Aerial Image Segmentation with PyTorch

December 8, 2024

1 Task 1 : Set up colab gpu runtime environment

[1]:
```
!pip install segmentation-models-pytorch
!pip install -U git+https://github.com/albumentations-team/albumentations
!pip install --upgrade opencv-contrib-python
```

```
Collecting segmentation-models-pytorch
  Downloading segmentation_models_pytorch-0.3.4-py3-none-any.whl.metadata (30
kB)
Collecting efficientnet-pytorch==0.7.1 (from segmentation-models-pytorch)
  Downloading efficientnet_pytorch-0.7.1.tar.gz (21 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: huggingface-hub>=0.24.6 in
/usr/local/lib/python3.10/dist-packages (from segmentation-models-pytorch)
(0.26.3)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages
(from segmentation-models-pytorch) (11.0.0)
Collecting pretrainedmodels==0.7.4 (from segmentation-models-pytorch)
  Downloading pretrainedmodels-0.7.4.tar.gz (58 kB)
                             58.8/58.8 kB
3.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from segmentation-models-pytorch) (1.16.0)
Collecting timm==0.9.7 (from segmentation-models-pytorch)
  Downloading timm-0.9.7-py3-none-any.whl.metadata (58 kB)
                             58.8/58.8 kB
5.1 MB/s eta 0:00:00
Requirement already satisfied: torchvision>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from segmentation-models-pytorch)
(0.20.1+cu121)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from segmentation-models-pytorch) (4.66.6)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(from efficientnet-pytorch==0.7.1->segmentation-models-pytorch) (2.5.1+cu121)
Collecting munch (from pretrainedmodels==0.7.4->segmentation-models-pytorch)
  Downloading munch-4.0.0-py2.py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages
```

```
(from timm==0.9.7->segmentation-models-pytorch) (6.0.2)
Requirement already satisfied: safetensors in /usr/local/lib/python3.10/dist-
packages (from timm==0.9.7->segmentation-models-pytorch) (0.4.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.24.6->segmentation-models-pytorch) (3.16.1)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub>=0.24.6->segmentation-models-pytorch) (2024.10.0)
Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub>=0.24.6->segmentation-models-pytorch) (24.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.24.6->segmentation-models-pytorch) (2.32.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub>=0.24.6->segmentation-models-pytorch) (4.12.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torchvision>=0.5.0->segmentation-models-pytorch) (1.26.4)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch->efficientnet-pytorch==0.7.1->segmentation-models-pytorch)
(3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch->efficientnet-pytorch==0.7.1->segmentation-models-pytorch) (3.1.4)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-
packages (from torch->efficientnet-pytorch==0.7.1->segmentation-models-pytorch)
(1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
sympy==1.13.1->torch->efficientnet-pytorch==0.7.1->segmentation-models-pytorch)
(1.3.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.24.6->segmentation-models-pytorch) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->huggingface-hub>=0.24.6->segmentation-models-pytorch)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.24.6->segmentation-models-pytorch) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.24.6->segmentation-models-pytorch) (2024.8.30)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch->efficientnet-
pytorch==0.7.1->segmentation-models-pytorch) (3.0.2)
Downloading segmentation_models_pytorch-0.3.4-py3-none-any.whl (109 kB)
                         109.5/109.5 kB
9.5 MB/s eta 0:00:00
```

```
Downloading timm-0.9.7-py3-none-any.whl (2.2 MB)
                           2.2/2.2 MB
78.0 MB/s eta 0:00:00
Downloading munch-4.0.0-py2.py3-none-any.whl (9.9 kB)
Building wheels for collected packages: efficientnet-pytorch, pretrainedmodels
  Building wheel for efficientnet-pytorch (setup.py) … done
  Created wheel for efficientnet-pytorch:
filename=efficientnet_pytorch-0.7.1-py3-none-any.whl size=16424
sha256=f69257e21b52a74e39ceee9725633c918fc6d993442b230fd1e867ec21150d59
  Stored in directory: /root/.cache/pip/wheels/03/3f/e9/911b1bc46869644912bda90a
56bcf7b960f20b5187feea3baf
  Building wheel for pretrainedmodels (setup.py) … done
  Created wheel for pretrainedmodels: filename=pretrainedmodels-0.7.4-py3-none-
any.whl size=60944
sha256=f2824bc394859d88e5b310633c766f165fcdad2115e1426a42309b3777746c1c
  Stored in directory: /root/.cache/pip/wheels/35/cb/a5/8f534c60142835bfc889f9a4
82e4a67e0b817032d9c6883b64
Successfully built efficientnet-pytorch pretrainedmodels
Installing collected packages: munch, efficientnet-pytorch, timm,
pretrainedmodels, segmentation-models-pytorch
  Attempting uninstall: timm
    Found existing installation: timm 1.0.12
    Uninstalling timm-1.0.12:
      Successfully uninstalled timm-1.0.12
Successfully installed efficientnet-pytorch-0.7.1 munch-4.0.0
pretrainedmodels-0.7.4 segmentation-models-pytorch-0.3.4 timm-0.9.7
Collecting git+https://github.com/albumentations-team/albumentations
  Cloning https://github.com/albumentations-team/albumentations to /tmp/pip-req-
build-khu4ej_r
  Running command git clone --filter=blob:none --quiet
https://github.com/albumentations-team/albumentations /tmp/pip-req-build-
khu4ej_r
  Resolved https://github.com/albumentations-team/albumentations to commit
47c24503e0636f258e2af2b18e552d52271308bf
  Installing build dependencies … done
  Getting requirements to build wheel … done
  Preparing metadata (pyproject.toml) … done
Requirement already satisfied: numpy>=1.24.4 in /usr/local/lib/python3.10/dist-
packages (from albumentations==1.4.22) (1.26.4)
Requirement already satisfied: scipy>=1.10.0 in /usr/local/lib/python3.10/dist-
packages (from albumentations==1.4.22) (1.13.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages
(from albumentations==1.4.22) (6.0.2)
Requirement already satisfied: pydantic>=2.9.2 in
/usr/local/lib/python3.10/dist-packages (from albumentations==1.4.22) (2.10.3)
Collecting albucore==0.0.21 (from albumentations==1.4.22)
  Downloading albucore-0.0.21-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: eval-type-backport in
```

```
/usr/local/lib/python3.10/dist-packages (from albumentations==1.4.22) (0.2.0)
Requirement already satisfied: opencv-python-headless>=4.9.0.80 in
/usr/local/lib/python3.10/dist-packages (from albumentations==1.4.22)
(4.10.0.84)
Requirement already satisfied: stringzilla>=3.10.4 in
/usr/local/lib/python3.10/dist-packages (from
albucore==0.0.21->albumentations==1.4.22) (3.11.0)
Collecting simsimd>=5.9.2 (from albucore==0.0.21->albumentations==1.4.22)
  Downloading simsimd-6.2.1-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (66
kB)
                           66.0/66.0 kB
4.8 MB/s eta 0:00:00
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from
pydantic>=2.9.2->albumentations==1.4.22) (0.7.0)
Requirement already satisfied: pydantic-core==2.27.1 in
/usr/local/lib/python3.10/dist-packages (from
pydantic>=2.9.2->albumentations==1.4.22) (2.27.1)
Requirement already satisfied: typing-extensions>=4.12.2 in
/usr/local/lib/python3.10/dist-packages (from
pydantic>=2.9.2->albumentations==1.4.22) (4.12.2)
Downloading albucore-0.0.21-py3-none-any.whl (12 kB)
Downloading simsimd-6.2.1-cp310-cp310-manylinux_2_28_x86_64.whl (632 kB)
                          632.7/632.7 kB
29.2 MB/s eta 0:00:00
Building wheels for collected packages: albumentations
  Building wheel for albumentations (pyproject.toml) … done
  Created wheel for albumentations: filename=albumentations-1.4.22-py3-none-
any.whl size=258445
sha256=2f81bb8a2bcbf79317834f0a3c3a922e22931ba15b80cb1e10381c36cfd060cf
  Stored in directory: /tmp/pip-ephem-wheel-cache-1f_5vqjo/wheels/51/4d/ab/5aafa
8b980086fbc362946de7da4aa3df33aacb3da0da29b93
Successfully built albumentations
Installing collected packages: simsimd, albucore, albumentations
  Attempting uninstall: albucore
    Found existing installation: albucore 0.0.19
    Uninstalling albucore-0.0.19:
      Successfully uninstalled albucore-0.0.19
  Attempting uninstall: albumentations
    Found existing installation: albumentations 1.4.20
    Uninstalling albumentations-1.4.20:
      Successfully uninstalled albumentations-1.4.20
Successfully installed albucore-0.0.21 albumentations-1.4.22 simsimd-6.2.1
Requirement already satisfied: opencv-contrib-python in
/usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-
packages (from opencv-contrib-python) (1.26.4)
```

2 About Dataset

2.0.1 Dataset

Here the dataset which we are going to use in this guided project is the subset(200 images and its masks) of the original dataset (Massachusetts Roads Dataset) consists of 1171 aerial images of the state of Massachusetts. Each image is 1500×1500 pixels in size, covering an area of 2.25 square kilometers

2.0.2 Full Dataset

After compeletion of this project you can try the same pipeline on full dataset

https://www.cs.toronto.edu/~vmnih/data/

```
@phdthesis{MnihThesis,
author = {Volodymyr Mnih},
title = {Machine Learning for Aerial Image Labeling},
school = {University of Toronto},
year = {2013}
}
```

3 Download Subset Dataset

[2]:
```
!git clone https://github.com/parth1620/Road_seg_dataset.git
```

```
Cloning into 'Road_seg_dataset'…
remote: Enumerating objects: 411, done.
remote: Total 411 (delta 0), reused 0 (delta 0), pack-reused 411 (from 1)
Receiving objects: 100% (411/411), 851.74 MiB | 16.51 MiB/s, done.
Resolving deltas: 100% (2/2), done.
Updating files: 100% (401/401), done.
```

4 Some Common Imports

[3]:
```
import sys
sys.path.append('/content/Road_seg_dataset')
```

[4]:
```
import torch
import cv2

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

```
from tqdm import tqdm

import helper
```

5 Task : 2 Setup Configurations

[5]:
```
CSV_FILE = '/content/Road_seg_dataset/train.csv'
DATA_DIR = '/content/Road_seg_dataset/'

DEVICE = 'cuda'
EPOCHS = 25
LR = 0.003
BATCH_SIZE = 8
IMG_SIZE = 512

ENCODER = 'timm-efficientnet-b0'
WEIGHTS = 'imagenet'
```

[6]:
```
df = pd.read_csv(CSV_FILE)
df.head()
```

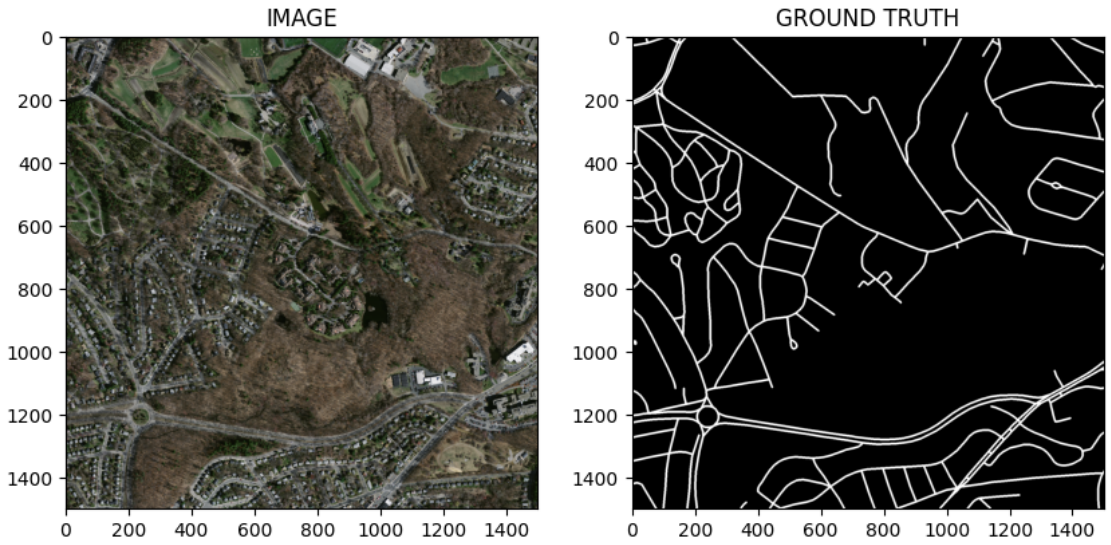
[6]:

	images	masks
0	images/17428750_15.png	masks/17428750_15.png
1	images/23279080_15.png	masks/23279080_15.png
2	images/24179185_15.png	masks/24179185_15.png
3	images/24179035_15.png	masks/24179035_15.png
4	images/11128810_15.png	masks/11128810_15.png

[7]:
```
idx = 17
row = df.iloc[idx]

image_path = DATA_DIR + row.images
mask_path = DATA_DIR + row.masks

image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE) / 255
```

[8]:
```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))

ax1.set_title('IMAGE')
ax1.imshow(image)

ax2.set_title('GROUND TRUTH')
ax2.imshow(mask,cmap = 'gray')
```

[8]: <matplotlib.image.AxesImage at 0x7ec08251f100>



[9]:
```
train_df, valid_df = train_test_split(df, test_size = 0.20, random_state = 42)
```

[10]:
```
print(len(valid_df))
print(len(train_df))
```

```
40
159
```

6 Task 3 : Augmentation Functions

albumentation documentation : https://albumentations.ai/docs/

[11]:
```
import albumentations as A
```

[12]:
```
def get_train_augs():
  return A.Compose([
      A.Resize(IMG_SIZE, IMG_SIZE),
      A.HorizontalFlip(p = 0.5),
      A.VerticalFlip(p = 0.5)
  ])

def get_valid_augs():
  return A.Compose([
      A.Resize(IMG_SIZE, IMG_SIZE)
  ])
```

7 Task 4 : Create Custom Dataset

[13]:
```
from torch.utils.data import Dataset
```

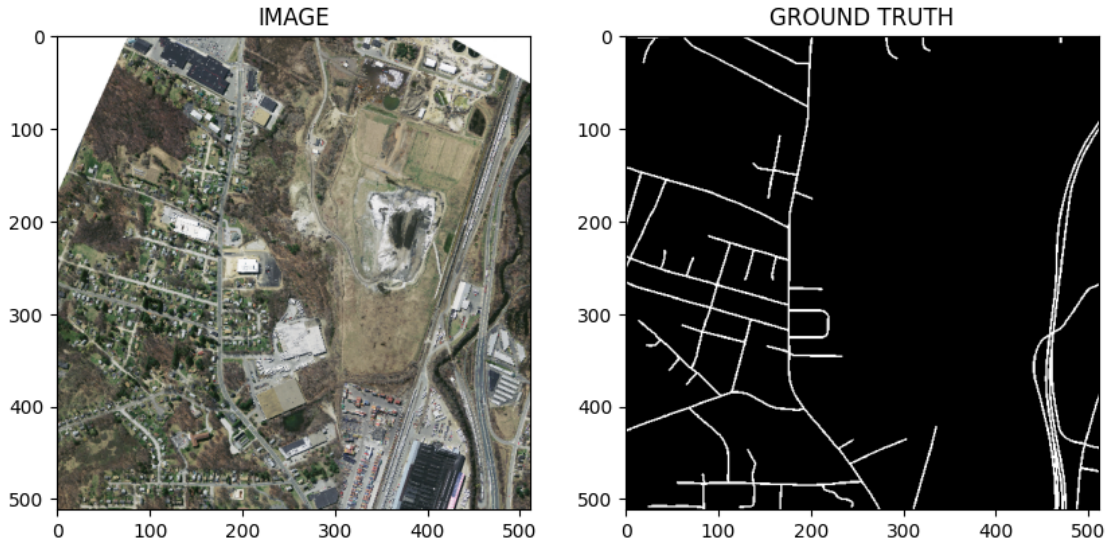
[14]:
```
class SegmentationDataset(Dataset):
  def __init__(self, df, augmentations):
    self.df = df
    self.augmentations = augmentations

  def __len__(self):
    return len(self.df)

  def __getitem__(self, idx):
    row = self.df.iloc[idx]

    image_path = DATA_DIR + row.images
    mask_path = DATA_DIR + row.masks

    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)  O U?- rV
    mask = np.expand_dims(mask, axis=-1)  O U?- r- +V

    if self.augmentations:
        data = self.augmentations(image=image, mask=mask)
        image = data['image']
        mask = data['mask']

    image = np.transpose(image, (2, 0, 1)).astype(np.float32)  O U+- ?- rV
    mask = np.transpose(mask, (2, 0, 1)).astype(np.float32)  O U+- ?- rV

    image = torch.Tensor(image) / 255.0
    mask = torch.round(torch.Tensor(mask) / 255.0)

    return image, mask
```

[15]:
```
trainset = SegmentationDataset(train_df, get_train_augs())
validset = SegmentationDataset(valid_df, get_valid_augs())
```

[16]:
```
print(f'Size of trainset: {len(trainset)}')
print(f'Size of validset: {len(validset)}')
```

```
Size of trainset: 159
Size of validset: 40
```

[17]:
```
idx = 5
image, mask = trainset[idx]

helper.show_image(image, mask)
```



8 Task 5 : Load dataset into batches

[18]:
```
from torch.utils.data import DataLoader
```

[19]:
```
trainloader = DataLoader(trainset, batch_size = BATCH_SIZE, shuffle = True)
validloader = DataLoader(validset, batch_size=BATCH_SIZE)
```

[20]:
```
print(f'Total no. of batches in trainloader: {len(trainloader)} and in␣
 ↪validloader: {len(validloader)}')
```

```
Total no. of batches in trainloader: 20 and in validloader: 5
```

[21]:
```
for images, masks in trainloader:
  print(f'One batch image shape: {images.shape} and mask shape: {masks.shape}')
  break
```

```
One batch image shape: torch.Size([8, 3, 512, 512]) and mask shape:
torch.Size([8, 1, 512, 512])
```

9 Task 6 : Create Segmentation Model

segmentation_models_pytorch documentation : https://smp.readthedocs.io/en/latest/

[22]:
```
import segmentation_models_pytorch as smp
from segmentation_models_pytorch.losses import DiceLoss

from torch import nn
```

[23]:
```
class SegmentationModel(nn.Module):
  def __init__(self):
    super(SegmentationModel, self).__init__()

    self.backbone = smp.Unet(
        encoder_name = ENCODER,
        encoder_weights = WEIGHTS,
        in_channels = 3,
        classes = 1,
        activation = None
    )

  def forward(self, images, masks = None):

    logits = self.backbone(images)

    if masks != None:
      return logits, DiceLoss(mode = 'binary')(logits, masks) + nn.
↪BCEWithLogitsLoss()(logits, masks)

    return logits
```

[24]:
```
model = SegmentationModel()
model.to(DEVICE)
```

```
Downloading: "https://github.com/huggingface/pytorch-image-
models/releases/download/v0.1-weights/tf_efficientnet_b0-0af12548.pth" to
/root/.cache/torch/hub/checkpoints/tf_efficientnet_b0-0af12548.pth
100%|      | 20.4M/20.4M [00:00<00:00, 37.9MB/s]
```

[24]:
```
SegmentationModel(
  (backbone): Unet(
    (encoder): EfficientNetEncoder(
      (conv_stem): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNormAct2d(
        32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
        (drop): Identity()
        (act): Swish()
      )
      (blocks): Sequential(
        (0): Sequential(
```

```
        UyV, .2Ti?rBb2a2T ` #H2*QMpU
          U+QMpn/rV, *QMpk/Ujk- jk- F2`M2HnbBx24Uj- jV- bi`B/24UR-
T //BM;4UR- RV- ;`QmTb4jk- #B b46 Hb2V
          U#MRV, " i+?LQ`K +ik/U
            jk- 2Tb4R2@y8- KQK2MimK4yXR- 77BM24h`m2- i` +Fn`mMI
            U/`QTV, A/2MiBivUV
            U +iV, arBb?UV
          V
          Ub2V, a[m22x21t+Bi2U
            U+QMpn`2/m+2V, *QMpk/Ujk- 3- F2`M2HnbBx24UR- RV- bi`I
            U +iRV, arBb?UV
            U+QMpn2tT M/V, *QMpk/U3- jk- F2`M2HnbBx24UR- RV- bi`B
            U; i2V, aB;KQB/UV
          V
          U+QMpnTrV, *QMpk/Ujk- Re- F2`M2HnbBx24UR- RV- bi`B/24U
#B b46 Hb2V
          U#MkV, " i+?LQ`K +ik/U
            Re- 2Tb4R2@y8- KQK2MimK4yXR- 77BM24h`m2- i` +Fn`mM
            U/`QTV, A/2MiBivUV
            U +iV, A/2MiBivUV
          V
          U/`QTnT i?V, A/2MiBivUV
        V
      V
      URV, a2[m2MiB HU
        UyV, AMp2`i2/_2bB/m HU
          U+QMpnTrV, *QMpk/URe- Ne- F2`M2HnbBx24UR- RV- bi`B/24
#B b46 Hb2V
          U#MRV, " i+?LQ`K +ik/U
            Ne- 2Tb4R2@y8- KQK2MimK4yXR- 77BM24h`m2- i` +Fn`mM
            U/`QTV, A/2MiBivUV
            U +iV, arBb?UV
          V
          U+QMpn/rV, *QMpk/UNe- Ne- F2`M2HnbBx24Uj- jV- bi`B/24UI
T //BM;4UR- RV- ;`QmTb4Ne- #B b46 Hb2V
          U#MkV, " i+?LQ`K +ik/U
            Ne- 2Tb4R2@y8- KQK2MimK4yXR- 77BM24h`m2- i` +Fn`mM
            U/`QTV, A/2MiBivUV
            U +iV, arBb?UV
          V
          Ub2V, a[m22x21t+Bi2U
            U+QMpn`2/m+2V, *QMpk/UNe- 9- F2`M2HnbBx24UR- RV- bi`
            U +iRV, arBb?UV
            U+QMpn2tT M/V, *QMpk/U9- Ne- F2`M2HnbBx24UR- RV- bi`
            U; i2V, aB;KQB/UV
          V
          U+QMpnTrHV, *QMpk/UNe- k9- F2`M2HnbBx24UR- RV- bi`B/2
```

```
bias=False)
            (bn3): BatchNormAct2d(
              24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.013)
          )
          (1): InvertedResidual(
            (conv_pw): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=144, bias=False)
            (bn2): BatchNormAct2d(
              144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.025)
          )
        )
        (2): Sequential(
          (0): InvertedResidual(
            (conv_pw): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              144, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(144, 144, kernel_size=(5, 5), stride=(2, 2),
padding=(2, 2), groups=144, bias=False)
            (bn2): BatchNormAct2d(
              144, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(144, 40, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              40, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.038)
          )
          (1): InvertedResidual(
            (conv_pw): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(240, 240, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=240, bias=False)
            (bn2): BatchNormAct2d(
              240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
```

```
              (conv_expand): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(240, 40, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              40, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.050)
          )
        )
        (3): Sequential(
          (0): InvertedResidual(
            (conv_pw): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(240, 240, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), groups=240, bias=False)
            (bn2): BatchNormAct2d(
              240, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(240, 80, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              80, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.062)
          )
          (1): InvertedResidual(
```

```
            (conv_pw): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=480, bias=False)
            (bn2): BatchNormAct2d(
              480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              80, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.075)
          )
          (2): InvertedResidual(
            (conv_pw): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=480, bias=False)
            (bn2): BatchNormAct2d(
              480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
```

```
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              80, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.088)
          )
        )
        (4): Sequential(
          (0): InvertedResidual(
            (conv_pw): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=480, bias=False)
            (bn2): BatchNormAct2d(
              480, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(480, 112, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
```

```
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.100)
          )
          (1): InvertedResidual(
            (conv_pw): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=672, bias=False)
            (bn2): BatchNormAct2d(
              672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.113)
          )
          (2): InvertedResidual(
            (conv_pw): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1),
```

```
padding=(2, 2), groups=672, bias=False)
            (bn2): BatchNormAct2d(
              672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              112, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.125)
          )
        )
        (5): Sequential(
          (0): InvertedResidual(
            (conv_pw): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(672, 672, kernel_size=(5, 5), stride=(2, 2),
padding=(2, 2), groups=672, bias=False)
            (bn2): BatchNormAct2d(
              672, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
```

```
            )
            (conv_pwl): Conv2d(672, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.138)
          )
          (1): InvertedResidual(
            (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (bn2): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.150)
          )
          (2): InvertedResidual(
            (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
```

```
            (bn1): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (bn2): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.163)
          )
          (3): InvertedResidual(
            (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1),
padding=(2, 2), groups=1152, bias=False)
            (bn2): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
```

```
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.175)
          )
        )
        (6): Sequential(
          (0): InvertedResidual(
            (conv_pw): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn1): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (conv_dw): Conv2d(1152, 1152, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), groups=1152, bias=False)
            (bn2): BatchNormAct2d(
              1152, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
              (act): Swish()
            )
            (se): SqueezeExcite(
              (conv_reduce): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
              (act1): Swish()
              (conv_expand): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
              (gate): Sigmoid()
            )
            (conv_pwl): Conv2d(1152, 320, kernel_size=(1, 1), stride=(1, 1),
bias=False)
            (bn3): BatchNormAct2d(
              320, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True
              (drop): Identity()
```

```
              (act): Identity()
            )
            (drop_path): DropPath(drop_prob=0.188)
          )
        )
      )
      (conv_head): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1),
bias=False)
      (bn2): BatchNormAct2d(
        1280, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
        (drop): Identity()
        (act): Swish()
      )
      (global_pool): SelectAdaptivePool2d (pool_type=avg,
flatten=Flatten(start_dim=1, end_dim=-1))
    )
    (decoder): UnetDecoder(
      (center): Identity()
      (blocks): ModuleList(
        (0): DecoderBlock(
          (conv1): Conv2dReLU(
            (0): Conv2d(432, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention1): Attention(
            (attention): Identity()
          )
          (conv2): Conv2dReLU(
            (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention2): Attention(
            (attention): Identity()
          )
        )
        (1): DecoderBlock(
          (conv1): Conv2dReLU(
            (0): Conv2d(296, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
            (2): ReLU(inplace=True)
          )
          (attention1): Attention(
            (attention): Identity()
          )
          (conv2): Conv2dReLU(
            (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention2): Attention(
            (attention): Identity()
          )
        )
        (2): DecoderBlock(
          (conv1): Conv2dReLU(
            (0): Conv2d(152, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention1): Attention(
            (attention): Identity()
          )
          (conv2): Conv2dReLU(
            (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention2): Attention(
            (attention): Identity()
          )
        )
        (3): DecoderBlock(
          (conv1): Conv2dReLU(
            (0): Conv2d(96, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention1): Attention(
```

```
            (attention): Identity()
          )
          (conv2): Conv2dReLU(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention2): Attention(
            (attention): Identity()
          )
        )
        (4): DecoderBlock(
          (conv1): Conv2dReLU(
            (0): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention1): Attention(
            (attention): Identity()
          )
          (conv2): Conv2dReLU(
            (0): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
            (2): ReLU(inplace=True)
          )
          (attention2): Attention(
            (attention): Identity()
          )
        )
      )
    )
    (segmentation_head): SegmentationHead(
      (0): Conv2d(16, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): Identity()
      (2): Activation(
        (activation): Identity()
      )
    )
  )
)
```

10 Task 7 : Create Train and Validation Function

[25]:
```
def train_fn(dataloader, model, optimizer):
  model.train() O im‘M QM /‘QTQmi- # i+?MQ‘K 2i+XXX
  total_loss = 0.0

  for images, masks in tqdm(dataloader):

    images = images.to(DEVICE)
    masks = masks.to(DEVICE)

    optimizer.zero_grad()
    logits, loss = model(images, masks)

    loss.backward()
    optimizer.step()

    total_loss += loss.item()
  return total_loss / len(dataloader)
```

[26]:
```
def eval_fn(dataloader, model):
  model.eval() O im‘M Q77 /‘QTQmi- # i+?MQ‘K 2i+XXX
  total_loss = 0.0

  with torch.no_grad():
    for images, masks in tqdm(dataloader):

      images = images.to(DEVICE)
      masks = masks.to(DEVICE)
      logits, loss = model(images, masks)
      total_loss += loss.item()

    return total_loss / len(dataloader)
```

11 Task 8 : Train Model

[27]:
```
optimizer = torch.optim.Adam(model.parameters(), lr = LR)
```

[28]:
```
best_loss = np.Inf

for i in range(EPOCHS):
  train_loss = train_fn(trainloader, model, optimizer)
  valid_loss = eval_fn(validloader, model)

  if valid_loss < best_loss:
    torch.save(model.state_dict(), 'best_model.pt')
```

```
    print("SAVED MODEL")
    best_loss = valid_loss

  print(f'Epoch : {i+1} Train Loss: {train_loss}, Valid Loss: {valid_loss}')
```

```
100%|      | 20/20 [00:23<00:00,  1.18s/it]
100%|      | 5/5 [00:03<00:00,  1.27it/s]

SAVED MODEL
Epoch : 1 Train Loss: 1.2603730082511901, Valid Loss: 1.1518649578094482

100%|      | 20/20 [00:22<00:00,  1.10s/it]
100%|      | 5/5 [00:03<00:00,  1.39it/s]

SAVED MODEL
Epoch : 2 Train Loss: 0.8315716087818146, Valid Loss: 0.8425767540931701

100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:03<00:00,  1.31it/s]

Epoch : 3 Train Loss: 0.7299677461385727, Valid Loss: 1.0247456073760985

100%|      | 20/20 [00:21<00:00,  1.06s/it]
100%|      | 5/5 [00:04<00:00,  1.10it/s]

SAVED MODEL
Epoch : 4 Train Loss: 0.692460685968399, Valid Loss: 0.7186045169830322

100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:03<00:00,  1.29it/s]

Epoch : 5 Train Loss: 0.6664293229579925, Valid Loss: 0.7480562210083008

100%|      | 20/20 [00:21<00:00,  1.08s/it]
100%|      | 5/5 [00:04<00:00,  1.23it/s]

SAVED MODEL
Epoch : 6 Train Loss: 0.6540595918893815, Valid Loss: 0.6771730542182922

100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:04<00:00,  1.22it/s]

SAVED MODEL
Epoch : 7 Train Loss: 0.6395488917827606, Valid Loss: 0.6658743858337403

100%|      | 20/20 [00:21<00:00,  1.06s/it]
100%|      | 5/5 [00:04<00:00,  1.18it/s]

Epoch : 8 Train Loss: 0.6321607053279876, Valid Loss: 0.7255025267601013

100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:04<00:00,  1.18it/s]

Epoch : 9 Train Loss: 0.6343806236982346, Valid Loss: 0.7000364661216736
```

```
100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:04<00:00,  1.18it/s]

Epoch : 10 Train Loss: 0.6313428521156311, Valid Loss: 0.698941707611084

100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:04<00:00,  1.14it/s]

Epoch : 11 Train Loss: 0.6345958858728409, Valid Loss: 0.6923960089683533

100%|      | 20/20 [00:21<00:00,  1.06s/it]
100%|      | 5/5 [00:04<00:00,  1.17it/s]

SAVED MODEL
Epoch : 12 Train Loss: 0.6171192198991775, Valid Loss: 0.6564433574676514

100%|      | 20/20 [00:21<00:00,  1.06s/it]
100%|      | 5/5 [00:04<00:00,  1.20it/s]

Epoch : 13 Train Loss: 0.5994628936052322, Valid Loss: 0.6569787979125976

100%|      | 20/20 [00:21<00:00,  1.07s/it]
100%|      | 5/5 [00:03<00:00,  1.25it/s]

Epoch : 14 Train Loss: 0.6034947812557221, Valid Loss: 0.6807732462882996

100%|      | 20/20 [00:21<00:00,  1.08s/it]
100%|      | 5/5 [00:03<00:00,  1.31it/s]

Epoch : 15 Train Loss: 0.6064578801393509, Valid Loss: 0.6575327754020691

100%|      | 20/20 [00:21<00:00,  1.08s/it]
100%|      | 5/5 [00:03<00:00,  1.37it/s]

SAVED MODEL
Epoch : 16 Train Loss: 0.5962778344750405, Valid Loss: 0.650817334651947

100%|      | 20/20 [00:21<00:00,  1.09s/it]
100%|      | 5/5 [00:03<00:00,  1.42it/s]

SAVED MODEL
Epoch : 17 Train Loss: 0.5893536135554314, Valid Loss: 0.6219703316688537

100%|      | 20/20 [00:21<00:00,  1.08s/it]
100%|      | 5/5 [00:03<00:00,  1.41it/s]

Epoch : 18 Train Loss: 0.5864942207932472, Valid Loss: 0.6668243646621704

100%|      | 20/20 [00:21<00:00,  1.09s/it]
100%|      | 5/5 [00:03<00:00,  1.41it/s]

Epoch : 19 Train Loss: 0.5898117691278457, Valid Loss: 0.6289428472518921

100%|      | 20/20 [00:21<00:00,  1.09s/it]
100%|      | 5/5 [00:03<00:00,  1.40it/s]

Epoch : 20 Train Loss: 0.587537744641304, Valid Loss: 0.6303063154220581
```

```
100%|      | 20/20 [00:21<00:00,  1.09s/it]
100%|      | 5/5 [00:03<00:00,  1.40it/s]

Epoch : 21 Train Loss: 0.577789568901062, Valid Loss: 0.6801318407058716

100%|      | 20/20 [00:21<00:00,  1.09s/it]
100%|      | 5/5 [00:03<00:00,  1.39it/s]

Epoch : 22 Train Loss: 0.5871293544769287, Valid Loss: 0.6669469118118286

100%|      | 20/20 [00:21<00:00,  1.09s/it]
100%|      | 5/5 [00:03<00:00,  1.37it/s]

SAVED MODEL
Epoch : 23 Train Loss: 0.5762363687157631, Valid Loss: 0.6158580541610718

100%|      | 20/20 [00:21<00:00,  1.10s/it]
100%|      | 5/5 [00:03<00:00,  1.40it/s]

Epoch : 24 Train Loss: 0.5749181523919106, Valid Loss: 0.6366091251373291

100%|      | 20/20 [00:21<00:00,  1.08s/it]
100%|      | 5/5 [00:03<00:00,  1.41it/s]

Epoch : 25 Train Loss: 0.5661695227026939, Valid Loss: 0.6501298427581788
```

12 Task 9 : Inference

[35]:
```
idx = 20

model.load_state_dict(torch.load('/content/best_model.pt'))

image, mask = validset[idx]

logits_mask = model(image.to(DEVICE).unsqueeze(0)) O U+- ?- rV @= U#- +- ?- rV
pred_mask = torch.sigmoid(logits_mask)
pred_mask = (pred_mask > 0.5)*1.0
```

```
<ipython-input-35-20e60f4aae86>:3: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
```

```
experimental feature.
  model.load_state_dict(torch.load('/content/best_model.pt'))
```

[36]:
```
helper.show_image(image, pred_mask.detach().cpu().squeeze(0))
```