TGS Salt Identification

This competition is about finding salt or no salt in the given seismic images.This task requires us to label every pixel thus this is a case for semantic segmentation.

Data Acquisition

https://www.kaggle.com/c/tgs-salt-identification-challenge/data

Evaluation Metric

Metric used is Mean Average Precision over different IOU Thresholds.

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [2]:

```
!pip install albumentations==0.4.6
!pip install tensorboardX
!pip install tensorboard
```

```
Collecting albumentations==0.4.6
  Downloading
https://files.pythonhosted.org/packages/92/33/1c459c2c9a4028ec75527eff88bc4e2d256555189f42af4baf4d7
233/albumentations-0.4.6.tar.gz (117kB)
     |████████████████████████████████| 122kB 9.0MB/s
Requirement already satisfied: numpy>=1.11.1 in /usr/local/lib/python3.6/dist-packages (from
albumentations==0.4.6) (1.19.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from
albumentations==0.4.6) (1.4.1)
Collecting imgaug>=0.4.0
  Downloading
https://files.pythonhosted.org/packages/66/b1/af3142c4a85cba6da9f4ebb5ff4e21e2616309552caca5e8acefe
622/imgaug-0.4.0-py2.py3-none-any.whl (948kB)
     |████████████████████████████████| 952kB 9.3MB/s
Requirement already satisfied: PyYAML in /usr/local/lib/python3.6/dist-packages (from
albumentations==0.4.6) (3.13)
Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/python3.6/dist-packages
(from albumentations==0.4.6) (4.1.2.30)
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from
imgaug>=0.4.0->albumentations==0.4.6) (7.0.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from
imgaug>=0.4.0->albumentations==0.4.6) (3.2.2)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from imgaug>=0.4.0->
albumentations==0.4.6) (1.15.0)
Requirement already satisfied: Shapely in /usr/local/lib/python3.6/dist-packages (from
imgaug>=0.4.0->albumentations==0.4.6) (1.7.1)
Requirement already satisfied: imageio in /usr/local/lib/python3.6/dist-packages (from
imgaug>=0.4.0->albumentations==0.4.6) (2.4.1)
Requirement already satisfied: scikit-image>=0.14.2 in /usr/local/lib/python3.6/dist-packages
(from imgaug>=0.4.0->albumentations==0.4.6) (0.16.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages
(from matplotlib->imgaug>=0.4.0->albumentations==0.4.6) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib->imgaug>=0.4.0->albumentations==0.4.6) (2.
4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from
matplotlib->imgaug>=0.4.0->albumentations==0.4.6) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from
matplotlib->imgaug>=0.4.0->albumentations==0.4.6) (0.10.0)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image>=0.14.2->imgaug>=0.4.0->albumentations==0.4.6) (2.5)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from
```

```
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image>=0.14.2->imgaug>=0.4.0->albumentations==0.4.6) (1.1.1)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from
networkx>=2.0->scikit-image>=0.14.2->imgaug>=0.4.0->albumentations==0.4.6) (4.4.2)
Building wheels for collected packages: albumentations
  Building wheel for albumentations (setup.py) ... done
  Created wheel for albumentations: filename=albumentations-0.4.6-cp36-none-any.whl size=65164
sha256=eef96a1f3cb123bc46a3165c7441a4c85ea2b4bc21ffd28dbf5cdc677d55d912
  Stored in directory:
/root/.cache/pip/wheels/c7/f4/89/56d1bee5c421c36c1a951eeb4adcc32fbb82f5344c086efa14
Successfully built albumentations
Installing collected packages: imgaug, albumentations
  Found existing installation: imgaug 0.2.9
    Uninstalling imgaug-0.2.9:
      Successfully uninstalled imgaug-0.2.9
  Found existing installation: albumentations 0.1.12
    Uninstalling albumentations-0.1.12:
      Successfully uninstalled albumentations-0.1.12
Successfully installed albumentations-0.4.6 imgaug-0.4.0
Collecting tensorboardX
  Downloading
https://files.pythonhosted.org/packages/af/0c/4f41bcd45db376e6fe5c619c01100e9b7531c55791b7244815bac
32c/tensorboardX-2.1-py2.py3-none-any.whl (308kB)
     |████████████████████████████████| 317kB 8.5MB/s
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from tensorboardX)
(1.15.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from tensorboardX)
(1.19.4)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.6/dist-packages (from
tensorboardX) (3.12.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from
protobuf>=3.8.0->tensorboardX) (51.0.0)
Installing collected packages: tensorboardX
Successfully installed tensorboardX-2.1
Requirement already satisfied: tensorboard in /usr/local/lib/python3.6/dist-packages (2.4.0)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (0.10.0)
Requirement already satisfied: numpy>=1.12.0 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (1.19.4)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.6/dist-p
ackages (from tensorboard) (0.4.2)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.6/dist-
packages (from tensorboard) (1.7.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (1.0.1)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (51.0.0)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (1.32.0)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in
/usr/local/lib/python3.6/dist-packages (from tensorboard) (0.36.2)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (2.23.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (1.15.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (3.3.3)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.6/dist-packages
(from tensorboard) (1.17.2)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.6/dist-packages (from
tensorboard) (3.12.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.6/dist-packages
(from google-auth-oauthlib<0.5,>=0.4.1->tensorboard) (1.3.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard) (2020.12.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tensorboard) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard) (2.10)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
/usr/local/lib/python3.6/dist-packages (from markdown>=2.6.8->tensorboard) (3.3.0)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages
(from google-auth<2,>=1.6.3->tensorboard) (4.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist-packages
(from google-auth<2,>=1.6.3->tensorboard) (0.2.8)
```

```
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in
/usr/local/lib/python3.6/dist-packages (from google-auth<2,>=1.6.3->tensorboard) (4.6)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from
requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in
/usr/local/lib/python3.6/dist-packages (from importlib-metadata; python_version < "3.8"-
>markdown>=2.6.8->tensorboard) (3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from
importlib-metadata; python_version < "3.8"->markdown>=2.6.8->tensorboard) (3.4.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.6/dist-packages
(from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard) (0.4.8)
```

In [3]:

```
import os
import numpy as np
import imageio
import matplotlib.pyplot as plt
import pandas as pd
import random
import math
import time
import pickle
from sklearn.model_selection import StratifiedKFold,train_test_split
from tqdm import tqdm
from datetime import datetime
import cv2
from pathlib import Path
import time
import torch
import torch.nn as nn
import torchvision
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms,models
from torch.utils import data
from torch.utils.data import random_split
from  torchsummary import summary
from torch.utils.tensorboard import SummaryWriter
from tensorboardX import SummaryWriter
from albumentations import Compose, RandomCrop, Normalize, HorizontalFlip, Resize,PadIfNeeded,Rando
mBrightness,Rotate,OneOf
from albumentations.pytorch import ToTensor
from matplotlib.image import imread
import PIL.Image
from albumentations.augmentations.transforms import
Blur,RandomContrast,ShiftScaleRotate,Cutout,VerticalFlip,RandomGamma,RandomResizedCrop
from albumentations.pytorch.transforms import ToTensorV2
```

In [4]:

```
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
```

In [5]:

```
%cd '/content/drive/My Drive/workspace'
```

```
/content/drive/My Drive/workspace
```

In [6]:

```
#for replicting results
seed=3
def set_seed(seed=3):
  torch.manual_seed(seed)
  torch.cuda.manual_seed(seed)
  np.random.seed(seed)   # Numpy module.
  random.seed(seed)  # Python random module.
  torch.manual_seed(seed)
  torch.backends.cudnn.benchmark = False
  torch.backends.cudnn.deterministic = True
  os.environ['PYTHONHASHSEED'] = str(seed)
```

```
os.environ['PYTHONHASHSEED'] = str(seed)
```

In [7]:

```
#used for reproducing results
def _init_fn(worker_id):
    np.random.seed(int(seed))
```

Common

Dataset

In [8]:

```
mask_path= '/content/drive/MyDrive/competition_data/train/masks/2ffea0c397.png'
image_path='/content/drive/MyDrive/competition_data/train/images/2ffea0c397.png'
image=PIL.Image.open(image_path).convert('RGB')
image= np.array(image).astype(np.float32)
print(imread(mask_path)[:5,:5]),print((image)[0,:5,:5])
```

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
[[168. 168. 168.]
 [174. 174. 174.]
 [176. 176. 176.]
 [177. 177. 177.]
 [179. 179. 179.]]
```

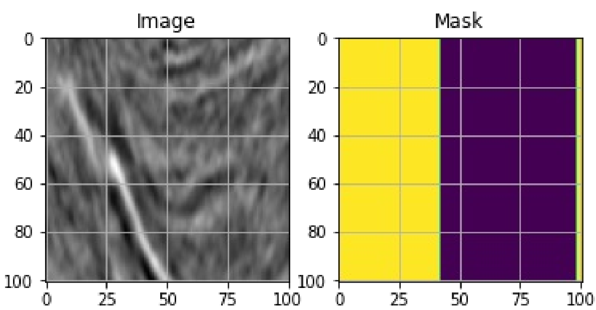
Out[8]:

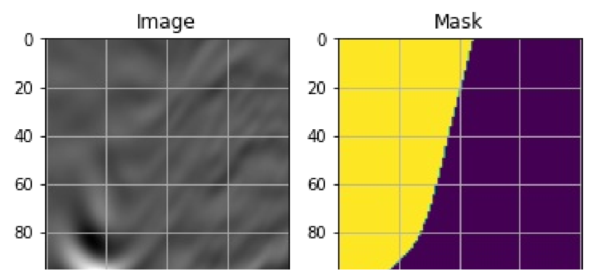
```
(None, None)
```

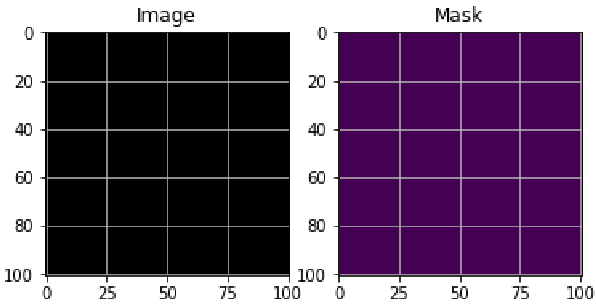
X represents 3x101x101 dimension images,Y represents with 101x101 binary masks,X has 0 to 255 range values, Y has 0,1

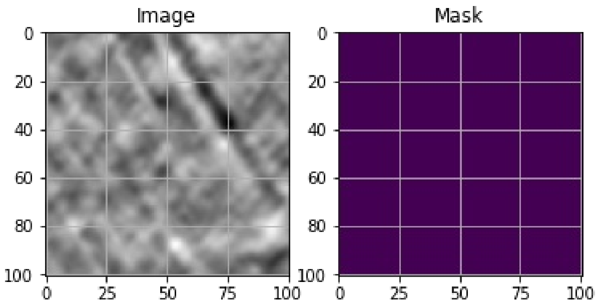
In [9]:

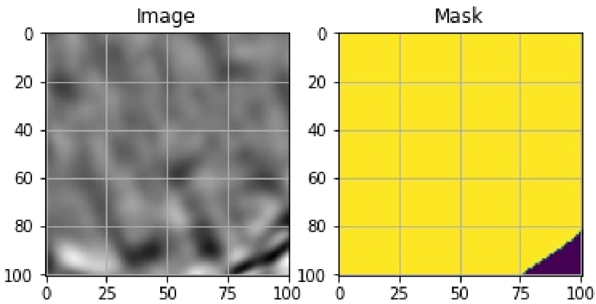
```
directory= "../competition_data"
depths_df = pd.read_csv(os.path.join(directory, 'trainanddepth.csv'))
depth= pd.read_csv(os.path.join(directory, 'depths.csv'))
train_path = os.path.join(directory, 'train')
file_list = list(depths_df['id'].values)
```











As above some anamolies(outliers) present such as vertical masks,black images with empty masks.

Change in brightness and contrast between images can be observed, blur of images can also be observed.

In [ ]:

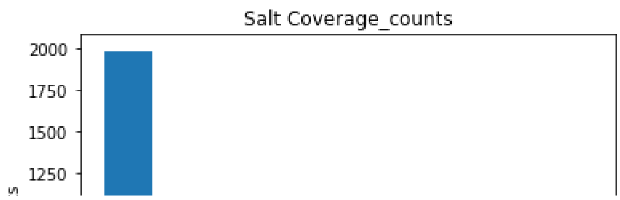
```
#salt content over the data 1 refers to <=10% salt
coverage= depths_df['salt_content'].values
```

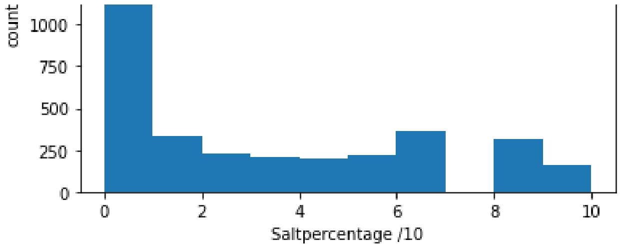
In [ ]:

```
plt.hist(coverage)
plt.title('Salt Coverage_counts')
plt.xlabel('Saltpercentage /10')
plt.ylabel('counts')
```

Out[ ]:

```
Text(0, 0.5, 'counts')
```





Abnormal amounts of no salt present.This problem can also be posed as finding no salt first and then applying semantic segmenation pipeline only on those masks for which salt is actually present.

In [10]:

```
#most basic transforms
def_transforms= Compose([
    PadIfNeeded(128,128,cv2.BORDER_REPLICATE),
    Normalize(mean=(0,0,0),std=(1,1,1,))])
```

In [11]:

```
class TGSSaltDataset(data.Dataset):
    def __init__(self, root_path, file_list, is_test = False,augment= def_transforms,dpsv=False):
        self.is_test = is_test
        self.root_path = root_path
        self.file_list = file_list
        self.augment= augment
        self.dpsv= dpsv
    def __len__(self):
        return len(self.file_list)

    def __getitem__(self, index):

        file_id = self.file_list[index]

        image_folder = os.path.join(self.root_path, "images")
        image_path = os.path.join(image_folder, file_id + ".png")

        mask_folder = os.path.join(self.root_path, "masks")
        mask_path = os.path.join(mask_folder, file_id + ".png")

        image = PIL.Image.open(image_path).convert('RGB')
        image= np.array(image).astype(np.float32)
        image=np.clip(image - np.median(image) +127, 0, 255)#remove all the noise which inherently
exists in the data
        if self.is_test:
            data= {"image":image}
            X= self.augment(**data)
            image= X["image"]
            return torch.FloatTensor(image).permute(2,0,1)
        else:
            mask= imread(mask_path)
            mask= np.array(mask)
            data= {"image":image,"mask":mask}
            X= self.augment(**data)
            if self.dpsv:
              others=get_others(X['mask'])
              return  (torch.FloatTensor(X["image"]).permute(2,0,1),*others)

            image,mask= torch.FloatTensor(X["image"]).permute(2,0,1),torch.FloatTensor(X["mask"]).u
nsqueeze(0)

            return (image, mask)
```

kaggle metric

In [12]:

```
#https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/67693
def iou_metric(outputs, labels,logits=True):
    outputs,labels= size_correct(outputs,labels)
```

```
    outputs= (outputs>0).detach().cpu().numpy() if logits else (outputs>.5).detach().cpu().numpy()
    labels= labels.detach().cpu().numpy()
    batch_size = outputs.shape[0]
    metric = 0.0
    for batch in range(batch_size):
        t, p = labels[batch], outputs[batch]
        true = np.sum(t)
        pred = np.sum(p)

        # deal with empty mask first
        if true == 0:
            metric += (pred == 0)
            continue

        # non empty mask case.  Union is never empty
        # hence it is safe to divide by its number of pixels
        intersection = np.sum(t * p)
        union = true + pred - intersection
        iou = intersection / union

        # iou metrric is a stepwise approximation of the real iou over 0.5
        iou = np.floor(max(0, (iou - 0.45)*20)) / 10
        iou= np.clip(iou,0,1.0)

        metric += iou

    # teake the average over all images in batch
    metric /= batch_size
    return metric
```

util

In [ ]:

```
#https://github.com/Bjarten/early-stopping-pytorch/blob/master/pytorchtools.py
class EarlyStopping:
    def __init__(self, patience=50, verbose=False, delta=0.003, path='./models/checkpoint.pth', tra
ce_func=print):

        self.patience = patience
        self.verbose = verbose
        self.counter = 0
        self.best_score = None
        self.early_stop = False
        self.val_metric_min = np.Inf
        self.best_model= None
        self.delta = delta
        self.path = path
        self.trace_func = trace_func

    def __call__(self, val_metric, model):

        score = val_metric

        if self.best_score is None:
            self.best_score = score
            self.save_checkpoint(val_metric, model)
        elif score < self.best_score + self.delta:
            self.counter += 1
            if self.verbose:
              self.trace_func(f'EarlyStopping counter: {self.counter} out of {self.patience}')
            if self.counter >= self.patience:
                self.early_stop = True
        else:
            self.best_score = score
            self.save_checkpoint(val_metric, model)
            self.counter = 0

    def save_checkpoint(self, val_metric, model):
        '''Saves model when validation loss decrease.'''
        if self.verbose:
            self.trace_func(f'Validation loss decreased ({self.val_metric_min:.6f} --> {val_metric:
.6f}).  Saving model ...')
        self.best_model= model.state_dict()
        torch.save(model.state_dict(), self.path)
```

```
        torch.save(model.state_dict(), self.path)
        self.val_metric_min = val_metric
```

Training

In [ ]:

```
train_ind= list(range(len(file_list)))
```

The following augmentations have been chosen as they are very close to the real data given.

In [ ]:

```
transform_train = Compose([
    HorizontalFlip(p=.5),
     Compose([RandomCrop(90,90),
     Resize(101,101)],p=.3),
    OneOf([RandomBrightness(.1),
    RandomContrast(.1),RandomGamma()],p=.2),
    PadIfNeeded(256//2,256//2,cv2.BORDER_REPLICATE),
    Normalize(mean=(0,0,0),std=(1,1,1,))])

transform_test= Compose([
PadIfNeeded(256//2,256//2,cv2.BORDER_REPLICATE),
Normalize(mean=(0,0,0),std=(1,1,1,))])
```

In [ ]:

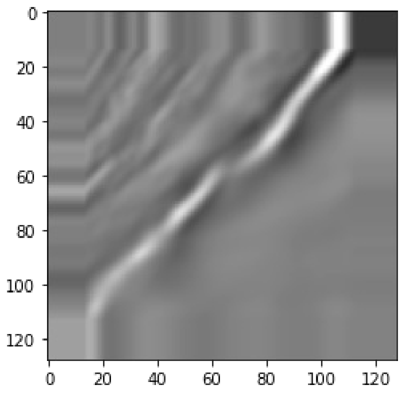
```
check= TGSSaltDataset(train_path,file_list,augment=transform_train)
```

In [ ]:

```
plt.imshow(check[0][0].permute(2,1,0))
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7fe2e3f0d780>
```



In [14]:

```
FLAGS = {}
FLAGS['data_dir'] = "../competition_data"
FLAGS['batch_size'] = 16
FLAGS['num_workers'] = 4
FLAGS['learning_rate'] = 1e-9
FLAGS['num_epochs'] = 3
FLAGS['log_dir']='./tensorboard/'
```

loss

In [ ]:

```
def acc(pred,mask,center=False):
  "calculates the accuracy"
  pred,mask= pred.detach().cpu().numpy(),mask.detach().cpu().numpy()
  if not center:
    pred,mask= size_correct(pred,mask)
  pred=pred>0
  res= (pred==mask).sum()
  return res
```

In [ ]:

```
"""
Lovasz-Softmax and Jaccard hinge loss in PyTorch
Maxim Berman 2018 ESAT-PSI KU Leuven (MIT License)
"""

from torch.autograd import Variable

def lovasz_grad(gt_sorted):
    """
    Computes gradient of the Lovasz extension w.r.t sorted errors
    See Alg. 1 in paper
    """
    p = len(gt_sorted)
    gts = gt_sorted.sum()
    intersection = gts - gt_sorted.float().cumsum(0)
    union = gts + (1 - gt_sorted).float().cumsum(0)
    jaccard = 1. - intersection / union
    if p > 1: # cover 1-pixel case
        jaccard[1:p] = jaccard[1:p] - jaccard[0:-1]
    return jaccard

# --------------------------- BINARY LOSSES ---------------------------


def lovasz_hinge(logits, labels, per_image=True, ignore=None):


    """
    Binary Lovasz hinge loss
      logits: [B, H, W] Variable, logits at each pixel (between -\infty and +\infty)
      labels: [B, H, W] Tensor, binary ground truth masks (0 or 1)
      per_image: compute the loss per image instead of per batch
      ignore: void class id
    """
    logits= logits.squeeze(1)
    labels= labels.squeeze(1)

    if per_image:
        loss = mean(lovasz_hinge_flat(*flatten_binary_scores(log.unsqueeze(0), lab.unsqueeze(0), ig
nore))
                          for log, lab in zip(logits, labels))
    else:
        loss = lovasz_hinge_flat(*flatten_binary_scores(logits, labels, ignore))
    return loss


def lovasz_hinge_flat(logits, labels):
    """
    Binary Lovasz hinge loss
      logits: [P] Variable, logits at each prediction (between -\infty and +\infty)
      labels: [P] Tensor, binary ground truth labels (0 or 1)
      ignore: label to ignore
    """
    if len(labels) == 0:
        # only void pixels, the gradients should be 0
        return logits.sum() * 0.
    signs = 2. * labels.float() - 1.
    errors = (1. - logits * Variable(signs))
    errors_sorted, perm = torch.sort(errors, dim=0, descending=True)
    perm = perm.data
    gt_sorted = labels[perm]
    grad = lovasz_grad(gt_sorted)
    loss = torch.dot(F.elu(errors_sorted)+1, Variable(grad))
    return loss
```

```
def flatten_binary_scores(scores, labels, ignore=None):
    """
    Flattens predictions in the batch (binary case)
    Remove labels equal to 'ignore'
    """
    scores = scores.view(-1)
    labels = labels.view(-1)
    if ignore is None:
        return scores, labels
    valid = (labels != ignore)
    vscores = scores[valid]
    vlabels = labels[valid]
    return vscores, vlabels


# --------------------------- HELPER FUNCTIONS ---------------------------
def isnan(x):
    return x != x


def mean(l, ignore_nan=False, empty=0):
    """
    nanmean compatible with generators.
    """
    l = iter(l)
    if ignore_nan:
        l = ifilterfalse(isnan, l)
    try:
        n = 1
        acc = next(l)
    except StopIteration:
        if empty == 'raise':
            raise ValueError('Empty mean')
        return empty
    for n, v in enumerate(l, 2):
        acc += v
    if n == 1:
        return acc
    return acc / n
```

In [ ]:

```
"this function makes sure that metric is computed on 101x101 so that there is a better correlation with kaggle score."
def size_correct(logits,mask):
  if logits.shape[-1]==128:
    logits_= logits[:,:,13:-14,13:-14]
    mask_= mask[:,:,13:-14,13:-14]
  if logits.shape[-1]==128*2:
    logits_= logits[:,:,27:-27,27:-27]
    mask_= mask[:,:,27:-27,27:-27]
  return logits_,mask_
```

In [ ]:

```
class LovaszLoss(nn.Module):
  def forward(self,logits,targets):
    assert(len(logits.shape)==4)
    return lovasz_hinge(logits,targets,per_image=True)
```

Model Preparation

In [ ]:

```
def train_loop_fn(loader):
  model.train()
  loss_,iou_,acc_=0.,0.,0.
  for x ,(image, mask) in enumerate(loader):
    mask=mask.to(device)
    image= image.to(device)
    y_pred = model(image)
    loss = loss_fn(y_pred, mask)
```

```
    loss = loss_fn(y_pred, mask)
    loss_+= loss.item()
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    ACC=acc(y_pred,mask)
    IOU= iou_metric(y_pred,mask)
    acc_+=ACC.item()
    iou_+=IOU.item()
    if scheduler:
      scheduler.step()

  loss_/=len(loader)
  iou_/=len(loader)
  acc_/=(len(train)*101*101)
  print('Train[{}]: Loss={:.5f} IOU={:.3f} ACC={:.3f}'.format(
        x, loss_,iou_,acc_))

  return loss_,iou_,acc_
```

In [ ]:

```
def test_loop_fn(loader):
  loss_,iou_,acc_=0.,0.,0.,
  model.eval()
  with torch.no_grad():
    for image, mask in  loader:
      image = image.to(device)
      mask=mask.to(device)
      y_pred= model(image)
      loss = loss_fn(y_pred, mask)
      loss_+= loss.item()
      ACC=acc(y_pred,mask)
      IOU= iou_metric(y_pred,mask)
      acc_+=ACC.item()
      iou_+=IOU.item()
  loss_ /=len(loader)
  iou_/=len(loader)
  acc_/=(len(val)*1.0*101*101)
  # if epoch>30:
  # scheduler.step(iou_)
  early_stopping(iou_,model)
  print('Validation: Loss={:.5f},IOU={:.3f} ACC={:.3f}'.format(
      loss_,iou_,acc_))
  return loss_,iou_,acc_
```

In [ ]:

```
#https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/67693
class ConvBn2d(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size=(3,3), stride=(1,1), padding=(1,1)):
        super(ConvBn2d, self).__init__()

        self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=kernel_size, stride=stride, pa
dding=padding, bias=False)
        self.bn = nn.BatchNorm2d(out_channels)

    def forward(self, z):
        x = self.conv(z)
        x = self.bn(x)
        return x

class Decoder(nn.Module):
    def __init__(self, in_channels, channels, out_channels ):
        super(Decoder, self).__init__()
        self.conv1 =  ConvBn2d(in_channels,  channels, kernel_size=3, padding=1)
        self.conv2 =  ConvBn2d(channels, out_channels, kernel_size=3, padding=1)

    def forward(self, x ):
        x = F.upsample(x, scale_factor=2, mode='bilinear', align_corners=True)#False
        x = F.relu(self.conv1(x),inplace=True)
        x = F.relu(self.conv2(x),inplace=True)
        return x

class Baseline(nn.Module):
```

```python
    def __init__(self ):
        super().__init__()
        self.resnet = torchvision.models.resnet34(pretrained=True)

        self.conv1 = nn.Sequential(
            self.resnet.conv1,
            self.resnet.bn1,
            self.resnet.relu,
        )# 64
        self.encoder2 = self.resnet.layer1  # 64
        self.encoder3 = self.resnet.layer2  #128
        self.encoder4 = self.resnet.layer3  #256
        self.encoder5=  self.resnet.layer4  #512

        self.center = nn.Sequential(
            nn.Conv2d(512, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
        )

        self.decoder5 = Decoder(512+64, 512, 64)
        self.decoder4 = Decoder(256+64, 256, 64)
        self.decoder3 = Decoder(128+64, 128,  64)
        self.decoder2 = Decoder(64+64  , 64,  64)


        self.logit    = nn.Sequential(
            nn.Conv2d(64, 32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(32,  1, kernel_size=1, padding=0),
        )


    def forward(self, x):

        x = self.conv1(x)
        e2 = self.encoder2( x)  #; print('e2',e2.size())
        e3 = self.encoder3(e2)  #; print('e3',e3.size())
        e4 = self.encoder4(e3)  #; print('e4',e4.size())
        e5 = self.encoder5(e4)  #; print('e5',e5.size())

        f = self.center(e5)

        f = self.decoder5(torch.cat([f, e5], 1))  #; print('d5',f.size())
        f = self.decoder4(torch.cat([f, e4], 1))  #; print('d4',f.size())
        f = self.decoder3(torch.cat([f, e3], 1))  #; print('d3',f.size())
        f = self.decoder2(torch.cat([f, e2], 1))  #; print('d2',f.size())

        logit = self.logit(f)                     #; print('logit',logit.size())
        return logit

model=Baseline()
```

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to
/root/.cache/torch/hub/checkpoints/resnet34-333f7ec4.pth
```

In [ ]:

```python
kf = StratifiedKFold(10,shuffle=True,random_state=seed)
for fold, (trn_idx,val_idx) in enumerate(kf.split(file_list,coverage)):
    if fold!=0 :
      continue
    print('****************************************************')
    print('********************   fold  %d   ********************' % fold)
    print('*********************************************')

    file_list_train= [x for i,x in enumerate(file_list) if i in trn_idx]
    file_list_val= [x for i,x in enumerate(file_list) if i in val_idx]
    train = TGSSaltDataset(train_path, file_list_train,augment=transform_test)#minimal augmentation
s for baseline model
    val = TGSSaltDataset(train_path, file_list_val,augment= transform_test)
    writer =SummaryWriter(FLAGS['log_dir']+'Baseline')

    train_loader = torch.utils.data.DataLoader(
```

```
    train_loader = torch.utils.data.DataLoader(
        train,
        batch_size=64,
        num_workers=4,
        drop_last=False,worker_init_fn=_init_fn)
    test_loader = torch.utils.data.DataLoader(
        val,
        batch_size=FLAGS['batch_size']*5,
        shuffle=False,
        num_workers=2,
        drop_last=False,worker_init_fn=_init_fn)

    set_seed()
    device = 'cuda'
    model = model.to(device)
    loss_fn = nn.BCEWithLogitsLoss()
    optimizer = torch.optim.Adam(model.parameters(),lr= 3e-4)

    for epoch in range(1,31):
      train_loss,train_iou,train_acc=train_loop_fn(train_loader)
      print("Finished training epoch {}".format(epoch))
      val_loss,val_iou,val_acc= test_loop_fn(test_loader)
      if epoch==30:
        torch.save(model.state_dict(), './models/baseline.pth')
      writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
      writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)
```

```
***************************************************
*******************    fold  0    *******************
********************************************
Train[56]: Loss=0.19045 IOU=0.671 ACC=0.935
Finished training epoch 1
Validation: Loss=0.18495,IOU=0.679 ACC=0.932
Train[56]: Loss=0.16968 IOU=0.705 ACC=0.941
Finished training epoch 2
Validation: Loss=0.15853,IOU=0.712 ACC=0.940
Train[56]: Loss=0.14113 IOU=0.727 ACC=0.951
Finished training epoch 3
Validation: Loss=0.14940,IOU=0.737 ACC=0.947
Train[56]: Loss=0.12935 IOU=0.755 ACC=0.955
Finished training epoch 4
Validation: Loss=0.16131,IOU=0.718 ACC=0.941
Train[56]: Loss=0.11506 IOU=0.760 ACC=0.960
Finished training epoch 5
Validation: Loss=0.35990,IOU=0.659 ACC=0.896
Train[56]: Loss=0.11235 IOU=0.756 ACC=0.962
Finished training epoch 6
Validation: Loss=0.17880,IOU=0.752 ACC=0.943
Train[56]: Loss=0.10761 IOU=0.771 ACC=0.963
Finished training epoch 7
Validation: Loss=0.19045,IOU=0.745 ACC=0.941
Train[56]: Loss=0.09349 IOU=0.788 ACC=0.967
Finished training epoch 8
Validation: Loss=0.15526,IOU=0.739 ACC=0.945
Train[56]: Loss=0.06920 IOU=0.818 ACC=0.977
Finished training epoch 9
Validation: Loss=0.13031,IOU=0.775 ACC=0.957
Train[56]: Loss=0.05859 IOU=0.835 ACC=0.980
Finished training epoch 10
Validation: Loss=0.17271,IOU=0.755 ACC=0.948
Train[56]: Loss=0.05477 IOU=0.839 ACC=0.980
Finished training epoch 11
Validation: Loss=0.15839,IOU=0.780 ACC=0.958
Train[56]: Loss=0.05399 IOU=0.839 ACC=0.981
Finished training epoch 12
Validation: Loss=0.20756,IOU=0.711 ACC=0.940
Train[56]: Loss=0.07217 IOU=0.807 ACC=0.975
Finished training epoch 13
Validation: Loss=0.15976,IOU=0.763 ACC=0.952
Train[56]: Loss=0.06129 IOU=0.828 ACC=0.978
Finished training epoch 14
Validation: Loss=0.16628,IOU=0.749 ACC=0.950
Train[56]: Loss=0.05043 IOU=0.848 ACC=0.982
Finished training epoch 15
Validation: Loss=0.16101,IOU=0.790 ACC=0.956
Train[56]: Loss=0.03837 IOU=0.862 ACC=0.985
Finished training epoch 16
```

```
Validation: Loss=0.15531,IOU=0.775 ACC=0.957
Train[56]: Loss=0.04000 IOU=0.864 ACC=0.984
Finished training epoch 17
Validation: Loss=0.21001,IOU=0.760 ACC=0.949
Train[56]: Loss=0.04044 IOU=0.862 ACC=0.985
Finished training epoch 18
Validation: Loss=0.17333,IOU=0.770 ACC=0.945
Train[56]: Loss=0.03149 IOU=0.881 ACC=0.988
Finished training epoch 19
Validation: Loss=0.15007,IOU=0.791 ACC=0.961
Train[56]: Loss=0.02510 IOU=0.893 ACC=0.990
Finished training epoch 20
Validation: Loss=0.16425,IOU=0.791 ACC=0.961
Train[56]: Loss=0.02287 IOU=0.900 ACC=0.990
Finished training epoch 21
Validation: Loss=0.18477,IOU=0.794 ACC=0.959
Train[56]: Loss=0.02049 IOU=0.906 ACC=0.991
Finished training epoch 22
Validation: Loss=0.18892,IOU=0.801 ACC=0.961
Train[56]: Loss=0.01858 IOU=0.912 ACC=0.992
Finished training epoch 23
Validation: Loss=0.21607,IOU=0.788 ACC=0.958
Train[56]: Loss=0.01750 IOU=0.913 ACC=0.992
Finished training epoch 24
Validation: Loss=0.20729,IOU=0.790 ACC=0.960
Train[56]: Loss=0.01658 IOU=0.916 ACC=0.993
Finished training epoch 25
Validation: Loss=0.21377,IOU=0.789 ACC=0.961
Train[56]: Loss=0.01623 IOU=0.919 ACC=0.993
Finished training epoch 26
Validation: Loss=0.21511,IOU=0.790 ACC=0.959
Train[56]: Loss=0.01529 IOU=0.923 ACC=0.993
Finished training epoch 27
Validation: Loss=0.22262,IOU=0.794 ACC=0.961
Train[56]: Loss=0.01415 IOU=0.926 ACC=0.993
Finished training epoch 28
Validation: Loss=0.23453,IOU=0.796 ACC=0.959
Train[56]: Loss=0.01463 IOU=0.924 ACC=0.993
Finished training epoch 29
Validation: Loss=0.23092,IOU=0.796 ACC=0.959
Train[56]: Loss=0.01507 IOU=0.920 ACC=0.993
Finished training epoch 30
Validation: Loss=0.21520,IOU=0.785 ACC=0.959
```

In [32]:

```
%reload_ext tensorboard
%tensorboard --logdir {FLAGS['log_dir']+'Baseline'}
```

Clearly overfitting as seen from above.lets add augmentations to regularise the training.Dropout wasn't chosen here as it slows the training.

In [ ]:

```
transform_train = Compose([
    HorizontalFlip(p=.5),
     Compose([RandomCrop(90,90),
     Resize(101,101)],p=.2),
    OneOf([RandomBrightness(.1),
    RandomContrast(.1),RandomGamma()],p=.2),
    PadIfNeeded(256//2,256//2,cv2.BORDER_REPLICATE),
    Normalize(mean=(0,0,0),std=(1,1,1,))])

transform_test= Compose([
PadIfNeeded(256//2,256//2,cv2.BORDER_REPLICATE),
Normalize(mean=(0,0,0),std=(1,1,1,))])

transform_train_lv = Compose([
    HorizontalFlip(p=.5),
     Compose([RandomCrop(80,80),
     Resize(101,101)],p=.4),
    OneOf([RandomBrightness(.1),
    RandomContrast(.1),RandomGamma()],p=.4),
```

```
    PadIfNeeded(256//2,256//2,cv2.BORDER_REPLICATE),
    Normalize(mean=(0,0,0),std=(1,1,1,))])
```

In [ ]:

```
kf = StratifiedKFold(10,shuffle=True,random_state=seed)
for fold, (trn_idx,val_idx) in enumerate(kf.split(file_list,coverage)):
    if fold!=0 :
      continue
    print('***************************************************')
    print('*******************   fold  %d   *******************' % fold)
    print('********************************************')

    file_list_train= [x for i,x in enumerate(file_list) if i in trn_idx]
    file_list_val= [x for i,x in enumerate(file_list) if i in val_idx]
    train = TGSSaltDataset(train_path, file_list_train,augment=transform_train)
    val = TGSSaltDataset(train_path, file_list_val,augment= transform_test)
    early_stopping = EarlyStopping(patience=80,path= './models/baseline_corrected.pth', verbose=Fal
se)
    writer =SummaryWriter(FLAGS['log_dir']+'Baseline_corrected')

    train_loader = torch.utils.data.DataLoader(
        train,
        batch_size=64,
        num_workers=4,
        drop_last=False,worker_init_fn=_init_fn)
    test_loader = torch.utils.data.DataLoader(
        val,
        batch_size=FLAGS['batch_size']*5,
        shuffle=False,
        num_workers=2,
        drop_last=False,worker_init_fn=_init_fn)

    set_seed()
    device = 'cuda'
    model=Baseline()
    model = model.to(device)
    loss_fn = nn.BCEWithLogitsLoss()
    optimizer = torch.optim.Adam(model.parameters(),lr= 3e-4)

    for epoch in range(1,51):
      train_loss,train_iou,train_acc=train_loop_fn(train_loader)
      print("Finished training epoch {}".format(epoch))
      val_loss,val_iou,val_acc= test_loop_fn(test_loader)
      writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
      writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)
```

```
***************************************************
*******************   fold  0   *******************
********************************************
Train[56]: Loss=0.36946 IOU=0.533 ACC=0.858
Finished training epoch 1
Validation: Loss=0.24362,IOU=0.586 ACC=0.910
Train[56]: Loss=0.26712 IOU=0.617 ACC=0.898
Finished training epoch 2
Validation: Loss=0.23556,IOU=0.614 ACC=0.917
Train[56]: Loss=0.24512 IOU=0.645 ACC=0.908
Finished training epoch 3
Validation: Loss=0.18234,IOU=0.709 ACC=0.932
Train[56]: Loss=0.22686 IOU=0.675 ACC=0.914
Finished training epoch 4
Validation: Loss=0.16146,IOU=0.733 ACC=0.934
Train[56]: Loss=0.21982 IOU=0.675 ACC=0.914
Finished training epoch 5
Validation: Loss=0.15068,IOU=0.747 ACC=0.940
Train[56]: Loss=0.21096 IOU=0.690 ACC=0.918
Finished training epoch 6
Validation: Loss=0.15916,IOU=0.744 ACC=0.938
Train[56]: Loss=0.21764 IOU=0.684 ACC=0.912
Finished training epoch 7
Validation: Loss=0.14960,IOU=0.741 ACC=0.941
Train[56]: Loss=0.20333 IOU=0.702 ACC=0.921
Finished training epoch 8
Validation: Loss=0.14906,IOU=0.753 ACC=0.940
Train[56]: Loss=0.19298 IOU=0.718 ACC=0.927
Finished training epoch 9
```

```
Finished training epoch 9
Validation: Loss=0.21187,IOU=0.664 ACC=0.926
Train[56]: Loss=0.19797 IOU=0.715 ACC=0.921
Finished training epoch 10
Validation: Loss=0.14262,IOU=0.752 ACC=0.939
Train[56]: Loss=0.19515 IOU=0.716 ACC=0.924
Finished training epoch 11
Validation: Loss=0.13798,IOU=0.749 ACC=0.946
Train[56]: Loss=0.19545 IOU=0.712 ACC=0.924
Finished training epoch 12
Validation: Loss=0.88848,IOU=0.251 ACC=0.732
Train[56]: Loss=0.20010 IOU=0.704 ACC=0.920
Finished training epoch 13
Validation: Loss=0.18466,IOU=0.761 ACC=0.933
Train[56]: Loss=0.18511 IOU=0.720 ACC=0.929
Finished training epoch 14
Validation: Loss=0.13656,IOU=0.783 ACC=0.949
Train[56]: Loss=0.17592 IOU=0.734 ACC=0.931
Finished training epoch 15
Validation: Loss=0.15923,IOU=0.746 ACC=0.936
Train[56]: Loss=0.18089 IOU=0.725 ACC=0.929
Finished training epoch 16
Validation: Loss=0.14439,IOU=0.787 ACC=0.949
Train[56]: Loss=0.17713 IOU=0.734 ACC=0.930
Finished training epoch 17
Validation: Loss=0.11912,IOU=0.794 ACC=0.951
Train[56]: Loss=0.17040 IOU=0.741 ACC=0.931
Finished training epoch 18
Validation: Loss=0.13446,IOU=0.751 ACC=0.947
Train[56]: Loss=0.18142 IOU=0.728 ACC=0.928
Finished training epoch 19
Validation: Loss=0.17079,IOU=0.719 ACC=0.936
Train[56]: Loss=0.16986 IOU=0.742 ACC=0.933
Finished training epoch 20
Validation: Loss=0.14466,IOU=0.774 ACC=0.946
Train[56]: Loss=0.17576 IOU=0.725 ACC=0.930
Finished training epoch 21
Validation: Loss=0.11839,IOU=0.782 ACC=0.954
Train[56]: Loss=0.16225 IOU=0.744 ACC=0.937
Finished training epoch 22
Validation: Loss=0.13484,IOU=0.799 ACC=0.957
Train[56]: Loss=0.16068 IOU=0.754 ACC=0.938
Finished training epoch 23
Validation: Loss=0.15041,IOU=0.792 ACC=0.950
Train[56]: Loss=0.15253 IOU=0.756 ACC=0.940
Finished training epoch 24
Validation: Loss=0.13413,IOU=0.797 ACC=0.949
Train[56]: Loss=0.14870 IOU=0.759 ACC=0.941
Finished training epoch 25
Validation: Loss=0.11775,IOU=0.819 ACC=0.959
Train[56]: Loss=0.15637 IOU=0.752 ACC=0.938
Finished training epoch 26
Validation: Loss=0.13028,IOU=0.798 ACC=0.950
Train[56]: Loss=0.14546 IOU=0.766 ACC=0.943
Finished training epoch 27
Validation: Loss=0.11104,IOU=0.800 ACC=0.956
Train[56]: Loss=0.13960 IOU=0.769 ACC=0.944
Finished training epoch 28
Validation: Loss=0.15722,IOU=0.814 ACC=0.954
Train[56]: Loss=0.14771 IOU=0.759 ACC=0.942
Finished training epoch 29
Validation: Loss=0.12517,IOU=0.799 ACC=0.953
Train[56]: Loss=0.14030 IOU=0.766 ACC=0.945
Finished training epoch 30
Validation: Loss=0.12547,IOU=0.746 ACC=0.953
Train[56]: Loss=0.13841 IOU=0.770 ACC=0.944
Finished training epoch 31
Validation: Loss=0.13757,IOU=0.805 ACC=0.953
Train[56]: Loss=0.13881 IOU=0.772 ACC=0.946
Finished training epoch 32
Validation: Loss=0.13378,IOU=0.795 ACC=0.952
Train[56]: Loss=0.13754 IOU=0.769 ACC=0.945
Finished training epoch 33
Validation: Loss=0.17814,IOU=0.792 ACC=0.946
Train[56]: Loss=0.14625 IOU=0.759 ACC=0.940
Finished training epoch 34
Validation: Loss=0.11106,IOU=0.808 ACC=0.958
Train[56]: Loss=0.14174 IOU=0.771 ACC=0.941
```

```
Train[56]: Loss=0.14174 IOU=0.771 ACC=0.941
Finished training epoch 35
Validation: Loss=0.10648,IOU=0.819 ACC=0.964
Train[56]: Loss=0.14531 IOU=0.768 ACC=0.943
Finished training epoch 36
Validation: Loss=0.15055,IOU=0.775 ACC=0.944
Train[56]: Loss=0.14705 IOU=0.761 ACC=0.942
Finished training epoch 37
Validation: Loss=0.14104,IOU=0.785 ACC=0.947
Train[56]: Loss=0.13623 IOU=0.776 ACC=0.946
Finished training epoch 38
Validation: Loss=0.14578,IOU=0.791 ACC=0.953
Train[56]: Loss=0.14186 IOU=0.772 ACC=0.943
Finished training epoch 39
Validation: Loss=0.10877,IOU=0.811 ACC=0.962
Train[56]: Loss=0.13596 IOU=0.779 ACC=0.947
Finished training epoch 40
Validation: Loss=0.14720,IOU=0.772 ACC=0.948
Train[56]: Loss=0.12732 IOU=0.782 ACC=0.949
Finished training epoch 41
Validation: Loss=0.11830,IOU=0.807 ACC=0.957
Train[56]: Loss=0.13743 IOU=0.774 ACC=0.943
Finished training epoch 42
Validation: Loss=0.11755,IOU=0.814 ACC=0.961
Train[56]: Loss=0.14553 IOU=0.757 ACC=0.939
Finished training epoch 43
Validation: Loss=0.13910,IOU=0.808 ACC=0.960
Train[56]: Loss=0.13990 IOU=0.769 ACC=0.942
Finished training epoch 44
Validation: Loss=0.13095,IOU=0.809 ACC=0.955
Train[56]: Loss=0.14876 IOU=0.752 ACC=0.940
Finished training epoch 45
Validation: Loss=0.13368,IOU=0.786 ACC=0.947
Train[56]: Loss=0.14483 IOU=0.768 ACC=0.942
Finished training epoch 46
Validation: Loss=0.12203,IOU=0.796 ACC=0.957
Train[56]: Loss=0.13473 IOU=0.782 ACC=0.947
Finished training epoch 47
Validation: Loss=0.13747,IOU=0.793 ACC=0.948
Train[56]: Loss=0.12944 IOU=0.785 ACC=0.947
Finished training epoch 48
Validation: Loss=0.12505,IOU=0.815 ACC=0.963
Train[56]: Loss=0.13839 IOU=0.778 ACC=0.945
Finished training epoch 49
Validation: Loss=0.10488,IOU=0.812 ACC=0.962
Train[56]: Loss=0.13599 IOU=0.785 ACC=0.944
Finished training epoch 50
Validation: Loss=0.11148,IOU=0.811 ACC=0.958
```

In [ ]:

```
kf = StratifiedKFold(10,shuffle=True,random_state=seed)
for fold, (trn_idx,val_idx) in enumerate(kf.split(file_list,coverage)):
    if fold!=0 :
      continue
    print('*****************************************************')
    print('*******************    fold  %d    *******************' % fold)
    print('*****************************************************')

    file_list_train= [x for i,x in enumerate(file_list) if i in trn_idx]
    file_list_val= [x for i,x in enumerate(file_list) if i in val_idx]
    train = TGSSaltDataset(train_path, file_list_train,augment=transform_train)
    val = TGSSaltDataset(train_path, file_list_val,augment= transform_test)
    writer =SummaryWriter(FLAGS['log_dir']+'Baseline_lovasz')

    train_loader = torch.utils.data.DataLoader(
        train,
        batch_size=64,
        num_workers=4,
        drop_last=False,worker_init_fn=_init_fn)
    test_loader = torch.utils.data.DataLoader(
        val,
        batch_size=FLAGS['batch_size']*5,
        shuffle=False,
        num_workers=2,
        drop_last=False,worker_init_fn=_init_fn)
```

```python
    set_seed()
    device = 'cuda'
    model=Baseline()
    model.load_state_dict(torch.load(f'./models/baseline_corrected.pth'),strict=False)
    model.to(device)
    loss_fn = LovaszLoss()
    optimizer = torch.optim.Adam(model.parameters(),lr= 1e-4)
    scheduler=torch.optim.lr_scheduler.CyclicLR(optimizer, base_lr=1e-4, max_lr=1e-3, step_size_up=
280, step_size_down=None, mode='triangular2', cycle_momentum=False,last_epoch=-1)
    early_stopping = EarlyStopping(patience=80,path= './models/baselinelovasz.pth', verbose=False)

    for epoch in range(1,51):############
      train_loss,train_iou,train_acc=train_loop_fn(train_loader)
      print("Finished training epoch {}".format(epoch))
      val_loss,val_iou,val_acc= test_loop_fn(test_loader)
      writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
      writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)
```

```
****************************************************
*******************   fold  0   *******************
****************************************************
Train[56]: Loss=0.91870 IOU=0.773 ACC=0.944
Finished training epoch 1
Validation: Loss=0.83149,IOU=0.798 ACC=0.947
Train[56]: Loss=0.91818 IOU=0.740 ACC=0.916
Finished training epoch 2
Validation: Loss=1.00649,IOU=0.746 ACC=0.938
Train[56]: Loss=1.07343 IOU=0.718 ACC=0.904
Finished training epoch 3
Validation: Loss=1.28495,IOU=0.718 ACC=0.930
Train[56]: Loss=1.15620 IOU=0.705 ACC=0.901
Finished training epoch 4
Validation: Loss=1.58985,IOU=0.625 ACC=0.871
Train[56]: Loss=1.26850 IOU=0.664 ACC=0.882
Finished training epoch 5
Validation: Loss=2.01842,IOU=0.551 ACC=0.889
Train[56]: Loss=1.17512 IOU=0.675 ACC=0.882
Finished training epoch 6
Validation: Loss=1.00265,IOU=0.757 ACC=0.936
Train[56]: Loss=1.10840 IOU=0.697 ACC=0.889
Finished training epoch 7
Validation: Loss=1.02445,IOU=0.716 ACC=0.895
Train[56]: Loss=1.00361 IOU=0.724 ACC=0.904
Finished training epoch 8
Validation: Loss=0.87646,IOU=0.778 ACC=0.937
Train[56]: Loss=0.92834 IOU=0.740 ACC=0.907
Finished training epoch 9
Validation: Loss=0.80448,IOU=0.808 ACC=0.945
Train[56]: Loss=0.86711 IOU=0.738 ACC=0.901
Finished training epoch 10
Validation: Loss=0.79452,IOU=0.806 ACC=0.945
Train[56]: Loss=0.83317 IOU=0.751 ACC=0.907
Finished training epoch 11
Validation: Loss=0.74026,IOU=0.819 ACC=0.954
Train[56]: Loss=0.83005 IOU=0.746 ACC=0.905
Finished training epoch 12
Validation: Loss=0.78383,IOU=0.805 ACC=0.951
Train[56]: Loss=0.85709 IOU=0.744 ACC=0.901
Finished training epoch 13
Validation: Loss=0.86906,IOU=0.792 ACC=0.941
Train[56]: Loss=0.91687 IOU=0.738 ACC=0.906
Finished training epoch 14
Validation: Loss=0.83306,IOU=0.779 ACC=0.950
Train[56]: Loss=0.91309 IOU=0.730 ACC=0.903
Finished training epoch 15
Validation: Loss=0.91869,IOU=0.778 ACC=0.945
Train[56]: Loss=0.99459 IOU=0.706 ACC=0.890
Finished training epoch 16
Validation: Loss=0.90241,IOU=0.775 ACC=0.939
Train[56]: Loss=0.90009 IOU=0.740 ACC=0.906
Finished training epoch 17
Validation: Loss=0.82508,IOU=0.798 ACC=0.945
Train[56]: Loss=0.90258 IOU=0.735 ACC=0.898
Finished training epoch 18
Validation: Loss=0.79614,IOU=0.799 ACC=0.946
Train[56]: Loss=0.83677 IOU=0.751 ACC=0.906
```

```
Finished training epoch 19
Validation: Loss=0.74858,IOU=0.807 ACC=0.950
Train[56]: Loss=0.78616 IOU=0.761 ACC=0.910
Finished training epoch 20
Validation: Loss=0.74820,IOU=0.811 ACC=0.947
Train[56]: Loss=0.77150 IOU=0.755 ACC=0.901
Finished training epoch 21
Validation: Loss=0.71922,IOU=0.823 ACC=0.952
Train[56]: Loss=0.78547 IOU=0.773 ACC=0.926
Finished training epoch 22
Validation: Loss=0.71711,IOU=0.818 ACC=0.955
Train[56]: Loss=0.79261 IOU=0.760 ACC=0.915
Finished training epoch 23
Validation: Loss=0.76466,IOU=0.804 ACC=0.945
Train[56]: Loss=0.77448 IOU=0.768 ACC=0.919
Finished training epoch 24
Validation: Loss=0.71651,IOU=0.811 ACC=0.950
Train[56]: Loss=0.78404 IOU=0.759 ACC=0.905
Finished training epoch 25
Validation: Loss=0.91615,IOU=0.790 ACC=0.944
Train[56]: Loss=0.82968 IOU=0.749 ACC=0.907
Finished training epoch 26
Validation: Loss=0.75401,IOU=0.806 ACC=0.950
Train[56]: Loss=0.75504 IOU=0.776 ACC=0.922
Finished training epoch 27
Validation: Loss=0.71156,IOU=0.822 ACC=0.957
Train[56]: Loss=0.73660 IOU=0.769 ACC=0.912
Finished training epoch 28
Validation: Loss=0.69343,IOU=0.825 ACC=0.958
Train[56]: Loss=0.69998 IOU=0.771 ACC=0.916
Finished training epoch 29
Validation: Loss=0.69052,IOU=0.825 ACC=0.959
Train[56]: Loss=0.68668 IOU=0.787 ACC=0.926
Finished training epoch 30
Validation: Loss=0.67700,IOU=0.825 ACC=0.960
Train[56]: Loss=0.65882 IOU=0.784 ACC=0.919
Finished training epoch 31
Validation: Loss=0.73328,IOU=0.823 ACC=0.951
Train[56]: Loss=0.67881 IOU=0.787 ACC=0.921
Finished training epoch 32
Validation: Loss=0.71977,IOU=0.822 ACC=0.962
Train[56]: Loss=0.66805 IOU=0.783 ACC=0.918
Finished training epoch 33
Validation: Loss=0.71129,IOU=0.812 ACC=0.939
Train[56]: Loss=0.70033 IOU=0.776 ACC=0.914
Finished training epoch 34
Validation: Loss=0.73540,IOU=0.822 ACC=0.957
Train[56]: Loss=0.72033 IOU=0.771 ACC=0.908
Finished training epoch 35
Validation: Loss=0.69569,IOU=0.821 ACC=0.953
Train[56]: Loss=0.69561 IOU=0.777 ACC=0.913
Finished training epoch 36
Validation: Loss=0.67759,IOU=0.828 ACC=0.958
Train[56]: Loss=0.78687 IOU=0.756 ACC=0.902
Finished training epoch 37
Validation: Loss=0.74570,IOU=0.823 ACC=0.953
Train[56]: Loss=0.71675 IOU=0.782 ACC=0.927
Finished training epoch 38
Validation: Loss=0.72783,IOU=0.819 ACC=0.952
Train[56]: Loss=0.68827 IOU=0.767 ACC=0.914
Finished training epoch 39
Validation: Loss=0.73267,IOU=0.823 ACC=0.958
Train[56]: Loss=0.65216 IOU=0.788 ACC=0.921
Finished training epoch 40
Validation: Loss=0.71486,IOU=0.811 ACC=0.957
Train[56]: Loss=0.64788 IOU=0.789 ACC=0.919
Finished training epoch 41
Validation: Loss=0.71061,IOU=0.832 ACC=0.961
Train[56]: Loss=0.65375 IOU=0.783 ACC=0.915
Finished training epoch 42
Validation: Loss=0.68795,IOU=0.836 ACC=0.957
Train[56]: Loss=0.66106 IOU=0.778 ACC=0.914
Finished training epoch 43
Validation: Loss=0.73356,IOU=0.818 ACC=0.948
Train[56]: Loss=0.71558 IOU=0.762 ACC=0.898
Finished training epoch 44
Validation: Loss=0.74040,IOU=0.817 ACC=0.951
```

```
Train[56]: Loss=0.65859 IOU=0.788 ACC=0.922
Finished training epoch 45
Validation: Loss=0.72289,IOU=0.824 ACC=0.955
Train[56]: Loss=0.63945 IOU=0.787 ACC=0.917
Finished training epoch 46
Validation: Loss=0.74686,IOU=0.824 ACC=0.953
Train[56]: Loss=0.65463 IOU=0.795 ACC=0.924
Finished training epoch 47
Validation: Loss=0.68323,IOU=0.834 ACC=0.957
Train[56]: Loss=0.63579 IOU=0.794 ACC=0.919
Finished training epoch 48
Validation: Loss=0.71840,IOU=0.817 ACC=0.950
Train[56]: Loss=0.63948 IOU=0.787 ACC=0.917
Finished training epoch 49
Validation: Loss=0.72707,IOU=0.832 ACC=0.956
Train[56]: Loss=0.63401 IOU=0.791 ACC=0.923
Finished training epoch 50
Validation: Loss=0.70955,IOU=0.833 ACC=0.955
```

In [34]:

```
%reload_ext tensorboard
%tensorboard --logdir {FLAGS['log_dir']+'Baseline_lovasz'}
```

In [ ]:

```
class ChannelAttentionGate(nn.Module):
    def __init__(self, channel, reduction=16):
        super(ChannelAttentionGate, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
                nn.Linear(channel, channel // reduction),
                nn.ReLU(inplace=True),
                nn.Linear(channel // reduction, channel),
                nn.Sigmoid()
                 )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return y


class SpatialAttentionGate(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SpatialAttentionGate, self).__init__()
        self.fc1 = nn.Conv2d(channel, reduction, kernel_size=1, padding=0)
        self.fc2 = nn.Conv2d(reduction, 1, kernel_size=1, padding=0)

    def forward(self, x):
        x = self.fc1(x)
        x = F.relu(x, inplace=True)
        x = self.fc2(x)
        x = torch.sigmoid(x)

        return x

class Decoder(nn.Module):
    def __init__(self, in_channels, channels, out_channels ):
        super(Decoder, self).__init__()
        self.conv1 =  ConvBn2d(in_channels,  channels, kernel_size=3, padding=1)
        self.conv2 =  ConvBn2d(channels, out_channels, kernel_size=3, padding=1)
        self.cg= ChannelAttentionGate(out_channels)
        self.sg= SpatialAttentionGate(out_channels)

    def forward(self, x ):
        x = F.upsample(x, scale_factor=2, mode='bilinear', align_corners=True)#False
        x = F.relu(self.conv1(x),inplace=True)
        x = F.relu(self.conv2(x),inplace=True)
        g1= self.sg(x)
        g2= self.cg(x)
        x= g1*x+g2*x
        return x
```

```
class UNetScseHypercol(nn.Module):

    def __init__(self ):
        super().__init__()
        self.resnet = torchvision.models.resnet34(pretrained=True)

        self.conv1 = nn.Sequential(
            self.resnet.conv1,
            self.resnet.bn1,
            self.resnet.relu,
        )# 64
        self.encoder2 = self.resnet.layer1  # 64
        self.encoder3 = self.resnet.layer2  #128
        self.encoder4 = self.resnet.layer3  #256
        self.encoder5 = self.resnet.layer4  #512

        self.center = nn.Sequential(
            nn.Conv2d(512, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
        )

        self.decoder5 = Decoder(512+64, 512, 64)
        self.decoder4 = Decoder(64+256, 256, 64)
        self.decoder3 = Decoder(64+128, 128,  64)
        self.decoder2 = Decoder( 64+ 64, 64, 64)


        self.logit    = nn.Sequential(
            nn.Conv2d(256, 32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(32,  1, kernel_size=1, padding=0),
        )


    def forward(self, x):

        x = self.conv1(x)

        e2 = self.encoder2( x)  #; print('e2',e2.size())
        e3 = self.encoder3(e2)  #; print('e3',e3.size())
        e4 = self.encoder4(e3)  #; print('e4',e4.size())
        e5 = self.encoder5(e4)  #; print('e5',e5.size())
                              #; print('center',f.size())
        f = self.center(e5)
        # print(e5.shape,f.shape)

        d5 = self.decoder5(torch.cat([f, e5], 1))  #; print('d5',f.size())
        d4 = self.decoder4(torch.cat([d5, e4], 1))  #; print('d4',f.size())
        d3= self.decoder3(torch.cat([d4, e3], 1))  #; print('d3',f.size())
        d2 = self.decoder2(torch.cat([d3, e2], 1))  #; print('d2',f.size())

        f = torch.cat((d2,
          F.upsample(d3,scale_factor=2,mode='bilinear',align_corners=False),
          F.upsample(d4,scale_factor=4,mode='bilinear',align_corners=False),
          F.upsample(d5,scale_factor=8,mode='bilinear',align_corners=False),
                    ),1)

        logit = self.logit(f)                    #; print('logit',logit.size())
        return logit

model= UNetScseHypercol()
```

In [ ]:

```
transform_train = Compose([
    HorizontalFlip(p=.5),
     Compose([RandomCrop(90,90),
     Resize(101,101)],p=.3),
    OneOf([RandomBrightness(.1),
    RandomContrast(.1),RandomGamma()],p=.3),
    PadIfNeeded(256//2,256//2,cv2.BORDER_REPLICATE),
    Normalize(mean=(0,0,0),std=(1,1,1,))])
```

In [ ]:

```python
kf = StratifiedKFold(10,shuffle=True,random_state=seed)
for fold, (trn_idx,val_idx) in enumerate(kf.split(file_list,coverage)):
    if fold!=0 :#######
      continue
    print('*****************************************************')
    print('*******************    fold  %d    *******************' % fold)
    print('*****************************************************')

    file_list_train= [x for i,x in enumerate(file_list) if i in trn_idx]
    file_list_val= [x for i,x in enumerate(file_list) if i in val_idx]
    train = TGSSaltDataset(train_path, file_list_train,augment=transform_train)
    val = TGSSaltDataset(train_path, file_list_val,augment= transform_test)
    writer =SummaryWriter(FLAGS['log_dir']+'scse')

    train_loader = torch.utils.data.DataLoader(
        train,
        batch_size=32,
        num_workers=4,
        drop_last=False,worker_init_fn=_init_fn)
    test_loader = torch.utils.data.DataLoader(
        val,
        batch_size=FLAGS['batch_size']*5,
        shuffle=False,
        num_workers=2,
        drop_last=False,worker_init_fn=_init_fn)

    set_seed()
    device = 'cuda'
    model = model.to(device)

    loss_fn= nn.BCEWithLogitsLoss()
    optimizer= torch.optim.Adam(model.parameters(),lr=3e-4)
    early_stopping = EarlyStopping(patience=80,path= './models/scse.pth', verbose=False)
    scheduler= None

    for epoch in range(1,21):############
      train_loss,train_iou,train_acc=train_loop_fn(train_loader)
      print("Finished training epoch {}".format(epoch))
      val_loss,val_iou,val_acc= test_loop_fn(test_loader)
      writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
      writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)


    loss_fn = LovaszLoss()
    optimizer= torch.optim.Adam(model.parameters(),lr=1e-4)

    for epoch in range(21,91):
      train_loss,train_iou,train_acc=train_loop_fn(train_loader)
      print("Finished training epoch {}".format(epoch))
      val_loss,val_iou,val_acc= test_loop_fn(test_loader)
      writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
      writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)

    scheduler=torch.optim.lr_scheduler.CyclicLR(optimizer, base_lr=1e-4, max_lr=1e-3, step_size_up=
560*2, step_size_down=None, mode='triangular2', cycle_momentum=False,last_epoch=-1)


    for epoch in range(91,171):
      train_loss,train_iou,train_acc=train_loop_fn(train_loader)
      print("Finished training epoch {}".format(epoch))
      val_loss,val_iou,val_acc= test_loop_fn(test_loader)
      writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
      writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)
```

```
*****************************************************
*******************    fold  0    *******************
*****************************************************
Train[112]: Loss=0.19873 IOU=0.730 ACC=0.916
Finished training epoch 1
Validation: Loss=0.16866,IOU=0.794 ACC=0.944
Train[112]: Loss=0.19705 IOU=0.728 ACC=0.917
Finished training epoch 2
Validation: Loss=0.12604,IOU=0.801 ACC=0.952
Train[112]: Loss=0.19345 IOU=0.733 ACC=0.920
Finished training epoch 3
```

```
Validation: Loss=0.17322,IOU=0.811 ACC=0.953
Train[112]: Loss=0.20115 IOU=0.730 ACC=0.916
Finished training epoch 4
Validation: Loss=0.14015,IOU=0.800 ACC=0.948
Train[112]: Loss=0.19690 IOU=0.723 ACC=0.918
Finished training epoch 5
Validation: Loss=0.12960,IOU=0.815 ACC=0.954
Train[112]: Loss=0.18784 IOU=0.735 ACC=0.922
Finished training epoch 6
Validation: Loss=0.16481,IOU=0.803 ACC=0.952
Train[112]: Loss=0.18446 IOU=0.739 ACC=0.923
Finished training epoch 7
Validation: Loss=0.13541,IOU=0.800 ACC=0.952
Train[112]: Loss=0.19176 IOU=0.736 ACC=0.921
Finished training epoch 8
Validation: Loss=0.15040,IOU=0.815 ACC=0.955
Train[112]: Loss=0.18802 IOU=0.735 ACC=0.926
Finished training epoch 9
Validation: Loss=0.14746,IOU=0.795 ACC=0.953
Train[112]: Loss=0.19867 IOU=0.724 ACC=0.916
Finished training epoch 10
Validation: Loss=0.14326,IOU=0.805 ACC=0.954
Train[112]: Loss=0.18391 IOU=0.739 ACC=0.923
Finished training epoch 11
Validation: Loss=0.14899,IOU=0.810 ACC=0.951
Train[112]: Loss=0.18247 IOU=0.745 ACC=0.928
Finished training epoch 12
Validation: Loss=0.16581,IOU=0.816 ACC=0.955
Train[112]: Loss=0.19047 IOU=0.736 ACC=0.922
Finished training epoch 13
Validation: Loss=0.17863,IOU=0.808 ACC=0.949
Train[112]: Loss=0.18662 IOU=0.739 ACC=0.924
Finished training epoch 14
Validation: Loss=0.17768,IOU=0.808 ACC=0.948
Train[112]: Loss=0.18291 IOU=0.745 ACC=0.926
Finished training epoch 15
Validation: Loss=0.15282,IOU=0.806 ACC=0.952
Train[112]: Loss=0.18845 IOU=0.728 ACC=0.921
Finished training epoch 16
Validation: Loss=0.15539,IOU=0.808 ACC=0.947
Train[112]: Loss=0.18480 IOU=0.738 ACC=0.922
Finished training epoch 17
Validation: Loss=0.17114,IOU=0.813 ACC=0.952
Train[112]: Loss=0.17832 IOU=0.740 ACC=0.927
Finished training epoch 18
Validation: Loss=0.15449,IOU=0.800 ACC=0.950
Train[112]: Loss=0.17400 IOU=0.749 ACC=0.931
Finished training epoch 19
Validation: Loss=0.17592,IOU=0.798 ACC=0.950
Train[112]: Loss=0.18975 IOU=0.732 ACC=0.922
Finished training epoch 20
Validation: Loss=0.17419,IOU=0.804 ACC=0.952
Train[112]: Loss=0.92290 IOU=0.758 ACC=0.930
Finished training epoch 21
Validation: Loss=0.83706,IOU=0.821 ACC=0.953
Train[112]: Loss=0.86943 IOU=0.760 ACC=0.931
Finished training epoch 22
Validation: Loss=0.81832,IOU=0.820 ACC=0.951
Train[112]: Loss=0.85852 IOU=0.761 ACC=0.931
Finished training epoch 23
Validation: Loss=0.79837,IOU=0.823 ACC=0.952
Train[112]: Loss=0.79804 IOU=0.779 ACC=0.937
Finished training epoch 24
Validation: Loss=0.86306,IOU=0.807 ACC=0.943
Train[112]: Loss=0.82398 IOU=0.768 ACC=0.932
Finished training epoch 25
Validation: Loss=0.80323,IOU=0.818 ACC=0.951
Train[112]: Loss=0.80026 IOU=0.770 ACC=0.934
Finished training epoch 26
Validation: Loss=0.81028,IOU=0.816 ACC=0.947
Train[112]: Loss=0.78998 IOU=0.781 ACC=0.937
Finished training epoch 27
Validation: Loss=0.82019,IOU=0.813 ACC=0.944
Train[112]: Loss=0.75929 IOU=0.778 ACC=0.939
Finished training epoch 28
Validation: Loss=0.85554,IOU=0.809 ACC=0.934
Train[112]: Loss=0.80684 IOU=0.759 ACC=0.926
```

```
Finished training epoch 29
Validation: Loss=0.85162,IOU=0.812 ACC=0.941
Train[112]: Loss=0.77612 IOU=0.770 ACC=0.935
Finished training epoch 30
Validation: Loss=0.76937,IOU=0.830 ACC=0.954
Train[112]: Loss=0.78202 IOU=0.771 ACC=0.929
Finished training epoch 31
Validation: Loss=0.78764,IOU=0.820 ACC=0.949
Train[112]: Loss=0.75347 IOU=0.761 ACC=0.907
Finished training epoch 32
Validation: Loss=0.79973,IOU=0.817 ACC=0.947
Train[112]: Loss=0.76464 IOU=0.756 ACC=0.905
Finished training epoch 33
Validation: Loss=0.83184,IOU=0.807 ACC=0.939
Train[112]: Loss=0.76455 IOU=0.753 ACC=0.898
Finished training epoch 34
Validation: Loss=0.80565,IOU=0.820 ACC=0.947
Train[112]: Loss=0.78444 IOU=0.760 ACC=0.907
Finished training epoch 35
Validation: Loss=0.80975,IOU=0.816 ACC=0.943
Train[112]: Loss=0.74849 IOU=0.754 ACC=0.897
Finished training epoch 36
Validation: Loss=0.85748,IOU=0.809 ACC=0.945
Train[112]: Loss=0.73262 IOU=0.760 ACC=0.903
Finished training epoch 37
Validation: Loss=0.82414,IOU=0.827 ACC=0.946
Train[112]: Loss=0.76753 IOU=0.750 ACC=0.895
Finished training epoch 38
Validation: Loss=0.78887,IOU=0.820 ACC=0.943
Train[112]: Loss=0.74085 IOU=0.759 ACC=0.903
Finished training epoch 39
Validation: Loss=0.81287,IOU=0.827 ACC=0.948
Train[112]: Loss=0.74629 IOU=0.763 ACC=0.908
Finished training epoch 40
Validation: Loss=0.79960,IOU=0.820 ACC=0.933
Train[112]: Loss=0.70904 IOU=0.756 ACC=0.893
Finished training epoch 41
Validation: Loss=0.78672,IOU=0.825 ACC=0.949
Train[112]: Loss=0.71620 IOU=0.764 ACC=0.901
Finished training epoch 42
Validation: Loss=0.83191,IOU=0.819 ACC=0.942
Train[112]: Loss=0.73205 IOU=0.758 ACC=0.898
Finished training epoch 43
Validation: Loss=0.75569,IOU=0.835 ACC=0.956
Train[112]: Loss=0.73728 IOU=0.766 ACC=0.906
Finished training epoch 44
Validation: Loss=0.75462,IOU=0.817 ACC=0.929
Train[112]: Loss=0.73011 IOU=0.762 ACC=0.900
Finished training epoch 45
Validation: Loss=0.75005,IOU=0.831 ACC=0.951
Train[112]: Loss=0.73604 IOU=0.762 ACC=0.903
Finished training epoch 46
Validation: Loss=0.76205,IOU=0.826 ACC=0.940
Train[112]: Loss=0.72662 IOU=0.762 ACC=0.902
Finished training epoch 47
Validation: Loss=0.74552,IOU=0.830 ACC=0.940
Train[112]: Loss=0.71556 IOU=0.770 ACC=0.909
Finished training epoch 48
Validation: Loss=0.76458,IOU=0.834 ACC=0.953
Train[112]: Loss=0.71276 IOU=0.761 ACC=0.897
Finished training epoch 49
Validation: Loss=0.77907,IOU=0.827 ACC=0.948
Train[112]: Loss=0.70110 IOU=0.765 ACC=0.900
Finished training epoch 50
Validation: Loss=0.82632,IOU=0.828 ACC=0.950
Train[112]: Loss=0.71932 IOU=0.756 ACC=0.892
Finished training epoch 51
Validation: Loss=0.78619,IOU=0.824 ACC=0.950
Train[112]: Loss=0.73062 IOU=0.767 ACC=0.906
Finished training epoch 52
Validation: Loss=0.80031,IOU=0.826 ACC=0.942
Train[112]: Loss=0.70593 IOU=0.766 ACC=0.902
Finished training epoch 53
Validation: Loss=0.82042,IOU=0.821 ACC=0.945
Train[112]: Loss=0.70758 IOU=0.764 ACC=0.902
Finished training epoch 54
Validation: Loss=0.84221,IOU=0.811 ACC=0.940
```

```
Train[112]: Loss=0.72676 IOU=0.762 ACC=0.897
Finished training epoch 55
Validation: Loss=0.75763,IOU=0.804 ACC=0.913
Train[112]: Loss=0.70696 IOU=0.779 ACC=0.912
Finished training epoch 56
Validation: Loss=0.76029,IOU=0.830 ACC=0.950
Train[112]: Loss=0.67173 IOU=0.776 ACC=0.908
Finished training epoch 57
Validation: Loss=0.82264,IOU=0.811 ACC=0.939
Train[112]: Loss=0.67483 IOU=0.780 ACC=0.908
Finished training epoch 58
Validation: Loss=0.73809,IOU=0.828 ACC=0.949
Train[112]: Loss=0.71641 IOU=0.763 ACC=0.899
Finished training epoch 59
Validation: Loss=0.79425,IOU=0.817 ACC=0.927
Train[112]: Loss=0.70382 IOU=0.762 ACC=0.893
Finished training epoch 60
Validation: Loss=0.79824,IOU=0.822 ACC=0.951
Train[112]: Loss=0.71644 IOU=0.768 ACC=0.904
Finished training epoch 61
Validation: Loss=0.86134,IOU=0.809 ACC=0.936
Train[112]: Loss=0.69315 IOU=0.769 ACC=0.904
Finished training epoch 62
Validation: Loss=0.82670,IOU=0.822 ACC=0.948
Train[112]: Loss=0.70490 IOU=0.781 ACC=0.915
Finished training epoch 63
Validation: Loss=0.82221,IOU=0.815 ACC=0.926
Train[112]: Loss=0.68114 IOU=0.772 ACC=0.903
Finished training epoch 64
Validation: Loss=0.82210,IOU=0.823 ACC=0.935
Train[112]: Loss=0.68553 IOU=0.768 ACC=0.897
Finished training epoch 65
Validation: Loss=0.81991,IOU=0.825 ACC=0.948
Train[112]: Loss=0.68608 IOU=0.778 ACC=0.907
Finished training epoch 66
Validation: Loss=0.79661,IOU=0.810 ACC=0.922
Train[112]: Loss=0.71284 IOU=0.758 ACC=0.895
Finished training epoch 67
Validation: Loss=0.83429,IOU=0.800 ACC=0.916
Train[112]: Loss=0.69603 IOU=0.769 ACC=0.899
Finished training epoch 68
Validation: Loss=0.77173,IOU=0.826 ACC=0.933
Train[112]: Loss=0.68639 IOU=0.764 ACC=0.895
Finished training epoch 69
Validation: Loss=0.76799,IOU=0.825 ACC=0.947
Train[112]: Loss=0.67631 IOU=0.778 ACC=0.907
Finished training epoch 70
Validation: Loss=0.79694,IOU=0.830 ACC=0.950
Train[112]: Loss=0.69784 IOU=0.771 ACC=0.904
Finished training epoch 71
Validation: Loss=0.82458,IOU=0.829 ACC=0.949
Train[112]: Loss=0.70469 IOU=0.758 ACC=0.887
Finished training epoch 72
Validation: Loss=0.80452,IOU=0.813 ACC=0.912
Train[112]: Loss=0.67569 IOU=0.767 ACC=0.896
Finished training epoch 73
Validation: Loss=0.78199,IOU=0.824 ACC=0.945
Train[112]: Loss=0.67106 IOU=0.778 ACC=0.904
Finished training epoch 74
Validation: Loss=0.86678,IOU=0.832 ACC=0.948
Train[112]: Loss=0.67419 IOU=0.771 ACC=0.897
Finished training epoch 75
Validation: Loss=0.78469,IOU=0.827 ACC=0.948
Train[112]: Loss=0.68631 IOU=0.774 ACC=0.901
Finished training epoch 76
Validation: Loss=0.77147,IOU=0.818 ACC=0.929
Train[112]: Loss=0.69124 IOU=0.771 ACC=0.898
Finished training epoch 77
Validation: Loss=0.75769,IOU=0.825 ACC=0.935
Train[112]: Loss=0.66362 IOU=0.780 ACC=0.902
Finished training epoch 78
Validation: Loss=0.83325,IOU=0.807 ACC=0.914
Train[112]: Loss=0.67793 IOU=0.772 ACC=0.898
Finished training epoch 79
Validation: Loss=0.80391,IOU=0.828 ACC=0.933
Train[112]: Loss=0.69465 IOU=0.774 ACC=0.905
Finished training epoch 80
```

```
Validation: Loss=0.74576,IOU=0.831 ACC=0.946
Train[112]: Loss=0.67491 IOU=0.775 ACC=0.903
Finished training epoch 81
Validation: Loss=0.83433,IOU=0.821 ACC=0.948
Train[112]: Loss=0.68615 IOU=0.768 ACC=0.895
Finished training epoch 82
Validation: Loss=0.80943,IOU=0.834 ACC=0.948
Train[112]: Loss=0.70420 IOU=0.768 ACC=0.895
Finished training epoch 83
Validation: Loss=0.77938,IOU=0.821 ACC=0.948
Train[112]: Loss=0.69483 IOU=0.768 ACC=0.893
Finished training epoch 84
Validation: Loss=0.82184,IOU=0.821 ACC=0.943
Train[112]: Loss=0.66498 IOU=0.776 ACC=0.902
Finished training epoch 85
Validation: Loss=0.85032,IOU=0.812 ACC=0.940
Train[112]: Loss=0.65731 IOU=0.782 ACC=0.904
Finished training epoch 86
Validation: Loss=0.85532,IOU=0.823 ACC=0.946
Train[112]: Loss=0.70093 IOU=0.764 ACC=0.893
Finished training epoch 87
Validation: Loss=0.84731,IOU=0.817 ACC=0.934
Train[112]: Loss=0.67806 IOU=0.768 ACC=0.895
Finished training epoch 88
Validation: Loss=0.85913,IOU=0.825 ACC=0.942
Train[112]: Loss=0.67013 IOU=0.778 ACC=0.901
Finished training epoch 89
Validation: Loss=0.82906,IOU=0.827 ACC=0.948
Train[112]: Loss=0.62226 IOU=0.786 ACC=0.907
Finished training epoch 90
Validation: Loss=0.83274,IOU=0.825 ACC=0.949
Train[112]: Loss=0.66502 IOU=0.773 ACC=0.900
Finished training epoch 91
Validation: Loss=0.80323,IOU=0.830 ACC=0.956
Train[112]: Loss=0.71386 IOU=0.769 ACC=0.899
Finished training epoch 92
Validation: Loss=0.78570,IOU=0.820 ACC=0.947
Train[112]: Loss=0.74010 IOU=0.763 ACC=0.902
Finished training epoch 93
Validation: Loss=0.86215,IOU=0.785 ACC=0.913
Train[112]: Loss=0.81896 IOU=0.755 ACC=0.899
Finished training epoch 94
Validation: Loss=0.94391,IOU=0.806 ACC=0.939
Train[112]: Loss=0.85208 IOU=0.743 ACC=0.897
Finished training epoch 95
Validation: Loss=0.93189,IOU=0.762 ACC=0.915
Train[112]: Loss=0.91100 IOU=0.723 ACC=0.884
Finished training epoch 96
Validation: Loss=0.93475,IOU=0.781 ACC=0.936
Train[112]: Loss=1.02924 IOU=0.706 ACC=0.878
Finished training epoch 97
Validation: Loss=0.85999,IOU=0.769 ACC=0.898
Train[112]: Loss=1.00074 IOU=0.711 ACC=0.881
Finished training epoch 98
Validation: Loss=0.82533,IOU=0.780 ACC=0.937
Train[112]: Loss=0.96241 IOU=0.715 ACC=0.883
Finished training epoch 99
Validation: Loss=0.89524,IOU=0.771 ACC=0.937
Train[112]: Loss=1.10808 IOU=0.679 ACC=0.863
Finished training epoch 100
Validation: Loss=0.84526,IOU=0.791 ACC=0.931
Train[112]: Loss=1.06144 IOU=0.686 ACC=0.865
Finished training epoch 101
Validation: Loss=0.77801,IOU=0.804 ACC=0.915
Train[112]: Loss=0.98157 IOU=0.711 ACC=0.883
Finished training epoch 102
Validation: Loss=0.88103,IOU=0.788 ACC=0.926
Train[112]: Loss=0.95873 IOU=0.721 ACC=0.887
Finished training epoch 103
Validation: Loss=0.78370,IOU=0.804 ACC=0.946
Train[112]: Loss=0.96367 IOU=0.717 ACC=0.880
Finished training epoch 104
Validation: Loss=0.76240,IOU=0.810 ACC=0.950
Train[112]: Loss=0.89453 IOU=0.719 ACC=0.875
Finished training epoch 105
Validation: Loss=0.74974,IOU=0.822 ACC=0.952
Train[112]: Loss=0.87212 IOU=0.733 ACC=0.881
```

```
Finished training epoch 106
Validation: Loss=0.73243,IOU=0.818 ACC=0.948
Train[112]: Loss=0.81415 IOU=0.743 ACC=0.883
Finished training epoch 107
Validation: Loss=0.82947,IOU=0.812 ACC=0.943
Train[112]: Loss=0.82539 IOU=0.738 ACC=0.882
Finished training epoch 108
Validation: Loss=0.78466,IOU=0.824 ACC=0.948
Train[112]: Loss=0.75996 IOU=0.750 ACC=0.888
Finished training epoch 109
Validation: Loss=0.80639,IOU=0.818 ACC=0.945
Train[112]: Loss=0.78358 IOU=0.740 ACC=0.882
Finished training epoch 110
Validation: Loss=0.78136,IOU=0.819 ACC=0.950
Train[112]: Loss=0.74310 IOU=0.751 ACC=0.887
Finished training epoch 111
Validation: Loss=0.80842,IOU=0.821 ACC=0.946
Train[112]: Loss=0.74043 IOU=0.758 ACC=0.892
Finished training epoch 112
Validation: Loss=0.81508,IOU=0.811 ACC=0.943
Train[112]: Loss=0.73113 IOU=0.770 ACC=0.908
Finished training epoch 113
Validation: Loss=0.79583,IOU=0.829 ACC=0.949
Train[112]: Loss=0.76206 IOU=0.751 ACC=0.891
Finished training epoch 114
Validation: Loss=0.80138,IOU=0.815 ACC=0.947
Train[112]: Loss=0.77106 IOU=0.752 ACC=0.891
Finished training epoch 115
Validation: Loss=0.85105,IOU=0.810 ACC=0.941
Train[112]: Loss=0.76438 IOU=0.751 ACC=0.890
Finished training epoch 116
Validation: Loss=0.76606,IOU=0.804 ACC=0.907
Train[112]: Loss=0.77897 IOU=0.756 ACC=0.896
Finished training epoch 117
Validation: Loss=0.81272,IOU=0.812 ACC=0.943
Train[112]: Loss=0.79955 IOU=0.740 ACC=0.878
Finished training epoch 118
Validation: Loss=0.79749,IOU=0.815 ACC=0.947
Train[112]: Loss=0.83937 IOU=0.732 ACC=0.879
Finished training epoch 119
Validation: Loss=0.78544,IOU=0.822 ACC=0.950
Train[112]: Loss=0.84149 IOU=0.728 ACC=0.874
Finished training epoch 120
Validation: Loss=0.79031,IOU=0.801 ACC=0.938
Train[112]: Loss=0.82402 IOU=0.746 ACC=0.892
Finished training epoch 121
Validation: Loss=0.73296,IOU=0.814 ACC=0.918
Train[112]: Loss=0.78795 IOU=0.747 ACC=0.887
Finished training epoch 122
Validation: Loss=0.80277,IOU=0.812 ACC=0.946
Train[112]: Loss=0.78047 IOU=0.745 ACC=0.883
Finished training epoch 123
Validation: Loss=0.79264,IOU=0.814 ACC=0.944
Train[112]: Loss=0.79412 IOU=0.740 ACC=0.883
Finished training epoch 124
Validation: Loss=0.86314,IOU=0.814 ACC=0.945
Train[112]: Loss=0.78726 IOU=0.754 ACC=0.897
Finished training epoch 125
Validation: Loss=0.80769,IOU=0.809 ACC=0.922
Train[112]: Loss=0.73968 IOU=0.764 ACC=0.900
Finished training epoch 126
Validation: Loss=0.77270,IOU=0.828 ACC=0.952
Train[112]: Loss=0.73282 IOU=0.765 ACC=0.900
Finished training epoch 127
Validation: Loss=0.79276,IOU=0.815 ACC=0.918
Train[112]: Loss=0.69474 IOU=0.760 ACC=0.890
Finished training epoch 128
Validation: Loss=0.83768,IOU=0.814 ACC=0.918
Train[112]: Loss=0.70230 IOU=0.763 ACC=0.895
Finished training epoch 129
Validation: Loss=0.84077,IOU=0.811 ACC=0.916
Train[112]: Loss=0.68654 IOU=0.766 ACC=0.891
Finished training epoch 130
Validation: Loss=0.80250,IOU=0.816 ACC=0.918
Train[112]: Loss=0.67457 IOU=0.775 ACC=0.899
Finished training epoch 131
Validation: Loss=0.82603,IOU=0.811 ACC=0.924
```

```
Train[112]: Loss=0.69225 IOU=0.771 ACC=0.900
Finished training epoch 132
Validation: Loss=0.84917,IOU=0.819 ACC=0.942
Train[112]: Loss=0.68514 IOU=0.772 ACC=0.900
Finished training epoch 133
Validation: Loss=0.87073,IOU=0.820 ACC=0.938
Train[112]: Loss=0.68104 IOU=0.778 ACC=0.901
Finished training epoch 134
Validation: Loss=0.82195,IOU=0.819 ACC=0.928
Train[112]: Loss=0.69040 IOU=0.775 ACC=0.902
Finished training epoch 135
Validation: Loss=0.85299,IOU=0.818 ACC=0.942
Train[112]: Loss=0.69424 IOU=0.770 ACC=0.900
Finished training epoch 136
Validation: Loss=0.90526,IOU=0.792 ACC=0.901
Train[112]: Loss=0.74117 IOU=0.765 ACC=0.898
Finished training epoch 137
Validation: Loss=0.81743,IOU=0.817 ACC=0.943
Train[112]: Loss=0.72164 IOU=0.767 ACC=0.898
Finished training epoch 138
Validation: Loss=0.83721,IOU=0.815 ACC=0.940
Train[112]: Loss=0.78579 IOU=0.748 ACC=0.888
Finished training epoch 139
Validation: Loss=0.79068,IOU=0.807 ACC=0.915
Train[112]: Loss=0.74286 IOU=0.751 ACC=0.885
Finished training epoch 140
Validation: Loss=0.81903,IOU=0.801 ACC=0.914
Train[112]: Loss=0.73851 IOU=0.763 ACC=0.895
Finished training epoch 141
Validation: Loss=0.80435,IOU=0.812 ACC=0.907
Train[112]: Loss=0.72082 IOU=0.757 ACC=0.886
Finished training epoch 142
Validation: Loss=0.76944,IOU=0.821 ACC=0.931
Train[112]: Loss=0.68936 IOU=0.772 ACC=0.901
Finished training epoch 143
Validation: Loss=0.75947,IOU=0.807 ACC=0.915
Train[112]: Loss=0.68633 IOU=0.767 ACC=0.896
Finished training epoch 144
Validation: Loss=0.83570,IOU=0.806 ACC=0.922
Train[112]: Loss=0.69332 IOU=0.768 ACC=0.893
Finished training epoch 145
Validation: Loss=0.76507,IOU=0.821 ACC=0.944
Train[112]: Loss=0.71916 IOU=0.759 ACC=0.890
Finished training epoch 146
Validation: Loss=0.73835,IOU=0.818 ACC=0.945
Train[112]: Loss=0.69273 IOU=0.771 ACC=0.896
Finished training epoch 147
Validation: Loss=0.74100,IOU=0.822 ACC=0.924
Train[112]: Loss=0.67874 IOU=0.773 ACC=0.895
Finished training epoch 148
Validation: Loss=0.76907,IOU=0.823 ACC=0.946
Train[112]: Loss=0.67583 IOU=0.770 ACC=0.895
Finished training epoch 149
Validation: Loss=0.80108,IOU=0.823 ACC=0.945
Train[112]: Loss=0.63864 IOU=0.778 ACC=0.897
Finished training epoch 150
Validation: Loss=0.77485,IOU=0.826 ACC=0.946
Train[112]: Loss=0.66408 IOU=0.770 ACC=0.889
Finished training epoch 151
Validation: Loss=0.75377,IOU=0.823 ACC=0.941
Train[112]: Loss=0.64874 IOU=0.784 ACC=0.906
Finished training epoch 152
Validation: Loss=0.74131,IOU=0.818 ACC=0.919
Train[112]: Loss=0.64817 IOU=0.776 ACC=0.895
Finished training epoch 153
Validation: Loss=0.76139,IOU=0.822 ACC=0.948
Train[112]: Loss=0.67574 IOU=0.775 ACC=0.896
Finished training epoch 154
Validation: Loss=0.81002,IOU=0.819 ACC=0.942
Train[112]: Loss=0.67528 IOU=0.781 ACC=0.901
Finished training epoch 155
Validation: Loss=0.75641,IOU=0.824 ACC=0.942
Train[112]: Loss=0.64577 IOU=0.780 ACC=0.896
Finished training epoch 156
Validation: Loss=0.80421,IOU=0.820 ACC=0.945
Train[112]: Loss=0.64578 IOU=0.786 ACC=0.902
Finished training epoch 157
```

```
Validation: Loss=0.81684,IOU=0.815 ACC=0.923
Train[112]: Loss=0.64385 IOU=0.777 ACC=0.893
Finished training epoch 158
Validation: Loss=0.83558,IOU=0.823 ACC=0.949
Train[112]: Loss=0.64098 IOU=0.775 ACC=0.893
Finished training epoch 159
Validation: Loss=0.86751,IOU=0.821 ACC=0.925
Train[112]: Loss=0.66931 IOU=0.785 ACC=0.905
Finished training epoch 160
Validation: Loss=0.89500,IOU=0.804 ACC=0.912
Train[112]: Loss=0.65437 IOU=0.782 ACC=0.904
Finished training epoch 161
Validation: Loss=0.77031,IOU=0.818 ACC=0.946
Train[112]: Loss=0.64987 IOU=0.778 ACC=0.897
Finished training epoch 162
Validation: Loss=0.79201,IOU=0.831 ACC=0.952
Train[112]: Loss=0.65245 IOU=0.774 ACC=0.889
Finished training epoch 163
Validation: Loss=0.76751,IOU=0.827 ACC=0.944
Train[112]: Loss=0.63981 IOU=0.783 ACC=0.901
Finished training epoch 164
Validation: Loss=0.82765,IOU=0.812 ACC=0.917
Train[112]: Loss=0.68192 IOU=0.783 ACC=0.905
Finished training epoch 165
Validation: Loss=0.77215,IOU=0.826 ACC=0.946
Train[112]: Loss=0.62338 IOU=0.794 ACC=0.911
Finished training epoch 166
Validation: Loss=0.79807,IOU=0.819 ACC=0.925
Train[112]: Loss=0.63910 IOU=0.780 ACC=0.897
Finished training epoch 167
Validation: Loss=0.82690,IOU=0.824 ACC=0.931
Train[112]: Loss=0.66418 IOU=0.778 ACC=0.899
Finished training epoch 168
Validation: Loss=0.81700,IOU=0.824 ACC=0.933
Train[112]: Loss=0.65341 IOU=0.788 ACC=0.908
Finished training epoch 169
Validation: Loss=0.79778,IOU=0.832 ACC=0.936
Train[112]: Loss=0.63172 IOU=0.782 ACC=0.895
Finished training epoch 170
Validation: Loss=0.82647,IOU=0.825 ACC=0.930
```

In [31]:

```
%reload_ext tensorboard
%tensorboard --logdir {FLAGS['log_dir']+'scse'}
```

```
Reusing TensorBoard on port 6007 (pid 359), started 1:03:53 ago. (Use '!kill 359' to kill it.)
```

lets try debugging the model to see the errors classified according to the classes.

In [ ]:

```
kf = StratifiedKFold(10,shuffle=True,random_state=seed)
for fold, (trn_idx,val_idx) in enumerate(kf.split(file_list,coverage)):
    if fold!=0 :#######
      continue
    file_list_train= [x for i,x in enumerate(file_list) if i in trn_idx]
    file_list_val= [x for i,x in enumerate(file_list) if i in val_idx]
train = TGSSaltDataset(train_path, file_list_train,augment=transform_test)
val = TGSSaltDataset(train_path, file_list_val,augment= transform_test)

train_loader = torch.utils.data.DataLoader(
    train,
    batch_size=32,
    num_workers=4,
    drop_last=False,worker_init_fn=_init_fn)
test_loader = torch.utils.data.DataLoader(
    val,
    batch_size=FLAGS['batch_size']*5,
    shuffle=False,
    num_workers=2,drop_last=False,worker_init_fn=_init_fn)
```

In [ ]:

```
all_predictions = []
for image,_ in tqdm(test_loader):
  with torch.no_grad():
    image = image.to(device)
    y_pred= model(image)
    y_pred = F.sigmoid(y_pred)
    all_predictions.append(y_pred)

val_predictions_stacked=torch.cat(all_predictions,0)
print(val_predictions_stacked.shape)
all_predictions=None
all_predictions = []
for image,_ in tqdm(train_loader):
  with torch.no_grad():
    image = image.to(device)
    bin,y_pred= model(image)
    y_pred = F.sigmoid(y_pred)
    all_predictions.append(y_pred)

train_predictions_stacked=torch.cat(all_predictions,0)
print(train_predictions_stacked.shape)
all_predictions=None
```

```
100%|██████████| 5/5 [00:02<00:00,  1.86it/s]
  0%|          | 0/113 [00:00<?, ?it/s]
```

```
torch.Size([400, 1, 128, 128])
```

```
100%|██████████| 113/113 [00:23<00:00,  4.87it/s]
```

```
torch.Size([3600, 1, 128, 128])
```

In [ ]:

```
train_predictions_stacked=train_predictions_stacked.cpu().numpy()
val_predictions_stacked=val_predictions_stacked.cpu().numpy()
```

In [ ]:

```
from scipy.special import logit
# with open('predictions_debug2.npy', 'wb') as f:
#   np.save(f,train_predictions_stacked)
#   np.save(f, val_predictions_stacked)
with open('predictions_debug2.npy', 'rb') as f:
  trainp = np.load(f)
  valp = np.load(f)
  trainp= logit(trainp)##return logits instead of sigomoid outputs
  valp= logit(valp)
```

In [ ]:

```
loose_iou=[]
loose_loss=[]
coverage=np.array((depths_df['salt_content'][val_idx]))
for i in tqdm(range(len(val))):
  img,mask=val[i]
  """avg iou calculation for each coverage class"""
  iou=iou_metric(torch.from_numpy(valp)[i].unsqueeze(0),mask.unsqueeze(0),logits=False)
  loss= lovasz_hinge(torch.from_numpy(valp)[i],mask).numpy()
  loose_iou.append(iou)
  loose_loss.append(loss)

loose_iou_tr=[]
loose_loss_tr=[]
coverage=np.array((depths_df['salt_content'][trn_idx]))
for i in tqdm(range(len(train))):
  img,mask=train[i]
  """avg iou calculation for each coverage class"""
```

```
    # avg iou calculation for each coverage class
    iou=iou_metric(torch.from_numpy(trainp)[i].unsqueeze(0),mask.unsqueeze(0))
    loss= lovasz_hinge(torch.from_numpy(trainp)[i],mask).numpy()
    loose_iou_tr.append(iou)
    loose_loss_tr.append(loss)
```

```
100%|██████████| 400/400 [00:02<00:00, 197.22it/s]
100%|██████████| 3600/3600 [00:18<00:00, 194.82it/s]
```

In [ ]:

```
coverage=np.array((depths_df['salt_content']))
csv_=[]
for ind,idx in enumerate(val_idx):
  block= [idx,loose_iou[ind],loose_loss[ind].item(),coverage[idx],1]
  csv_.append(block)
for ind,idx in enumerate(trn_idx):
  block= [idx,loose_iou_tr[ind],loose_loss_tr[ind].item(),coverage[idx],0]
  csv_.append(block)
```

In [ ]:

```
import csv

with open("debug.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(['idx', 'iou', 'loss_lovasz','coverage','is_val'])
    writer.writerows(csv_)
```

In [18]:

```
debug= pd.read_csv('debug.csv')
debug.head()
```

Out[18]:

	idx	iou	loss_lovasz	coverage	is_val
0	38	0.7	0.764614	4	1
1	44	0.5	1.192690	0	1
2	46	0.0	2.006062	0	1
3	47	1.0	0.063098	8	1
4	50	1.0	0.001246	0	1

In [19]:

```
debug.groupby(['iou']).count()['idx']
```

Out[19]:

```
iou
0.0     187
0.1       6
0.2      10
0.3      16
0.4      35
0.5      37
0.6      68
0.7     103
0.8     198
0.9     418
1.0    2922
Name: idx, dtype: int64
```

In [20]:

```
iou_0_val=debug[(debug['is_val']==1) & (debug['iou']==0.0)]
iou_0_trn=debug[(debug['is_val']==0) & (debug['iou']==0.0)]
```

In [21]:

```
iou_0_val.groupby(['coverage']).count()
```

Out[21]:

coverage	idx	iou	loss_lovasz	is_val
0	21	21	21	21
1	6	6	6	6
2	4	4	4	4
3	4	4	4	4
4	1	1	1	1
5	2	2	2	2
6	5	5	5	5
8	3	3	3	3
10	1	1	1	1

In [22]:

```
iou_0_trn.groupby(['coverage']).count()
```

Out[22]:

coverage	idx	iou	loss_lovasz	is_val
0	65	65	65	65
1	20	20	20	20
2	11	11	11	11
3	6	6	6	6
4	11	11	11	11
5	7	7	7	7
6	7	7	7	7
8	10	10	10	10
10	3	3	3	3

Both train and val are suffering the most when the salt content is 0-5%

In [ ]:

```
valnp= np.array([x[1].numpy() for x in val])
trainnp= np.array([x[1].numpy() for x in train])
```

In [ ]:

```
legend1= np.ones([1,128,128])
legend1[:,64:,:]=0
legend2= np.ones([1,128,128])
legend2[:,:,64:]=0
```

In [ ]:

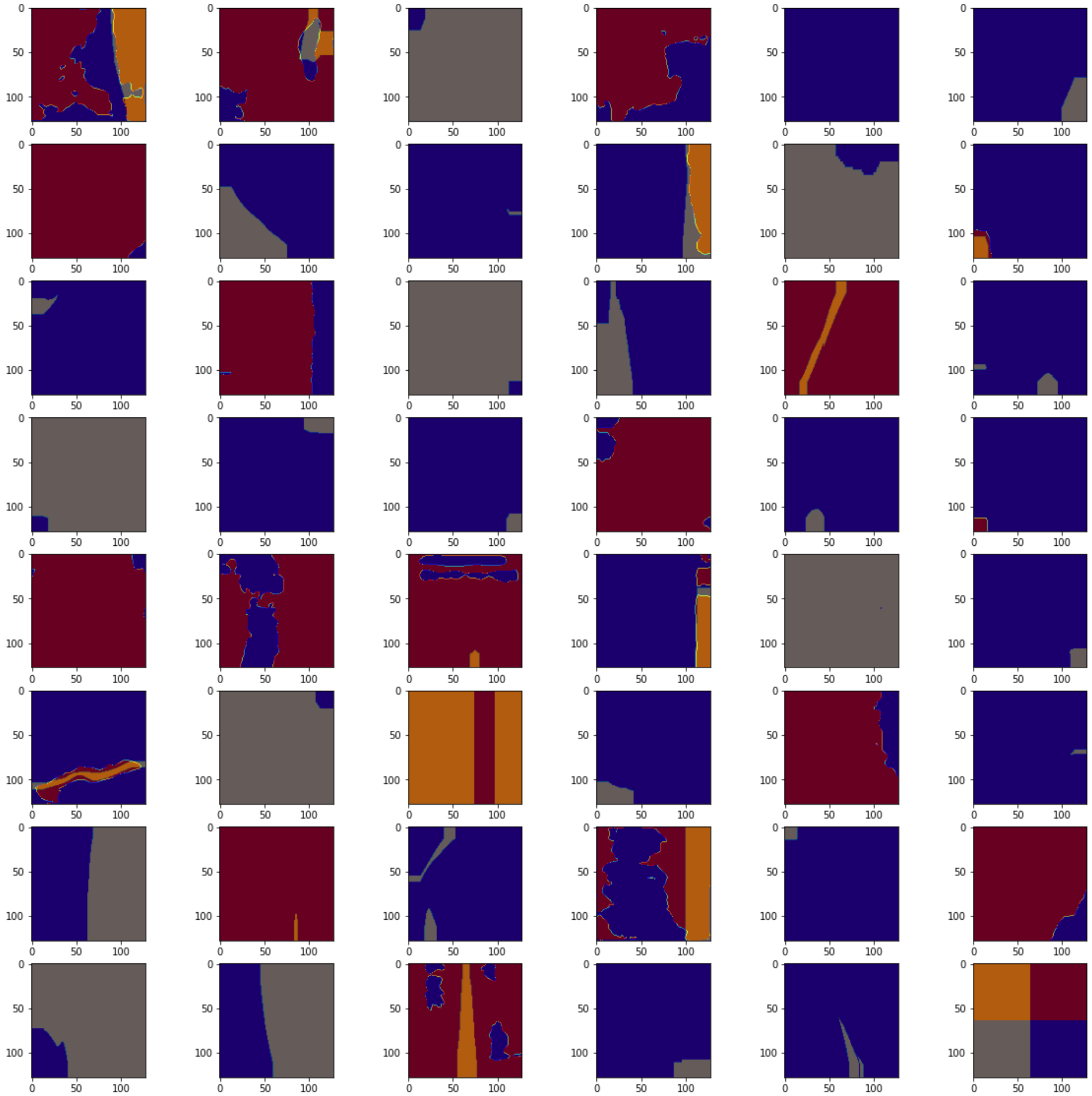
```
v_slice= (valp[iou_0_val.index.values]>0).squeeze(1)
h_slice= (valnp[iou_0_val.index.values]).squeeze(1)
h_slice=np.append(h_slice,legend1,0)
v_slice=np.append(v_slice,legend2,0)

# v_slice= (trainp[iou_0_trn.index.values<400]>0).squeeze(1)
```

```
# v_slice= (trainp[iou_0_trn.index.values-400]>0).squeeze(1)
# h_slice= (trainnp[iou_0_trn.index.values-400]).squeeze(1)
# v_slice=np.append(v_slice,legend1,0)
# h_slice=np.append(h_slice,legend2,0)
```

In [ ]:

```
w=20
h=20
fig=plt.figure(figsize=(20, 20))
columns = 6
rows = 8
for i in range(1, columns*rows +1):
    fig.add_subplot(rows, columns, i)
    plt.imshow(h_slice[i-1])
    plt.imshow(v_slice[i-1],cmap='jet',alpha=.6)
plt.show()
```



This shows small masks around corners and edges are hard for model to predict, some masks are predicted salt as nosalt.i.e false negatives are more.

In [23]:

```
Fprate= sum([(valp[x]>0).sum()>0 and valnp[x].sum()==0 for x in range(len(valnp))])
```

```
Fnrate= sum([(valp[x]>0).sum()==0 and valnp[x].sum()>0 for x in range(len(valnp))])
Fprate,Fnrate
```

```
(11,24)
```

Answer

In [ ]:

```
from skimage.transform import resize
def get_others(m):
    new_img_size=m.shape[0]
    y6 = torch.tensor(np.amax(m, axis=(0,1)),dtype=torch.float32)
    m=torch.FloatTensor(m).unsqueeze(0)
    return [m,y6]
```

In [ ]:

```
class ChannelAttentionGate(nn.Module):
    def __init__(self, channel, reduction=16):
        super(ChannelAttentionGate, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
                nn.Linear(channel, channel // reduction),
                nn.ReLU(inplace=True),
                nn.Linear(channel // reduction, channel),
                nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return y


class SpatialAttentionGate(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SpatialAttentionGate, self).__init__()
        self.fc1 = nn.Conv2d(channel, reduction, kernel_size=1, padding=0)
        self.fc2 = nn.Conv2d(reduction, 1, kernel_size=1, padding=0)

    def forward(self, x):
        x = self.fc1(x)
        x = F.relu(x, inplace=True)
        x = self.fc2(x)
        x = torch.sigmoid(x)

        return x

class Decoder(nn.Module):
    def __init__(self, in_channels, channels, out_channels ):
        super(Decoder, self).__init__()
        self.conv1 =  ConvBn2d(in_channels,  channels, kernel_size=3, padding=1)
        self.conv2 =  ConvBn2d(channels, out_channels, kernel_size=3, padding=1)
        self.cg= ChannelAttentionGate(out_channels)
        self.sg= SpatialAttentionGate(out_channels)

    def forward(self, x ):
        x = F.upsample(x, scale_factor=2, mode='bilinear', align_corners=True)#False
        x = F.relu(self.conv1(x),inplace=True)
        x = F.relu(self.conv2(x),inplace=True)
        g1= self.sg(x)
        g2= self.cg(x)
        x= g1*x+g2*x
        return x

class UNetDPSV2(nn.Module):

    def __init__(self ):
        super().__init__()
        self.resnet = torchvision.models.resnet34(pretrained=True)
```

```
        self.conv_one = nn.Sequential(
            self.resnet.conv1,
            self.resnet.bn1,
            self.resnet.relu,
        )# 64
        self.encoder2 = self.resnet.layer1  # 64
        self.encoder3 = self.resnet.layer2  #128
        self.encoder4 = self.resnet.layer3  #256
        self.encoder5 = self.resnet.layer4  #512

        self.center = nn.Sequential(
            nn.Conv2d(512, 512, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(512, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
        )

        self.decoder5 = Decoder(512+256, 512, 64)
        self.decoder4 = Decoder(64+256, 256, 64)
        self.decoder3 = Decoder(64+128, 128,  64)
        self.decoder2 = Decoder( 64+ 64, 64, 64)

        self.conv0= nn.Conv2d(256,1,1)
        self.pool= nn.AdaptiveAvgPool2d((1,1))
        self.dense=nn.Linear(512,1)


        self.logit    = nn.Sequential(
            nn.Conv2d(256, 32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(32,  1, kernel_size=1, padding=0),
        )


    def forward(self, x):
        #batch_size,C,H,W = x.shape

        x = self.conv_one(x)

        e2 = self.encoder2( x)  #; print('e2',e2.size())
        e3 = self.encoder3(e2)  #; print('e3',e3.size())
        e4 = self.encoder4(e3)  #; print('e4',e4.size())
        e5 = self.encoder5(e4)  #; print('e5',e5.size())
                            #; print('center',f.size())
        f = self.center(e5)
        # print(e5.shape,f.shape)

        d5 = self.decoder5(torch.cat([f, e5], 1))  #; print('d5',d5.size(),f.size())
        d4 = self.decoder4(torch.cat([d5, e4], 1))  #; print('d4',d4.size())
        d3= self.decoder3(torch.cat([d4, e3], 1))  #; print('d3',d3.size())
        d2 = self.decoder2(torch.cat([d3, e2], 1))  #; print('d2',d2.size())

        d2 = torch.cat((d2,
          F.upsample(d3,scale_factor=2,mode='bilinear',align_corners=False),
          F.upsample(d4,scale_factor=4,mode='bilinear',align_corners=False),
          F.upsample(d5,scale_factor=8,mode='bilinear',align_corners=False),
                      ),1)


        final = self.logit(d2)                      #; print('logit',logit.size())
        no_mask= self.dense(self.pool(e5).view(-1,512))

        return [no_mask,final]

model= UNetDPSV2()
sum([i.numel() for i in model.parameters()])
```

Out[ ]:

30545023

In [ ]:

```
def get_losslv_simple(model,inp):
  no_mask,final=model(inp[0])
  final=final*inp[-1].view(-1,1,1,1)
```

```
  mfl=nn.BCEWithLogitsLoss()
  lv=LovaszLoss()
  weights=[.1/2.1,2/2.1]
  loss=[mfl(no_mask.view(-1,1,1,1),inp[-1].view(-1,1,1,1)),lv(final,inp[1])]
  accuracy=acc(no_mask.reshape(-1),inp[-1],center=True)/len(inp[-1])
  loss=sum(x_i*y_i for x_i, y_i in zip(weights, loss))
  return loss,accuracy,final
```

In [ ]:

```
kf = StratifiedKFold(10,shuffle=True,random_state=seed)
train_ind= list(range(len(file_list)))
for fold, (trn_idx, val_idx) in enumerate(kf.split(train_ind,coverage)):
    if fold!=0 :
      continue
    print('****************************************************')
    print('*******************    fold  %d   *******************' % fold)
    print('****************************************************')

    file_list_train= [x for i,x in enumerate(file_list) if i in trn_idx]
    file_list_val= [x for i,x in enumerate(file_list) if i in val_idx]

    train = TGSSaltDataset(train_path, file_list_train,augment=transform_train,dpsv=True)
    val = TGSSaltDataset(train_path, file_list_val,augment= transform_test,dpsv=True)
    writer =SummaryWriter(FLAGS['log_dir']+'binary')
    train_loader = torch.utils.data.DataLoader(
        train,
        batch_size=FLAGS['batch_size']*3,
        num_workers=FLAGS['num_workers'],
        drop_last=False)
    test_loader = torch.utils.data.DataLoader(
        val,
        batch_size=FLAGS['batch_size']*5,
        shuffle=False,
        num_workers=FLAGS['num_workers'],
        drop_last=False)

    set_seed()
    device = 'cuda'

    model= model.to(device)
    loss_fn=get_losslv_simple
    optimizer= torch.optim.Adam(model.parameters(),lr= 3e-4)
    early_stopping = EarlyStopping(patience=300,
verbose=False,delta=.001,path=f'./models/binary_{str(fold)}.pth')
    scheduler=None
    def train_loop_fn(loader):
      model.train()
      loss_=[]
      iou_=[]
      acc_=[]
      for x, inp in enumerate(loader):
        for i in range(len(inp)):
          inp[i]=inp[i].to(device)
        loss,acc__,y_pred=loss_fn(model,inp)
        loss_.append(loss.item())
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        if scheduler:
          scheduler.step()
        IOU= iou_metric(y_pred,inp[1])
        acc_.append(acc__)
        iou_.append(IOU.item())
      print('Train[{}]: Loss={:.5f} IOU={:.5f} ACC{:.3f}'.format(x, np.average(loss_),np.average(io
u_),np.average(acc_)))
      return np.average(loss_),np.average(iou_)

    def test_loop_fn(loader):
      model.eval()
      loss_=[]
      iou_=[]
      acc_=[]
      with torch.no_grad():
        for x,inp in enumerate(loader):
          for i in range(len(inp)):
```

```
          inp[i]=inp[i].to(device)
        loss, acc__,y_pred = loss_fn(model,inp)
        IOU= iou_metric(y_pred,inp[1])
        loss_.append(loss.item())
        iou_.append(IOU.item())
        acc_.append(acc__)

    print('Validation: Loss={:.5f},IOU={:.5f},ACC={:.3f}'.format(
      np.average(loss_),np.average(iou_),np.average(acc_)))

    early_stopping(np.average(iou_), model)
    return np.average(loss_),np.average(iou_)

  for epoch in range(1,120):
    train_loss,train_iou=train_loop_fn(train_loader)
    print("Finished training epoch {}".format(epoch))
    val_loss,val_iou = test_loop_fn(test_loader)
    writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
    writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)

  scheduler=torch.optim.lr_scheduler.CyclicLR(optimizer, base_lr=5e-5, max_lr=1e-3, step_size_up=
370, step_size_down=None, mode='triangular2', cycle_momentum=False,last_epoch=-1)
  for epoch in range(120,201):
    train_loss,train_iou=train_loop_fn(train_loader)
    print("Finished training epoch {}".format(epoch))
    val_loss,val_iou = test_loop_fn(test_loader)
    writer.add_scalars('loss_exp',{'train':train_loss,'val':val_loss},epoch)
    writer.add_scalars('IOU',{'train':train_iou,'val':val_iou},epoch)
```

```
***************************************************
*******************    fold  0    *******************
***************************************************
Train[74]: Loss=1.64843 IOU=0.67006 ACC0.759
Finished training epoch 1
Validation: Loss=1.42927,IOU=0.73050,ACC=0.847
Train[74]: Loss=1.47471 IOU=0.73967 ACC0.793
Finished training epoch 2
Validation: Loss=1.36712,IOU=0.77550,ACC=0.817
Train[74]: Loss=1.43764 IOU=0.74919 ACC0.801
Finished training epoch 3
Validation: Loss=1.36281,IOU=0.77600,ACC=0.770
Train[74]: Loss=1.41660 IOU=0.75603 ACC0.799
Finished training epoch 4
Validation: Loss=1.35070,IOU=0.77800,ACC=0.802
Train[74]: Loss=1.37920 IOU=0.77014 ACC0.819
Finished training epoch 5
Validation: Loss=1.18907,IOU=0.82225,ACC=0.863
Train[74]: Loss=1.37373 IOU=0.76950 ACC0.819
Finished training epoch 6
Validation: Loss=1.19452,IOU=0.82775,ACC=0.872
Train[74]: Loss=1.32145 IOU=0.78361 ACC0.833
Finished training epoch 7
Validation: Loss=1.16653,IOU=0.83500,ACC=0.870
Train[74]: Loss=1.33122 IOU=0.79158 ACC0.829
Finished training epoch 8
Validation: Loss=1.21460,IOU=0.82200,ACC=0.863
Train[74]: Loss=1.33474 IOU=0.78472 ACC0.825
Finished training epoch 9
Validation: Loss=1.24452,IOU=0.82225,ACC=0.843
Train[74]: Loss=1.33636 IOU=0.79019 ACC0.828
Finished training epoch 10
Validation: Loss=1.14308,IOU=0.83975,ACC=0.880
Train[74]: Loss=1.31953 IOU=0.79886 ACC0.832
Finished training epoch 11
Validation: Loss=1.14251,IOU=0.84900,ACC=0.890
Train[74]: Loss=1.28548 IOU=0.80192 ACC0.849
Finished training epoch 12
Validation: Loss=1.19135,IOU=0.83925,ACC=0.895
Train[74]: Loss=1.29375 IOU=0.79581 ACC0.836
Finished training epoch 13
Validation: Loss=1.15191,IOU=0.83950,ACC=0.890
Train[74]: Loss=1.29387 IOU=0.79753 ACC0.861
Finished training epoch 14
Validation: Loss=1.18766,IOU=0.83350,ACC=0.855
Train[74]: Loss=1.26897 IOU=0.81294 ACC0.850
Finished training epoch 15
```

```
Validation: Loss=1.11497,IOU=0.85325,ACC=0.875
Train[74]: Loss=1.26123 IOU=0.81069 ACC0.853
Finished training epoch 16
Validation: Loss=1.12403,IOU=0.85025,ACC=0.905
Train[74]: Loss=1.26667 IOU=0.80064 ACC0.846
Finished training epoch 17
Validation: Loss=1.22279,IOU=0.83550,ACC=0.832
Train[74]: Loss=1.26574 IOU=0.81064 ACC0.849
Finished training epoch 18
Validation: Loss=1.10693,IOU=0.86000,ACC=0.898
Train[74]: Loss=1.29576 IOU=0.80014 ACC0.844
Finished training epoch 19
Validation: Loss=1.13578,IOU=0.84575,ACC=0.890
Train[74]: Loss=1.26165 IOU=0.80781 ACC0.841
Finished training epoch 20
Validation: Loss=1.10535,IOU=0.85300,ACC=0.895
Train[74]: Loss=1.25330 IOU=0.81081 ACC0.853
Finished training epoch 21
Validation: Loss=1.15257,IOU=0.83400,ACC=0.873
Train[74]: Loss=1.27492 IOU=0.80417 ACC0.850
Finished training epoch 22
Validation: Loss=1.11575,IOU=0.85175,ACC=0.890
Train[74]: Loss=1.25817 IOU=0.80942 ACC0.867
Finished training epoch 23
Validation: Loss=1.10025,IOU=0.85250,ACC=0.890
Train[74]: Loss=1.25649 IOU=0.80800 ACC0.847
Finished training epoch 24
Validation: Loss=1.13194,IOU=0.84900,ACC=0.885
Train[74]: Loss=1.27984 IOU=0.80064 ACC0.845
Finished training epoch 25
Validation: Loss=1.14775,IOU=0.84150,ACC=0.897
Train[74]: Loss=1.29496 IOU=0.80367 ACC0.847
Finished training epoch 26
Validation: Loss=1.10429,IOU=0.85300,ACC=0.895
Train[74]: Loss=1.24944 IOU=0.81239 ACC0.866
Finished training epoch 27
Validation: Loss=1.12205,IOU=0.85475,ACC=0.897
Train[74]: Loss=1.24694 IOU=0.81453 ACC0.861
Finished training epoch 28
Validation: Loss=1.10273,IOU=0.86325,ACC=0.903
Train[74]: Loss=1.24146 IOU=0.81933 ACC0.862
Finished training epoch 29
Validation: Loss=1.13101,IOU=0.84050,ACC=0.887
Train[74]: Loss=1.23598 IOU=0.81547 ACC0.846
Finished training epoch 30
Validation: Loss=1.14264,IOU=0.84800,ACC=0.910
Train[74]: Loss=1.23575 IOU=0.81775 ACC0.858
Finished training epoch 31
Validation: Loss=1.09352,IOU=0.85725,ACC=0.890
Train[74]: Loss=1.23517 IOU=0.82111 ACC0.860
Finished training epoch 32
Validation: Loss=1.11847,IOU=0.85475,ACC=0.885
Train[74]: Loss=1.24167 IOU=0.81539 ACC0.845
Finished training epoch 33
Validation: Loss=1.11743,IOU=0.85700,ACC=0.887
Train[74]: Loss=1.25751 IOU=0.81033 ACC0.856
Finished training epoch 34
Validation: Loss=1.11475,IOU=0.85475,ACC=0.885
Train[74]: Loss=1.27117 IOU=0.80678 ACC0.836
Finished training epoch 35
Validation: Loss=1.11138,IOU=0.85925,ACC=0.895
Train[74]: Loss=1.23731 IOU=0.81656 ACC0.859
Finished training epoch 36
Validation: Loss=1.07922,IOU=0.86875,ACC=0.905
Train[74]: Loss=1.22059 IOU=0.82500 ACC0.852
Finished training epoch 37
Validation: Loss=1.09905,IOU=0.86625,ACC=0.915
Train[74]: Loss=1.22315 IOU=0.82600 ACC0.864
Finished training epoch 38
Validation: Loss=1.07710,IOU=0.87150,ACC=0.908
Train[74]: Loss=1.21744 IOU=0.82683 ACC0.871
Finished training epoch 39
Validation: Loss=1.11030,IOU=0.85175,ACC=0.903
Train[74]: Loss=1.23619 IOU=0.82675 ACC0.858
Finished training epoch 40
Validation: Loss=1.13522,IOU=0.85125,ACC=0.895
Train[74]: Loss=1.22001 IOU=0.83003 ACC0.871
```

```
Finished training epoch 41
Validation: Loss=1.10086,IOU=0.85725,ACC=0.910
Train[74]: Loss=1.22858 IOU=0.81853 ACC0.865
Finished training epoch 42
Validation: Loss=1.09713,IOU=0.86225,ACC=0.905
Train[74]: Loss=1.20951 IOU=0.82511 ACC0.867
Finished training epoch 43
Validation: Loss=1.10249,IOU=0.85850,ACC=0.897
Train[74]: Loss=1.19406 IOU=0.83686 ACC0.866
Finished training epoch 44
Validation: Loss=1.07870,IOU=0.86475,ACC=0.905
Train[74]: Loss=1.21522 IOU=0.82156 ACC0.878
Finished training epoch 45
Validation: Loss=1.07788,IOU=0.86525,ACC=0.925
Train[74]: Loss=1.19440 IOU=0.83508 ACC0.860
Finished training epoch 46
Validation: Loss=1.09241,IOU=0.86775,ACC=0.903
Train[74]: Loss=1.20551 IOU=0.82653 ACC0.864
Finished training epoch 47
Validation: Loss=1.07937,IOU=0.86100,ACC=0.917
Train[74]: Loss=1.22250 IOU=0.81975 ACC0.867
Finished training epoch 48
Validation: Loss=1.12888,IOU=0.85875,ACC=0.887
Train[74]: Loss=1.20564 IOU=0.82722 ACC0.861
Finished training epoch 49
Validation: Loss=1.08424,IOU=0.87600,ACC=0.903
Train[74]: Loss=1.17152 IOU=0.84336 ACC0.875
Finished training epoch 50
Validation: Loss=1.05359,IOU=0.87575,ACC=0.905
Train[74]: Loss=1.20632 IOU=0.83294 ACC0.866
Finished training epoch 51
Validation: Loss=1.07988,IOU=0.87200,ACC=0.912
Train[74]: Loss=1.19187 IOU=0.83572 ACC0.869
Finished training epoch 52
Validation: Loss=1.11566,IOU=0.85325,ACC=0.912
Train[74]: Loss=1.20493 IOU=0.82808 ACC0.866
Finished training epoch 53
Validation: Loss=1.09716,IOU=0.86500,ACC=0.893
Train[74]: Loss=1.20759 IOU=0.82483 ACC0.876
Finished training epoch 54
Validation: Loss=1.08566,IOU=0.87150,ACC=0.918
Train[74]: Loss=1.20310 IOU=0.83172 ACC0.876
Finished training epoch 55
Validation: Loss=1.08495,IOU=0.86925,ACC=0.912
Train[74]: Loss=1.19665 IOU=0.83242 ACC0.859
Finished training epoch 56
Validation: Loss=1.09039,IOU=0.87000,ACC=0.900
Train[74]: Loss=1.20137 IOU=0.83119 ACC0.856
Finished training epoch 57
Validation: Loss=1.11640,IOU=0.86175,ACC=0.907
Train[74]: Loss=1.18654 IOU=0.83736 ACC0.874
Finished training epoch 58
Validation: Loss=1.10156,IOU=0.86025,ACC=0.925
Train[74]: Loss=1.18985 IOU=0.83306 ACC0.873
Finished training epoch 59
Validation: Loss=1.10410,IOU=0.86550,ACC=0.923
Train[74]: Loss=1.20779 IOU=0.82503 ACC0.870
Finished training epoch 60
Validation: Loss=1.08688,IOU=0.86800,ACC=0.895
Train[74]: Loss=1.20202 IOU=0.82569 ACC0.873
Finished training epoch 61
Validation: Loss=1.08357,IOU=0.86750,ACC=0.910
Train[74]: Loss=1.20490 IOU=0.82797 ACC0.865
Finished training epoch 62
Validation: Loss=1.11241,IOU=0.85925,ACC=0.912
Train[74]: Loss=1.18579 IOU=0.83261 ACC0.868
Finished training epoch 63
Validation: Loss=1.08089,IOU=0.86725,ACC=0.900
Train[74]: Loss=1.17454 IOU=0.84167 ACC0.865
Finished training epoch 64
Validation: Loss=1.05288,IOU=0.87900,ACC=0.918
Train[74]: Loss=1.19430 IOU=0.83083 ACC0.874
Finished training epoch 65
Validation: Loss=1.07881,IOU=0.87250,ACC=0.920
Train[74]: Loss=1.19180 IOU=0.83756 ACC0.880
Finished training epoch 66
Validation: Loss=1.07245,IOU=0.87525,ACC=0.908
```

```
Train[74]: Loss=1.18133 IOU=0.84122 ACC0.866
Finished training epoch 67
Validation: Loss=1.05875,IOU=0.87875,ACC=0.918
Train[74]: Loss=1.18585 IOU=0.83639 ACC0.878
Finished training epoch 68
Validation: Loss=1.09632,IOU=0.86700,ACC=0.903
Train[74]: Loss=1.16961 IOU=0.83883 ACC0.881
Finished training epoch 69
Validation: Loss=1.06061,IOU=0.87875,ACC=0.925
Train[74]: Loss=1.17430 IOU=0.84228 ACC0.886
Finished training epoch 70
Validation: Loss=1.10052,IOU=0.87025,ACC=0.905
Train[74]: Loss=1.14994 IOU=0.84469 ACC0.881
Finished training epoch 71
Validation: Loss=1.03403,IOU=0.88150,ACC=0.910
Train[74]: Loss=1.17740 IOU=0.84275 ACC0.875
Finished training epoch 72
Validation: Loss=1.06681,IOU=0.87000,ACC=0.912
Train[74]: Loss=1.16143 IOU=0.84628 ACC0.872
Finished training epoch 73
Validation: Loss=1.08371,IOU=0.87875,ACC=0.915
Train[74]: Loss=1.16147 IOU=0.84031 ACC0.878
Finished training epoch 74
Validation: Loss=1.06086,IOU=0.88050,ACC=0.922
Train[74]: Loss=1.14336 IOU=0.84867 ACC0.884
Finished training epoch 75
Validation: Loss=1.04431,IOU=0.87700,ACC=0.933
Train[74]: Loss=1.17690 IOU=0.84208 ACC0.877
Finished training epoch 76
Validation: Loss=1.07238,IOU=0.86875,ACC=0.910
Train[74]: Loss=1.14321 IOU=0.85419 ACC0.884
Finished training epoch 77
Validation: Loss=1.07485,IOU=0.87700,ACC=0.922
Train[74]: Loss=1.16617 IOU=0.84292 ACC0.877
Finished training epoch 78
Validation: Loss=1.09327,IOU=0.86925,ACC=0.887
Train[74]: Loss=1.17165 IOU=0.84425 ACC0.873
Finished training epoch 79
Validation: Loss=1.08683,IOU=0.87600,ACC=0.905
Train[74]: Loss=1.17568 IOU=0.83792 ACC0.881
Finished training epoch 80
Validation: Loss=1.06650,IOU=0.87450,ACC=0.908
Train[74]: Loss=1.16631 IOU=0.84528 ACC0.874
Finished training epoch 81
Validation: Loss=1.08713,IOU=0.87125,ACC=0.915
Train[74]: Loss=1.16420 IOU=0.84203 ACC0.882
Finished training epoch 82
Validation: Loss=1.07929,IOU=0.87075,ACC=0.912
Train[74]: Loss=1.17712 IOU=0.83464 ACC0.886
Finished training epoch 83
Validation: Loss=1.08493,IOU=0.86825,ACC=0.907
Train[74]: Loss=1.16283 IOU=0.84192 ACC0.868
Finished training epoch 84
Validation: Loss=1.07019,IOU=0.87650,ACC=0.908
Train[74]: Loss=1.19396 IOU=0.83311 ACC0.879
Finished training epoch 85
Validation: Loss=1.09140,IOU=0.86775,ACC=0.900
Train[74]: Loss=1.17343 IOU=0.83872 ACC0.888
Finished training epoch 86
Validation: Loss=1.07866,IOU=0.87225,ACC=0.915
Train[74]: Loss=1.17028 IOU=0.84183 ACC0.874
Finished training epoch 87
Validation: Loss=1.04578,IOU=0.88025,ACC=0.907
Train[74]: Loss=1.16331 IOU=0.84122 ACC0.882
Finished training epoch 88
Validation: Loss=1.05359,IOU=0.87925,ACC=0.918
Train[74]: Loss=1.15064 IOU=0.85278 ACC0.886
Finished training epoch 89
Validation: Loss=1.08120,IOU=0.87800,ACC=0.905
Train[74]: Loss=1.15282 IOU=0.85042 ACC0.891
Finished training epoch 90
Validation: Loss=1.06305,IOU=0.87575,ACC=0.915
Train[74]: Loss=1.14974 IOU=0.84742 ACC0.894
Finished training epoch 91
Validation: Loss=1.06937,IOU=0.88275,ACC=0.912
Train[74]: Loss=1.13126 IOU=0.85883 ACC0.882
Finished training epoch 92
```

```
Validation: Loss=1.03949,IOU=0.88500,ACC=0.910
Train[74]: Loss=1.14068 IOU=0.85156 ACC0.886
Finished training epoch 93
Validation: Loss=1.06076,IOU=0.88150,ACC=0.918
Train[74]: Loss=1.14151 IOU=0.84781 ACC0.874
Finished training epoch 94
Validation: Loss=1.05125,IOU=0.87575,ACC=0.915
Train[74]: Loss=1.13565 IOU=0.85122 ACC0.891
Finished training epoch 95
Validation: Loss=1.04558,IOU=0.87675,ACC=0.903
Train[74]: Loss=1.13938 IOU=0.85656 ACC0.886
Finished training epoch 96
Validation: Loss=1.04249,IOU=0.88400,ACC=0.920
Train[74]: Loss=1.13136 IOU=0.85672 ACC0.878
Finished training epoch 97
Validation: Loss=1.07618,IOU=0.87600,ACC=0.917
Train[74]: Loss=1.13127 IOU=0.85508 ACC0.897
Finished training epoch 98
Validation: Loss=1.02816,IOU=0.88750,ACC=0.915
Train[74]: Loss=1.13811 IOU=0.85417 ACC0.897
Finished training epoch 99
Validation: Loss=1.06224,IOU=0.87850,ACC=0.902
Train[74]: Loss=1.12434 IOU=0.86283 ACC0.889
Finished training epoch 100
Validation: Loss=1.04147,IOU=0.88650,ACC=0.912
Train[74]: Loss=1.17422 IOU=0.83875 ACC0.889
Finished training epoch 101
Validation: Loss=1.06978,IOU=0.86025,ACC=0.897
Train[74]: Loss=1.13539 IOU=0.85911 ACC0.881
Finished training epoch 102
Validation: Loss=1.11681,IOU=0.86950,ACC=0.918
Train[74]: Loss=1.14107 IOU=0.85064 ACC0.882
Finished training epoch 103
Validation: Loss=1.05681,IOU=0.88350,ACC=0.920
Train[74]: Loss=1.12596 IOU=0.86278 ACC0.887
Finished training epoch 104
Validation: Loss=1.07996,IOU=0.87925,ACC=0.907
Train[74]: Loss=1.16252 IOU=0.84389 ACC0.881
Finished training epoch 105
Validation: Loss=1.05664,IOU=0.87125,ACC=0.900
Train[74]: Loss=1.14404 IOU=0.85042 ACC0.889
Finished training epoch 106
Validation: Loss=1.06342,IOU=0.87800,ACC=0.897
Train[74]: Loss=1.12929 IOU=0.85969 ACC0.892
Finished training epoch 107
Validation: Loss=1.08030,IOU=0.87925,ACC=0.915
Train[74]: Loss=1.15455 IOU=0.85125 ACC0.883
Finished training epoch 108
Validation: Loss=1.06689,IOU=0.87650,ACC=0.897
Train[74]: Loss=1.12928 IOU=0.85697 ACC0.894
Finished training epoch 109
Validation: Loss=1.06222,IOU=0.87975,ACC=0.918
Train[74]: Loss=1.12857 IOU=0.85883 ACC0.903
Finished training epoch 110
Validation: Loss=1.06324,IOU=0.86675,ACC=0.927
Train[74]: Loss=1.14097 IOU=0.85583 ACC0.887
Finished training epoch 111
Validation: Loss=1.05529,IOU=0.88175,ACC=0.915
Train[74]: Loss=1.12295 IOU=0.86203 ACC0.889
Finished training epoch 112
Validation: Loss=1.06920,IOU=0.87400,ACC=0.920
Train[74]: Loss=1.12847 IOU=0.85392 ACC0.891
Finished training epoch 113
Validation: Loss=1.07265,IOU=0.87150,ACC=0.917
Train[74]: Loss=1.10538 IOU=0.86553 ACC0.900
Finished training epoch 114
Validation: Loss=1.06718,IOU=0.87875,ACC=0.932
Train[74]: Loss=1.10324 IOU=0.86081 ACC0.899
Finished training epoch 115
Validation: Loss=1.05264,IOU=0.87725,ACC=0.903
Train[74]: Loss=1.12223 IOU=0.85447 ACC0.897
Finished training epoch 116
Validation: Loss=1.04635,IOU=0.88750,ACC=0.903
Train[74]: Loss=1.13045 IOU=0.85542 ACC0.887
Finished training epoch 117
Validation: Loss=1.06721,IOU=0.87700,ACC=0.907
Train[74]: Loss=1.10996 IOU=0.86700 ACC0.895
```

```
Finished training epoch 118
Validation: Loss=1.06328,IOU=0.88375,ACC=0.897
Train[74]: Loss=1.12288 IOU=0.86153 ACC0.897
Finished training epoch 119
Validation: Loss=1.06516,IOU=0.87975,ACC=0.905
Train[74]: Loss=1.11530 IOU=0.86114 ACC0.903
Finished training epoch 120
Validation: Loss=1.04122,IOU=0.88250,ACC=0.917
Train[74]: Loss=1.12024 IOU=0.86228 ACC0.896
Finished training epoch 121
Validation: Loss=1.10945,IOU=0.87175,ACC=0.922
Train[74]: Loss=1.15379 IOU=0.84706 ACC0.887
Finished training epoch 122
Validation: Loss=1.10877,IOU=0.86675,ACC=0.895
Train[74]: Loss=1.16668 IOU=0.84158 ACC0.878
Finished training epoch 123
Validation: Loss=1.08616,IOU=0.86550,ACC=0.910
Train[74]: Loss=1.23799 IOU=0.82208 ACC0.854
Finished training epoch 124
Validation: Loss=1.18849,IOU=0.83450,ACC=0.878
Train[74]: Loss=1.26269 IOU=0.81581 ACC0.862
Finished training epoch 125
Validation: Loss=1.09935,IOU=0.86075,ACC=0.915
Train[74]: Loss=1.19940 IOU=0.83411 ACC0.874
Finished training epoch 126
Validation: Loss=1.09076,IOU=0.86825,ACC=0.900
Train[74]: Loss=1.16096 IOU=0.84567 ACC0.882
Finished training epoch 127
Validation: Loss=1.06391,IOU=0.87150,ACC=0.903
Train[74]: Loss=1.15187 IOU=0.84686 ACC0.879
Finished training epoch 128
Validation: Loss=1.05096,IOU=0.88100,ACC=0.912
Train[74]: Loss=1.11922 IOU=0.85947 ACC0.901
Finished training epoch 129
Validation: Loss=1.02762,IOU=0.88450,ACC=0.918
Train[74]: Loss=1.12695 IOU=0.85914 ACC0.895
Finished training epoch 130
Validation: Loss=1.01860,IOU=0.89275,ACC=0.915
Train[74]: Loss=1.11036 IOU=0.86661 ACC0.897
Finished training epoch 131
Validation: Loss=1.05913,IOU=0.88550,ACC=0.912
Train[74]: Loss=1.11993 IOU=0.85981 ACC0.892
Finished training epoch 132
Validation: Loss=1.06362,IOU=0.87625,ACC=0.917
Train[74]: Loss=1.15083 IOU=0.84894 ACC0.885
Finished training epoch 133
Validation: Loss=1.09388,IOU=0.87975,ACC=0.918
Train[74]: Loss=1.15090 IOU=0.85131 ACC0.875
Finished training epoch 134
Validation: Loss=1.09754,IOU=0.87100,ACC=0.895
Train[74]: Loss=1.17861 IOU=0.83917 ACC0.884
Finished training epoch 135
Validation: Loss=1.03735,IOU=0.88850,ACC=0.915
Train[74]: Loss=1.10993 IOU=0.85819 ACC0.891
Finished training epoch 136
Validation: Loss=1.02776,IOU=0.88925,ACC=0.910
Train[74]: Loss=1.12042 IOU=0.85911 ACC0.891
Finished training epoch 137
Validation: Loss=1.04227,IOU=0.87850,ACC=0.915
Train[74]: Loss=1.12971 IOU=0.85592 ACC0.904
Finished training epoch 138
Validation: Loss=1.04802,IOU=0.88925,ACC=0.910
Train[74]: Loss=1.11213 IOU=0.86650 ACC0.906
Finished training epoch 139
Validation: Loss=1.03685,IOU=0.88525,ACC=0.920
Train[74]: Loss=1.10881 IOU=0.86156 ACC0.897
Finished training epoch 140
Validation: Loss=1.05138,IOU=0.89025,ACC=0.922
Train[74]: Loss=1.08909 IOU=0.86986 ACC0.899
Finished training epoch 141
Validation: Loss=1.06222,IOU=0.88200,ACC=0.915
Train[74]: Loss=1.10543 IOU=0.86150 ACC0.900
Finished training epoch 142
Validation: Loss=1.04166,IOU=0.88750,ACC=0.922
Train[74]: Loss=1.10241 IOU=0.86417 ACC0.909
Finished training epoch 143
Validation: Loss=1.04501,IOU=0.88550,ACC=0.922
```

```
Train[74]: Loss=1.10623 IOU=0.86464 ACC0.900
Finished training epoch 144
Validation: Loss=1.04579,IOU=0.88725,ACC=0.920
Train[74]: Loss=1.09048 IOU=0.87322 ACC0.903
Finished training epoch 145
Validation: Loss=1.04614,IOU=0.88775,ACC=0.922
Train[74]: Loss=1.08923 IOU=0.87250 ACC0.903
Finished training epoch 146
Validation: Loss=1.04781,IOU=0.88450,ACC=0.925
Train[74]: Loss=1.09217 IOU=0.86608 ACC0.917
Finished training epoch 147
Validation: Loss=1.04883,IOU=0.88750,ACC=0.912
Train[74]: Loss=1.07248 IOU=0.87933 ACC0.907
Finished training epoch 148
Validation: Loss=1.04654,IOU=0.88475,ACC=0.927
Train[74]: Loss=1.08002 IOU=0.87550 ACC0.904
Finished training epoch 149
Validation: Loss=1.03993,IOU=0.89175,ACC=0.920
Train[74]: Loss=1.07004 IOU=0.87925 ACC0.919
Finished training epoch 150
Validation: Loss=1.03948,IOU=0.89175,ACC=0.927
Train[74]: Loss=1.09251 IOU=0.86994 ACC0.912
Finished training epoch 151
Validation: Loss=1.03609,IOU=0.89400,ACC=0.915
Train[74]: Loss=1.09299 IOU=0.87078 ACC0.923
Finished training epoch 152
Validation: Loss=1.04356,IOU=0.88925,ACC=0.907
Train[74]: Loss=1.07872 IOU=0.87536 ACC0.911
Finished training epoch 153
Validation: Loss=1.03153,IOU=0.89375,ACC=0.920
Train[74]: Loss=1.08082 IOU=0.87528 ACC0.913
Finished training epoch 154
Validation: Loss=1.03692,IOU=0.88850,ACC=0.910
Train[74]: Loss=1.09410 IOU=0.87369 ACC0.914
Finished training epoch 155
Validation: Loss=1.03828,IOU=0.88350,ACC=0.915
Train[74]: Loss=1.08425 IOU=0.87228 ACC0.917
Finished training epoch 156
Validation: Loss=1.04744,IOU=0.88300,ACC=0.910
Train[74]: Loss=1.06568 IOU=0.88181 ACC0.916
Finished training epoch 157
Validation: Loss=1.04847,IOU=0.88800,ACC=0.912
Train[74]: Loss=1.07395 IOU=0.87669 ACC0.923
Finished training epoch 158
Validation: Loss=1.04038,IOU=0.88300,ACC=0.918
Train[74]: Loss=1.06728 IOU=0.87903 ACC0.921
Finished training epoch 159
Validation: Loss=1.04539,IOU=0.88900,ACC=0.920
Train[74]: Loss=1.06498 IOU=0.88031 ACC0.914
Finished training epoch 160
Validation: Loss=1.04835,IOU=0.89150,ACC=0.910
Train[74]: Loss=1.07368 IOU=0.87631 ACC0.911
Finished training epoch 161
Validation: Loss=1.04222,IOU=0.88675,ACC=0.922
Train[74]: Loss=1.08048 IOU=0.87619 ACC0.921
Finished training epoch 162
Validation: Loss=1.04532,IOU=0.89375,ACC=0.923
Train[74]: Loss=1.08463 IOU=0.87469 ACC0.910
Finished training epoch 163
Validation: Loss=1.04750,IOU=0.88900,ACC=0.925
Train[74]: Loss=1.06860 IOU=0.88308 ACC0.919
Finished training epoch 164
Validation: Loss=1.04104,IOU=0.88475,ACC=0.922
Train[74]: Loss=1.07873 IOU=0.87569 ACC0.920
Finished training epoch 165
Validation: Loss=1.04891,IOU=0.89025,ACC=0.918
Train[74]: Loss=1.06078 IOU=0.88314 ACC0.919
Finished training epoch 166
Validation: Loss=1.04063,IOU=0.89075,ACC=0.920
Train[74]: Loss=1.06424 IOU=0.88519 ACC0.918
Finished training epoch 167
Validation: Loss=1.04594,IOU=0.89000,ACC=0.915
Train[74]: Loss=1.06269 IOU=0.88314 ACC0.928
Finished training epoch 168
Validation: Loss=1.05347,IOU=0.89125,ACC=0.920
Train[74]: Loss=1.04730 IOU=0.88939 ACC0.921
Finished training epoch 169
```

```
Finished training epoch 169
Validation: Loss=1.04819,IOU=0.89275,ACC=0.910
Train[74]: Loss=1.06877 IOU=0.88028 ACC0.929
Finished training epoch 170
Validation: Loss=1.05018,IOU=0.89475,ACC=0.917
Train[74]: Loss=1.06232 IOU=0.87731 ACC0.926
Finished training epoch 171
Validation: Loss=1.05337,IOU=0.89225,ACC=0.933
Train[74]: Loss=1.05454 IOU=0.88525 ACC0.934
Finished training epoch 172
Validation: Loss=1.04656,IOU=0.89075,ACC=0.920
Train[74]: Loss=1.06241 IOU=0.88417 ACC0.921
Finished training epoch 173
Validation: Loss=1.03725,IOU=0.89075,ACC=0.918
Train[74]: Loss=1.07201 IOU=0.87825 ACC0.919
Finished training epoch 174
Validation: Loss=1.05750,IOU=0.88525,ACC=0.918
Train[74]: Loss=1.06280 IOU=0.88478 ACC0.928
Finished training epoch 175
Validation: Loss=1.05579,IOU=0.88650,ACC=0.920
Train[74]: Loss=1.07919 IOU=0.87656 ACC0.925
Finished training epoch 176
Validation: Loss=1.05913,IOU=0.89050,ACC=0.925
Train[74]: Loss=1.06593 IOU=0.88150 ACC0.926
Finished training epoch 177
Validation: Loss=1.05516,IOU=0.88400,ACC=0.918
Train[74]: Loss=1.03638 IOU=0.89386 ACC0.925
Finished training epoch 178
Validation: Loss=1.05615,IOU=0.88825,ACC=0.930
Train[74]: Loss=1.05473 IOU=0.88147 ACC0.928
Finished training epoch 179
Validation: Loss=1.05059,IOU=0.88950,ACC=0.935
Train[74]: Loss=1.04392 IOU=0.89208 ACC0.932
Finished training epoch 180
Validation: Loss=1.04794,IOU=0.88900,ACC=0.925
Train[74]: Loss=1.06810 IOU=0.88006 ACC0.923
Finished training epoch 181
Validation: Loss=1.06143,IOU=0.88950,ACC=0.930
Train[74]: Loss=1.05487 IOU=0.88742 ACC0.923
Finished training epoch 182
Validation: Loss=1.05841,IOU=0.89050,ACC=0.927
Train[74]: Loss=1.06110 IOU=0.88453 ACC0.924
Finished training epoch 183
Validation: Loss=1.05533,IOU=0.89050,ACC=0.927
Train[74]: Loss=1.05847 IOU=0.88344 ACC0.923
Finished training epoch 184
Validation: Loss=1.05486,IOU=0.89075,ACC=0.927
Train[74]: Loss=1.06302 IOU=0.88503 ACC0.928
Finished training epoch 185
Validation: Loss=1.05066,IOU=0.88975,ACC=0.922
Train[74]: Loss=1.04822 IOU=0.88628 ACC0.934
Finished training epoch 186
Validation: Loss=1.05462,IOU=0.88900,ACC=0.920
Train[74]: Loss=1.05264 IOU=0.88567 ACC0.926
Finished training epoch 187
Validation: Loss=1.05190,IOU=0.88200,ACC=0.923
Train[74]: Loss=1.04840 IOU=0.88833 ACC0.929
Finished training epoch 188
Validation: Loss=1.05527,IOU=0.88775,ACC=0.920
Train[74]: Loss=1.06702 IOU=0.88269 ACC0.930
Finished training epoch 189
Validation: Loss=1.05540,IOU=0.89250,ACC=0.927
Train[74]: Loss=1.05580 IOU=0.88392 ACC0.932
Finished training epoch 190
Validation: Loss=1.05127,IOU=0.89200,ACC=0.920
Train[74]: Loss=1.05841 IOU=0.88892 ACC0.936
Finished training epoch 191
Validation: Loss=1.05130,IOU=0.89050,ACC=0.927
Train[74]: Loss=1.06149 IOU=0.88206 ACC0.937
Finished training epoch 192
Validation: Loss=1.05309,IOU=0.88875,ACC=0.933
Train[74]: Loss=1.06093 IOU=0.88217 ACC0.932
Finished training epoch 193
Validation: Loss=1.04961,IOU=0.88975,ACC=0.933
Train[74]: Loss=1.06141 IOU=0.88686 ACC0.934
Finished training epoch 194
Validation: Loss=1.05484,IOU=0.88725,ACC=0.927
Train[74]: Loss=1.03768 IOU=0.89439 ACC0.932
```

```
Finished training epoch 195
Validation: Loss=1.04868,IOU=0.88950,ACC=0.930
Train[74]: Loss=1.04580 IOU=0.89225 ACC0.928
Finished training epoch 196
Validation: Loss=1.06082,IOU=0.88700,ACC=0.920
Train[74]: Loss=1.03719 IOU=0.89206 ACC0.929
Finished training epoch 197
Validation: Loss=1.06045,IOU=0.88675,ACC=0.920
Train[74]: Loss=1.05556 IOU=0.89025 ACC0.931
Finished training epoch 198
Validation: Loss=1.05632,IOU=0.88925,ACC=0.922
Train[74]: Loss=1.06179 IOU=0.88531 ACC0.928
Finished training epoch 199
Validation: Loss=1.06039,IOU=0.88400,ACC=0.922
Train[74]: Loss=1.05415 IOU=0.88625 ACC0.929
Finished training epoch 200
Validation: Loss=1.05480,IOU=0.88825,ACC=0.922
```

In [30]:

```
%reload_ext tensorboard
%tensorboard --logdir {FLAGS['log_dir']+'binary'}
```

In [25]:

```
debug= pd.read_csv('debug2.csv')
debug.groupby(['iou']).count()['idx']#187 to 148 0 iou count;2922 to 3304 1 iou count
```

Out[25]:

```
iou
0.0     148
0.1       4
0.2       5
0.3       5
0.4       6
0.5       8
0.6      18
0.7      41
0.8     103
0.9     358
1.0    3304
Name: idx, dtype: int64
```

In [ ]:

```
iou_0_val.groupby(['coverage']).count() #every coverage benefitted by this little change eg 21->19
0 coverage
```

Out[ ]:

	idx	iou	loss_lovasz	is_val
coverage				
0	19	19	19	19
1	5	5	5	5
2	2	2	2	2
3	4	4	4	4
4	1	1	1	1
5	1	1	1	1
6	4	4	4	4
8	1	1	1	1
10	1	1	1	1

In [ ]:

```
iou_0_trn.groupby(['coverage']).count()
```

Out[ ]:

coverage	idx	iou	loss_lovasz	is_val
0	51	51	51	51
1	16	16	15	16
2	12	12	11	12
3	4	4	4	4
4	9	9	9	9
5	3	3	3	3
6	2	2	2	2
8	9	9	8	9
10	4	4	4	4

In [29]:

```
Fprate= sum([(valp[x]>0).sum()>0 and valnp[x].sum()==0 for x in range(len(valnp))])
Fnrate= sum([(valp[x]>0).sum()==0 and valnp[x].sum()>0 for x in range(len(valnp))])
Fprate,Fnrate# Fprate reduced from 11 to 6
```

```
(6,24)
```

Inference

In [ ]:

```
#single model
device = 'cuda'
model = UNetDPSV2()
model.load_state_dict(torch.load(f'./models/binary_0.pth'),strict=False)
model.eval()
model.to(device)
print('done')
```

```
done
```

In [ ]:

```
all_predictions_stacked=None
all_predictions=None
```

In [ ]:

```
test_path = os.path.join(directory, 'test')
test_path2= os.path.join(test_path,'images')
while(5):
  try:
    test_path_list= os.listdir(test_path2)
    break
  except:
    pass

test_file_list = [f.split('.')[0] for f in test_path_list]
```

In [ ]:

```
tests_dataset = TGSSaltDataset(test_path, test_file_list, is_test = True,augment= transform_test)
tests_loader = torch.utils.data.DataLoader(
       tests_dataset,
       batch_size=32*3,
```

```
        shuffle=False,
        num_workers=10)
```

In [ ]:

```
############## FOR SINGLE MODEL
all_predictions = []
for image in tqdm(tests_loader):
  with torch.no_grad():
    image = image.to(device)
    bin,y_pred= model(image)
    y_pred = F.sigmoid(y_pred)*(bin.view(-1,1,1,1)>0)
    all_predictions.append(y_pred)

all_predictions_stacked=torch.cat(all_predictions,0)
print(all_predictions_stacked.shape)
all_predictions=None
```

```
100%|██████████| 188/188 [06:35<00:00,  2.10s/it]
```

```
torch.Size([18000, 1, 128, 128])
```

In [ ]:

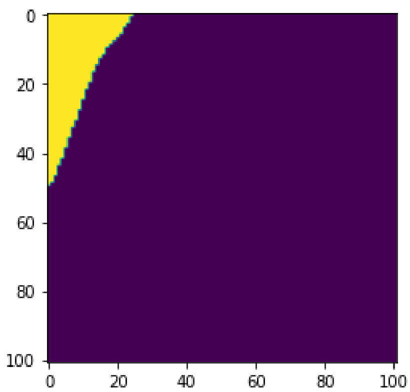
```
all_predictions_stacked=all_predictions_stacked.squeeze(1).cpu().numpy()
all_predictions_stacked= all_predictions_stacked[:,13:-14,13:-14]
```

In [ ]:

```
plt.imshow(all_predictions_stacked[363]>.5)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7f6de43e9f60>
```



In [ ]:

```
#https://stackoverflow.com/questions/42798659/how-to-remove-small-connected-objects-using-opencv
def remove_smasks(array_):
  """ PostProcessing Removes small masks of 10 pixels, as it  is most likely a result of
overfitting."""
  array_= list(array_)
  for ind,curr in enumerate(array_):
    nb_components, output, stats, centroids = cv2.connectedComponentsWithStats(curr, connectivity=8
)
    sizes = stats[1:, -1]; nb_components = nb_components - 1
    min_size = 10
    img2 = np.zeros((output.shape))
    for i in range(0, nb_components):
        if sizes[i] >= min_size:
            img2[output == i + 1] = 255
    array_[ind]= img2
  return np.array(array_)
```

In [ ]:

```
threshold = best_threshold=0.5
binary_prediction = (all_predictions_stacked > threshold).astype(np.uint8)
binary_prediction=remove_smasks(binary_prediction)
binary_prediction= (binary_prediction==255)

def rle_encoding(x):
    dots = np.where(x.T.flatten() == 1)[0]
    run_lengths = []
    prev = -2
    for b in dots:
        if (b > prev+1): run_lengths.extend((b + 1, 0))
        run_lengths[-1] += 1
        prev = b
    return run_lengths


all_masks = []
for p_mask in list(binary_prediction):
    p_mask = rle_encoding(p_mask)
    all_masks.append(' '.join(map(str, p_mask)))
```

In [ ]:

```
submit = pd.DataFrame([test_file_list, all_masks]).T
submit.columns = ['id', 'rle_mask']
submit.to_csv('/content/sample_data/dec16.csv', index = False)
```

Conclusion

In [28]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model",'CV', "PublicLB","PrivateLB"]
x.add_row(["Baseline",.800, .7880,.8210])
x.add_row(["Baseline_corrected",.8150, .7993,.8242])
x.add_row(["Baseline_lovasz",.8330, .8246,.8471])
x.add_row(["Scse_hypercol",.8350, .8339,.8482])
x.add_row(["Binary_judge",.8920, .8449,.8666])

print(x)
```

```
+--------------------+-------+----------+-----------+
|       Model        |   CV  | PublicLB | PrivateLB |
+--------------------+-------+----------+-----------+
|      Baseline      |  0.8  |  0.788   |   0.821   |
| Baseline_corrected | 0.815 |  0.7993  |   0.8242  |
|  Baseline_lovasz   | 0.833 |  0.8246  |   0.8471  |
|   Scse_hypercol    | 0.835 |  0.8339  |   0.8482  |
|    Binary_judge    | 0.892 |  0.8449  |   0.8666  |
+--------------------+-------+----------+-----------+
```