Aerial_Dataset_Feature_Extraction_Segmentation_UNet

May 29, 2021

[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__)
```

[2]:
```
# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
#!pip install tifffile
```

[4]:
```
#!pip install imagecodecs
```

[5]:
```
#files  = os.listdir('/content/drive/My Drive/AerialImageDataset/train/images')
#files.sort()
```

[6]:
```
#print(len(files))
```

[7]:
```
#print(files[1])
```

[8]:
```
#import tifffile
#img  = tifffile.imread('/content/drive/My Drive/AerialImageDataset/train/
 ↪images/' + files[1])
```

[9]:
```
!pip install rasterio
```

```
Requirement already satisfied: rasterio in /usr/local/lib/python3.7/dist-
packages (1.2.3)
Requirement already satisfied: snuggs>=1.4.1 in /usr/local/lib/python3.7/dist-
packages (from rasterio) (1.4.7)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from rasterio) (1.19.5)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages
(from rasterio) (2020.12.5)
Requirement already satisfied: click<8,>=4.0 in /usr/local/lib/python3.7/dist-
packages (from rasterio) (7.1.2)
Requirement already satisfied: click-plugins in /usr/local/lib/python3.7/dist-
packages (from rasterio) (1.1.1)
Requirement already satisfied: affine in /usr/local/lib/python3.7/dist-packages
(from rasterio) (2.3.0)
Requirement already satisfied: attrs in /usr/local/lib/python3.7/dist-packages
(from rasterio) (21.2.0)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-
packages (from rasterio) (0.7.2)
Requirement already satisfied: pyparsing>=2.1.6 in /usr/local/lib/python3.7
/dist-packages (from snuggs>=1.4.1->rasterio) (2.4.7)
```

[10]:
```
import rasterio as rio
```

[11]:
```
tqdm = partial(tqdm, position=0, leave=True)
```

[12]:
```
'''
src = '/content/drive/My Drive/AerialImageDataset/train/gt/'
target = '/content/drive/MyDrive/AerialImageDataset_rgb/train/gt/'
files  = os.listdir('/content/drive/My Drive/AerialImageDataset/train/gt')
files.sort()
#print(files[0])
for file in tqdm(files):
  if file.split('.')[1]=='tif':
    with rio.open(src + file) as img :
        imgnp= img.read()
    #img  = rio.imread(src + files[0])
    #profile['driver']='JPEG'
    jpeg_filename  = target + file.split('.')[0] + '.jpg'
    #jpeg_filename=tif_filename.with_suffix('.jpeg')
    with rio.open(jpeg_filename, 'w'␣
 ↪,width=5000,height=5000,count=1,dtype='uint8') as dst:
        dst.write(imgnp)
#print(imgnp.shape)
print(ok)
for file in tqdm(files):
  try:
    img  = tifffile.imread(src + file)
    print(img.shape)
    #img_bgr = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    #img = img_bgr[::-1]

    #cv2.imwrite(target+file.split('.')[0]+'.jpg',img_bgr)
    print(ok)
  except:
    continue
'''
```

[12]:
```
"\nsrc = '/content/drive/My Drive/AerialImageDataset/train/gt/'\ntarget =
'/content/drive/MyDrive/AerialImageDataset_rgb/train/gt/'\nfiles  =
os.listdir('/content/drive/My
Drive/AerialImageDataset/train/gt')\nfiles.sort()\n#print(files[0])\nfor file in
tqdm(files):\n  if file.split('.')[1]=='tif':\n    with rio.open(src + file) as
img :\n        imgnp= img.read()\n    #img  = rio.imread(src + files[0])\n
#profile['driver']='JPEG'\n    jpeg_filename  = target + file.split('.')[0] +
'.jpg'\n    #jpeg_filename=tif_filename.with_suffix('.jpeg')\n    with
rio.open(jpeg_filename, 'w' ,width=5000,height=5000,count=1,dtype='uint8') as
dst:\n        dst.write(imgnp)\n#print(imgnp.shape)\nprint(ok)\nfor file in
tqdm(files):\n  try:\n    img  = tifffile.imread(src + file)\n
print(img.shape)\n    #img_bgr = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)\n    #img
= img_bgr[::-1]\n\n    #cv2.imwrite(target+file.split('.')[0]+'.jpg',img_bgr)\n
print(ok)\n  except:\n    continue\n"
```

[13]:
```
train_path = '/content/drive/MyDrive/AerialImageDataset_rgb/train/images/'␣
↪#Inria Aerial Image Dataset
train_label = '/content/drive/MyDrive/AerialImageDataset_rgb/train/gt/'
```

[14]:
```
import os
import torch
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
from matplotlib.patches import Ellipse
from skimage.draw import ellipse
import glob
import random
import torchvision.transforms.functional as TF

class AerialData(Dataset):
  def __init__(self, root_dir,train=True,test=False,␣
↪transform=None,transform_test=None):
    self.root_dir = root_dir
    self.root_dir_train = root_dir + '/images/'
    self.labels_path = self.root_dir+'/gt/'
    self.train = train
    self.all_imgs_dir = os.listdir(self.root_dir_train)
    self.all_labels_dir = os.listdir(self.labels_path)
    self.transform = transform


  def __len__(self):
    if self.train==True:
      return len(self.all_imgs_dir)


  def transform_img_msk(self, image, mask,test=False):
      #preprocessor = transforms.Compose([transforms.ToTensor(),resize,␣
↪transforms.RandomHorizontalFlip(),transforms.RandomVerticalFlip(),transforms.
↪RandomRotation(45),transforms.RandomPerspective(), transforms.
↪RandomAffine(degrees=20),normalize])
      normalize = transforms.Normalize(mean = [0.485,0.456,0.406],
                                       std = [0.229,0.224,0.225])
      # Transform to tensor
      image = TF.to_tensor(image)
      mask = TF.to_tensor(mask)

      # Resize
      resize = transforms.Resize(size=(224, 224))
```

```
    image = resize(image)
    mask = resize(mask)

    if test==True:
      image = normalize(image)

      return image,mask
    '''


    # Random horizontal flipping
    if random.random() > 0.5:
        image = TF.rotate(image,45)
        mask = TF.rotate(mask,45)


    # Random horizontal flipping
    if random.random() > 0.5:
        image = TF.hflip(image)
        mask = TF.hflip(mask)

    # Random vertical flipping
    if random.random() > 0.5:
        image = TF.vflip(image)
        mask = TF.vflip(mask)

    '''
    #image = normalize(image)


    return image, mask



def __getitem__(self,index):
  #index+=1
  if torch.is_tensor(index):
    index = index.tolist()
  #print(index)



  try:
    img_loc = os.path.join(self.root_dir_train, self.all_imgs_dir[index])
    img_name = img_loc.split('/')[-1]
    mask = os.path.join(self.labels_path, img_name)
```

```
    except FileNotFoundError:
      pass


    #print(img_name)
    #print(ok)
    image = io.imread(img_loc,as_gray=False)
    mask = io.imread(mask,as_gray=True)
    #print(image.shape)

    if self.transform:
      image,mask = self.transform_img_msk(image,mask)

    sample = {'image': image,'mask': mask,'name':img_name}

    #print(sample['label'])
    #plt.imshow(sample['image'])
    #plt.show()

    return sample
```

[15]:
```
#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 = AerialData(root_dir='/content/drive/MyDrive/
 ↪AerialImageDataset_rgb/train',transform=True)

# 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=2,
                                                  sampler=train_sampler)
aerial_validation_loader = torch.utils.data.DataLoader(aerial_dataset_full,␣
 ↪batch_size=2,
                                                       sampler=valid_sampler)
```

0.1 UNet-Model

[16]:
```
import torch
import torch.nn as nn
from torchvision import models

class DoubleConv(nn.Module):
    """(convolution => [BN] => ReLU) * 2"""

    def __init__(self, in_channels, out_channels, mid_channels=None):
        super().__init__()
        if not mid_channels:
            mid_channels = out_channels
        self.double_conv = nn.Sequential(
            nn.Conv2d(in_channels, mid_channels, kernel_size=3, padding=1),
            nn.BatchNorm2d(mid_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(mid_channels, out_channels, kernel_size=3, padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        return self.double_conv(x)


class Down(nn.Module):
    """Downscaling with maxpool then double conv"""

    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.maxpool_conv = nn.Sequential(
            nn.MaxPool2d(2),
            DoubleConv(in_channels, out_channels)
        )
```

```
    def forward(self, x):
        return self.maxpool_conv(x)


class Up(nn.Module):
    """Upscaling then double conv"""

    def __init__(self, in_channels, out_channels, bilinear=True):
        super().__init__()

        # if bilinear, use the normal convolutions to reduce the number of␣
 ↪channels
        if bilinear:
            self.up = nn.Upsample(scale_factor=2, mode='bilinear',␣
 ↪align_corners=True)
            self.conv = DoubleConv(in_channels, out_channels, in_channels // 2)
        else:
            self.up = nn.ConvTranspose2d(in_channels , in_channels // 2,␣
 ↪kernel_size=2, stride=2)
            self.conv = DoubleConv(in_channels, out_channels)


    def forward(self, x1, x2):
        x1 = self.up(x1)
        # input is CHW
        diffY = x2.size()[2] - x1.size()[2]
        diffX = x2.size()[3] - x1.size()[3]

        x1 = F.pad(x1, [diffX // 2, diffX - diffX // 2,
                        diffY // 2, diffY - diffY // 2])
        # if you have padding issues, see
        # https://github.com/HaiyongJiang/U-Net-Pytorch-Unstructured-Buggy/
 ↪commit/0e854509c2cea854e247a9c615f175f76fbb2e3a
        # https://github.com/xiaopeng-liao/Pytorch-UNet/commit/
 ↪8ebac70e633bac59fc22bb5195e513d5832fb3bd
        x = torch.cat([x2, x1], dim=1)
        return self.conv(x)


class OutConv(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(OutConv, self).__init__()
        self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=1)

    def forward(self, x):
        return self.conv(x)
```

[17]:
```
class UNet_main(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=True):
        super(UNet_main, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = DoubleConv(n_channels, 64)
        self.down1 = Down(64, 128)
        self.down2 = Down(128, 256)
        self.down3 = Down(256, 512)
        factor = 2 if bilinear else 1
        self.down4 = Down(512, 1024 // factor)
        self.up1 = Up(1024, 512 // factor, bilinear)
        self.up2 = Up(512, 256 // factor, bilinear)
        self.up3 = Up(256, 128 // factor, bilinear)
        self.up4 = Up(128, 64, bilinear)
        self.outc = OutConv(64, n_classes)



    def forward(self, x):
        x1 = self.inc(x)

        x2 = self.down1(x1)

        x3 = self.down2(x2)

        x4 = self.down3(x3)

        x5 = self.down4(x4)

        x = self.up1(x5, x4)

        x = self.up2(x, x3)


        x = self.up3(x, x2)


        x = self.up4(x, x1)


        #print(x.shape)
        logits = self.outc(x)
```

```
        #print(logits.shape)
        #print(ok)
        return logits,[],[]
```

0.2 For Feature Extraction

I just created a copy of the original UNet class because, if I had to run this model, which extracts the intermediate layers and outputs of each stage, this would casue memory error during training.

[29]:
```
class UNet(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=True):
        super(UNet, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = DoubleConv(n_channels, 64)
        self.down1 = Down(64, 128)
        self.down2 = Down(128, 256)
        self.down3 = Down(256, 512)
        factor = 2 if bilinear else 1
        self.down4 = Down(512, 1024 // factor)
        self.up1 = Up(1024, 512 // factor, bilinear)
        self.up2 = Up(512, 256 // factor, bilinear)
        self.up3 = Up(256, 128 // factor, bilinear)
        self.up4 = Up(128, 64, bilinear)
        self.outc = OutConv(64, n_classes)

        self.layers = []
        self.all_out= []



    def forward(self, x):
        x1 = self.inc(x)
        self.all_out.append(x1)
        self.layers.append(self.inc)

        x2 = self.down1(x1)
        self.layers.append(self.down1)
        self.all_out.append(x2)

        x3 = self.down2(x2)
        self.all_out.append(x3)
        self.layers.append(self.down2)

        x4 = self.down3(x3)
```

```
        self.all_out.append(x4)
        self.layers.append(self.down3)


        x5 = self.down4(x4)
        self.all_out.append(x5)
        self.layers.append(self.down4)

        x = self.up1(x5, x4)
        self.all_out.append(x)
        self.layers.append(self.up1)

        x = self.up2(x, x3)
        self.all_out.append(x)
        self.layers.append(self.up2)

        x = self.up3(x, x2)
        self.all_out.append(x)
        self.layers.append(self.up3)

        x = self.up4(x, x1)
        self.all_out.append(x)
        self.layers.append(self.up4)

        #print(x.shape)
        logits = self.outc(x)
        self.all_out.append(logits)
        self.layers.append(self.outc)

        #print(logits.shape)
        #print(ok)
        return logits,self.all_out,self.layers
```

[18]:
```
class IoULoss(nn.Module):
    def __init__(self, weight=None, size_average=True):
        super(IoULoss, self).__init__()

    def forward(self, inputs, targets, smooth=1):

        #comment out if your model contains a sigmoid or equivalent activation␣
↪layer
        inputs = F.sigmoid(inputs)

        #flatten label and prediction tensors
        inputs = inputs.view(-1)
        targets = targets.view(-1)
```

```
        #intersection is equivalent to True Positive count
        #union is the mutually inclusive area of all labels & predictions
        intersection = (inputs * targets).sum()
        total = (inputs + targets).sum()
        union = total - intersection

        IoU = (intersection + smooth)/(union + smooth)

        return 1 - IoU
```

[19]:
```
from collections import defaultdict
import torch.nn.functional as F
import copy
from sklearn.metrics import jaccard_similarity_score


def dice_loss(pred, target, smooth = 1.):
    pred = pred.contiguous()
    target = target.contiguous()

    intersection = (pred * target).sum(dim=2).sum(dim=2)

    loss = (1 - ((2. * intersection + smooth) / (pred.sum(dim=2).sum(dim=2) +␣
 ↪target.sum(dim=2).sum(dim=2) + smooth)))

    return loss.mean()

smooth = 1e-12
def jaccard_approx(pred, target, smooth=1e-12 ):
    #intersection = K.sum(y_true * y_pred, axis=[0, -1, -2])
    intersection = (pred * target).sum(dim=2).sum(dim=2)
    #sum_ = K.sum(y_true + y_pred, axis=[0, -1, -2])
    sum_ = (pred.sum(dim=2).sum(dim=2) + target.sum(dim=2).sum(dim=2))

    jac = (intersection + smooth) / (sum_ - intersection + smooth)

    return jac.mean()

def calc_loss(pred, target, metrics, bce_weight=0.5):
    bce = F.binary_cross_entropy_with_logits(pred, target)

    #pred = F.sigmoid(pred)

    #dice = dice_loss(pred, target)
    #loss = bce * bce_weight + dice * (1 - bce_weight)
    #loss = torch.log(jaccard_approx(pred,target))
    #loss = jaccard_approx(pred, target )
```

```
    loss = bce

    #iou_loss = IoULoss()
    #loss = iou_loss.forward(pred,target)




    #metrics['bce'] += bce.data.cpu().numpy() * target.size(0)
    #metrics['dice'] += dice.data.cpu().numpy() * target.size(0)
    metrics['loss'] += loss.data.cpu().numpy() * target.size(0)
    #metrics['iouloss'] += iouloss.data.cpu().numpy() * target.size(0)
    #metrics['jaccloss'] += jaccloss.data.cpu().numpy() * target.size(0)

    return loss

def calc_jacc_img_msk(model, prd, msk, batch_size, n_classes=1):
    #prd = model.predict(img, batch_size= batch_size)
    #print("prd.shape {0}, msk.shape {1}". format(prd.shape, msk.shape))
     #prd.shape, msk.shape (16, 2, 256, 256) (16, 2, 256, 256)
    avg, trs = [], []

    for i in range(n_classes):
        t_msk = msk[:, i, :, :] # t_mask shape is (Npredictions, H, W)
        t_prd = prd[:, i, :, :]
        t_msk = t_msk.reshape(msk.shape[0] * msk.shape[2], msk.shape[3]) #␣
 ↪shape is Npredictions*W, H
        t_prd = t_prd.reshape(msk.shape[0] * msk.shape[2], msk.shape[3])

        m, b_tr = 0, 0
        for j in range(10):
            tr = j / 10
            pred_binary_mask = t_prd > tr

            jk = jaccard_similarity_score(t_msk, pred_binary_mask)
            if jk > m:
                m = jk
                b_tr = tr

        print("i, m, b_tr", i, m, b_tr)
        avg.append(m)
        trs.append(b_tr)

    score = sum(avg) / n_classes
    return score, trs

def print_metrics(metrics, epoch_samples, phase):
```

```
    outputs = []
    for k in metrics.keys():
        outputs.append("{}: {:4f}".format(k, metrics[k] / epoch_samples))

    print("{}: {}".format(phase, ", ".join(outputs)))


train_loss = []
test_loss = []

def train_model(model, optimizer, scheduler, num_epochs=10):
    #best_model_wts = copy.deepcopy(model.state_dict())
    best_loss = 1e10

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        since = time.time()

      # Each epoch has a training and validation phase

        scheduler.step()
        for param_group in optimizer.param_groups:
            print("LR", param_group['lr'])
        print('Started training')

        model.train()  # Set model to training mode


        metrics = defaultdict(float)
        epoch_samples = 0

        for inputs in aerial_train_loader:
            #print(dataloaders['train'])
            #print('Entered')
            inputs['image'] = inputs['image'].to(device)
            inputs['mask'] =  inputs['mask'].to(device)

            # zero the parameter gradients
            optimizer.zero_grad()

            # forward
            # track history if only in train

            outputs,_,_ = model(inputs['image'])
            loss = calc_loss(outputs, inputs['mask'], metrics)
```

```
            # backward + optimize only if in training phase
            loss.backward()
            optimizer.step()

            # statistics
            epoch_samples += inputs['image'].size(0)

        print_metrics(metrics, epoch_samples, 'train')
        epoch_loss = metrics['loss'] / epoch_samples
        train_loss.append(epoch_loss)

        #torch.save(model,f'/content/drive/My Drive/cars_latest/
↪resunet50_epoch_{e}_1.pt')

        #train_score, trs = calc_jacc_img_msk(model, inputs['image'].cpu().
↪numpy(), inputs['segmented_mask'].cpu().numpy(), 8, 1)


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

        metrics = defaultdict(float)
        epoch_samples=0

        for inputs in aerial_validation_loader:

            #print(dataloaders['train'])
            inputs['image'] = inputs['image'].to(device)
            #print(inputs['image'].shape)
            inputs['mask'] =  inputs['mask'].to(device)


            outputs,_,_ = model(inputs['image'])
            loss = calc_loss(outputs, inputs['mask'], metrics)
            epoch_samples += inputs['image'].size(0)


        print_metrics(metrics, epoch_samples, 'val')
        epoch_loss = metrics['loss'] / epoch_samples
        test_loss.append(epoch_loss)

        #test_score, trs = calc_jacc_img_msk(model, outputs.cpu().data.numpy(),␣
↪inputs['segmented_mask'].cpu().numpy(), 8, 1)

        #print("\n")
```

```
        print("Epoch: ", epoch, "Train Loss: ", train_loss[-1], "Test Loss: ",␣
↪test_loss[-1] )

        time_elapsed = time.time() - since
        print('{:.0f}m {:.0f}s'.format(time_elapsed // 60, time_elapsed % 60))
        print("\n")
        if epoch%5==0:
          torch.save({
              'epoch': epoch,
              'model_state_dict': model.state_dict(),
              'optimizer_state_dict': optimizer.state_dict(),
              'train_loss': train_loss[-1],
              }, f'/content/drive/My Drive/AerialImageDataset_rgb/train/
↪epoch_{epoch}_trainloss_{train_loss[-1]}_testloss_{test_loss[-1]}_unet_01.
↪pt')

        torch.save({
              'epoch': epoch,
              'model_state_dict': model.state_dict(),
              'optimizer_state_dict': optimizer.state_dict(),
              'train_loss': train_loss[-1],
              }, f'/content/drive/My Drive/AerialImageDataset_rgb/train/
↪epoch_{epoch}_trainloss_{train_loss[-1]}_testloss_{test_loss[-1]}_unet_01.
↪pt')



    #print('Best val loss: {:4f}'.format(best_loss))

    # load best model weights
    #model.load_state_dict(best_model_wts)
    return model,train_loss,test_loss
```

[55]:
```
model = UNet_main(n_channels=3, n_classes=1)

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


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

gpu_flag = torch.cuda.is_available()
print(gpu_flag)
if gpu_flag:
    model = model.cuda()
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
True
```

[21]:
```
print(model)
```

```
UNet(
  (inc): DoubleConv(
    (double_conv): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (5): ReLU(inplace=True)
    )
  )
  (down1): Down(
    (maxpool_conv): Sequential(
      (0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (1): DoubleConv(
        (double_conv): Sequential(
          (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (5): ReLU(inplace=True)
        )
      )
    )
  )
  (down2): Down(
    (maxpool_conv): Sequential(
      (0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (1): DoubleConv(
        (double_conv): Sequential(
          (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
```

```
1))
          (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (5): ReLU(inplace=True)
        )
      )
    )
  )
  (down3): Down(
    (maxpool_conv): Sequential(
      (0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (1): DoubleConv(
        (double_conv): Sequential(
          (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (5): ReLU(inplace=True)
        )
      )
    )
  )
  (down4): Down(
    (maxpool_conv): Sequential(
      (0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (1): DoubleConv(
        (double_conv): Sequential(
          (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (2): ReLU(inplace=True)
          (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
          (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
          (5): ReLU(inplace=True)
        )
      )
    )
  )
```

```
  (up1): Up(
    (up): Upsample(scale_factor=2.0, mode=bilinear)
    (conv): DoubleConv(
      (double_conv): Sequential(
        (0): Conv2d(1024, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (up2): Up(
    (up): Upsample(scale_factor=2.0, mode=bilinear)
    (conv): DoubleConv(
      (double_conv): Sequential(
        (0): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (up3): Up(
    (up): Upsample(scale_factor=2.0, mode=bilinear)
    (conv): DoubleConv(
      (double_conv): Sequential(
        (0): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (up4): Up(
    (up): Upsample(scale_factor=2.0, mode=bilinear)
```

```
    (conv): DoubleConv(
      (double_conv): Sequential(
        (0): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (5): ReLU(inplace=True)
      )
    )
  )
  (outc): OutConv(
    (conv): Conv2d(64, 1, kernel_size=(1, 1), stride=(1, 1))
  )
)
```

0.3 Summary of how an example image (224,224,3) is processed through the U-Net Model Pipeline

[48]:
```
summary(model, (3, 224, 224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 224, 224]           1,792
       BatchNorm2d-2         [-1, 64, 224, 224]             128
              ReLU-3         [-1, 64, 224, 224]               0
            Conv2d-4         [-1, 64, 224, 224]          36,928
       BatchNorm2d-5         [-1, 64, 224, 224]             128
              ReLU-6         [-1, 64, 224, 224]               0
        DoubleConv-7         [-1, 64, 224, 224]               0
         MaxPool2d-8         [-1, 64, 112, 112]               0
            Conv2d-9        [-1, 128, 112, 112]          73,856
      BatchNorm2d-10        [-1, 128, 112, 112]             256
             ReLU-11        [-1, 128, 112, 112]               0
           Conv2d-12        [-1, 128, 112, 112]         147,584
      BatchNorm2d-13        [-1, 128, 112, 112]             256
             ReLU-14        [-1, 128, 112, 112]               0
       DoubleConv-15        [-1, 128, 112, 112]               0
             Down-16        [-1, 128, 112, 112]               0
        MaxPool2d-17          [-1, 128, 56, 56]               0
           Conv2d-18          [-1, 256, 56, 56]         295,168
      BatchNorm2d-19          [-1, 256, 56, 56]             512
             ReLU-20          [-1, 256, 56, 56]               0
           Conv2d-21          [-1, 256, 56, 56]         590,080
```

```
BatchNorm2d-22          [-1, 256, 56, 56]             512
       ReLU-23          [-1, 256, 56, 56]               0
 DoubleConv-24          [-1, 256, 56, 56]               0
       Down-25          [-1, 256, 56, 56]               0
  MaxPool2d-26          [-1, 256, 28, 28]               0
     Conv2d-27          [-1, 512, 28, 28]       1,180,160
BatchNorm2d-28          [-1, 512, 28, 28]           1,024
       ReLU-29          [-1, 512, 28, 28]               0
     Conv2d-30          [-1, 512, 28, 28]       2,359,808
BatchNorm2d-31          [-1, 512, 28, 28]           1,024
       ReLU-32          [-1, 512, 28, 28]               0
 DoubleConv-33          [-1, 512, 28, 28]               0
       Down-34          [-1, 512, 28, 28]               0
  MaxPool2d-35          [-1, 512, 14, 14]               0
     Conv2d-36          [-1, 512, 14, 14]       2,359,808
BatchNorm2d-37          [-1, 512, 14, 14]           1,024
       ReLU-38          [-1, 512, 14, 14]               0
     Conv2d-39          [-1, 512, 14, 14]       2,359,808
BatchNorm2d-40          [-1, 512, 14, 14]           1,024
       ReLU-41          [-1, 512, 14, 14]               0
 DoubleConv-42          [-1, 512, 14, 14]               0
       Down-43          [-1, 512, 14, 14]               0
   Upsample-44          [-1, 512, 28, 28]               0
     Conv2d-45          [-1, 512, 28, 28]       4,719,104
BatchNorm2d-46          [-1, 512, 28, 28]           1,024
       ReLU-47          [-1, 512, 28, 28]               0
     Conv2d-48          [-1, 256, 28, 28]       1,179,904
BatchNorm2d-49          [-1, 256, 28, 28]             512
       ReLU-50          [-1, 256, 28, 28]               0
 DoubleConv-51          [-1, 256, 28, 28]               0
         Up-52          [-1, 256, 28, 28]               0
   Upsample-53          [-1, 256, 56, 56]               0
     Conv2d-54          [-1, 256, 56, 56]       1,179,904
BatchNorm2d-55          [-1, 256, 56, 56]             512
       ReLU-56          [-1, 256, 56, 56]               0
     Conv2d-57          [-1, 128, 56, 56]         295,040
BatchNorm2d-58          [-1, 128, 56, 56]             256
       ReLU-59          [-1, 128, 56, 56]               0
 DoubleConv-60          [-1, 128, 56, 56]               0
         Up-61          [-1, 128, 56, 56]               0
   Upsample-62        [-1, 128, 112, 112]               0
     Conv2d-63        [-1, 128, 112, 112]         295,040
BatchNorm2d-64        [-1, 128, 112, 112]             256
       ReLU-65        [-1, 128, 112, 112]               0
     Conv2d-66         [-1, 64, 112, 112]          73,792
BatchNorm2d-67         [-1, 64, 112, 112]             128
       ReLU-68         [-1, 64, 112, 112]               0
 DoubleConv-69         [-1, 64, 112, 112]               0
```

```
              Up-70         [-1, 64, 112, 112]               0
          Upsample-71         [-1, 64, 224, 224]               0
            Conv2d-72         [-1, 64, 224, 224]          73,792
       BatchNorm2d-73         [-1, 64, 224, 224]             128
              ReLU-74         [-1, 64, 224, 224]               0
            Conv2d-75         [-1, 64, 224, 224]          36,928
       BatchNorm2d-76         [-1, 64, 224, 224]             128
              ReLU-77         [-1, 64, 224, 224]               0
        DoubleConv-78         [-1, 64, 224, 224]               0
              Up-79         [-1, 64, 224, 224]               0
            Conv2d-80          [-1, 1, 224, 224]              65
          OutConv-81          [-1, 1, 224, 224]               0
================================================================
Total params: 17,267,393
Trainable params: 17,267,393
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 721.22
Params size (MB): 65.87
Estimated Total Size (MB): 787.66
----------------------------------------------------------------
```

0.4 Feature-Extraction and Display of Feature Maps of Intermediate Layers for an example image tensor (Without Model-training)

[58]:
```
#plt.figure(figsize = (20,10))


for i, sample in enumerate(aerial_validation_loader):
  img_var = Variable(sample['image'][1,:,:,:].unsqueeze(0)).cuda()

  out,img_feat_all,layers = model(img_var)


  for j,img_feat_batch in enumerate(img_feat_all):

    img_feat_1 = img_feat_batch[0,:,:,:]
    plt.figure(figsize = (20,10))


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

    print(layers[j])

    if j>=9:
      print(img_numpy.shape)
```

```
      plt.imshow(img_numpy[0, :, :-1], cmap='gray')
      plt.show()
      print('Feature Extraction done for first image (without any training)')

      break

    # 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[ind-1, :, :-1], cmap='gray')
        ind += 1
    # show the figure


    plt.show()
  break
```

```
Output hidden; open in https://colab.research.google.com to view.
```

1 U-Net Model Training and Validation

[21]:
```
model_trained,train_loss,test_loss = train_model(model, optimizer,␣
↪exp_lr_scheduler, num_epochs=20)
```

```
Epoch 0/19
----------
LR 0.001
Started training

/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)

train: loss: 0.418618
val: loss: 0.509129
Epoch:  0 Train Loss:  0.41861796006560326 Test Loss:  0.5091291492184004
3m 31s


Epoch 1/19
----------
LR 0.001
Started training
train: loss: 0.336834
val: loss: 0.369428
Epoch:  1 Train Loss:  0.3368339124653075 Test Loss:  0.369428426027298
2m 4s


Epoch 2/19
----------
LR 0.001
Started training
train: loss: 0.325826
val: loss: 0.328942
Epoch:  2 Train Loss:  0.3258264135155413 Test Loss:  0.3289418112900522
2m 4s


Epoch 3/19
----------
LR 0.001
Started training
train: loss: 0.310911
val: loss: 0.305409
Epoch:  3 Train Loss:  0.31091133194665116 Test Loss:  0.305409019605981
2m 4s


Epoch 4/19
----------
LR 0.001
Started training
train: loss: 0.303118
val: loss: 0.291869
Epoch:  4 Train Loss:  0.3031180656204621 Test Loss:  0.2918694524301423
2m 3s
```

```
Epoch 5/19
----------
LR 0.001
Started training
train: loss: 0.295602
val: loss: 0.283152
Epoch:  5 Train Loss:  0.29560218565165997 Test Loss:  0.2831515793999036
2m 3s


Epoch 6/19
----------
LR 0.0001
Started training
train: loss: 0.285498
val: loss: 0.276510
Epoch:  6 Train Loss:  0.2854979105500711 Test Loss:  0.2765097961657577
2m 3s


Epoch 7/19
----------
LR 0.0001
Started training
train: loss: 0.283355
val: loss: 0.277207
Epoch:  7 Train Loss:  0.2833553554697169 Test Loss:  0.27720696354905766
2m 3s


Epoch 8/19
----------
LR 0.0001
Started training
train: loss: 0.277915
val: loss: 0.275680
Epoch:  8 Train Loss:  0.2779150376510289 Test Loss:  0.2756797932088375
2m 3s


Epoch 9/19
----------
LR 0.0001
Started training
train: loss: 0.273814
val: loss: 0.276405
```

```
Epoch:  9 Train Loss:  0.27381387259811163 Test Loss:  0.27640525127450627
2m 2s


Epoch 10/19
----------
LR 0.0001
Started training
train: loss: 0.271043
val: loss: 0.268948
Epoch:  10 Train Loss:  0.2710431701400214 Test Loss:  0.2689480251736111
2m 3s


Epoch 11/19
----------
LR 0.0001
Started training
train: loss: 0.272479
val: loss: 0.268828
Epoch:  11 Train Loss:  0.2724792833129565 Test Loss:  0.2688278829058011
2m 4s


Epoch 12/19
----------
LR 0.0001
Started training
train: loss: 0.270399
val: loss: 0.269348
Epoch:  12 Train Loss:  0.27039889215181273 Test Loss:  0.2693480894797378
2m 3s


Epoch 13/19
----------
LR 1e-05
Started training
train: loss: 0.265672
val: loss: 0.268601
Epoch:  13 Train Loss:  0.26567246909770703 Test Loss:  0.26860059135489994
2m 2s


Epoch 14/19
----------
LR 1e-05
Started training
```

```
train: loss: 0.263904
val: loss: 0.266437
Epoch:  14 Train Loss:  0.2639044345253044 Test Loss:  0.2664370822409789
2m 4s


Epoch 15/19
----------
LR 1e-05
Started training
train: loss: 0.263245
val: loss: 0.267151
Epoch:  15 Train Loss:  0.26324479323294425 Test Loss:  0.26715131600697833
2m 2s


Epoch 16/19
----------
LR 1e-05
Started training


      ␣
↪---------------------------------------------------------------------------

       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-21-49ee48c5846b> in <module>()
   ----> 1 model_trained,train_loss,test_loss = train_model(model, optimizer,␣
↪exp_lr_scheduler, num_epochs=20)


       <ipython-input-19-c4b96b9be0d1> in train_model(model, optimizer,␣
↪scheduler, num_epochs)
       114         epoch_samples = 0
       115
   --> 116         for inputs in aerial_train_loader:
       117             #print(dataloaders['train'])
       118             #print('Entered')


       /usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py in␣
↪__next__(self)
       515             if self._sampler_iter is None:
       516                 self._reset()
   --> 517             data = self._next_data()
       518             self._num_yielded += 1
```

```
      519             if self._dataset_kind == _DatasetKind.Iterable and \


      /usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py in␣
↪_next_data(self)
      555     def _next_data(self):
      556         index = self._next_index()  # may raise StopIteration
  --> 557         data = self._dataset_fetcher.fetch(index)  # may raise␣
↪StopIteration
      558         if self._pin_memory:
      559             data = _utils.pin_memory.pin_memory(data)


      /usr/local/lib/python3.7/dist-packages/torch/utils/data/_utils/fetch.py␣
↪in fetch(self, possibly_batched_index)
       42     def fetch(self, possibly_batched_index):
       43         if self.auto_collation:
  ---> 44             data = [self.dataset[idx] for idx in␣
↪possibly_batched_index]
       45         else:
       46             data = self.dataset[possibly_batched_index]


      /usr/local/lib/python3.7/dist-packages/torch/utils/data/_utils/fetch.py␣
↪in <listcomp>(.0)
       42     def fetch(self, possibly_batched_index):
       43         if self.auto_collation:
  ---> 44             data = [self.dataset[idx] for idx in␣
↪possibly_batched_index]
       45         else:
       46             data = self.dataset[possibly_batched_index]


      <ipython-input-14-623950ba22a8> in __getitem__(self, index)
       91     #print(img_name)
       92     #print(ok)
  ---> 93     image = io.imread(img_loc,as_gray=False)
       94     mask = io.imread(mask,as_gray=True)
       95     #print(image.shape)


      /usr/local/lib/python3.7/dist-packages/skimage/io/_io.py in␣
↪imread(fname, as_gray, plugin, **plugin_args)
       46
       47     with file_or_url_context(fname) as fname:
  ---> 48         img = call_plugin('imread', fname, plugin=plugin,␣
↪**plugin_args)
```

```
     49
     50     if not hasattr(img, 'ndim'):


/usr/local/lib/python3.7/dist-packages/skimage/io/manage_plugins.py in␣
↪call_plugin(kind, *args, **kwargs)
    208                                 (plugin, kind))
    209
--> 210     return func(*args, **kwargs)
    211
    212


/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/
↪imageio_plugin.py in imread(*args, **kwargs)
      8 @wraps(imageio_imread)
      9 def imread(*args, **kwargs):
---> 10     return np.asarray(imageio_imread(*args, **kwargs))


/usr/local/lib/python3.7/dist-packages/imageio/core/functions.py in␣
↪imread(uri, format, **kwargs)
    219
    220     # Get reader and read first
--> 221     reader = read(uri, format, "i", **kwargs)
    222     with reader:
    223         return reader.get_data(0)


/usr/local/lib/python3.7/dist-packages/imageio/core/functions.py in␣
↪get_reader(uri, format, mode, **kwargs)
    141
    142     # Return its reader object
--> 143     return format.get_reader(request)
    144
    145


/usr/local/lib/python3.7/dist-packages/imageio/core/format.py in␣
↪get_reader(self, request)
    172                 "Format %s cannot read in mode %r" % (self.name,␣
↪select_mode)
    173             )
--> 174         return self.Reader(self, request)
    175
    176     def get_writer(self, request):
```

```
      /usr/local/lib/python3.7/dist-packages/imageio/core/format.py in␣
↪__init__(self, format, request)
      222             self._request = request
      223             # Open the reader/writer
  --> 224             self._open(**self.request.kwargs.copy())
      225
      226         @property


      /usr/local/lib/python3.7/dist-packages/imageio/plugins/pillow.py in␣
↪_open(self, pilmode, as_gray, exifrotate)
      404     class Reader(PillowFormat.Reader):
      405         def _open(self, pilmode=None, as_gray=False,␣
↪exifrotate=True):
  --> 406             return PillowFormat.Reader._open(self, pilmode=pilmode,␣
↪as_gray=as_gray)
      407
      408         def _get_file(self):


      /usr/local/lib/python3.7/dist-packages/imageio/plugins/pillow.py in␣
↪_open(self, pilmode, as_gray)
      123             if hasattr(Image, "_decompression_bomb_check"):
      124                 Image._decompression_bomb_check(self._im.size)
  --> 125             pil_try_read(self._im)
      126             # Store args
      127             self._kwargs = dict(


      /usr/local/lib/python3.7/dist-packages/imageio/plugins/pillow.py in␣
↪pil_try_read(im)
      499     try:
      500         # this will raise an IOError if the file is not readable
  --> 501         im.getdata()[0]
      502     except IOError as e:
      503         site = "http://pillow.readthedocs.io/en/latest/installation.
↪html"


      /usr/local/lib/python3.7/dist-packages/PIL/Image.py in getdata(self,␣
↪band)
     1275         """
     1276
  -> 1277         self.load()
     1278         if band is not None:
     1279             return self.im.getband(band)
```

```
    /usr/local/lib/python3.7/dist-packages/PIL/ImageFile.py in load(self)
    249
    250                             b = b + s
--> 251                             n, err_code = decoder.decode(b)
    252                             if n < 0:
    253                                 break


    KeyboardInterrupt:
```

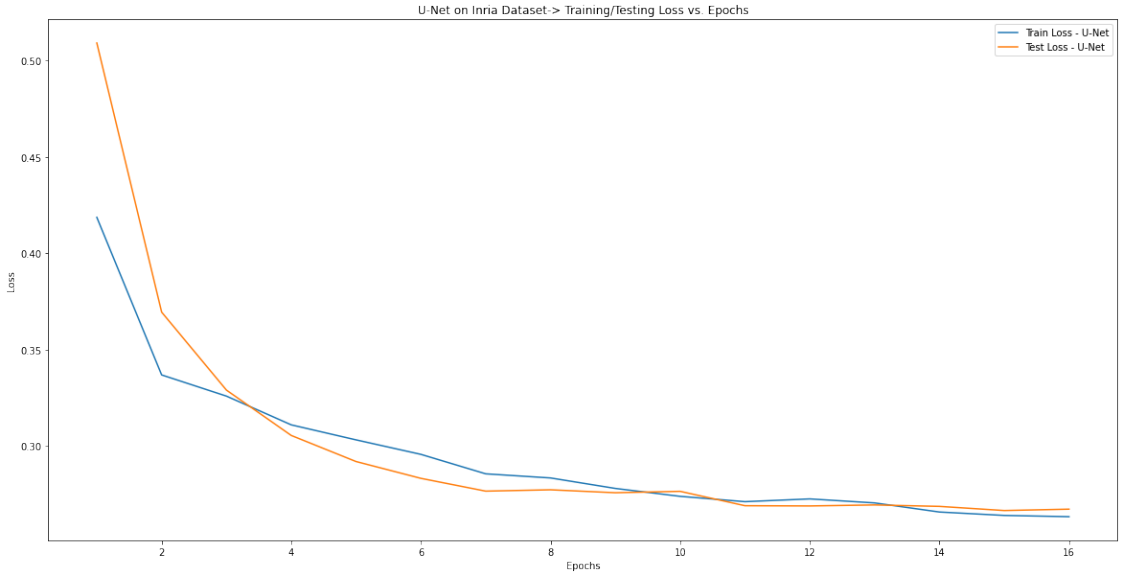
[40]:
```
list_epochs = [i+1 for i in range(16)]

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


plt.plot(list_epochs,train_loss,label='Train Loss - U-Net')
plt.plot(list_epochs,test_loss,label='Test Loss - U-Net')

plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('U-Net on Inria Dataset-> Training/Testing Loss vs. Epochs')
plt.legend()
```

[40]:
```
<matplotlib.legend.Legend at 0x7f701e825b50>
```



1.1 Result/Example

[42]:
```
model.eval()
for inputs in aerial_validation_loader:

    #print(dataloaders['train'])
    inputs['image'] = inputs['image'].to(device)
    #print(inputs['image'].shape)
    inputs['mask'] =  inputs['mask'].to(device)


    outputs,_,_ = model(inputs['image'])




    outputs_numpy = outputs.cpu().detach().numpy()[1,:,:,:]
    mask_numpy =   inputs['mask'].cpu().detach().numpy()[1,:,:,:]

    print(outputs_numpy.reshape(224,224,1).shape)
    print(mask_numpy.reshape(224,224,1).shape)
    plt.figure(figsize = (20,10))

    plt.subplot(121)
    plt.imshow(outputs_numpy.reshape(224,224),cmap='gray')
    plt.subplot(122)
    plt.imshow(mask_numpy.reshape(224,224),cmap='gray')

    print(ok)
```

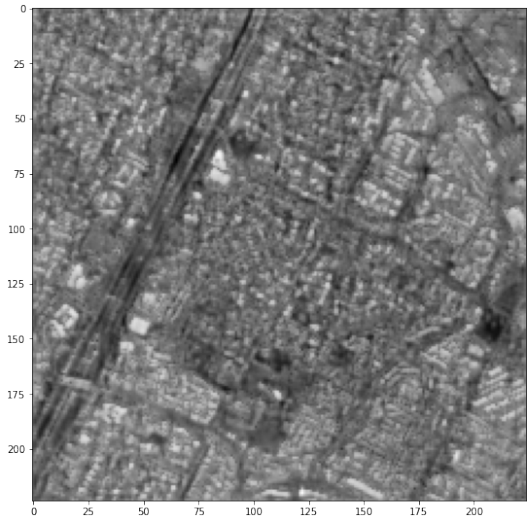
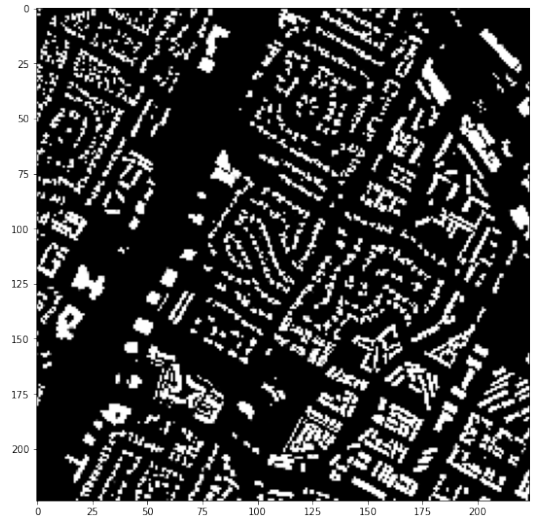
```
(224, 224, 1)
(224, 224, 1)
```

```
      ␣
↪---------------------------------------------------------------------------

       NameError                                 Traceback (most recent call␣
↪last)

       <ipython-input-42-cda8bcfc660d> in <module>()
        25     plt.imshow(mask_numpy.reshape(224,224),cmap='gray')
        26
   ---> 27     print(ok)


       NameError: name 'ok' is not defined
```

1.2 Feature-Extraction and Display of Feature Maps of Intermediate Layers for an example image tensor (After Model-training)

[54]:
```
model_feat = UNet(n_channels=3, n_classes=1)
```

[50]:
```
model_feat.load_state_dict(model.state_dict())
```

[50]:
```
<All keys matched successfully>
```

[46]:
```
plt.figure(figsize = (20,10))


for i, sample in enumerate(aerial_validation_loader):
  img_var = Variable(sample['image'][1,:,:,:].unsqueeze(0))
  print(img_var.type())

  out,img_feat_all,layers = model_feat(img_var)


  for j,img_feat_batch in enumerate(img_feat_all):

    img_feat_1 = img_feat_batch[0,:,:,:]
    plt.figure(figsize = (20,10))


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

    print(layers[j])

    if j>=9:
```

```
      print(img_numpy.shape)

      plt.imshow(img_numpy[0, :, :-1], cmap='gray')
      plt.show()
      print('Feature Extraction done for first image (without any training)')

      break

    # 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[ind-1, :, :-1], cmap='gray')
        ind += 1
    # show the figure


    plt.show()
  break
```

```
Output hidden; open in https://colab.research.google.com to view.
```

1.3 Obeservation

- This is of-course not a great model, the point was to show how image-features developed over-the course of training.
- The Model can be improved by either maybe using a pretrained-decoder like Resnet/VGG and/or a better loss function (tried dice,iou,bce,jaccard, combo of the previous ones), which can decrease and learn faster & better and/or augmentation. As you can notice, the BCE loss-function is decreasing, which is great, but not at a large-amount.

[63]:
```
!jupyter nbconvert --to html 'drive/My Drive/Colab Notebooks/
↪Aerial_Dataset_Feature_Extraction_Segmentation_UNet.ipynb'
```

```
[NbConvertApp] Converting notebook drive/My Drive/Colab
Notebooks/Aerial_Dataset_Feature_Extraction_Segmentation_UNet.ipynb to html
[NbConvertApp] Writing 1152538 bytes to drive/My Drive/Colab
Notebooks/Aerial_Dataset_Feature_Extraction_Segmentation_UNet.html
```

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Aerial_Dataset_Feature_Extraction_Segmentation_UNet.ipynb')
```

```
File colab_pdf.py already there; not retrieving.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

[ ]:
```

```