**PROJECT CODE:**

!pip install torchmetrics

from google.colab import drive

drive.mount('/content/drive')

import torch

import torch.nn as nn

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import torchvision

from torchvision import datasets,transforms

from tqdm import tqdm

import cv2

from torch.utils.data import Dataset, DataLoader

import torch.optim as optim

from PIL import Image

import torchvision.transforms as transforms

import os

import torch.nn.functional as F

from torchmetrics import Dice, JaccardIndex

# config

LEARNING\_RATE = 3e-5

SPLIT=0.2

DEVICE = "cuda" if torch.cuda.is\_available() else "cpu"

BATCH\_SIZE = 16

EPOCHS = 10 # changes 30

NUM\_WORKERS = 4

IMAGE\_HEIGHT = 210

IMAGE\_WIDTH = 290

PIN\_MEMORY = True

TRAIN\_IMG\_DIR = r'/content/drive/MyDrive/leela1999/ultrasound-nerve-segmentation/image'

TRAIN\_MASK\_DIR = r'/content/drive/MyDrive/leela1999/ultrasound-nerve-segmentation/mask'

import torch

import torch.nn as nn

def double\_conv(in\_c, out\_c):

    conv = nn.Sequential(

        nn.Conv2d(in\_c, out\_c, kernel\_size=3, stride=1, padding=1),

        nn.BatchNorm2d(out\_c),

        nn.ReLU(inplace=True),

        nn.Conv2d(out\_c, out\_c, kernel\_size=3, stride=1, padding=1),

        nn.BatchNorm2d(out\_c),

        nn.ReLU(inplace=True)

    )

    return conv.to(DEVICE)

# def crop\_img(tensor, target\_tensor):

#     target\_size = target\_tensor.size()[2]

#     tensor\_size = tensor.size()[2]

#     delta = tensor\_size-target\_size

#     delta = delta//2

#     # all batch, all channels, heightModified,widthModified

#     return tensor[:, :, delta:tensor\_size-delta, delta:tensor\_size-delta]

def addPadding(srcShapeTensor, tensor\_whose\_shape\_isTobechanged):

    if(srcShapeTensor.shape != tensor\_whose\_shape\_isTobechanged.shape):

        target = torch.zeros(srcShapeTensor.shape)

        target[:, :, :tensor\_whose\_shape\_isTobechanged.shape[2],

               :tensor\_whose\_shape\_isTobechanged.shape[3]] = tensor\_whose\_shape\_isTobechanged

        return target.to(DEVICE)

    return tensor\_whose\_shape\_isTobechanged.to(DEVICE)

class UNet(nn.Module):

    def \_\_init\_\_(self):

        super(UNet, self).\_\_init\_\_()

        self.max\_pool\_2x2 = nn.MaxPool2d(kernel\_size=2, stride=2)

        self.down\_conv\_1 = double\_conv(3, 64)

        self.down\_conv\_2 = double\_conv(64, 128)

        self.down\_conv\_3 = double\_conv(128, 256)

        self.down\_conv\_4 = double\_conv(256, 512)

        self.down\_conv\_5 = double\_conv(512, 1024)

        self.up\_trans\_1 = nn.ConvTranspose2d(

            in\_channels=1024,

            out\_channels=512,

            kernel\_size=2,

            stride=2

        )

        self.up\_conv\_1 = double\_conv(1024, 512)

        self.up\_trans\_2 = nn.ConvTranspose2d(

            in\_channels=512,

            out\_channels=256,

            kernel\_size=2,

            stride=2

        )

        self.up\_conv\_2 = double\_conv(512, 256)

        self.up\_trans\_3 = nn.ConvTranspose2d(

            in\_channels=256,

            out\_channels=128,

            kernel\_size=2,

            stride=2

        )

        self.up\_conv\_3 = double\_conv(256, 128)

        self.up\_trans\_4 = nn.ConvTranspose2d(

            in\_channels=128,

            out\_channels=64,

            kernel\_size=2,

            stride=2

        )

        self.up\_conv\_4 = double\_conv(128, 64)

        self.out = nn.Conv2d(

            in\_channels=64,

            out\_channels=1,

            kernel\_size=1

        )

    def forward(self, image):

        # expected size

        # encoder (Normal convolutions decrease the size)

        x1 = self.down\_conv\_1(image)

        # print("x1 "+str(x1.shape))

        x2 = self.max\_pool\_2x2(x1)

        # print("x2 "+str(x2.shape))

        x3 = self.down\_conv\_2(x2)

        # print("x3 "+str(x3.shape))

        x4 = self.max\_pool\_2x2(x3)

        # print("x4 "+str(x4.shape))

        x5 = self.down\_conv\_3(x4)

        # print("x5 "+str(x5.shape))

        x6 = self.max\_pool\_2x2(x5)

        # print("x6 "+str(x6.shape))

        x7 = self.down\_conv\_4(x6)

        # print("x7 "+str(x7.shape))

        x8 = self.max\_pool\_2x2(x7)

        # print("x8 "+str(x8.shape))

        x9 = self.down\_conv\_5(x8)

        # print("x9 "+str(x9.shape))

        # decoder (transposed convolutions increase the size)

        x = self.up\_trans\_1(x9)

        x = addPadding(x7, x)

        x = self.up\_conv\_1(torch.cat([x7, x], 1))

        x = self.up\_trans\_2(x)

        x = addPadding(x5, x)

        x = self.up\_conv\_2(torch.cat([x5, x], 1))

        x = self.up\_trans\_3(x)

        x = addPadding(x3, x)

        x = self.up\_conv\_3(torch.cat([x3, x], 1))

        x = self.up\_trans\_4(x)

        x = addPadding(x1, x)

        x = self.up\_conv\_4(torch.cat([x1, x], 1))

        x = self.out(x)

        # print(x.shape)

        return x.to(DEVICE)

# if \_\_name\_\_ == "\_\_main\_\_":

#     image = torch.rand((3, 3, 572, 572))

#     model = UNet()

#     print(image.shape)

#     model(image)

class UltrasoundDataset(Dataset):

    def \_\_init\_\_(self,images,image\_dir,mask\_dir,transform=None,train=True):

        self.image\_dir = image\_dir

        self.mask\_dir = mask\_dir

        self.transform = transform

        self.isTrain = train

        self.images = images

    def \_\_len\_\_(self):

        return len(self.images)

    def \_\_getitem\_\_(self,index):

        img\_path = os.path.join(self.image\_dir,self.images[index])

        mask\_path = os.path.join(self.mask\_dir,self.images[index].replace(".tif","\_mask.tif"))

        image = np.array(Image.open(img\_path).convert("RGB"))

        mask = np.array(Image.open(mask\_path).convert("L"),dtype=np.float32)

        mask[mask == 255.0] = 1.0

        if self.transform is not None:

            augmentations = self.transform(image=image,mask=mask)

            image = augmentations['image']

            mask = augmentations['mask']

        return {"image":image,"mask":mask}

images = os.listdir(TRAIN\_IMG\_DIR)

masks = os.listdir(TRAIN\_MASK\_DIR)

images

lst\_mask = [file\_name.split("\_mask")[0] for file\_name in masks]

lst = [file\_name.split(".")[0] for file\_name in images]

common\_elements = list(set(lst\_mask) & set(lst))

cm\_images = [str(item) + '.tif' for item in common\_elements]

images=cm\_images

pip install --user albumentations

def fit(model,dataloader,data,optimizer,criterion):

    print('-------------Training---------------')

    model.train()

    train\_running\_loss = 0.0

    counter=0

    # num of batches

    num\_batches = int(len(data)/dataloader.batch\_size)

    for i,data in tqdm(enumerate(dataloader),total=num\_batches):

        counter+=1

        image,mask = data["image"].to(DEVICE),data["mask"].to(DEVICE)

        optimizer.zero\_grad()

        outputs = model(image)

        outputs = outputs.squeeze(1)

        loss = criterion(outputs,mask)

        train\_running\_loss += loss.item()

        loss.backward()

        optimizer.step()

    train\_loss = train\_running\_loss/counter

    return train\_loss

# def validate(model,dataloader,data,criterion):

#     print("\n--------Validating---------\n")

#     model.eval()

#     ddice = JaccardIndex(num\_classes=2,task='binary')

#     valid\_running\_loss = 0.0

#     dice\_overall = 0.0

#     counter = 0

#     # number of batches

#     num\_batches = int(len(data)/dataloader.batch\_size)

#     with torch.no\_grad():

#         for i,data in tqdm(enumerate(dataloader),total=num\_batches):

#             counter+=1

#             image,mask = data["image"].to(DEVICE),data["mask"].to(DEVICE)

#             outputs = model(image)

#             outputs =outputs.squeeze(1)

#             loss = criterion(outputs,mask)

#             valid\_running\_loss += loss.item()

#             #calculate dice coef for each image

#             dice\_batch = 0.0

#             for k in range(mask.shape[0]):

#                 target = mask[k].int().cpu()

#                 pred = outputs[k].cpu()

#                 dice\_batch += ddice(pred, target)

#             dice\_batch = dice\_batch / dataloader.batch\_size

#             dice\_overall += dice\_batch

#     valid\_loss = valid\_running\_loss/counter

#     dice\_overall = dice\_overall/counter

#     return valid\_loss, dice\_overall

import torch

from tqdm import tqdm

def validate(model, dataloader, data, criterion):

    print("\n--------Validating---------\n")

    model.eval()

    ddice = JaccardIndex(num\_classes=2, task='binary')

    valid\_running\_loss = 0.0

    dice\_overall = 0.0

    counter = 0

    # number of batches

    num\_batches = int(len(data) / dataloader.batch\_size)

    with torch.no\_grad():

        epsilon = 1e-7  # Small epsilon value to avoid division by zero

        for i, data in tqdm(enumerate(dataloader), total=num\_batches):

            counter += 1

            image, mask = data["image"].to(DEVICE), data["mask"].to(DEVICE)

            outputs = model(image)

            outputs = outputs.squeeze(1)

            loss = criterion(outputs, mask)

            valid\_running\_loss += loss.item()

            # calculate dice coef for each image

            dice\_batch = 0.0

            for k in range(mask.shape[0]):

                target = mask[k].int().cpu()

                pred = outputs[k].cpu()

                intersection = (pred \* target).sum()

                union = pred.sum() + target.sum()

                dice\_batch += (2.0 \* intersection + epsilon) / (union + epsilon)

            dice\_batch = dice\_batch / dataloader.batch\_size

            dice\_overall += dice\_batch

    valid\_loss = valid\_running\_loss / counter

    dice\_overall = dice\_overall / counter

    return valid\_loss, dice\_overall

import albumentations as A

from albumentations.pytorch import ToTensorV2

train\_transform = A.Compose([

    A.Resize(IMAGE\_HEIGHT,IMAGE\_WIDTH),

    A.Rotate(limit=35,p=1.0),

    A.HorizontalFlip(p=0.5),

    A.VerticalFlip(p=0.1),

    A.Normalize(

        mean=[0.0,0.0,0.0],

        std = [1.0,1.0,1.0],

        max\_pixel\_value=255.0

    ),

    ToTensorV2()

])

validation\_transform = A.Compose([

    A.Resize(IMAGE\_HEIGHT,IMAGE\_WIDTH),

    A.Normalize(

        mean = [0.0,0.0,0.0],

        std = [1.0,1.0,1.0],

        max\_pixel\_value=255.0,

    ),

    ToTensorV2()

])

def train\_test\_split(images,splitSize):

    imageLen = len(images)

    val\_len = int(splitSize\*imageLen)

    train\_len = imageLen - val\_len

    train\_images,val\_images = images[:train\_len],images[train\_len:]

    return train\_images,val\_images

train\_images\_path,val\_images\_path = train\_test\_split(images,SPLIT)

train\_data = UltrasoundDataset(train\_images\_path,TRAIN\_IMG\_DIR,TRAIN\_MASK\_DIR,train\_transform,True)

valid\_data = UltrasoundDataset(val\_images\_path,TRAIN\_IMG\_DIR,TRAIN\_MASK\_DIR,validation\_transform,True)

train\_dataloader = DataLoader(train\_data,batch\_size=BATCH\_SIZE,shuffle=True)

valid\_dataloader = DataLoader(valid\_data,batch\_size=BATCH\_SIZE,shuffle=True)

model = UNet().to(DEVICE)

optimizer = optim.Adam(model.parameters(),lr=LEARNING\_RATE)

criterion = nn.BCEWithLogitsLoss()

# Val Loss: 0.6500

# Dice Coefficient: 0.0000

val\_epoch\_loss, dice\_coef\_ = validate(model, valid\_dataloader, valid\_data, criterion)

dice\_coef\_

train\_loss = []

val\_loss = []

dice = []

val\_loss\_min = 0.022717

for epoch in range(EPOCHS):

    print(f"Epoch {epoch+1} of {EPOCHS}")

    train\_epoch\_loss = fit(model, train\_dataloader, train\_data, optimizer, criterion)

    val\_epoch\_loss, dice\_coef\_ = validate(model, valid\_dataloader, valid\_data, criterion)

    train\_loss.append(train\_epoch\_loss)

    val\_loss.append(val\_epoch\_loss)

    dice.append(dice\_coef\_)

    if val\_epoch\_loss <= val\_loss\_min:

            print('val\_epoch\_loss metrics improved ({:.6f} --> {:.6f}).  Saving model ...'.format(val\_loss\_min, val\_epoch\_loss))

            torch.save(model.state\_dict(), 'UNET\_ultra.pt')

            val\_loss\_min = val\_epoch\_loss

    print(f"Train Loss: {train\_epoch\_loss:.4f}")

    print(f'Val Loss: {val\_epoch\_loss:.4f}')

    print(f'Dice Coefficient: {dice\_coef\_:.4f}')

# loss plots

plt.figure(figsize=(10, 7))

plt.plot(train\_loss, color="orange", label='train loss')

plt.plot(val\_loss, color="red", label='validation loss')

plt.plot(dice, color="blue", label='dice coefficient')

plt.xlabel("Epochs")

plt.ylabel("Loss, Metrics")

plt.legend()

# plt.savefig(f"../input/loss.png")

plt.show()

torch.save({

    'epoch': EPOCHS,

    'model\_state\_dict': model.state\_dict(),

    'optimizer\_state\_dict': optimizer.state\_dict(),

    'loss': criterion,

}, "./model.pth")

print("\n---------DONE TRAINING----------\n")

train\_dataloader

train\_data

model.load\_state\_dict(torch.load('/content/drive/MyDrive/leela1999/UNET\_ultra.pt/UNET\_ultra.pt',map\_location=torch.device('cpu')))

val\_epoch\_loss, dice\_coef\_ = validate(model, valid\_dataloader, valid\_data, criterion)

print(f'Val Loss: {val\_epoch\_loss:.4f}')

print(f'Dice Coefficient: {dice\_coef\_:.4f}')

# bincount = lambda inds, arr: torch.scatter\_reduce(arr, 0, inds, reduce="sum")

# bincount = lambda inds, arr: torch.scatter\_reduce(arr, 0, inds, reduce="sum")

# bincount

data = train\_data.\_\_getitem\_\_(125)

plt.imshow(data['mask'],cmap="gray")

plt.show()

# for Testing on Single datapoint after training

# plt.imshow(np.transpose(np.array(data['image']),(1,2,0)),cmap="gray")

img = data['image'].unsqueeze(0).to(device="cuda")

plt.imshow(data['image'][0],cmap="gray")

plt.show()

# model = UNet()

output = model(img)

output = torch.squeeze(output)

output[output>0.0] = 1.0

output[output<=0.0]=0

print(torch.max(output))

disp = output.detach().cpu()

plt.imshow(disp,cmap="gray")

**OUTPUT:**

**Installation of torchmetrics:**

Collecting torchmetrics:

Downloading torchmetrics-1.2.0-py3-none-any.whl (805 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 805.2/805.2 kB 11.3 MB/s eta 0:00:00

Requirement already satisfied: numpy>1.20.0 in /usr/local/lib/python3.10/dist-packages (from torchmetrics) (1.23.5)

Requirement already satisfied: torch>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from torchmetrics) (2.1.0+cu118)

Collecting lightning-utilities>=0.8.0 (from torchmetrics)

Downloading lightning\_utilities-0.10.0-py3-none-any.whl (24 kB)

Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.8.0->torchmetrics) (23.2)

Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.8.0->torchmetrics) (67.7.2)

Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.8.0->torchmetrics) (4.5.0)

Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.1->torchmetrics) (3.13.1)

Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.1->torchmetrics) (1.12)

Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.1->torchmetrics) (3.2.1)

Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.1->torchmetrics) (3.1.2)

Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.1->torchmetrics) (2023.6.0)

Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.1->torchmetrics) (2.1.0)

Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.1->torchmetrics) (2.1.3)

Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.1->torchmetrics) (1.3.0)

Installing collected packages: lightning-utilities, torchmetrics

Successfully installed lightning-utilities-0.10.0 torchmetrics-1.2.0

Mounted at /content/drive

['45\_49.tif', '45\_72.tif', '45\_74.tif', '45\_65.tif', '45\_80.tif', '45\_64.tif', '45\_79.tif', '45\_67.tif', '45\_73.tif', '45\_68.tif', '45\_71.tif', '45\_70.tif', '45\_63.tif', '45\_69.tif', '45\_75.tif', '45\_76.tif', '45\_8.tif', '45\_7.tif', '45\_66.tif', '45\_77.tif', '45\_78.tif', '45\_86.tif', '45\_92.tif', '45\_90.tif', '45\_93.tif', '45\_83.tif', '45\_88.tif', '45\_91.tif', '45\_81.tif', '45\_85.tif', '45\_87.tif', '45\_82.tif', '45\_89.tif', '45\_94.tif', '45\_9.tif', '45\_84.tif', '46\_10.tif', '46\_1.tif', '45\_99.tif', '46\_103.tif', '45\_95.tif', '46\_101.tif', '46\_100.tif', '45\_98.tif', '45\_97.tif', '46\_102.tif', '45\_96.tif', '46\_104.tif', '46\_11.tif', '46\_113.tif', '46\_105.tif', '46\_109.tif', '46\_108.tif', '46\_112.tif', '46\_114.tif', '46\_106.tif', '46\_115.tif', '46\_107.tif', '46\_111.tif', '46\_110.tif', '46\_14.tif', '46\_119.tif', '46\_16.tif', '46\_116.tif', '46\_18.tif', '46\_19.tif', '46\_120.tif', '46\_17.tif', '46\_2.tif', '46\_15.tif', '46\_13.tif', '46\_117.tif', '46\_12.tif', '46\_118.tif', '46\_25.tif', '46\_30.tif', '46\_24.tif', '46\_31.tif', '46\_23.tif', '46\_28.tif', '46\_22.tif', '46\_29.tif', '46\_26.tif', '46\_3.tif', '46\_27.tif', '46\_20.tif', '46\_21.tif', '46\_41.tif', '46\_37.tif', '46\_44.tif', '46\_40.tif', '46\_32.tif', '46\_34.tif', '46\_4.tif', '46\_38.tif', '46\_43.tif', '46\_35.tif', '46\_39.tif', '46\_42.tif', '46\_33.tif', '46\_36.tif', '46\_47.tif', '46\_53.tif', '46\_52.tif', '46\_51.tif', '46\_46.tif', '46\_5.tif', '46\_54.tif', '46\_57.tif', '46\_49.tif', '46\_56.tif', '46\_45.tif', '46\_48.tif', '46\_55.tif', '46\_50.tif', '46\_59.tif', '46\_6.tif', '46\_62.tif', '46\_64.tif', '46\_63.tif', '46\_69.tif', '46\_66.tif', '46\_68.tif', '46\_61.tif', '46\_65.tif', '46\_67.tif', '46\_58.tif', '46\_60.tif', '46\_8.tif', '46\_73.tif', '46\_77.tif', '46\_71.tif', '46\_74.tif', '46\_79.tif', '46\_70.tif', '46\_7.tif', '46\_72.tif', '46\_75.tif', '46\_80.tif', '46\_78.tif', '46\_76.tif', '46\_86.tif', '46\_9.tif', '46\_92.tif', '46\_88.tif', '46\_82.tif', '46\_89.tif', '46\_83.tif', '46\_87.tif', '46\_81.tif', '46\_84.tif', '46\_91.tif', '46\_90.tif', '46\_85.tif', '46\_96.tif', '47\_100.tif', '46\_93.tif', '47\_104.tif', '46\_98.tif', '47\_103.tif', '46\_99.tif', '47\_102.tif', '46\_94.tif', '46\_97.tif', '47\_1.tif', '47\_10.tif', '47\_101.tif', '46\_95.tif', '47\_108.tif', '47\_105.tif', '47\_110.tif', '47\_115.tif', '47\_11.tif', '47\_107.tif', '47\_106.tif', '47\_113.tif', '47\_111.tif', '47\_112.tif', '47\_116.tif', '47\_109.tif', '47\_114.tif', '47\_18.tif', '47\_120.tif', '47\_17.tif', '47\_14.tif', '47\_19.tif', '47\_12.tif', '47\_117.tif', '47\_15.tif', '47\_119.tif', '47\_16.tif', '47\_118.tif', '47\_13.tif', '47\_20.tif', '47\_27.tif', '47\_21.tif', '47\_28.tif', '47\_24.tif', '47\_32.tif', '47\_23.tif', '47\_25.tif', '47\_2.tif', '47\_31.tif', '47\_3.tif', '47\_26.tif', '47\_35.tif', '47\_33.tif', '47\_22.tif', '47\_34.tif', '47\_30.tif', '47\_29.tif', '47\_51.tif', '47\_37.tif', '47\_52.tif', '47\_42.tif', '47\_43.tif', '47\_48.tif', '47\_4.tif', '47\_36.tif', '47\_46.tif', '47\_44.tif', '47\_5.tif', '47\_40.tif', '47\_38.tif', '47\_49.tif', '47\_50.tif', '47\_47.tif', '47\_39.tif', '47\_45.tif', '47\_41.tif', '47\_7.tif', '47\_55.tif', '47\_6.tif', '47\_68.tif', '47\_54.tif', '47\_59.tif', '47\_70.tif', '47\_63.tif', '47\_60.tif', '47\_57.tif', '47\_62.tif', '47\_61.tif', '47\_64.tif', '47\_66.tif', '47\_53.tif', '47\_58.tif', '47\_65.tif', '47\_69.tif', '47\_67.tif', '47\_56.tif', '47\_71.tif', '47\_72.tif', '47\_77.tif', '47\_82.tif', '47\_74.tif', '47\_81.tif', '47\_80.tif', '47\_79.tif', '47\_76.tif', '47\_84.tif', '47\_75.tif', '47\_83.tif', '47\_8.tif', '47\_73.tif', '47\_78.tif', '47\_93.tif', '47\_97.tif', '47\_9.tif', '47\_92.tif', '47\_85.tif', '47\_94.tif', '47\_96.tif', '47\_91.tif', '47\_88.tif', '47\_87.tif', '47\_95.tif', '47\_86.tif', '47\_89.tif', '47\_90.tif', '4\_108.tif', '4\_10.tif', '47\_98.tif', '4\_106.tif', '4\_104.tif', '47\_99.tif', '4\_100.tif', '4\_105.tif', '4\_107.tif', '4\_102.tif', '4\_101.tif', '4\_1.tif', '4\_103.tif', '4\_112.tif', '4\_11.tif', '4\_110.tif', '4\_109.tif', '4\_116.tif', '4\_111.tif', '4\_118.tif', '4\_114.tif', '4\_115.tif', '4\_119.tif', '4\_113.tif', '4\_12.tif', '4\_117.tif', '4\_13.tif', '4\_120.tif', '4\_24.tif', '4\_22.tif', '4\_25.tif', '4\_16.tif', '4\_2.tif', '4\_14.tif', '4\_20.tif', '4\_23.tif', '4\_15.tif', '4\_18.tif', '4\_19.tif', '4\_17.tif', '4\_21.tif', '4\_26.tif', '4\_34.tif', '4\_40.tif', '4\_28.tif', '4\_41.tif', '4\_33.tif', '4\_30.tif', '4\_36.tif', '4\_27.tif', '4\_37.tif', '4\_29.tif', '4\_39.tif', '4\_31.tif', '4\_32.tif', '4\_38.tif', '4\_4.tif', '4\_3.tif', '4\_35.tif', '4\_52.tif', '4\_47.tif', '4\_50.tif', '4\_48.tif', '4\_55.tif', '4\_42.tif', '4\_51.tif', '4\_5.tif', '4\_45.tif', '4\_49.tif', '4\_43.tif', '4\_44.tif', '4\_53.tif', '4\_54.tif', '4\_46.tif', '4\_68.tif', '4\_57.tif', '4\_6.tif', '4\_67.tif', '4\_64.tif', '4\_66.tif', '4\_61.tif', '4\_58.tif', '4\_69.tif', '4\_60.tif', '4\_56.tif', '4\_63.tif', '4\_62.tif', '4\_65.tif', '4\_59.tif', '4\_8.tif', '4\_80.tif', '4\_7.tif', '4\_75.tif', '4\_76.tif', '4\_71.tif', '4\_79.tif', '4\_70.tif', '4\_72.tif', '4\_82.tif', '4\_77.tif', '4\_81.tif', '4\_74.tif', '4\_73.tif', '4\_78.tif', '4\_94.tif', '4\_9.tif', '4\_86.tif', '4\_85.tif', '4\_84.tif', '4\_92.tif', '4\_90.tif', '4\_88.tif', '4\_87.tif', '4\_95.tif', '4\_89.tif', '4\_83.tif', '4\_91.tif', '4\_93.tif', '5\_10.tif', '4\_97.tif', '4\_96.tif', '5\_103.tif', '4\_98.tif', '5\_1.tif', '5\_105.tif', '5\_106.tif', '5\_101.tif', '5\_100.tif', '4\_99.tif', '5\_102.tif', '5\_104.tif', '5\_113.tif', '5\_110.tif', '5\_108.tif', '5\_115.tif', '5\_117.tif', '5\_112.tif', '5\_109.tif', '5\_116.tif', '5\_111.tif', '5\_114.tif', '5\_107.tif', '5\_118.tif', '5\_11.tif', '5\_119.tif', '5\_120.tif', '5\_13.tif', '5\_2.tif', '5\_20.tif', '5\_14.tif', '5\_19.tif', '5\_15.tif', '5\_12.tif', '5\_18.tif', '5\_16.tif', '5\_17.tif', '5\_33.tif', '5\_21.tif', '5\_28.tif', '5\_35.tif', '5\_23.tif', '5\_31.tif', '5\_30.tif', '5\_22.tif', '5\_25.tif', '5\_27.tif', '5\_3.tif', '5\_36.tif', '5\_34.tif', '5\_29.tif', '5\_26.tif', '5\_24.tif', '5\_32.tif', '5\_37.tif', '5\_41.tif', '5\_44.tif', '5\_50.tif', '5\_39.tif', '5\_45.tif', '5\_47.tif', '5\_42.tif', '5\_40.tif', '5\_5.tif', '5\_4.tif', '5\_49.tif', '5\_55.tif', '5\_38.tif', '5\_51.tif', '5\_46.tif', '5\_52.tif', '5\_53.tif', '5\_43.tif', '5\_54.tif', '5\_48.tif', '5\_73.tif', '5\_57.tif', '5\_56.tif', '5\_70.tif', '5\_71.tif', '5\_58.tif', '5\_65.tif', '5\_59.tif', '5\_69.tif', '5\_60.tif', '5\_7.tif', '5\_63.tif', '5\_72.tif', '5\_64.tif', '5\_61.tif', '5\_6.tif', '5\_62.tif', '5\_68.tif', '5\_67.tif', '5\_66.tif', '5\_78.tif', '5\_85.tif', '5\_80.tif', '5\_76.tif', '5\_8.tif', '5\_81.tif', '5\_75.tif', '5\_83.tif', '5\_84.tif', '5\_74.tif', '5\_79.tif', '5\_82.tif', '5\_77.tif', '5\_96.tif', '5\_9.tif', '5\_95.tif', '5\_87.tif', '5\_94.tif', '5\_89.tif', '5\_98.tif', '5\_97.tif', '5\_91.tif', '5\_88.tif', '5\_92.tif', '5\_90.tif', '5\_93.tif', '5\_86.tif', '6\_103.tif', '6\_10.tif', '6\_107.tif', '6\_105.tif', '6\_101.tif', '6\_102.tif', '6\_1.tif', '5\_99.tif', '6\_106.tif', '6\_109.tif', '6\_108.tif', '6\_100.tif', '6\_104.tif', '6\_111.tif', '6\_116.tif', '6\_119.tif', '6\_110.tif', '6\_117.tif', '6\_112.tif', '6\_113.tif', '6\_114.tif', '6\_115.tif', '6\_118.tif', '6\_11.tif', '6\_14.tif', '6\_2.tif', '6\_19.tif', '6\_16.tif', '6\_15.tif', '6\_20.tif', '6\_120.tif', '6\_13.tif', '6\_21.tif', '6\_17.tif', '6\_18.tif', '6\_12.tif', '6\_25.tif', '6\_31.tif', '6\_23.tif', '6\_29.tif', '6\_35.tif', '6\_28.tif', '6\_34.tif', '6\_27.tif', '6\_3.tif', '6\_33.tif', '6\_36.tif', '6\_30.tif', '6\_26.tif', '6\_24.tif', '6\_22.tif', '6\_32.tif', '6\_37.tif', '6\_42.tif', '6\_46.tif', '6\_44.tif', '6\_47.tif', '6\_41.tif', '6\_43.tif', '6\_48.tif', '6\_45.tif', '6\_5.tif', '6\_49.tif', '6\_40.tif', '6\_4.tif', '6\_38.tif', '6\_39.tif', '6\_53.tif', '6\_54.tif', '6\_52.tif', '6\_51.tif', '6\_57.tif', '6\_50.tif', '6\_58.tif', '6\_55.tif', '6\_56.tif', '6\_66.tif', '6\_62.tif', '6\_64.tif', '6\_67.tif', '6\_63.tif', '6\_60.tif', '6\_6.tif', '6\_68.tif', '6\_65.tif', '6\_61.tif', '6\_59.tif', '6\_80.tif', '6\_79.tif', '6\_69.tif', '6\_72.tif', '6\_8.tif', '6\_77.tif', '6\_7.tif', '6\_70.tif', '6\_71.tif', '6\_78.tif', '6\_76.tif', '6\_73.tif', '6\_75.tif', '6\_74.tif', '6\_89.tif', '6\_83.tif', '6\_91.tif', '6\_84.tif', '6\_86.tif', '6\_87.tif', '6\_81.tif', '6\_90.tif', '6\_85.tif', '6\_82.tif', '6\_88.tif', '6\_9.tif', '6\_98.tif', '6\_96.tif', '7\_102.tif', '6\_93.tif', '7\_10.tif', '6\_94.tif', '6\_92.tif', '7\_1.tif', '6\_97.tif', '7\_103.tif', '6\_95.tif', '6\_99.tif', '7\_100.tif', '7\_101.tif', '7\_113.tif', '7\_104.tif', '7\_114.tif', '7\_109.tif', '7\_106.tif', '7\_111.tif', '7\_112.tif', '7\_110.tif', '7\_105.tif', '7\_107.tif', '7\_11.tif', '7\_115.tif', '7\_108.tif', '7\_12.tif', '7\_116.tif', '7\_23.tif', '7\_17.tif', '7\_119.tif', '7\_19.tif', '7\_117.tif', '7\_18.tif', '7\_2.tif', '7\_24.tif', '7\_15.tif', '7\_20.tif', '7\_26.tif', '7\_25.tif', '7\_13.tif', '7\_16.tif', '7\_118.tif', '7\_22.tif', '7\_21.tif', '7\_14.tif', '7\_3.tif', '7\_27.tif', '7\_35.tif', '7\_31.tif', '7\_34.tif', '7\_38.tif', '7\_40.tif', '7\_39.tif', '7\_36.tif', '7\_33.tif', '7\_42.tif', '7\_4.tif', '7\_29.tif', '7\_32.tif', '7\_37.tif', '7\_41.tif', '7\_30.tif', '7\_28.tif', '7\_48.tif', '7\_44.tif', '7\_45.tif', '7\_5.tif', '7\_59.tif', '7\_51.tif', '7\_46.tif', '7\_58.tif', '7\_6.tif', '7\_43.tif', '7\_54.tif', '7\_55.tif', '7\_47.tif', '7\_49.tif', '7\_50.tif', '7\_56.tif', '7\_53.tif', '7\_57.tif', '7\_52.tif', '7\_69.tif', '7\_71.tif', '7\_64.tif', '7\_65.tif', '7\_63.tif', '7\_67.tif', '7\_61.tif', '7\_70.tif', '7\_62.tif', '7\_66.tif', '7\_7.tif', '7\_68.tif', '7\_60.tif', '7\_80.tif', '7\_76.tif', '7\_84.tif', '7\_74.tif', '7\_77.tif', '7\_73.tif', '7\_75.tif', '7\_82.tif', '7\_83.tif', '7\_81.tif', '7\_78.tif', '7\_85.tif', '7\_79.tif', '7\_72.tif', '7\_86.tif', '7\_8.tif', '7\_90.tif', '7\_9.tif', '7\_97.tif', '7\_98.tif', '7\_89.tif', '7\_95.tif', '7\_99.tif', '7\_88.tif', '7\_96.tif', '7\_92.tif', '7\_94.tif', '7\_91.tif', '7\_87.tif', '7\_93.tif', '8\_1.tif', '8\_100.tif', '8\_109.tif', '8\_108.tif', '8\_10.tif', '8\_105.tif', '8\_103.tif', '8\_106.tif', '8\_101.tif', '8\_102.tif', '8\_104.tif', '8\_107.tif', '8\_116.tif', '8\_11.tif', '8\_111.tif', '8\_112.tif', '8\_12.tif', '8\_110.tif', '8\_117.tif', '8\_115.tif', '8\_113.tif', '8\_114.tif', '8\_118.tif', '8\_119.tif', '8\_14.tif', '8\_20.tif', '8\_21.tif', '8\_16.tif', '8\_15.tif', '8\_23.tif', '8\_2.tif', '8\_13.tif', '8\_17.tif', '8\_120.tif', '8\_18.tif', '8\_19.tif', '8\_22.tif', '8\_26.tif', '8\_30.tif', '8\_32.tif', '8\_29.tif', '8\_31.tif', '8\_25.tif', '8\_3.tif', '8\_27.tif', '8\_24.tif', '8\_28.tif', '8\_34.tif', '8\_33.tif', '8\_42.tif', '8\_40.tif', '8\_45.tif', '8\_4.tif', '8\_38.tif', '8\_39.tif', '8\_44.tif', '8\_36.tif', '8\_37.tif', '8\_35.tif', '8\_41.tif', '8\_43.tif', '8\_48.tif', '8\_54.tif', '8\_55.tif', '8\_50.tif', '8\_5.tif', '8\_56.tif', '8\_51.tif', '8\_46.tif', '8\_57.tif', '8\_49.tif', '8\_52.tif', '8\_53.tif', '8\_47.tif', '8\_62.tif', '8\_58.tif', '8\_59.tif', '8\_66.tif', '8\_64.tif', '8\_63.tif', '8\_6.tif', '8\_67.tif', '8\_61.tif', '8\_65.tif', '8\_68.tif', '8\_60.tif', '8\_71.tif', '8\_8.tif', '8\_76.tif', '8\_7.tif', '8\_78.tif', '8\_70.tif', '8\_77.tif', '8\_73.tif', '8\_72.tif', '8\_69.tif', '8\_75.tif', '8\_79.tif', '8\_74.tif', '8\_83.tif', '8\_87.tif', '8\_9.tif', '8\_86.tif', '8\_85.tif', '8\_84.tif', '8\_90.tif', '8\_81.tif', '8\_82.tif', '8\_80.tif', '8\_88.tif', '8\_89.tif', '9\_103.tif', '8\_93.tif', '8\_98.tif', '9\_1.tif', '9\_101.tif', '8\_91.tif', '9\_102.tif', '8\_94.tif', '8\_97.tif', '9\_100.tif', '8\_96.tif', '9\_10.tif', '8\_92.tif', '8\_99.tif', '8\_95.tif', '9\_12.tif', '9\_104.tif', '9\_11.tif', '9\_115.tif', '9\_110.tif', '9\_111.tif', '9\_113.tif', '9\_107.tif', '9\_119.tif', '9\_117.tif', '9\_112.tif', '9\_106.tif', '9\_105.tif', '9\_116.tif', '9\_114.tif', '9\_118.tif', '9\_108.tif', '9\_109.tif', '9\_20.tif', '9\_3.tif', '9\_13.tif', '9\_21.tif', '9\_29.tif', '9\_18.tif', '9\_14.tif', '9\_27.tif', '9\_16.tif', '9\_15.tif', '9\_17.tif', '9\_2.tif', '9\_23.tif', '9\_22.tif', '9\_24.tif', '9\_26.tif', '9\_28.tif', '9\_19.tif', '9\_25.tif', '9\_30.tif', '9\_120.tif', '9\_46.tif', '9\_41.tif', '9\_38.tif', '9\_36.tif', '9\_31.tif', '9\_45.tif', '9\_35.tif', '9\_34.tif', '9\_40.tif', '9\_4.tif', '9\_33.tif', '9\_37.tif', '9\_43.tif', '9\_44.tif', '9\_42.tif', '9\_32.tif', '9\_39.tif', '9\_5.tif', '9\_50.tif', '9\_52.tif', '9\_54.tif', '9\_48.tif', '9\_56.tif', '9\_55.tif', '9\_57.tif', '9\_53.tif', '9\_49.tif', '9\_51.tif', '9\_47.tif', '9\_63.tif', '9\_6.tif', '9\_65.tif', '9\_67.tif', '9\_58.tif', '9\_66.tif', '9\_62.tif', '9\_68.tif', '9\_61.tif', '9\_59.tif', '9\_64.tif', '9\_60.tif', '9\_79.tif', '9\_8.tif', '9\_76.tif', '9\_71.tif', '9\_75.tif', '9\_72.tif', '9\_77.tif', '9\_74.tif', '9\_69.tif', '9\_78.tif', '9\_7.tif', '9\_70.tif', '9\_73.tif', '9\_86.tif', '9\_88.tif', '9\_85.tif', '9\_89.tif', '9\_80.tif', '9\_9.tif', '9\_84.tif', '9\_83.tif', '9\_90.tif', '9\_81.tif', '9\_87.tif', '9\_82.tif', '9\_99.tif', '9\_92.tif', '9\_95.tif', '9\_94.tif', '9\_93.tif', '9\_96.tif', '9\_98.tif', '9\_97.tif', '9\_91.tif', ...]

Requirement already satisfied: albumentations in /usr/local/lib/python3.10/dist-packages (1.3.1)

Requirement already satisfied: numpy>=1.11.1 in /usr/local/lib/python3.10/dist-packages (from albumentations) (1.23.5)

Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from albumentations) (1.11.3)

Requirement already satisfied: scikit-image>=0.16.1 in /usr/local/lib/python3.10/dist-packages (from albumentations) (0.19.3)

Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from albumentations) (6.0.1)

Requirement already satisfied: qudida>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from albumentations) (0.0.4)

Requirement already satisfied: opencv-python-headless>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from albumentations) (4.8.1.78)

Requirement already satisfied: scikit-learn>=0.19.1 in /usr/local/lib/python3.10/dist-packages (from qudida>=0.0.4->albumentations) (1.2.2)

Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from qudida>=0.0.4->albumentations) (4.5.0)

Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.16.1->albumentations) (3.2.1)

Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.16.1->albumentations) (9.4.0)

Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.16.1->albumentations) (2.31.6)

Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.16.1->albumentations) (2023.9.26)

Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.16.1->albumentations) (1.4.1)

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.16.1->albumentations) (23.2)

Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.19.1->qudida>=0.0.4->albumentations) (1.3.2)

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.19.1->qudida>=0.0.4->albumentations) (3.2.0)

--------Validating---------

100%|██████████| 24/24 [06:10<00:00, 15.44s/it]

tensor(0.0392)

Epoch 1 of 10

-------------Training---------------

97it [23:00, 14.24s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.76it/s]

Train Loss: 0.4596

Val Loss: 0.3557

Dice Coefficient: 0.0271

Epoch 2 of 10

-------------Training---------------

97it [01:50, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.77it/s]

Train Loss: 0.3300

Val Loss: 0.3165

Dice Coefficient: 0.0266

Epoch 3 of 10

-------------Training---------------

97it [01:50, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.80it/s]

Train Loss: 0.3061

Val Loss: 0.2974

Dice Coefficient: 0.0269

Epoch 4 of 10

-------------Training---------------

97it [01:50, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.76it/s]

Train Loss: 0.2875

Val Loss: 0.2845

Dice Coefficient: 0.0268

Epoch 5 of 10

-------------Training---------------

97it [01:50, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.76it/s]

Train Loss: 0.2719

Val Loss: 0.2660

Dice Coefficient: 0.0261

Epoch 6 of 10

-------------Training---------------

97it [01:50, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.77it/s]

Train Loss: 0.2584

Val Loss: 0.2475

Dice Coefficient: 0.0231

Epoch 7 of 10

-------------Training---------------

97it [01:50, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.79it/s]

Train Loss: 0.2435

Val Loss: 0.2381

Dice Coefficient: 0.0127

Epoch 8 of 10

-------------Training---------------

97it [01:51, 1.14s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.78it/s]

Train Loss: 0.2314

Val Loss: 0.2350

Dice Coefficient: 0.0067

Epoch 9 of 10

-------------Training---------------

97it [01:51, 1.15s/it]

--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.77it/s]

Train Loss: 0.2183

Val Loss: 0.2116

Dice Coefficient: 0.0118

Epoch 10 of 10

-------------Training---------------

97it [01:51, 1.14s/it]

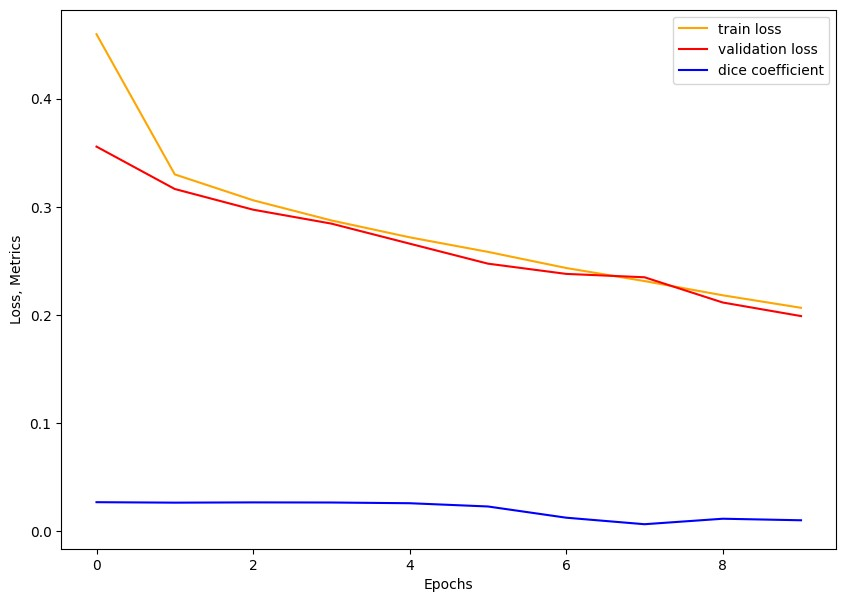
--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.78it/s]

Train Loss: 0.2067

Val Loss: 0.1991

Dice Coefficient: 0.0103



---------DONE TRAINING----------

<torch.utils.data.dataloader.DataLoader at 0x7eeee2206d40>

<\_\_main\_\_.UltrasoundDataset at 0x7eeee22055d0>

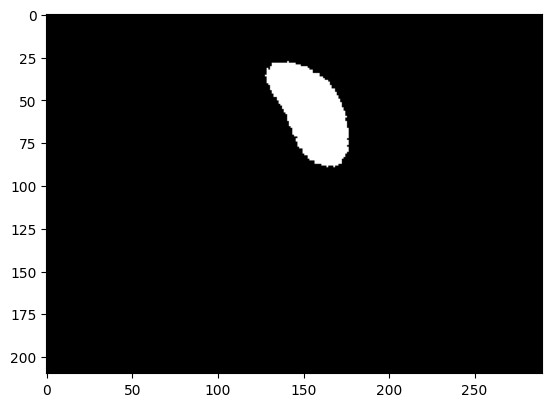
<All keys matched successfully>

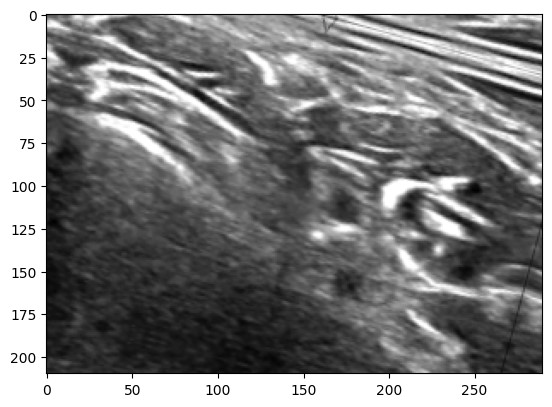
--------Validating---------

100%|██████████| 24/24 [00:13<00:00, 1.81it/s]

Val Loss: 0.0312

Dice Coefficient: 0.0039





tensor(1., device='cuda:0', grad\_fn=<MaxBackward1>)

<matplotlib.image.AxesImage at 0x7eeed9cb4b20>

