MB): 54.53
----------------------------------------------------------------
```

In [30]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_squeezenet,test_loss_squeezenet,accuracy_squeezenet = training_and_validation_loop(epochs,step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_loader,best_acc,best_model_wts,'squeezenet')
```

```
0 2.370865821838379
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Detected call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later, you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step()`.  Failure to do this will result in PyTorch skipping the first value of the learning rate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
10 1.1117057800292969
20 1.5032753944396973
```

```
Done Training
Epoch:  0 Train Loss:  1.348187925723883 Test Loss:  1.1427829662958782 Accuracy:  61.386
138613861384
Saved model with accuracy:  61.386138613861384
0 1.107326626777649
10 0.8771452903747559
20 1.3786516189575195
Done Training
Epoch:  1 Train Loss:  1.0833288568716783 Test Loss:  0.9319753547509512 Accuracy:  70.29
702970297029
Saved model with accuracy:  70.29702970297029
0 1.202566146850586
10 1.6049587726593018
20 1.324761986732483
Done Training
Epoch:  2 Train Loss:  1.0277675848740797 Test Loss:  1.3383516371250153 Accuracy:  44.55
4455445544555
0 1.2520273923873901
10 0.8197417259216309
20 1.0175464153289795
Done Training
Epoch:  3 Train Loss:  1.0174243851349904 Test Loss:  1.7177240663052846 Accuracy:  60.39
6039603960396
0 1.5640711784362793
10 0.664365291595459
20 1.0088777542114258
Done Training
Epoch:  4 Train Loss:  1.0759205222129822 Test Loss:  1.1147720714410145 Accuracy:  64.35
643564356435
0 0.7130188345909119
10 0.9761123657226562
20 0.3531895875930786
Done Training
Epoch:  5 Train Loss:  0.72105640402207 Test Loss:  0.7286671002705892 Accuracy:  78.2178
2178217822
Saved model with accuracy:  78.21782178217822
0 0.3942648470401764
10 0.5363904237747192
20 0.539560079574585
Done Training
Epoch:  6 Train Loss:  0.47219447734264225 Test Loss:  0.4005001311500867 Accuracy:  88.1
1881188118812
Saved model with accuracy:  88.11881188118812
0 0.10826339572668076
10 0.3169958293437958
20 0.6232759356498718
Done Training
Epoch:  7 Train Loss:  0.4527901282104162 Test Loss:  0.435536486407121 Accuracy:  87.128
71287128714
0 0.31860998272895813
10 0.4506802260875702
20 1.015223503112793
Done Training
Epoch:  8 Train Loss:  0.3495920321975763 Test Loss:  0.9637555330991745 Accuracy:  74.25
742574257426
0 0.5802138447761536
10 0.20019979774951935
20 0.2354196310043335
Done Training
Epoch:  9 Train Loss:  0.3119526357891468 Test Loss:  0.31179084690908593 Accuracy:  90.0
990099009901
Saved model with accuracy:  90.0990099009901
```

Mobilenet-V3 Model Training and Validation

In [31]:

```
gpu_flag = torch.cuda.is_available()

#preloading Resnet18
```

```
model = models.mobilenet_v3_large(pretrained = True)
#append a new last layer


# Freeze training for all layers
for param in model.features.parameters():
    param.require_grad = False

model.classifier[3] = nn.Linear(1280,num_classes)
# num_features = model.classifier[6].in_features
#model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1), stride=(1,1))
#model.num_classes = num_classes

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
Downloading: "https://download.pytorch.org/models/mobilenet_v3_large-8738ca79.pth" to /ro
ot/.cache/torch/hub/checkpoints/mobilenet_v3_large-8738ca79.pth
```

```
True
MobileNetV3(
  (features): Sequential(
    (0): ConvBNActivation(
      (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(16, eps=0.001, momentum=0.01, affine=True, track_running_stats=Tru
e)
      (2): Hardswish()
    )
    (1): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1
6, bias=False)
          (1): BatchNorm2d(16, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (1): ConvBNActivation(
          (0): Conv2d(16, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(16, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (2): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(16, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(64, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (1): ConvBNActivation(
          (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=6
4, bias=False)
          (1): BatchNorm2d(64, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (2): ConvBNActivation(
          (0): Conv2d(64, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(24, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (3): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(24, 72, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(72, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (1): ConvBNActivation(
          (0): Conv2d(72, 72, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=7
2, bias=False)
          (1): BatchNorm2d(72, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (2): ConvBNActivation(
          (0): Conv2d(72, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(24, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (4): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(24, 72, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(72, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (1): ConvBNActivation(
          (0): Conv2d(72, 72, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), groups=7
2, bias=False)
          (1): BatchNorm2d(72, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): ReLU(inplace=True)
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(72, 24, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(24, 72, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(72, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(40, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (5): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(40, 120, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(120, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): ReLU(inplace=True)
        )
        (1): ConvBNActivation(
          (0): Conv2d(120, 120, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups
=120, bias=False)
          (1): BatchNorm2d(120, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
```

```
          (2): ReLU(inplace=True)
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(120, 32, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(32, 120, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(120, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(40, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (6): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(40, 120, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(120, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): ReLU(inplace=True)
        )
        (1): ConvBNActivation(
          (0): Conv2d(120, 120, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups
=120, bias=False)
          (1): BatchNorm2d(120, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): ReLU(inplace=True)
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(120, 32, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(32, 120, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(120, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(40, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (7): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(240, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(240, 240, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups
=240, bias=False)
          (1): BatchNorm2d(240, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): ConvBNActivation(
          (0): Conv2d(240, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (8): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(80, 200, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(200, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
```

```
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(200, 200, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=200, bias=False)
          (1): BatchNorm2d(200, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): ConvBNActivation(
          (0): Conv2d(200, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (9): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(80, 184, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(184, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(184, 184, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=184, bias=False)
          (1): BatchNorm2d(184, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): ConvBNActivation(
          (0): Conv2d(184, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (10): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(80, 184, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(184, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(184, 184, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=184, bias=False)
          (1): BatchNorm2d(184, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): ConvBNActivation(
          (0): Conv2d(184, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(80, eps=0.001, momentum=0.01, affine=True, track_running_stats
=True)
          (2): Identity()
        )
      )
    )
    (11): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=480, bias=False)
          (1): BatchNorm2d(480, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(480, 120, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(120, 480, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(480, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(112, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Identity()
        )
      )
    )
    (12): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(672, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(672, 672, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=672, bias=False)
          (1): BatchNorm2d(672, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(672, 168, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(168, 672, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(112, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Identity()
        )
      )
    )
    (13): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(672, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), groups
=672, bias=False)
          (1): BatchNorm2d(672, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(672, 168, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(168, 672, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(672, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Identity()
          (2): Identity()
        )
      )
    )
    (14): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(960, 960, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups
=960, bias=False)
          (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(960, 240, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(240, 960, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Identity()
        )
      )
    )
    (15): InvertedResidual(
      (block): Sequential(
        (0): ConvBNActivation(
          (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (1): ConvBNActivation(
          (0): Conv2d(960, 960, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups
=960, bias=False)
          (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Hardswish()
        )
        (2): SqueezeExcitation(
          (fc1): Conv2d(960, 240, kernel_size=(1, 1), stride=(1, 1))
          (relu): ReLU(inplace=True)
          (fc2): Conv2d(240, 960, kernel_size=(1, 1), stride=(1, 1))
        )
        (3): ConvBNActivation(
          (0): Conv2d(960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(160, eps=0.001, momentum=0.01, affine=True, track_running_stat
s=True)
          (2): Identity()
        )
      )
    )
    (16): ConvBNActivation(
      (0): Conv2d(160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(960, eps=0.001, momentum=0.01, affine=True, track_running_stats=Tr
ue)
      (2): Hardswish()
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=1)
  (classifier): Sequential(
    (0): Linear(in_features=960, out_features=1280, bias=True)
    (1): Hardswish()
    (2): Dropout(p=0.2, inplace=True)
    (3): Linear(in_features=1280, out_features=5, bias=True)
  )
)
```

Summary of how an example image (224,224,3) is processed through the model-pipleine

In [32]:

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

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 112, 112]             432
       BatchNorm2d-2         [-1, 16, 112, 112]              32
         Hardswish-3         [-1, 16, 112, 112]               0
            Conv2d-4         [-1, 16, 112, 112]             144
       BatchNorm2d-5         [-1, 16, 112, 112]              32
              ReLU-6         [-1, 16, 112, 112]               0
            Conv2d-7         [-1, 16, 112, 112]             256
       BatchNorm2d-8         [-1, 16, 112, 112]              32
          Identity-9         [-1, 16, 112, 112]               0
 InvertedResidual-10         [-1, 16, 112, 112]               0
           Conv2d-11         [-1, 64, 112, 112]           1,024
      BatchNorm2d-12         [-1, 64, 112, 112]             128
             ReLU-13         [-1, 64, 112, 112]               0
           Conv2d-14           [-1, 64, 56, 56]             576
      BatchNorm2d-15           [-1, 64, 56, 56]             128
             ReLU-16           [-1, 64, 56, 56]               0
           Conv2d-17           [-1, 24, 56, 56]           1,536
      BatchNorm2d-18           [-1, 24, 56, 56]              48
         Identity-19           [-1, 24, 56, 56]               0
 InvertedResidual-20           [-1, 24, 56, 56]               0
           Conv2d-21           [-1, 72, 56, 56]           1,728
      BatchNorm2d-22           [-1, 72, 56, 56]             144
             ReLU-23           [-1, 72, 56, 56]               0
           Conv2d-24           [-1, 72, 56, 56]             648
      BatchNorm2d-25           [-1, 72, 56, 56]             144
             ReLU-26           [-1, 72, 56, 56]               0
           Conv2d-27           [-1, 24, 56, 56]           1,728
      BatchNorm2d-28           [-1, 24, 56, 56]              48
         Identity-29           [-1, 24, 56, 56]               0
 InvertedResidual-30           [-1, 24, 56, 56]               0
           Conv2d-31           [-1, 72, 56, 56]           1,728
      BatchNorm2d-32           [-1, 72, 56, 56]             144
             ReLU-33           [-1, 72, 56, 56]               0
           Conv2d-34           [-1, 72, 28, 28]           1,800
      BatchNorm2d-35           [-1, 72, 28, 28]             144
             ReLU-36           [-1, 72, 28, 28]               0
           Conv2d-37             [-1, 24, 1, 1]           1,752
             ReLU-38             [-1, 24, 1, 1]               0
           Conv2d-39             [-1, 72, 1, 1]           1,800
SqueezeExcitation-40           [-1, 72, 28, 28]               0
           Conv2d-41           [-1, 40, 28, 28]           2,880
      BatchNorm2d-42           [-1, 40, 28, 28]              80
         Identity-43           [-1, 40, 28, 28]               0
 InvertedResidual-44           [-1, 40, 28, 28]               0
           Conv2d-45          [-1, 120, 28, 28]           4,800
      BatchNorm2d-46          [-1, 120, 28, 28]             240
             ReLU-47          [-1, 120, 28, 28]               0
           Conv2d-48          [-1, 120, 28, 28]           3,000
      BatchNorm2d-49          [-1, 120, 28, 28]             240
             ReLU-50          [-1, 120, 28, 28]               0
           Conv2d-51             [-1, 32, 1, 1]           3,872
             ReLU-52             [-1, 32, 1, 1]               0
           Conv2d-53            [-1, 120, 1, 1]           3,960
SqueezeExcitation-54          [-1, 120, 28, 28]               0
           Conv2d-55           [-1, 40, 28, 28]           4,800
      BatchNorm2d-56           [-1, 40, 28, 28]              80
         Identity-57           [-1, 40, 28, 28]               0
 InvertedResidual-58           [-1, 40, 28, 28]               0
           Conv2d-59          [-1, 120, 28, 28]           4,800
      BatchNorm2d-60          [-1, 120, 28, 28]             240
             ReLU-61          [-1, 120, 28, 28]               0
           Conv2d-62          [-1, 120, 28, 28]           3,000
      BatchNorm2d-63          [-1, 120, 28, 28]             240
             ReLU-64          [-1, 120, 28, 28]               0
           Conv2d-65             [-1, 32, 1, 1]           3,872
             ReLU-66             [-1, 32, 1, 1]               0
           Conv2d-67            [-1, 120, 1, 1]           3,960
SqueezeExcitation-68          [-1, 120, 28, 28]               0
           Conv2d-69           [-1, 40, 28, 28]           4,800
      BatchNorm2d-70           [-1, 40, 28, 28]              80
         Identity-71           [-1, 40, 28, 28]               0
 InvertedResidual-72           [-1, 40, 28, 28]               0
           Conv2d-73          [-1, 240, 28, 28]           9,600
      BatchNorm2d-74          [-1, 240, 28, 28]             480
        Hardswish-75          [-1, 240, 28, 28]               0
           Conv2d-76          [-1, 240, 14, 14]           2,160
      BatchNorm2d-77          [-1, 240, 14, 14]             480
        Hardswish-78          [-1, 240, 14, 14]               0
           Conv2d-79           [-1, 80, 14, 14]          19,200
      BatchNorm2d-80           [-1, 80, 14, 14]             160
         Identity-81           [-1, 80, 14, 14]               0
 InvertedResidual-82           [-1, 80, 14, 14]               0
           Conv2d-83          [-1, 200, 14, 14]          16,000
      BatchNorm2d-84          [-1, 200, 14, 14]             400
        Hardswish-85          [-1, 200, 14, 14]               0
           Conv2d-86          [-1, 200, 14, 14]           1,800
      BatchNorm2d-87          [-1, 200, 14, 14]             400
        Hardswish-88          [-1, 200, 14, 14]               0
           Conv2d-89           [-1, 80, 14, 14]          16,000
      BatchNorm2d-90           [-1, 80, 14, 14]             160
         Identity-91           [-1, 80, 14, 14]               0
 InvertedResidual-92           [-1, 80, 14, 14]               0
           Conv2d-93          [-1, 184, 14, 14]          14,720
      BatchNorm2d-94          [-1, 184, 14, 14]             368
        Hardswish-95          [-1, 184, 14, 14]               0
           Conv2d-96          [-1, 184, 14, 14]           1,656
      BatchNorm2d-97          [-1, 184, 14, 14]             368
        Hardswish-98          [-1, 184, 14, 14]               0
           Conv2d-99           [-1, 80, 14, 14]          14,720
     BatchNorm2d-100           [-1, 80, 14, 14]             160
        Identity-101           [-1, 80, 14, 14]               0
InvertedResidual-102           [-1, 80, 14, 14]               0
          Conv2d-103          [-1, 184, 14, 14]          14,720
     BatchNorm2d-104          [-1, 184, 14, 14]             368
       Hardswish-105          [-1, 184, 14, 14]               0
          Conv2d-106          [-1, 184, 14, 14]           1,656
     BatchNorm2d-107          [-1, 184, 14, 14]             368
       Hardswish-108          [-1, 184, 14, 14]               0
          Conv2d-109           [-1, 80, 14, 14]          14,720
     BatchNorm2d-110           [-1, 80, 14, 14]             160
        Identity-111           [-1, 80, 14, 14]               0
InvertedResidual-112           [-1, 80, 14, 14]               0
          Conv2d-113          [-1, 480, 14, 14]          38,400
     BatchNorm2d-114          [-1, 480, 14, 14]             960
       Hardswish-115          [-1, 480, 14, 14]               0
          Conv2d-116          [-1, 480, 14, 14]           4,320
     BatchNorm2d-117          [-1, 480, 14, 14]             960
       Hardswish-118          [-1, 480, 14, 14]               0
          Conv2d-119            [-1, 120, 1, 1]          57,720
            ReLU-120            [-1, 120, 1, 1]               0
          Conv2d-121            [-1, 480, 1, 1]          58,080
SqueezeExcitation-122          [-1, 480, 14, 14]               0
          Conv2d-123          [-1, 112, 14, 14]          53,760
     BatchNorm2d-124          [-1, 112, 14, 14]             224
        Identity-125          [-1, 112, 14, 14]               0
InvertedResidual-126          [-1, 112, 14, 14]               0
          Conv2d-127          [-1, 672, 14, 14]          75,264
     BatchNorm2d-128          [-1, 672, 14, 14]           1,344
       Hardswish-129          [-1, 672, 14, 14]               0
```

```
          Conv2d-130         [-1, 672, 14, 14]           6,048
     BatchNorm2d-131         [-1, 672, 14, 14]           1,344
       Hardswish-132         [-1, 672, 14, 14]               0
          Conv2d-133           [-1, 168, 1, 1]         113,064
            ReLU-134           [-1, 168, 1, 1]               0
          Conv2d-135           [-1, 672, 1, 1]         113,568
SqueezeExcitation-136         [-1, 672, 14, 14]               0
          Conv2d-137         [-1, 112, 14, 14]          75,264
     BatchNorm2d-138         [-1, 112, 14, 14]             224
        Identity-139         [-1, 112, 14, 14]               0
InvertedResidual-140         [-1, 112, 14, 14]               0
          Conv2d-141         [-1, 672, 14, 14]          75,264
     BatchNorm2d-142         [-1, 672, 14, 14]           1,344
       Hardswish-143         [-1, 672, 14, 14]               0
          Conv2d-144           [-1, 672, 7, 7]          16,800
     BatchNorm2d-145           [-1, 672, 7, 7]           1,344
       Hardswish-146           [-1, 672, 7, 7]               0
          Conv2d-147           [-1, 168, 1, 1]         113,064
            ReLU-148           [-1, 168, 1, 1]               0
          Conv2d-149           [-1, 672, 1, 1]         113,568
SqueezeExcitation-150           [-1, 672, 7, 7]               0
          Conv2d-151           [-1, 160, 7, 7]         107,520
     BatchNorm2d-152           [-1, 160, 7, 7]             320
        Identity-153           [-1, 160, 7, 7]               0
InvertedResidual-154           [-1, 160, 7, 7]               0
          Conv2d-155           [-1, 960, 7, 7]         153,600
     BatchNorm2d-156           [-1, 960, 7, 7]           1,920
       Hardswish-157           [-1, 960, 7, 7]               0
          Conv2d-158           [-1, 960, 7, 7]          24,000
     BatchNorm2d-159           [-1, 960, 7, 7]           1,920
       Hardswish-160           [-1, 960, 7, 7]               0
          Conv2d-161           [-1, 240, 1, 1]         230,640
            ReLU-162           [-1, 240, 1, 1]               0
          Conv2d-163           [-1, 960, 1, 1]         231,360
SqueezeExcitation-164           [-1, 960, 7, 7]               0
          Conv2d-165           [-1, 160, 7, 7]         153,600
     BatchNorm2d-166           [-1, 160, 7, 7]             320
        Identity-167           [-1, 160, 7, 7]               0
InvertedResidual-168           [-1, 160, 7, 7]               0
          Conv2d-169           [-1, 960, 7, 7]         153,600
     BatchNorm2d-170           [-1, 960, 7, 7]           1,920
       Hardswish-171           [-1, 960, 7, 7]               0
          Conv2d-172           [-1, 960, 7, 7]          24,000
     BatchNorm2d-173           [-1, 960, 7, 7]           1,920
       Hardswish-174           [-1, 960, 7, 7]               0
          Conv2d-175           [-1, 240, 1, 1]         230,640
            ReLU-176           [-1, 240, 1, 1]               0
          Conv2d-177           [-1, 960, 1, 1]         231,360
SqueezeExcitation-178           [-1, 960, 7, 7]               0
          Conv2d-179           [-1, 160, 7, 7]         153,600
     BatchNorm2d-180           [-1, 160, 7, 7]             320
        Identity-181           [-1, 160, 7, 7]               0
InvertedResidual-182           [-1, 160, 7, 7]               0
          Conv2d-183           [-1, 960, 7, 7]         153,600
     BatchNorm2d-184           [-1, 960, 7, 7]           1,920
       Hardswish-185           [-1, 960, 7, 7]               0
AdaptiveAvgPool2d-186           [-1, 960, 1, 1]               0
          Linear-187                [-1, 1280]       1,230,080
       Hardswish-188                [-1, 1280]               0
         Dropout-189                [-1, 1280]               0
          Linear-190                   [-1, 5]           6,405
================================================================
Total params: 4,208,437
Trainable params: 4,208,437
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 109.74
Params size (MB): 16.05
Estimated Total Size (MB): 126.36
----------------------------------------------------------------
```

In [33]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_mobilenetv3,test_loss_mobilenetv3,accuracy_mobilenetv3 = training_and_validati
on_loop(epochs,step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_lo
ader,best_acc,best_model_wts,'mobilenetv3')
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
0 1.7044938802719116
10 0.8560001850128174
20 0.6709185838699341
Done Training
Epoch:  0 Train Loss:  0.9067165920367608 Test Loss:  0.6518133531014124 Accuracy:  87.12
871287128714
Saved model with accuracy:  87.12871287128714
0 0.23527871072292328
10 0.120726577937603
20 0.13641296327114105
Done Training
Epoch:  1 Train Loss:  0.19453917006747082 Test Loss:  0.3477473681171735 Accuracy:  91.0
8910891089108
Saved model with accuracy:  91.08910891089108
0 0.08034613728523254
10 0.11464028060436249
20 0.0319516621530056
Done Training
Epoch:  2 Train Loss:  0.09244648238214162 Test Loss:  0.9670339624087015 Accuracy:  64.3
5643564356435
0 0.02152995392680168
10 0.06685855239629745
20 0.14557349681854248
Done Training
Epoch:  3 Train Loss:  0.06261978070968045 Test Loss:  1.0001689592997234 Accuracy:  65.3
4653465346534
0 0.05367603525519371
10 0.013469028286635876
20 0.010422355495393276
Done Training
Epoch:  4 Train Loss:  0.036673401339122884 Test Loss:  0.615285669763883 Accuracy:  81.1
8811881188118
0 0.009298174642026424
10 0.008970660157501698
20 0.005466326139867306
Done Training
Epoch:  5 Train Loss:  0.04260593713619388 Test Loss:  0.490286427239577 Accuracy:  84.15
841584158416
0 0.024849897250533104
10 0.008556940592825413
20 0.005676905158907175
Done Training
Epoch:  6 Train Loss:  0.029832732845814183 Test Loss:  0.4429765964547793 Accuracy:  90.
0990099009901
0 0.07684990018606186
10 0.014088056050240993
20 0.020855512470006943
Done Training
Epoch:  7 Train Loss:  0.08120686394753508 Test Loss:  0.2988235444451372 Accuracy:  94.0
5940594059406
Saved model with accuracy:  94.05940594059406
0 0.004049375653266907
10 0.01076989434659481
20 0.0017321212217211723
Done Training
```

```
Epoch:  8 Train Loss:  0.04226264553681876 Test Loss:  0.19566461816430092 Accuracy:  95.
04950495049505
Saved model with accuracy:  95.04950495049505
0 0.005631010048091412
10 0.010514802299439907
20 0.00878622941672802
Done Training
Epoch:  9 Train Loss:  0.060993892971265055 Test Loss:  0.18364583087774614 Accuracy:  94
.05940594059406
```

Resnext50 Model Training and Validation

In [34]:

```
gpu_flag = torch.cuda.is_available()

#preloading Resnet18
model = models.resnext50_32x4d(pretrained = True)
#append a new last layer


# Freeze training for all layers
#for param in model.features.parameters():
#    param.require_grad = False

model.fc = nn.Linear(2048,num_classes)
# num_features = model.classifier[6].in_features
#model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1), stride=(1,1))
#model.num_classes = num_classes

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
Downloading: "https://download.pytorch.org/models/resnext50_32x4d-7cdf4587.pth" to /root/
.cache/torch/hub/checkpoints/resnext50_32x4d-7cdf4587.pth
```

```
True
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
  )
  (layer2): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(256, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (relu): ReLU(inplace=True)
    )
  )
  (layer3): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (3): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (4): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups
=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=T
rue)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), grou
ps=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), grou
ps=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), grou
ps=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=
True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=5, bias=True)
)
```

Summary of how an example image (224,224,3) is processed through the model-pipleine

In [35]:

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

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 112, 112]           9,408
       BatchNorm2d-2         [-1, 64, 112, 112]             128
              ReLU-3         [-1, 64, 112, 112]               0
         MaxPool2d-4           [-1, 64, 56, 56]               0
            Conv2d-5          [-1, 128, 56, 56]           8,192
       BatchNorm2d-6          [-1, 128, 56, 56]             256
              ReLU-7          [-1, 128, 56, 56]               0
            Conv2d-8          [-1, 128, 56, 56]           4,608
       BatchNorm2d-9          [-1, 128, 56, 56]             256
             ReLU-10          [-1, 128, 56, 56]               0
           Conv2d-11          [-1, 256, 56, 56]          32,768
      BatchNorm2d-12          [-1, 256, 56, 56]             512
           Conv2d-13          [-1, 256, 56, 56]          16,384
      BatchNorm2d-14          [-1, 256, 56, 56]             512
             ReLU-15          [-1, 256, 56, 56]               0
       Bottleneck-16          [-1, 256, 56, 56]               0
           Conv2d-17          [-1, 128, 56, 56]          32,768
      BatchNorm2d-18          [-1, 128, 56, 56]             256
             ReLU-19          [-1, 128, 56, 56]               0
           Conv2d-20          [-1, 128, 56, 56]           4,608
      BatchNorm2d-21          [-1, 128, 56, 56]             256
             ReLU-22          [-1, 128, 56, 56]               0
           Conv2d-23          [-1, 256, 56, 56]          32,768
      BatchNorm2d-24          [-1, 256, 56, 56]             512
             ReLU-25          [-1, 256, 56, 56]               0
       Bottleneck-26          [-1, 256, 56, 56]               0
           Conv2d-27          [-1, 128, 56, 56]          32,768
      BatchNorm2d-28          [-1, 128, 56, 56]             256
             ReLU-29          [-1, 128, 56, 56]               0
           Conv2d-30          [-1, 128, 56, 56]           4,608
      BatchNorm2d-31          [-1, 128, 56, 56]             256
             ReLU-32          [-1, 128, 56, 56]               0
           Conv2d-33          [-1, 256, 56, 56]          32,768
      BatchNorm2d-34          [-1, 256, 56, 56]             512
             ReLU-35          [-1, 256, 56, 56]               0
       Bottleneck-36          [-1, 256, 56, 56]               0
           Conv2d-37          [-1, 256, 56, 56]          65,536
      BatchNorm2d-38          [-1, 256, 56, 56]             512
             ReLU-39          [-1, 256, 56, 56]               0
           Conv2d-40          [-1, 256, 28, 28]          18,432
      BatchNorm2d-41          [-1, 256, 28, 28]             512
             ReLU-42          [-1, 256, 28, 28]               0
           Conv2d-43          [-1, 512, 28, 28]         131,072
      BatchNorm2d-44          [-1, 512, 28, 28]           1,024
           Conv2d-45          [-1, 512, 28, 28]         131,072
      BatchNorm2d-46          [-1, 512, 28, 28]           1,024
             ReLU-47          [-1, 512, 28, 28]               0
       Bottleneck-48          [-1, 512, 28, 28]               0
           Conv2d-49          [-1, 256, 28, 28]         131,072
      BatchNorm2d-50          [-1, 256, 28, 28]             512
             ReLU-51          [-1, 256, 28, 28]               0
           Conv2d-52          [-1, 256, 28, 28]          18,432
      BatchNorm2d-53          [-1, 256, 28, 28]             512
             ReLU-54          [-1, 256, 28, 28]               0
           Conv2d-55          [-1, 512, 28, 28]         131,072
      BatchNorm2d-56          [-1, 512, 28, 28]           1,024
             ReLU-57          [-1, 512, 28, 28]               0
       Bottleneck-58          [-1, 512, 28, 28]               0
           Conv2d-59          [-1, 256, 28, 28]         131,072
      BatchNorm2d-60          [-1, 256, 28, 28]             512
             ReLU-61          [-1, 256, 28, 28]               0
           Conv2d-62          [-1, 256, 28, 28]          18,432
      BatchNorm2d-63          [-1, 256, 28, 28]             512
             ReLU-64          [-1, 256, 28, 28]               0
           Conv2d-65          [-1, 512, 28, 28]         131,072
      BatchNorm2d-66          [-1, 512, 28, 28]           1,024
             ReLU-67          [-1, 512, 28, 28]               0
       Bottleneck-68          [-1, 512, 28, 28]               0
           Conv2d-69          [-1, 256, 28, 28]         131,072
      BatchNorm2d-70          [-1, 256, 28, 28]             512
             ReLU-71          [-1, 256, 28, 28]               0
           Conv2d-72          [-1, 256, 28, 28]          18,432
      BatchNorm2d-73          [-1, 256, 28, 28]             512
             ReLU-74          [-1, 256, 28, 28]               0
           Conv2d-75          [-1, 512, 28, 28]         131,072
      BatchNorm2d-76          [-1, 512, 28, 28]           1,024
             ReLU-77          [-1, 512, 28, 28]               0
       Bottleneck-78          [-1, 512, 28, 28]               0
           Conv2d-79          [-1, 512, 28, 28]         262,144
      BatchNorm2d-80          [-1, 512, 28, 28]           1,024
             ReLU-81          [-1, 512, 28, 28]               0
           Conv2d-82          [-1, 512, 14, 14]          73,728
      BatchNorm2d-83          [-1, 512, 14, 14]           1,024
             ReLU-84          [-1, 512, 14, 14]               0
           Conv2d-85         [-1, 1024, 14, 14]         524,288
      BatchNorm2d-86         [-1, 1024, 14, 14]           2,048
           Conv2d-87         [-1, 1024, 14, 14]         524,288
      BatchNorm2d-88         [-1, 1024, 14, 14]           2,048
             ReLU-89         [-1, 1024, 14, 14]               0
       Bottleneck-90         [-1, 1024, 14, 14]               0
           Conv2d-91          [-1, 512, 14, 14]         524,288
      BatchNorm2d-92          [-1, 512, 14, 14]           1,024
             ReLU-93          [-1, 512, 14, 14]               0
           Conv2d-94          [-1, 512, 14, 14]          73,728
      BatchNorm2d-95          [-1, 512, 14, 14]           1,024
             ReLU-96          [-1, 512, 14, 14]               0
           Conv2d-97         [-1, 1024, 14, 14]         524,288
      BatchNorm2d-98         [-1, 1024, 14, 14]           2,048
             ReLU-99         [-1, 1024, 14, 14]               0
      Bottleneck-100         [-1, 1024, 14, 14]               0
          Conv2d-101          [-1, 512, 14, 14]         524,288
     BatchNorm2d-102          [-1, 512, 14, 14]           1,024
            ReLU-103          [-1, 512, 14, 14]               0
          Conv2d-104          [-1, 512, 14, 14]          73,728
     BatchNorm2d-105          [-1, 512, 14, 14]           1,024
            ReLU-106          [-1, 512, 14, 14]               0
          Conv2d-107         [-1, 1024, 14, 14]         524,288
     BatchNorm2d-108         [-1, 1024, 14, 14]           2,048
            ReLU-109         [-1, 1024, 14, 14]               0
      Bottleneck-110         [-1, 1024, 14, 14]               0
          Conv2d-111          [-1, 512, 14, 14]         524,288
     BatchNorm2d-112          [-1, 512, 14, 14]           1,024
            ReLU-113          [-1, 512, 14, 14]               0
          Conv2d-114          [-1, 512, 14, 14]          73,728
     BatchNorm2d-115          [-1, 512, 14, 14]           1,024
            ReLU-116          [-1, 512, 14, 14]               0
          Conv2d-117         [-1, 1024, 14, 14]         524,288
     BatchNorm2d-118         [-1, 1024, 14, 14]           2,048
            ReLU-119         [-1, 1024, 14, 14]               0
      Bottleneck-120         [-1, 1024, 14, 14]               0
          Conv2d-121          [-1, 512, 14, 14]         524,288
     BatchNorm2d-122          [-1, 512, 14, 14]           1,024
            ReLU-123          [-1, 512, 14, 14]               0
          Conv2d-124          [-1, 512, 14, 14]          73,728
     BatchNorm2d-125          [-1, 512, 14, 14]           1,024
            ReLU-126          [-1, 512, 14, 14]               0
          Conv2d-127         [-1, 1024, 14, 14]         524,288
     BatchNorm2d-128         [-1, 1024, 14, 14]           2,048
            ReLU-129         [-1, 1024, 14, 14]               0
      Bottleneck-130         [-1, 1024, 14, 14]               0
          Conv2d-131          [-1, 512, 14, 14]         524,288
     BatchNorm2d-132          [-1, 512, 14, 14]           1,024
            ReLU-133          [-1, 512, 14, 14]               0
          Conv2d-134          [-1, 512, 14, 14]          73,728
     BatchNorm2d-135          [-1, 512, 14, 14]           1,024
            ReLU-136          [-1, 512, 14, 14]               0
          Conv2d-137         [-1, 1024, 14, 14]         524,288
     BatchNorm2d-138         [-1, 1024, 14, 14]           2,048
            ReLU-139         [-1, 1024, 14, 14]               0
      Bottleneck-140         [-1, 1024, 14, 14]               0
          Conv2d-141         [-1, 1024, 14, 14]       1,048,576
     BatchNorm2d-142         [-1, 1024, 14, 14]           2,048
            ReLU-143         [-1, 1024, 14, 14]               0
          Conv2d-144           [-1, 1024, 7, 7]         294,912
     BatchNorm2d-145           [-1, 1024, 7, 7]           2,048
            ReLU-146           [-1, 1024, 7, 7]               0
          Conv2d-147           [-1, 2048, 7, 7]       2,097,152
     BatchNorm2d-148           [-1, 2048, 7, 7]           4,096
          Conv2d-149           [-1, 2048, 7, 7]       2,097,152
     BatchNorm2d-150           [-1, 2048, 7, 7]           4,096
            ReLU-151           [-1, 2048, 7, 7]               0
      Bottleneck-152           [-1, 2048, 7, 7]               0
          Conv2d-153           [-1, 1024, 7, 7]       2,097,152
     BatchNorm2d-154           [-1, 1024, 7, 7]           2,048
            ReLU-155           [-1, 1024, 7, 7]               0
          Conv2d-156           [-1, 1024, 7, 7]         294,912
     BatchNorm2d-157           [-1, 1024, 7, 7]           2,048
            ReLU-158           [-1, 1024, 7, 7]               0
          Conv2d-159           [-1, 2048, 7, 7]       2,097,152
     BatchNorm2d-160           [-1, 2048, 7, 7]           4,096
            ReLU-161           [-1, 2048, 7, 7]               0
      Bottleneck-162           [-1, 2048, 7, 7]               0
          Conv2d-163           [-1, 1024, 7, 7]       2,097,152
     BatchNorm2d-164           [-1, 1024, 7, 7]           2,048
            ReLU-165           [-1, 1024, 7, 7]               0
          Conv2d-166           [-1, 1024, 7, 7]         294,912
     BatchNorm2d-167           [-1, 1024, 7, 7]           2,048
            ReLU-168           [-1, 1024, 7, 7]               0
          Conv2d-169           [-1, 2048, 7, 7]       2,097,152
     BatchNorm2d-170           [-1, 2048, 7, 7]           4,096
            ReLU-171           [-1, 2048, 7, 7]               0
      Bottleneck-172           [-1, 2048, 7, 7]               0
AdaptiveAvgPool2d-173           [-1, 2048, 1, 1]               0
          Linear-174                    [-1, 5]          10,245
================================================================
Total params: 22,990,149
Trainable params: 22,990,149
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 361.77
Params size (MB): 87.70
Estimated Total Size (MB): 450.05
----------------------------------------------------------------
```

In [36]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_resnext50,test_loss_resnext50,accuracy_resnext50 = training_and_validation_loop(epochs,step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_loader,best_acc,best_model_wts,'resnext50')
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Detected call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
0 1.602941870689392
10 1.0839041471481323
20 0.28791990876197815
Done Training
Epoch:  0 Train Loss:  0.9075254436868888 Test Loss:  0.5383091866970062 Accuracy:  83.16
831683168317
Saved model with accuracy:  83.16831683168317
0 0.042323045432567596
10 0.6951889991760254
20 1.3334811925888062
Done Training
Epoch:  1 Train Loss:  0.3572772375236337 Test Loss:  0.7578728844722112 Accuracy:  81.18
811881188118
0 0.2989128828048706
10 0.21760797500610352
20 0.6661592721939087
Done Training
Epoch:  2 Train Loss:  0.4727291588922246 Test Loss:  1.5387743314107258 Accuracy:  76.23
762376237623
0 0.7490621209144592
10 0.09535327553749084
20 0.5054001212120056
Done Training
Epoch:  3 Train Loss:  0.491560452671435 Test Loss:  0.385557191252398 Accuracy:  91.0891
0891089108
Saved model with accuracy:  91.08910891089108
0 0.03432445973157883
10 0.3373035788536072
20 0.01228652335703373
Done Training
Epoch:  4 Train Loss:  0.23032332039796388 Test Loss:  0.3899754925320546 Accuracy:  91.0
8910891089108
0 0.030821867287158966
10 0.09242334961891174
20 0.06880614161491394
Done Training
Epoch:  5 Train Loss:  0.1384196400266284 Test Loss:  0.15067613179174563 Accuracy:  97.0
2970297029702
Saved model with accuracy:  97.02970297029702
0 0.004101904574781656
10 0.03237317502498627
20 0.11073403060436249
Done Training
Epoch:  6 Train Loss:  0.11684230749065486 Test Loss:  0.18141781887970865 Accuracy:  95.
04950495049505
0 0.00604900112375617
10 0.01307608187198639
20 0.13812753558158875
Done Training
Epoch:  7 Train Loss:  0.15135580154422384 Test Loss:  0.22845622423725823 Accuracy:  98.
01980198019803
Saved model with accuracy:  98.01980198019803
0 0.009679892100393772
10 0.0070915380492806435
20 0.007535798475146294
Done Training
Epoch:  8 Train Loss:  0.04722554238441472 Test Loss:  0.15797216346254572 Accuracy:  97.
02970297029702
0 0.09250009059906006
10 0.008186898194253445
20 0.013614282011985779
Done Training
Epoch:  9 Train Loss:  0.10282710103694206 Test Loss:  0.1499561067127312 Accuracy:  95.0
4950495049505
```

In [37]:

In [37]:

```
#Pre processing the data
normalize = transforms.Normalize(mean = [0.485,0.456,0.406],
                                std = [0.229,0.224,0.225])
resize = transforms.Resize((299,299))



preprocessor = transforms.Compose([ resize, transforms.ToTensor(), normalize
                                   ])

aerial_dataset_full = get_sat_data(aerial_path,preprocessor)

# Creating data indices for training and validation splits:
dataset_size = len(aerial_dataset_full)
indices = list(range(dataset_size))
validation_split = 0.2
split = int(np.floor(validation_split * dataset_size))
shuffle_dataset = True
random_seed= 101

if shuffle_dataset :
    np.random.seed(random_seed)
    np.random.shuffle(indices)
train_indices, val_indices = indices[split:], indices[:split]

# Creating PT data samplers and loaders:
train_sampler = SubsetRandomSampler(train_indices)
valid_sampler = SubsetRandomSampler(val_indices)

aerial_train_loader = torch.utils.data.DataLoader(aerial_dataset_full, batch_size=16,
                                           sampler=train_sampler)
aerial_validation_loader = torch.utils.data.DataLoader(aerial_dataset_full, batch_size=16
,
                                                sampler=valid_sampler)



gpu_flag = torch.cuda.is_available()

#preloading Resnet18
model = models.inception_v3(pretrained = True)
#append a new last layer


# Freeze training for all layers
#for param in model.features.parameters():
#    param.require_grad = False

model.fc = nn.Linear(2048,num_classes)
# num_features = model.classifier[6].in_features
#model.classifier[1] = nn.Conv2d(512, num_classes, kernel_size=(1,1), stride=(1,1))
#model.num_classes = num_classes
model.aux_logits=False

# define loss function
criterion = nn.CrossEntropyLoss()

# setup SGD
optimizer = torch.optim.SGD(model.parameters(), lr=0.004, momentum=0.9)

step_lr_scheduler = lr_scheduler.StepLR(optimizer,5,.3)

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()

print(model)
```

```
Downloading: "https://download.pytorch.org/models/inception_v3_google-1a9a5a14.pth" to /r
oot/.cache/torch/hub/checkpoints/inception_v3_google-1a9a5a14.pth
```

```
True
Inception3(
  (Conv2d_1a_3x3): BasicConv2d(
    (conv): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_2a_3x3): BasicConv2d(
    (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_2b_3x3): BasicConv2d(
    (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (maxpool1): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (Conv2d_3b_1x1): BasicConv2d(
    (conv): Conv2d(64, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(80, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_4a_3x3): BasicConv2d(
    (conv): Conv2d(80, 192, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True
)
  )
  (maxpool2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (Mixed_5b): InceptionA(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch5x5_1): BasicConv2d(
      (conv): Conv2d(192, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(48, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch5x5_2): BasicConv2d(
      (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), bias=Fals
e)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_3): BasicConv2d(
      (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
  )
  (Mixed_5c): InceptionA(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    )
    (branch5x5_1): BasicConv2d(
      (conv): Conv2d(256, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(48, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch5x5_2): BasicConv2d(
      (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), bias=Fals
e)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_3): BasicConv2d(
      (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
  )
  (Mixed_5d): InceptionA(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch5x5_1): BasicConv2d(
      (conv): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(48, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch5x5_2): BasicConv2d(
      (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), bias=Fals
e)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_3): BasicConv2d(
      (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
```

```
e)
    )
  )
  (Mixed_6a): InceptionB(
    (branch3x3): BasicConv2d(
      (conv): Conv2d(288, 384, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fals
e)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
    (branch3x3dbl_3): BasicConv2d(
      (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tru
e)
    )
  )
  (Mixed_6b): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(128, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_5): BasicConv2d(
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(128, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (Mixed_6c): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (Mixed_6d): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (Mixed_6e): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (AuxLogits): InceptionAux(
    (conv0): BasicConv2d(
      (conv): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (conv1): BasicConv2d(
      (conv): Conv2d(128, 768, kernel_size=(5, 5), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(768, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (fc): Linear(in_features=768, out_features=1000, bias=True)
  )
  (Mixed_7a): InceptionD(
    (branch3x3_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_2): BasicConv2d(
      (conv): Conv2d(192, 320, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7x3_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7x3_2): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7x3_3): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=Fa
lse)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch7x7x3_4): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (Mixed_7b): InceptionE(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(1280, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_1): BasicConv2d(
      (conv): Conv2d(1280, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_2a): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_2b): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(1280, 448, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(448, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(448, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_3a): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_3b): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(1280, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (Mixed_7c): InceptionE(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(2048, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
      (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_1): BasicConv2d(
      (conv): Conv2d(2048, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_2a): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3_2b): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(2048, 448, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(448, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(448, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_3a): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch3x3dbl_3b): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=Fa
lse)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(2048, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=Tr
ue)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=2048, out_features=5, bias=True)
)
```

Summary of how an example image (299,299,3) is processed through the model-pipleine

In [38]:

```
summary(model, (3, 299, 299))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 149, 149]             864
       BatchNorm2d-2         [-1, 32, 149, 149]              64
       BasicConv2d-3         [-1, 32, 149, 149]               0
            Conv2d-4         [-1, 32, 147, 147]           9,216
       BatchNorm2d-5         [-1, 32, 147, 147]              64
       BasicConv2d-6         [-1, 32, 147, 147]               0
            Conv2d-7         [-1, 64, 147, 147]          18,432
            Conv2d-7         [-1, 64, 147, 147]          18,432
       BatchNorm2d-8         [-1, 64, 147, 147]             128
       BasicConv2d-9         [-1, 64, 147, 147]               0
        MaxPool2d-10           [-1, 64, 73, 73]               0
           Conv2d-11           [-1, 80, 73, 73]           5,120
      BatchNorm2d-12           [-1, 80, 73, 73]             160
      BasicConv2d-13           [-1, 80, 73, 73]               0
           Conv2d-14          [-1, 192, 71, 71]         138,240
      BatchNorm2d-15          [-1, 192, 71, 71]             384
      BasicConv2d-16          [-1, 192, 71, 71]               0
        MaxPool2d-17          [-1, 192, 35, 35]               0
           Conv2d-18           [-1, 64, 35, 35]          12,288
      BatchNorm2d-19           [-1, 64, 35, 35]             128
      BasicConv2d-20           [-1, 64, 35, 35]               0
           Conv2d-21           [-1, 48, 35, 35]           9,216
      BatchNorm2d-22           [-1, 48, 35, 35]              96
      BasicConv2d-23           [-1, 48, 35, 35]               0
           Conv2d-24           [-1, 64, 35, 35]          76,800
      BatchNorm2d-25           [-1, 64, 35, 35]             128
      BasicConv2d-26           [-1, 64, 35, 35]               0
           Conv2d-27           [-1, 64, 35, 35]          12,288
      BatchNorm2d-28           [-1, 64, 35, 35]             128
      BasicConv2d-29           [-1, 64, 35, 35]               0
           Conv2d-30           [-1, 96, 35, 35]          55,296
      BatchNorm2d-31           [-1, 96, 35, 35]             192
      BasicConv2d-32           [-1, 96, 35, 35]               0
           Conv2d-33           [-1, 96, 35, 35]          82,944
      BatchNorm2d-34           [-1, 96, 35, 35]             192
      BasicConv2d-35           [-1, 96, 35, 35]               0
           Conv2d-36           [-1, 32, 35, 35]           6,144
      BatchNorm2d-37           [-1, 32, 35, 35]              64
      BasicConv2d-38           [-1, 32, 35, 35]               0
       InceptionA-39          [-1, 256, 35, 35]               0
           Conv2d-40           [-1, 64, 35, 35]          16,384
      BatchNorm2d-41           [-1, 64, 35, 35]             128
      BasicConv2d-42           [-1, 64, 35, 35]               0
           Conv2d-43           [-1, 48, 35, 35]          12,288
      BatchNorm2d-44           [-1, 48, 35, 35]              96
      BasicConv2d-45           [-1, 48, 35, 35]               0
           Conv2d-46           [-1, 64, 35, 35]          76,800
      BatchNorm2d-47           [-1, 64, 35, 35]             128
      BasicConv2d-48           [-1, 64, 35, 35]               0
           Conv2d-49           [-1, 64, 35, 35]          16,384
      BatchNorm2d-50           [-1, 64, 35, 35]             128
      BasicConv2d-51           [-1, 64, 35, 35]               0
           Conv2d-52           [-1, 96, 35, 35]          55,296
      BatchNorm2d-53           [-1, 96, 35, 35]             192
      BasicConv2d-54           [-1, 96, 35, 35]               0
           Conv2d-55           [-1, 96, 35, 35]          82,944
      BatchNorm2d-56           [-1, 96, 35, 35]             192
      BasicConv2d-57           [-1, 96, 35, 35]               0
           Conv2d-58           [-1, 64, 35, 35]          16,384
      BatchNorm2d-59           [-1, 64, 35, 35]             128
      BasicConv2d-60           [-1, 64, 35, 35]               0
       InceptionA-61          [-1, 288, 35, 35]               0
           Conv2d-62           [-1, 64, 35, 35]          18,432
      BatchNorm2d-63           [-1, 64, 35, 35]             128
      BasicConv2d-64           [-1, 64, 35, 35]               0
           Conv2d-65           [-1, 48, 35, 35]          13,824
      BatchNorm2d-66           [-1, 48, 35, 35]              96
      BasicConv2d-67           [-1, 48, 35, 35]               0
           Conv2d-68           [-1, 64, 35, 35]          76,800
      BatchNorm2d-69           [-1, 64, 35, 35]             128
      BasicConv2d-70           [-1, 64, 35, 35]               0
           Conv2d-71           [-1, 64, 35, 35]          18,432
      BatchNorm2d-72           [-1, 64, 35, 35]             128
      BasicConv2d-73           [-1, 64, 35, 35]               0
           Conv2d-74           [-1, 96, 35, 35]          55,296
      BatchNorm2d-75           [-1, 96, 35, 35]             192
      BasicConv2d-76           [-1, 96, 35, 35]               0
           Conv2d-77           [-1, 96, 35, 35]          82,944
      BatchNorm2d-78           [-1, 96, 35, 35]             192
      BasicConv2d-79           [-1, 96, 35, 35]               0
      BasicConv2d-79           [-1, 96, 35, 35]               0
           Conv2d-80           [-1, 64, 35, 35]          18,432
      BatchNorm2d-81           [-1, 64, 35, 35]             128
      BasicConv2d-82           [-1, 64, 35, 35]               0
       InceptionA-83          [-1, 288, 35, 35]               0
           Conv2d-84          [-1, 384, 17, 17]         995,328
      BatchNorm2d-85          [-1, 384, 17, 17]             768
      BasicConv2d-86          [-1, 384, 17, 17]               0
           Conv2d-87           [-1, 64, 35, 35]          18,432
      BatchNorm2d-88           [-1, 64, 35, 35]             128
      BasicConv2d-89           [-1, 64, 35, 35]               0
           Conv2d-90           [-1, 96, 35, 35]          55,296
      BatchNorm2d-91           [-1, 96, 35, 35]             192
      BasicConv2d-92           [-1, 96, 35, 35]               0
           Conv2d-93           [-1, 96, 17, 17]          82,944
      BatchNorm2d-94           [-1, 96, 17, 17]             192
      BasicConv2d-95           [-1, 96, 17, 17]               0
       InceptionB-96          [-1, 768, 17, 17]               0
           Conv2d-97          [-1, 192, 17, 17]         147,456
      BatchNorm2d-98          [-1, 192, 17, 17]             384
      BasicConv2d-99          [-1, 192, 17, 17]               0
          Conv2d-100          [-1, 128, 17, 17]          98,304
     BatchNorm2d-101          [-1, 128, 17, 17]             256
     BasicConv2d-102          [-1, 128, 17, 17]               0
          Conv2d-103          [-1, 128, 17, 17]         114,688
     BatchNorm2d-104          [-1, 128, 17, 17]             256
     BasicConv2d-105          [-1, 128, 17, 17]               0
          Conv2d-106          [-1, 192, 17, 17]         172,032
     BatchNorm2d-107          [-1, 192, 17, 17]             384
     BasicConv2d-108          [-1, 192, 17, 17]               0
          Conv2d-109          [-1, 128, 17, 17]          98,304
     BatchNorm2d-110          [-1, 128, 17, 17]             256
     BasicConv2d-111          [-1, 128, 17, 17]               0
          Conv2d-112          [-1, 128, 17, 17]         114,688
     BatchNorm2d-113          [-1, 128, 17, 17]             256
     BasicConv2d-114          [-1, 128, 17, 17]               0
          Conv2d-115          [-1, 128, 17, 17]         114,688
     BatchNorm2d-116          [-1, 128, 17, 17]             256
     BasicConv2d-117          [-1, 128, 17, 17]               0
          Conv2d-118          [-1, 128, 17, 17]         114,688
     BatchNorm2d-119          [-1, 128, 17, 17]             256
     BasicConv2d-120          [-1, 128, 17, 17]               0
          Conv2d-121          [-1, 192, 17, 17]         172,032
     BatchNorm2d-122          [-1, 192, 17, 17]             384
     BasicConv2d-123          [-1, 192, 17, 17]               0
          Conv2d-124          [-1, 192, 17, 17]         147,456
     BatchNorm2d-125          [-1, 192, 17, 17]             384
     BasicConv2d-126          [-1, 192, 17, 17]               0
      InceptionC-127          [-1, 768, 17, 17]               0
          Conv2d-128          [-1, 192, 17, 17]         147,456
     BatchNorm2d-129          [-1, 192, 17, 17]             384
     BasicConv2d-130          [-1, 192, 17, 17]               0
          Conv2d-131          [-1, 160, 17, 17]         122,880
     BatchNorm2d-132          [-1, 160, 17, 17]             320
     BasicConv2d-133          [-1, 160, 17, 17]               0
          Conv2d-134          [-1, 160, 17, 17]         179,200
     BatchNorm2d-135          [-1, 160, 17, 17]             320
     BasicConv2d-136          [-1, 160, 17, 17]               0
          Conv2d-137          [-1, 192, 17, 17]         215,040
     BatchNorm2d-138          [-1, 192, 17, 17]             384
     BasicConv2d-139          [-1, 192, 17, 17]               0
          Conv2d-140          [-1, 160, 17, 17]         122,880
     BatchNorm2d-141          [-1, 160, 17, 17]             320
     BasicConv2d-142          [-1, 160, 17, 17]               0
          Conv2d-143          [-1, 160, 17, 17]         179,200
     BatchNorm2d-144          [-1, 160, 17, 17]             320
     BasicConv2d-145          [-1, 160, 17, 17]               0
          Conv2d-146          [-1, 160, 17, 17]         179,200
     BatchNorm2d-147          [-1, 160, 17, 17]             320
     BasicConv2d-148          [-1, 160, 17, 17]               0
          Conv2d-149          [-1, 160, 17, 17]         179,200
     BatchNorm2d-150          [-1, 160, 17, 17]             320
     BasicConv2d-151          [-1, 160, 17, 17]               0
     BasicConv2d-151          [-1, 160, 17, 17]               0
          Conv2d-152          [-1, 192, 17, 17]         215,040
     BatchNorm2d-153          [-1, 192, 17, 17]             384
     BasicConv2d-154          [-1, 192, 17, 17]               0
          Conv2d-155          [-1, 192, 17, 17]         147,456
     BatchNorm2d-156          [-1, 192, 17, 17]             384
     BasicConv2d-157          [-1, 192, 17, 17]               0
      InceptionC-158          [-1, 768, 17, 17]               0
          Conv2d-159          [-1, 192, 17, 17]         147,456
     BatchNorm2d-160          [-1, 192, 17, 17]             384
     BasicConv2d-161          [-1, 192, 17, 17]               0
          Conv2d-162          [-1, 160, 17, 17]         122,880
     BatchNorm2d-163          [-1, 160, 17, 17]             320
     BasicConv2d-164          [-1, 160, 17, 17]               0
          Conv2d-165          [-1, 160, 17, 17]         179,200
     BatchNorm2d-166          [-1, 160, 17, 17]             320
     BasicConv2d-167          [-1, 160, 17, 17]               0
          Conv2d-168          [-1, 192, 17, 17]         215,040
     BatchNorm2d-169          [-1, 192, 17, 17]             384
     BasicConv2d-170          [-1, 192, 17, 17]               0
          Conv2d-171          [-1, 160, 17, 17]         122,880
     BatchNorm2d-172          [-1, 160, 17, 17]             320
     BasicConv2d-173          [-1, 160, 17, 17]               0
          Conv2d-174          [-1, 160, 17, 17]         179,200
     BatchNorm2d-175          [-1, 160, 17, 17]             320
     BasicConv2d-176          [-1, 160, 17, 17]               0
          Conv2d-177          [-1, 160, 17, 17]         179,200
     BatchNorm2d-178          [-1, 160, 17, 17]             320
     BasicConv2d-179          [-1, 160, 17, 17]               0
          Conv2d-180          [-1, 160, 17, 17]         179,200
     BatchNorm2d-181          [-1, 160, 17, 17]             320
     BasicConv2d-182          [-1, 160, 17, 17]               0
          Conv2d-183          [-1, 192, 17, 17]         215,040
     BatchNorm2d-184          [-1, 192, 17, 17]             384
     BasicConv2d-185          [-1, 192, 17, 17]               0
          Conv2d-186          [-1, 192, 17, 17]         147,456
     BatchNorm2d-187          [-1, 192, 17, 17]             384
     BasicConv2d-188          [-1, 192, 17, 17]               0
      InceptionC-189          [-1, 768, 17, 17]               0
          Conv2d-190          [-1, 192, 17, 17]         147,456
     BatchNorm2d-191          [-1, 192, 17, 17]             384
     BasicConv2d-192          [-1, 192, 17, 17]               0
          Conv2d-193          [-1, 192, 17, 17]         147,456
     BatchNorm2d-194          [-1, 192, 17, 17]             384
     BasicConv2d-195          [-1, 192, 17, 17]               0
          Conv2d-196          [-1, 192, 17, 17]         258,048
     BatchNorm2d-197          [-1, 192, 17, 17]             384
     BasicConv2d-198          [-1, 192, 17, 17]               0
          Conv2d-199          [-1, 192, 17, 17]         258,048
     BatchNorm2d-200          [-1, 192, 17, 17]             384
     BasicConv2d-201          [-1, 192, 17, 17]               0
          Conv2d-202          [-1, 192, 17, 17]         147,456
     BatchNorm2d-203          [-1, 192, 17, 17]             384
     BasicConv2d-204          [-1, 192, 17, 17]               0
          Conv2d-205          [-1, 192, 17, 17]         258,048
     BatchNorm2d-206          [-1, 192, 17, 17]             384
     BasicConv2d-207          [-1, 192, 17, 17]               0
          Conv2d-208          [-1, 192, 17, 17]         258,048
     BatchNorm2d-209          [-1, 192, 17, 17]             384
     BasicConv2d-210          [-1, 192, 17, 17]               0
          Conv2d-211          [-1, 192, 17, 17]         258,048
     BatchNorm2d-212          [-1, 192, 17, 17]             384
     BasicConv2d-213          [-1, 192, 17, 17]               0
          Conv2d-214          [-1, 192, 17, 17]         258,048
     BatchNorm2d-215          [-1, 192, 17, 17]             384
     BasicConv2d-216          [-1, 192, 17, 17]               0
          Conv2d-217          [-1, 192, 17, 17]         147,456
     BatchNorm2d-218          [-1, 192, 17, 17]             384
     BasicConv2d-219          [-1, 192, 17, 17]               0
      InceptionC-220          [-1, 768, 17, 17]               0
          Conv2d-221            [-1, 128, 5, 5]          98,304
     BatchNorm2d-222            [-1, 128, 5, 5]             256
     BasicConv2d-223            [-1, 128, 5, 5]               0
     BasicConv2d-223            [-1, 128, 5, 5]               0
          Conv2d-224            [-1, 768, 1, 1]       2,457,600
     BatchNorm2d-225            [-1, 768, 1, 1]           1,536
     BasicConv2d-226            [-1, 768, 1, 1]               0
          Linear-227                 [-1, 1000]         769,000
    InceptionAux-228                 [-1, 1000]               0
          Conv2d-229          [-1, 192, 17, 17]         147,456
     BatchNorm2d-230          [-1, 192, 17, 17]             384
     BasicConv2d-231          [-1, 192, 17, 17]               0
          Conv2d-232            [-1, 320, 8, 8]         552,960
     BatchNorm2d-233            [-1, 320, 8, 8]             640
     BasicConv2d-234            [-1, 320, 8, 8]               0
          Conv2d-235          [-1, 192, 17, 17]         147,456
     BatchNorm2d-236          [-1, 192, 17, 17]             384
     BasicConv2d-237          [-1, 192, 17, 17]               0
          Conv2d-238          [-1, 192, 17, 17]         258,048
     BatchNorm2d-239          [-1, 192, 17, 17]             384
     BasicConv2d-240          [-1, 192, 17, 17]               0
          Conv2d-241          [-1, 192, 17, 17]         258,048
     BatchNorm2d-242          [-1, 192, 17, 17]             384
     BasicConv2d-243          [-1, 192, 17, 17]               0
          Conv2d-244            [-1, 192, 8, 8]         331,776
     BatchNorm2d-245            [-1, 192, 8, 8]             384
     BasicConv2d-246            [-1, 192, 8, 8]               0
      InceptionD-247           [-1, 1280, 8, 8]               0
          Conv2d-248            [-1, 320, 8, 8]         409,600
     BatchNorm2d-249            [-1, 320, 8, 8]             640
     BasicConv2d-250            [-1, 320, 8, 8]               0
          Conv2d-251            [-1, 384, 8, 8]         491,520
     BatchNorm2d-252            [-1, 384, 8, 8]             768
     BasicConv2d-253            [-1, 384, 8, 8]               0
          Conv2d-254            [-1, 384, 8, 8]         442,368
     BatchNorm2d-255            [-1, 384, 8, 8]             768
     BasicConv2d-256            [-1, 384, 8, 8]               0
          Conv2d-257            [-1, 384, 8, 8]         442,368
     BatchNorm2d-258            [-1, 384, 8, 8]             768
     BasicConv2d-259            [-1, 384, 8, 8]               0
          Conv2d-260            [-1, 448, 8, 8]         573,440
     BatchNorm2d-261            [-1, 448, 8, 8]             896
     BasicConv2d-262            [-1, 448, 8, 8]               0
          Conv2d-263            [-1, 384, 8, 8]       1,548,288
     BatchNorm2d-264            [-1, 384, 8, 8]             768
     BasicConv2d-265            [-1, 384, 8, 8]               0
          Conv2d-266            [-1, 384, 8, 8]         442,368
     BatchNorm2d-267            [-1, 384, 8, 8]             768
     BasicConv2d-268            [-1, 384, 8, 8]               0
          Conv2d-269            [-1, 384, 8, 8]         442,368
     BatchNorm2d-270            [-1, 384, 8, 8]             768
     BasicConv2d-271            [-1, 384, 8, 8]               0
          Conv2d-272            [-1, 192, 8, 8]         245,760
     BatchNorm2d-273            [-1, 192, 8, 8]             384
     BasicConv2d-274            [-1, 192, 8, 8]               0
      InceptionE-275           [-1, 2048, 8, 8]               0
          Conv2d-276            [-1, 320, 8, 8]         655,360
     BatchNorm2d-277            [-1, 320, 8, 8]             640
     BasicConv2d-278            [-1, 320, 8, 8]               0
          Conv2d-279            [-1, 384, 8, 8]         786,432
     BatchNorm2d-280            [-1, 384, 8, 8]             768
     BasicConv2d-281            [-1, 384, 8, 8]               0
          Conv2d-282            [-1, 384, 8, 8]         442,368
     BatchNorm2d-283            [-1, 384, 8, 8]             768
     BasicConv2d-284            [-1, 384, 8, 8]               0
          Conv2d-285            [-1, 384, 8, 8]         442,368
     BatchNorm2d-286            [-1, 384, 8, 8]             768
     BasicConv2d-287            [-1, 384, 8, 8]               0
          Conv2d-288            [-1, 448, 8, 8]         917,504
     BatchNorm2d-289            [-1, 448, 8, 8]             896
     BasicConv2d-290            [-1, 448, 8, 8]               0
          Conv2d-291            [-1, 384, 8, 8]       1,548,288
     BatchNorm2d-292            [-1, 384, 8, 8]             768
     BasicConv2d-293            [-1, 384, 8, 8]               0
          Conv2d-294            [-1, 384, 8, 8]         442,368
     BatchNorm2d-295            [-1, 384, 8, 8]             768
     BatchNorm2d-295          [-1, 384, 8, 8]             768
     BasicConv2d-296          [-1, 384, 8, 8]               0
          Conv2d-297          [-1, 384, 8, 8]         442,368
     BatchNorm2d-298          [-1, 384, 8, 8]             768
     BasicConv2d-299          [-1, 384, 8, 8]               0
          Conv2d-300          [-1, 192, 8, 8]         393,216
     BatchNorm2d-301          [-1, 192, 8, 8]             384
     BasicConv2d-302          [-1, 192, 8, 8]               0
      InceptionE-303         [-1, 2048, 8, 8]               0
AdaptiveAvgPool2d-304          [-1, 2048, 1, 1]               0
         Dropout-305         [-1, 2048, 1, 1]               0
          Linear-306                  [-1, 5]          10,245
================================================================
Total params: 25,122,509
Trainable params: 25,122,509
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.02
Forward/backward pass size (MB): 228.65
Params size (MB): 95.83
Estimated Total Size (MB): 325.51
----------------------------------------------------------------
```

In [39]:

```
epochs=10
best_model_wts = copy.deepcopy(model.state_dict())
best_acc = 0.0
train_loss_inceptionv3,test_loss_inceptionv3,accuracy_inceptionv3= training_and_validatio
n_loop(epochs,step_lr_scheduler,model,optimizer,aerial_train_loader,aerial_validation_loa
der,best_acc,best_model_wts,'inceptionv3')
```

```
/usr/local/lib/python3.7/dist-packages/torch/optim/lr_scheduler.py:134: UserWarning: Dete
cted call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later,
you should call them in the opposite order: `optimizer.step()` before `lr_scheduler.step(
)`.  Failure to do this will result in PyTorch skipping the first value of the learning r
ate schedule. See more details at https://pytorch.org/docs/stable/optim.html#how-to-adjus
t-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
```

```
0 1.5312782526016235
10 1.0991261005401611
20 0.4819774627685547
Done Training
Epoch:  0 Train Loss:  1.09126678109169 Test Loss:  0.4952628513177236 Accuracy:  88.1188
1188118812
Saved model with accuracy:  88.11881188118812
0 0.5206536650657654
10 0.5972175002098083
20 0.20286329090595245
Done Training
Epoch:  1 Train Loss:  0.43909166810604244 Test Loss:  0.5022932936747869 Accuracy:  83.1
6831683168317
0 0.029663583263754845
10 0.2688331604003906
20 0.7232166528701782
Done Training
Epoch:  2 Train Loss:  0.35298985393288046 Test Loss:  0.35934006919463474 Accuracy:  92.
07920792079207
Saved model with accuracy:  92.07920792079207
0 0.043572623282670975
10 0.7370339035987854
20 0.1756567656993866
Done Training
Epoch:  3 Train Loss:  0.26234725762445193 Test Loss:  0.2054745890200138 Accuracy:  94.0
5940594059406
Saved model with accuracy:  94.05940594059406
0 1.0982884168624878
10 0.013487825170159340
20 0.05854685232043266
Done Training
Epoch:  4 Train Loss:  0.1312421516228754 Test Loss:  0.08735690467680494 Accuracy:  97.0
2970297029702
```

```
Saved model with accuracy:  97.02970297029702
0 0.06621582061052322
10 0.009867901913821697
20 0.18910717964172363
Done Training
Epoch:  5 Train Loss:  0.175221745032244 Test Loss:  0.11520770378410816 Accuracy:  97.02
970297029702
0 0.2957690954208374
10 0.044243186712265015
20 0.004880521912127733
Done Training
Epoch:  6 Train Loss:  0.13165851562427214 Test Loss:  0.08269767711559932 Accuracy:  98.
01980198019803
Saved model with accuracy:  98.01980198019803
0 0.07399868965148926
10 0.014481846243143082
20 0.4246997833251953
Done Training
Epoch:  7 Train Loss:  0.11945701920642303 Test Loss:  0.10821722504139568 Accuracy:  95.
04950495049505
0 0.10328452289104462
10 0.23128166794776917
20 0.017757482826709747
Done Training
Epoch:  8 Train Loss:  0.16414010986829033 Test Loss:  0.13939137143703798 Accuracy:  97.
02970297029702
0 0.0059217968955636024
10 0.08050955832004547
20 0.02656821347773075
Done Training
Epoch:  9 Train Loss:  0.08340879411508258 Test Loss:  0.21572157717309892 Accuracy:  96.
03960396039604
```

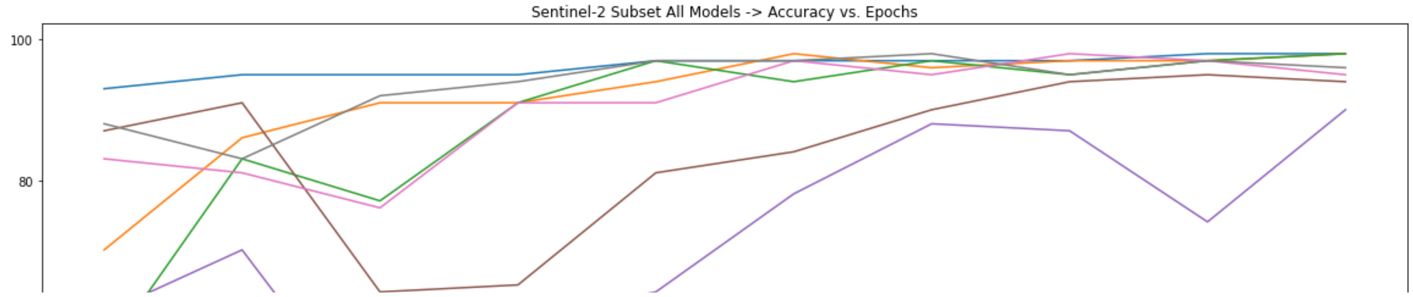
Accuracy Plot vs Epochs for All Models

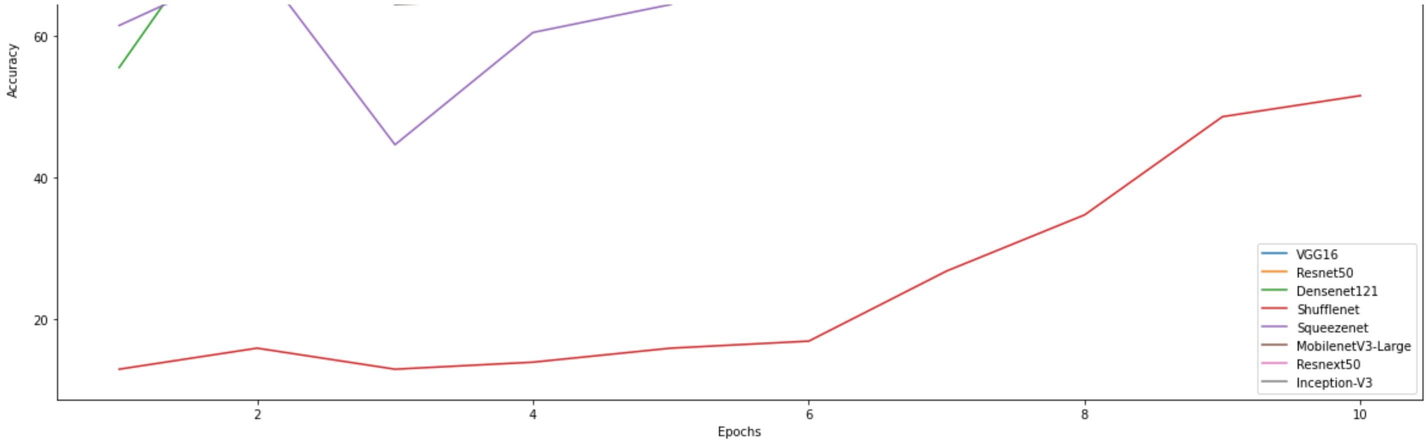
In [40]:

```
list_epochs = [i+1 for i in range(10)]

plt.figure(figsize = (20,10))


plt.plot(list_epochs,accuracy_vgg,label='VGG16')
plt.plot(list_epochs,accuracy_resnet,label='Resnet50')
plt.plot(list_epochs,accuracy_densenet,label='Densenet121')
plt.plot(list_epochs,accuracy_shufflenet,label='Shufflenet')
plt.plot(list_epochs,accuracy_squeezenet,label='Squeezenet')
plt.plot(list_epochs,accuracy_mobilenetv3,label='MobilenetV3-Large')
plt.plot(list_epochs,accuracy_resnext50,label='Resnext50')
plt.plot(list_epochs,accuracy_inceptionv3,label='Inception-V3')


plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Sentinel-2 Subset All Models -> Accuracy vs. Epochs')
plt.legend()
```

Out[40]:

```
<matplotlib.legend.Legend at 0x7f2070e84910>
```





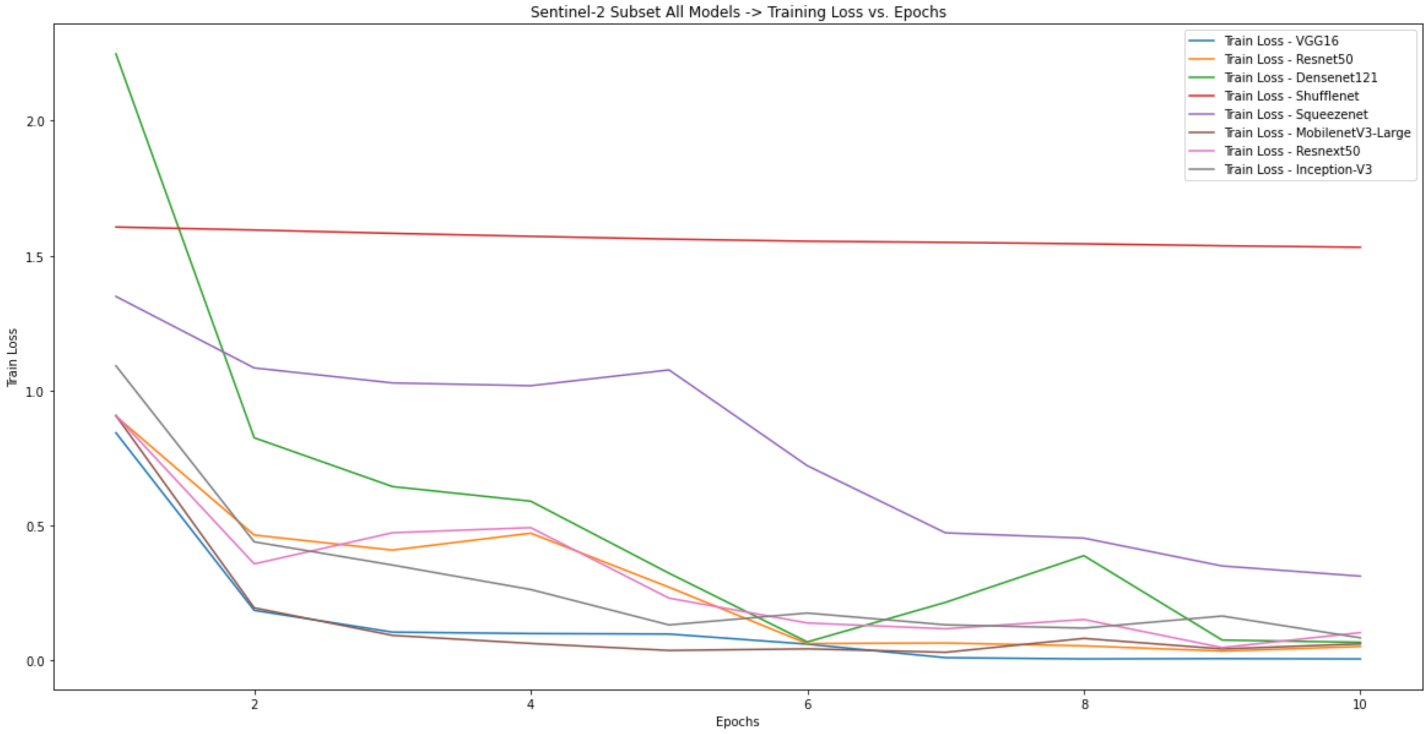
Training Loss Plot vs Epochs for All Models

In [41]:

```
plt.figure(figsize = (20,10))


plt.plot(list_epochs,train_loss_vgg,label='Train Loss - VGG16')
plt.plot(list_epochs,train_loss_resnet,label='Train Loss - Resnet50')
plt.plot(list_epochs,train_loss_densenet,label='Train Loss - Densenet121')
plt.plot(list_epochs,train_loss_shufflenet,label='Train Loss - Shufflenet')
plt.plot(list_epochs,train_loss_squeezenet,label='Train Loss - Squeezenet')
plt.plot(list_epochs,train_loss_mobilenetv3,label='Train Loss - MobilenetV3-Large')
plt.plot(list_epochs,train_loss_resnext50,label='Train Loss - Resnext50')
plt.plot(list_epochs,train_loss_inceptionv3,label='Train Loss - Inception-V3')


plt.xlabel('Epochs')
plt.ylabel('Train Loss')
plt.title('Sentinel-2 Subset All Models -> Training Loss vs. Epochs')
plt.legend()
```

Out[41]:

```
<matplotlib.legend.Legend at 0x7f2076092d50>
```



Test Loss Plot vs Epochs for All Models

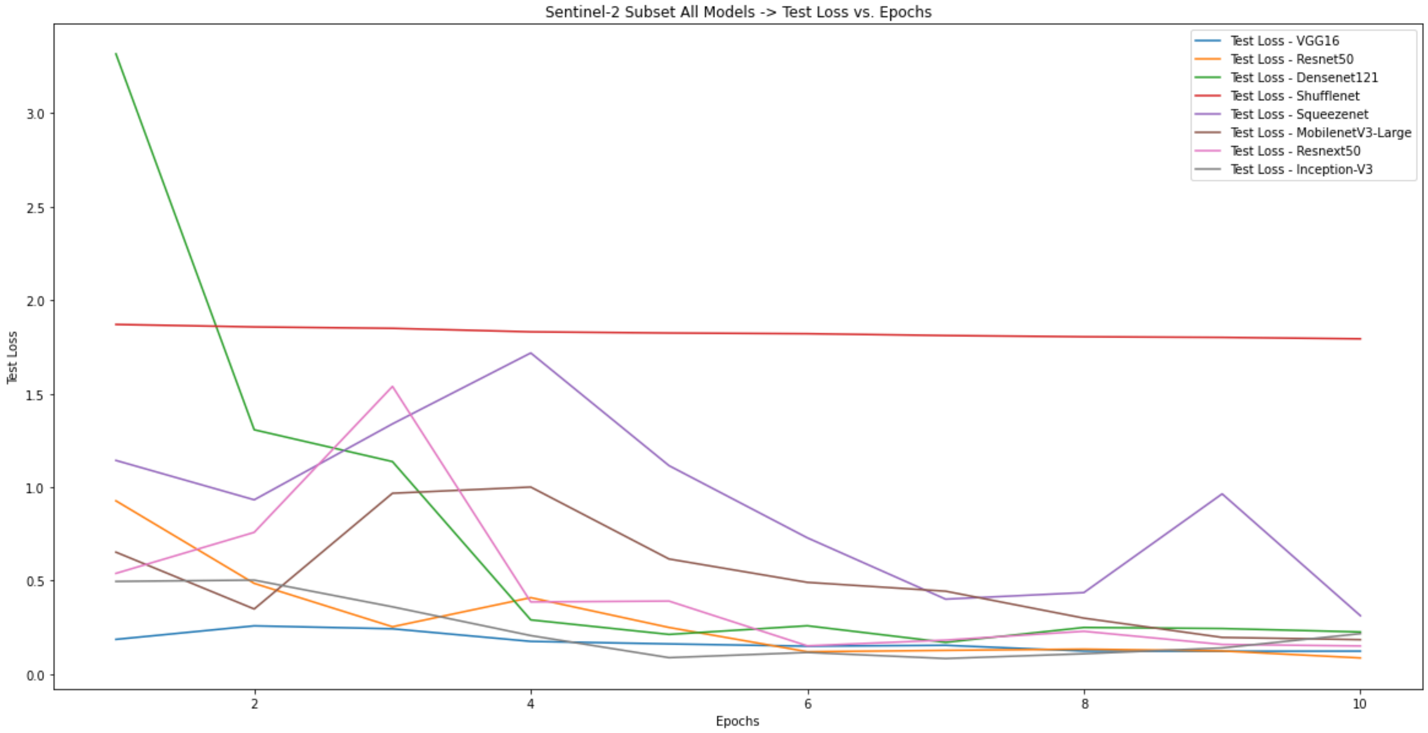
In [42]:

In [42]:

```
plt.figure(figsize = (20,10))


plt.plot(list_epochs,test_loss_vgg,label='Test Loss - VGG16')
plt.plot(list_epochs,test_loss_resnet,label='Test Loss - Resnet50')
plt.plot(list_epochs,test_loss_densenet,label='Test Loss - Densenet121')
plt.plot(list_epochs,test_loss_shufflenet,label='Test Loss - Shufflenet')
plt.plot(list_epochs,test_loss_squeezenet,label='Test Loss - Squeezenet')
plt.plot(list_epochs,test_loss_mobilenetv3,label='Test Loss - MobilenetV3-Large')
plt.plot(list_epochs,test_loss_resnext50,label='Test Loss - Resnext50')
plt.plot(list_epochs,test_loss_inceptionv3,label='Test Loss - Inception-V3')


plt.xlabel('Epochs')
plt.ylabel('Test Loss')
plt.title('Sentinel-2 Subset All Models -> Test Loss vs. Epochs')

plt.legend()
```

Out[42]:

```
<matplotlib.legend.Legend at 0x7f20760aeb50>
```



In [43]:

```
#!curl -k https://files.inria.fr/aerialimagelabeling/getAerial.sh | bash
```

In [51]:

```
#!ls
```

In [52]:

```
#import shutil
#shutil.copytree('sample_data/','/content/drive/My Drive/AerialImageDataset_sample_data/'
)
```

Reference

https://pytorch.org/vision/stable/models.html

In [ ]: