```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="FmCTXKicdAcgH2ZRPNlJ")
project = rf.workspace("jim-ps1mt").project("jim_sem")
version = project.version(2)
dataset = version.download("png-mask-semantic")
```

```
Collecting roboflow
  Downloading roboflow-1.1.61-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from roboflow) (2025
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.11/dist-packages (from roboflow) (0.12.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from robof
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from roboflow) (3
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Collecting opencv-python-headless==4.10.0.84 (from roboflow)
  Downloading opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Collecting pillow-heif>=0.18.0 (from roboflow)
  Downloading pillow_heif-0.22.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-packages (from roboflo
Collecting python-dotenv (from roboflow)
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.3
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.17.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.11/dist-packages (from roboflo
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.11/dist-packages (from robof
Collecting filetype (from roboflow)
  Downloading filetype-1.2.0-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matpl
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplot
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (fro
Downloading roboflow-1.1.61-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.2/85.2 kB 6.7 MB/s eta 0:00:00
Downloading idna-3.7-py3-none-any.whl (66 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.8/66.8 kB 5.0 MB/s eta 0:00:00
Downloading opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.9/49.9 MB 13.9 MB/s eta 0:00:00
Downloading pillow_heif-0.22.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 64.0 MB/s eta 0:00:00
Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Downloading python_dotenv-1.1.0-py3-none-any.whl (20 kB)
Installing collected packages: filetype, python-dotenv, pillow-heif, opencv-python-headless, idna, robo
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.11.0.86
    Uninstalling opencv-python-headless-4.11.0.86:
      Successfully uninstalled opencv-python-headless-4.11.0.86
  Attempting uninstall: idna
    Found existing installation: idna 3.10
    Uninstalling idna-3.10:
      Successfully uninstalled idna-3.10
Successfully installed filetype-1.2.0 idna-3.7 opencv-python-headless-4.10.0.84 pillow-heif-0.22.0 pyth
loading Roboflow workspace...
loading Roboflow project...
```

```
Downloading Dataset Version Zip in Jim_sem-2 to png-mask-semantic:: 100%|██████████| 22769/22769 [00:00

Extracting Dataset Version Zip to Jim_sem-2 in png-mask-semantic:: 100%|██████████| 1048/1048 [00:00<00
```

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="FmCTXKicdAcgH2ZRPNlJ")
project = rf.workspace("jim-ps1mt").project("jim_sem")
version = project.version(3)
dataset = version.download("png-mask-semantic")
```

```
Collecting roboflow
  Downloading roboflow-1.1.62-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from roboflow) (2025
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.11/dist-packages (from roboflow) (0.12.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from robof
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from roboflow) (3
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Collecting opencv-python-headless==4.10.0.84 (from roboflow)
  Downloading opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Collecting pillow-heif>=0.18.0 (from roboflow)
  Downloading pillow_heif-0.22.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-packages (from roboflo
Collecting python-dotenv (from roboflow)
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.3
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.17.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.11/dist-packages (from roboflo
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.11/dist-packages (from roboflow)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.11/dist-packages (from robof
Collecting filetype (from roboflow)
  Downloading filetype-1.2.0-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matpl
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplot
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplo
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (fro
Downloading roboflow-1.1.62-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.3/85.3 kB 3.3 MB/s eta 0:00:00
Downloading idna-3.7-py3-none-any.whl (66 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.8/66.8 kB 5.1 MB/s eta 0:00:00
Downloading opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.9/49.9 MB 19.2 MB/s eta 0:00:00
Downloading pillow_heif-0.22.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 110.4 MB/s eta 0:00:00
Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Downloading python_dotenv-1.1.0-py3-none-any.whl (20 kB)
Installing collected packages: filetype, python-dotenv, pillow-heif, opencv-python-headless, idna, robo
  Attempting uninstall: opencv-python-headless
    Found existing installation: opencv-python-headless 4.11.0.86
    Uninstalling opencv-python-headless-4.11.0.86:
      Successfully uninstalled opencv-python-headless-4.11.0.86
  Attempting uninstall: idna
    Found existing installation: idna 3.10
    Uninstalling idna-3.10:
      Successfully uninstalled idna-3.10
```

```
Successfully installed filetype-1.2.0 idna-3.7 opencv-python-headless-4.10.0.84 pillow-heif-0.22.0 pyth
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Jim_sem-3 to png-mask-semantic:: 100%|██████████| 22638/22638 [00:00

Extracting Dataset Version Zip to Jim_sem-3 in png-mask-semantic:: 100%|██████████| 1048/1048 [00:00<00
```

```
import cv2
```

```
!pip install segmentation-models-pytorch
```

```
Collecting segmentation-models-pytorch
  Downloading segmentation_models_pytorch-0.5.0-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: huggingface-hub>=0.24 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: numpy>=1.19.3 in /usr/local/lib/python3.11/dist-packages (from segmen
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from segmentati
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.11/dist-packages (from s
Requirement already satisfied: timm>=0.9 in /usr/local/lib/python3.11/dist-packages (from segmentati
Requirement already satisfied: torch>=1.8 in /usr/local/lib/python3.11/dist-packages (from segmentat
Requirement already satisfied: torchvision>=0.9 in /usr/local/lib/python3.11/dist-packages (from seg
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from segment
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from hug
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-packages (from hugg
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingf
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.8-
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8->s
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from s
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinj
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from request
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from r
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from r
```

```
Downloading segmentation_models_pytorch-0.5.0-py3-none-any.whl (154 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.8/154.8 kB 4.0 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.3 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 49.8 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 57.0 MB/s eta 0:00:00
```

```
import os

base_path = "/content/Jim_sem-3/train"
print("Train Images:", os.listdir(os.path.join(base_path))[:5])
print("Train Masks:", os.listdir(os.path.join(base_path))[:5])
```

```
Train Images: ['IMG_0988_jpg.rf.2c2364074ff019f7626ef70bd9d79733.jpg', 'IMG_0994_jpg.rf.9ef1e2831c8a2be
Train Masks: ['IMG_0988_jpg.rf.2c2364074ff019f7626ef70bd9d79733.jpg', 'IMG_0994_jpg.rf.9ef1e2831c8a2be6
```

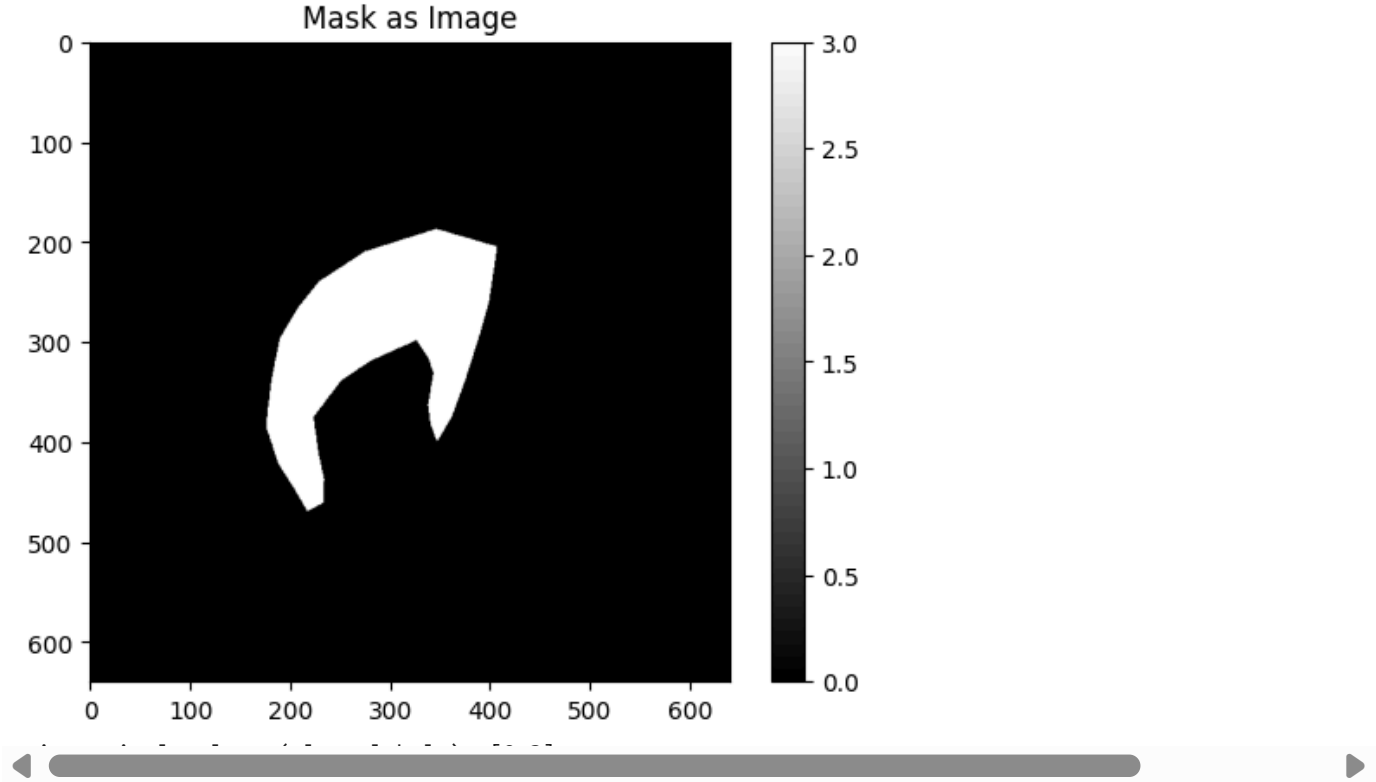
```
import os
import shutil

def sort_images_masks(folder_path):
    images_path = os.path.join(folder_path)
    masks_path = os.path.join(folder_path)

    for filename in os.listdir(images_path):
        if "_mask.png" in filename:
            shutil.move(os.path.join(images_path, filename), os.path.join(masks_path, filename))

    for filename in os.listdir(masks_path):
        if "_mask.png" not in filename:
            shutil.move(os.path.join(masks_path, filename), os.path.join(images_path, filename))

sort_images_masks("/content/Jim_sem-3/train")
sort_images_masks("/content/Jim_sem-3/valid")
sort_images_masks("/content/Jim_sem-3/test")
```

```
import os
import shutil

def sort_images_masks(folder_path):
    images_path = os.path.join(folder_path, "images")
    masks_path = os.path.join(folder_path, "masks")


    os.makedirs(images_path, exist_ok=True)
    os.makedirs(masks_path, exist_ok=True)

    for filename in os.listdir(folder_path):
        file_path = os.path.join(folder_path, filename)
        if os.path.isfile(file_path):
            if "_mask.png" in filename:
                shutil.move(file_path, os.path.join(masks_path, filename))
            else:
```

```
                shutil.move(file_path, os.path.join(images_path, filename))

sort_images_masks("/content/Jim_sem-3/train")
sort_images_masks("/content/Jim_sem-3/valid")
sort_images_masks("/content/Jim_sem-3/test")



os.listdir("/content/Jim_sem-3/train/images")[:5]
os.listdir("/content/Jim_sem-3/train/masks")[:5]
```

```
['IMG_0980_jpg.rf.a28e5b034139bff74b3958ff8ade9433_mask.png',
 'IMG_0963_jpg.rf.a8123b1bd726db8c07e8530247d97ae1_mask.png',
 'IMG_1018_jpg.rf.0ba7a96411f867abd6599e4f86d8bb9d_mask.png',
 'IMG_0994_jpg.rf.ebf1ff1a1468d018c2490d2bf0e05498_mask.png',
 'IMG_0992_jpg.rf.1ec21a48476fa73f877858a29ad24463_mask.png']
```

```
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

mask_path = "/content/Jim_sem-3/train/masks/IMG_0879_jpg.rf.a3264c77d05c2a975bf47b9bf61485e9_mask.png"
mask = Image.open(mask_path)
mask_np = np.array(mask)

plt.imshow(mask_np, cmap='gray')
plt.title("Mask as Image")
plt.colorbar()
plt.show()

print("Unique pixel values (class labels):", np.unique(mask_np))
```



```
!pip install -q albumentations
```

```
import albumentations as A
from albumentations.pytorch import ToTensorV2

train_transform = A.Compose([
    A.Resize(256, 256),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.RandomRotate90(p=0.5),
    A.ShiftScaleRotate(shift_limit=0.05, scale_limit=0.1, rotate_limit=10, p=0.5),
    A.ColorJitter(p=0.3),
    A.Normalize(mean=(0.485, 0.456, 0.406),
                std=(0.229, 0.224, 0.225)),
    ToTensorV2()
])

val_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=(0.485, 0.456, 0.406),
                std=(0.229, 0.224, 0.225)),
    ToTensorV2()
])
```

```
ShiftScaleRotate is a special case of Affine transform. Please use Affine transform instead.
```

```
import os
import cv2
import torch
from torch.utils.data import Dataset

class SegmentationDataset(Dataset):
    def __init__(self, image_dir, mask_dir, transform=None):
        self.image_dir = image_dir
        self.mask_dir = mask_dir
        self.transform = transform

        self.image_paths = sorted([
            os.path.join(image_dir, fname) for fname in os.listdir(image_dir)
            if fname.endswith(('.jpg', '.jpeg', '.png'))
        ])
        self.mask_paths = sorted([
            os.path.join(mask_dir, fname) for fname in os.listdir(mask_dir)
            if fname.endswith(('.jpg', '.jpeg', '.png'))
        ])

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

    def __getitem__(self, idx):
        image = cv2.imread(self.image_paths[idx])
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        mask = cv2.imread(self.mask_paths[idx], cv2.IMREAD_GRAYSCALE)

        if self.transform:
```

```
            augmented = self.transform(image=image, mask=mask)
            image = augmented['image']
            mask = augmented['mask']

        return image, mask.long()



Start coding or generate with AI.


train_dataset = SegmentationDataset(
    image_dir="/content/Jim_sem-3/train/images",
    mask_dir="/content/Jim_sem-3/train/masks",
    #transform=train_transform
)

val_dataset = SegmentationDataset(
    image_dir="/content/Jim_sem-3/valid/images",
    mask_dir="/content/Jim_sem-3/valid/masks",
    #transform=val_transform
)



import torch
import torch.nn as nn
import torch.nn.functional as F

class UNet(nn.Module):
    def __init__(self, num_classes):
        super(UNet, self).__init__()

        def conv_block(in_channels, out_channels):
            return nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
                nn.ReLU(inplace=True),
                nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
                nn.ReLU(inplace=True),
            )

        self.encoder1 = conv_block(3, 64)
        self.pool1 = nn.MaxPool2d(2)
        self.encoder2 = conv_block(64, 128)
        self.pool2 = nn.MaxPool2d(2)
        self.encoder3 = conv_block(128, 256)
        self.pool3 = nn.MaxPool2d(2)
        self.encoder4 = conv_block(256, 512)
        self.pool4 = nn.MaxPool2d(2)

        self.bottleneck = conv_block(512, 1024)

        self.upconv4 = nn.ConvTranspose2d(1024, 512, kernel_size=2, stride=2)
        self.decoder4 = conv_block(1024, 512)
        self.upconv3 = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
        self.decoder3 = conv_block(512, 256)
        self.upconv2 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
        self.decoder2 = conv_block(256, 128)
        self.upconv1 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
        self.decoder1 = conv_block(128, 64)
```

```
        self.final_conv = nn.Conv2d(64, num_classes, kernel_size=1)

    def forward(self, x):
        enc1 = self.encoder1(x)
        enc2 = self.encoder2(self.pool1(enc1))
        enc3 = self.encoder3(self.pool2(enc2))
        enc4 = self.encoder4(self.pool3(enc3))

        bottleneck = self.bottleneck(self.pool4(enc4))

        dec4 = self.upconv4(bottleneck)
        dec4 = torch.cat((dec4, enc4), dim=1)
        dec4 = self.decoder4(dec4)

        dec3 = self.upconv3(dec4)
        dec3 = torch.cat((dec3, enc3), dim=1)
        dec3 = self.decoder3(dec3)

        dec2 = self.upconv2(dec3)
        dec2 = torch.cat((dec2, enc2), dim=1)
        dec2 = self.decoder2(dec2)

        dec1 = self.upconv1(dec2)
        dec1 = torch.cat((dec1, enc1), dim=1)
        dec1 = self.decoder1(dec1)

        return self.final_conv(dec1)
```

```
!pip install segmentation-models-pytorch
```

```
Collecting segmentation-models-pytorch
  Downloading segmentation_models_pytorch-0.5.0-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: huggingface-hub>=0.24 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: numpy>=1.19.3 in /usr/local/lib/python3.11/dist-packages (from segmen
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from segmentati
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.11/dist-packages (from s
Requirement already satisfied: timm>=0.9 in /usr/local/lib/python3.11/dist-packages (from segmentati
Requirement already satisfied: torch>=1.8 in /usr/local/lib/python3.11/dist-packages (from segmentat
Requirement already satisfied: torchvision>=0.9 in /usr/local/lib/python3.11/dist-packages (from seg
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from segment
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from hug
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-packages (from hugg
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingf
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.8-
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8->s
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.8->segmentation-models-pytorch)
```

```
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.8->segmentation-models-pytorch)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from s
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinj
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from request
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from r
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from r
Downloading segmentation_models_pytorch-0.5.0-py3-none-any.whl (154 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.8/154.8 kB 4.5 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 1.9 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 96.1 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
```

```
import segmentation_models_pytorch as smp
import torch

NUM_CLASSES = 4

model = smp.Unet(
    encoder_name="resnet34",
    encoder_weights="imagenet",
    in_channels=3,
    classes=NUM_CLASSES
)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
```

```
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or dataset
```

config.json: 100% 156/156 [00:00<00:00, 9.52kB/s]

model.safetensors: 100% 87.3M/87.3M [00:00<00:00, 118MB/s]

```
Unet(
  (encoder): ResNetEncoder(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (2): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (2): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (3): BasicBlock(
```

```
    (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer3): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (2): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (3): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (4): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (5): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer4): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
```

```
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (2): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
  (decoder): UnetDecoder(
    (center): Identity()
    (blocks): ModuleList(
      (0): UnetDecoderBlock(
        (conv1): Conv2dReLU(
          (0): Conv2d(768, 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): UnetDecoderBlock(
        (conv1): Conv2dReLU(
          (0): Conv2d(384, 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): UnetDecoderBlock(
        (conv1): Conv2dReLU(
          (0): Conv2d(192, 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): UnetDecoderBlock(
          (conv1): Conv2dReLU(
            (0): Conv2d(128, 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): UnetDecoderBlock(
          (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, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): Identity()
      (2): Activation(
        (activation): Identity()
      )
    )
  )
```

```
import torch.nn as nn

loss_fn = nn.CrossEntropyLoss()



import torch.optim as optim

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



Start coding or generate with AI.


def train_one_epoch(model, loader, optimizer, loss_fn, device):
    model.train()
    total_loss = 0

    for images, masks in loader:
        images = images.to(device)
        masks = masks.to(device)

        outputs = model(images)
        loss = loss_fn(outputs, masks)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

    return total_loss / len(loader)


def evaluate(model, loader, loss_fn, device):
    model.eval()
    total_loss = 0

    with torch.no_grad():
        for images, masks in loader:
            images = images.to(device)
            masks = masks.to(device)

            outputs = model(images)
            loss = loss_fn(outputs, masks)

            total_loss += loss.item()

    return total_loss / len(loader)



import os
import cv2
import torch
from torch.utils.data import Dataset

class SegmentationDataset(Dataset):
    def __init__(self, image_dir, mask_dir, transform=None):
```

```
            self.image_dir = image_dir
            self.mask_dir = mask_dir
            self.transform = transform

            self.image_paths = sorted([
                os.path.join(image_dir, fname) for fname in os.listdir(image_dir)
                if fname.endswith(('.jpg', '.jpeg', '.png'))
            ])
            self.mask_paths = sorted([
                os.path.join(mask_dir, fname) for fname in os.listdir(mask_dir)
                if fname.endswith(('.jpg', '.jpeg', '.png'))
            ])

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

        def __getitem__(self, idx):
            image = cv2.imread(self.image_paths[idx])
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

            mask = cv2.imread(self.mask_paths[idx], cv2.IMREAD_GRAYSCALE)

            if self.transform:
                augmented = self.transform(image=image, mask=mask)
                image = augmented['image']
                mask = augmented['mask']

            return image, mask.long()
```

```
import albumentations as A
from albumentations.pytorch import ToTensorV2
imagenet_mean = (0.485, 0.456, 0.406)
imagenet_std = (0.229, 0.224, 0.225)

train_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=imagenet_mean, std=imagenet_std),
    ToTensorV2()
])

val_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=imagenet_mean, std=imagenet_std),
    ToTensorV2()
])
```

```
A new version of Albumentations is available: '2.0.6' (you have '2.0.5'). Upgrade using: pip install -U
```

```
from torch.utils.data import DataLoader
```

```
train_dataset = SegmentationDataset(
    image_dir="/content/Jim_sem-3/train/images",
    mask_dir="/content/Jim_sem-3/train/masks",
```

```
    transform=train_transform
)

val_dataset = SegmentationDataset(
    image_dir="/content/Jim_sem-3/valid/images",
    mask_dir="/content/Jim_sem-3/valid/masks",
    transform=val_transform
)

train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=8, shuffle=False)



!pip install torchmetrics
```

```
Collecting torchmetrics
  Downloading torchmetrics-1.7.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: numpy>1.20.0 in /usr/local/lib/python3.11/dist-packages (from torchmetri
Requirement already satisfied: packaging>17.1 in /usr/local/lib/python3.11/dist-packages (from torchmet
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from torchmetri
Collecting lightning-utilities>=0.8.0 (from torchmetrics)
  Downloading lightning_utilities-0.14.3-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from lightning-ut
Requirement already satisfied: typing_extensions in /usr/local/lib/python3.11/dist-packages (from light
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->to
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch>=2.0.0->to
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=2.
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=2.
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from symp
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2-
Downloading torchmetrics-1.7.1-py3-none-any.whl (961 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 961.5/961.5 kB 14.6 MB/s eta 0:00:00
Downloading lightning_utilities-0.14.3-py3-none-any.whl (28 kB)
Installing collected packages: lightning-utilities, torchmetrics
Successfully installed lightning-utilities-0.14.3 torchmetrics-1.7.1
```

```
from torchmetrics.classification import MulticlassAccuracy

def evaluate_accuracy(model, loader, device, num_classes=4):
    model.eval()
    acc_metric = MulticlassAccuracy(num_classes=num_classes, average='macro').to(device)

    with torch.no_grad():
        for images, masks in loader:
```

```
            images = images.to(device)
            masks = masks.to(device)

            outputs = model(images)
            preds = torch.argmax(outputs, dim=1)

            acc_metric.update(preds, masks)

    return acc_metric.compute().item()



num_epochs = 10
train_losses = []
val_losses = []

for epoch in range(num_epochs):
    train_loss = train_one_epoch(model, train_loader, optimizer, loss_fn, device)
    val_loss = evaluate(model, val_loader, loss_fn, device)

    train_losses.append(train_loss)
    val_losses.append(val_loss)

    print(f"Epoch {epoch+1}/{num_epochs} - Train Loss: {train_loss:.4f} - Val Loss: {val_loss:.4f}")
```

```
Epoch 1/10 - Train Loss: 1.3227 - Val Loss: 0.8911
Epoch 2/10 - Train Loss: 0.7824 - Val Loss: 0.6249
Epoch 3/10 - Train Loss: 0.5179 - Val Loss: 0.4059
Epoch 4/10 - Train Loss: 0.3453 - Val Loss: 0.2785
Epoch 5/10 - Train Loss: 0.2480 - Val Loss: 0.2078
Epoch 6/10 - Train Loss: 0.1878 - Val Loss: 0.1759
Epoch 7/10 - Train Loss: 0.1479 - Val Loss: 0.1543
Epoch 8/10 - Train Loss: 0.1190 - Val Loss: 0.1119
Epoch 9/10 - Train Loss: 0.0978 - Val Loss: 0.0968
Epoch 10/10 - Train Loss: 0.0831 - Val Loss: 0.0831
```

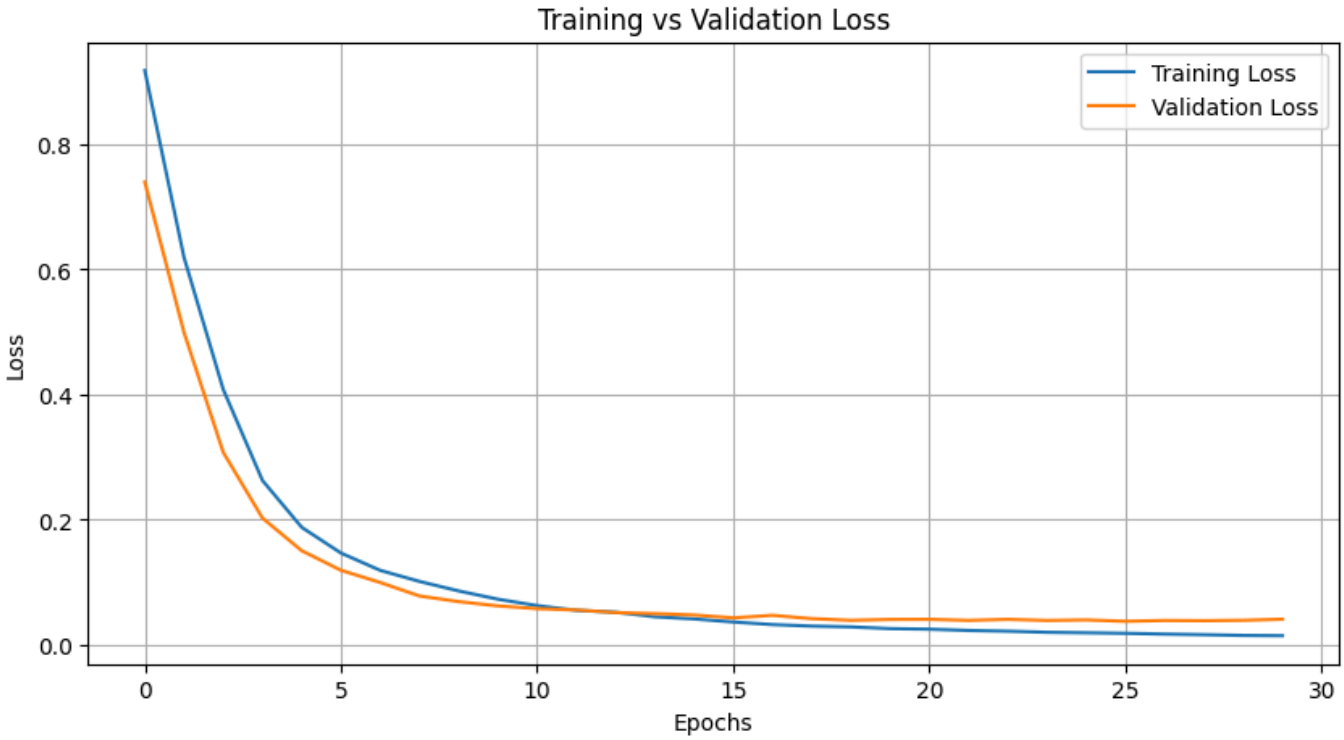
```
import torch.optim as optim
from torch.optim.lr_scheduler import ReduceLROnPlateau

optimizer = optim.Adam(model.parameters(), lr=1e-4)
scheduler = ReduceLROnPlateau(optimizer, mode='min', patience=3, factor=0.5)

loss_fn = nn.CrossEntropyLoss()

num_epochs = 30
train_losses = []
val_losses = []
train_accuracies = []
val_accuracies = []

best_val_loss = float('inf')
save_path = "best_model_checkpoint.pth"

for epoch in range(num_epochs):
    train_loss = train_one_epoch(model, train_loader, optimizer, loss_fn, device)
    val_loss = evaluate(model, val_loader, loss_fn, device)

    scheduler.step(val_loss)
    train_losses.append(train_loss)
```

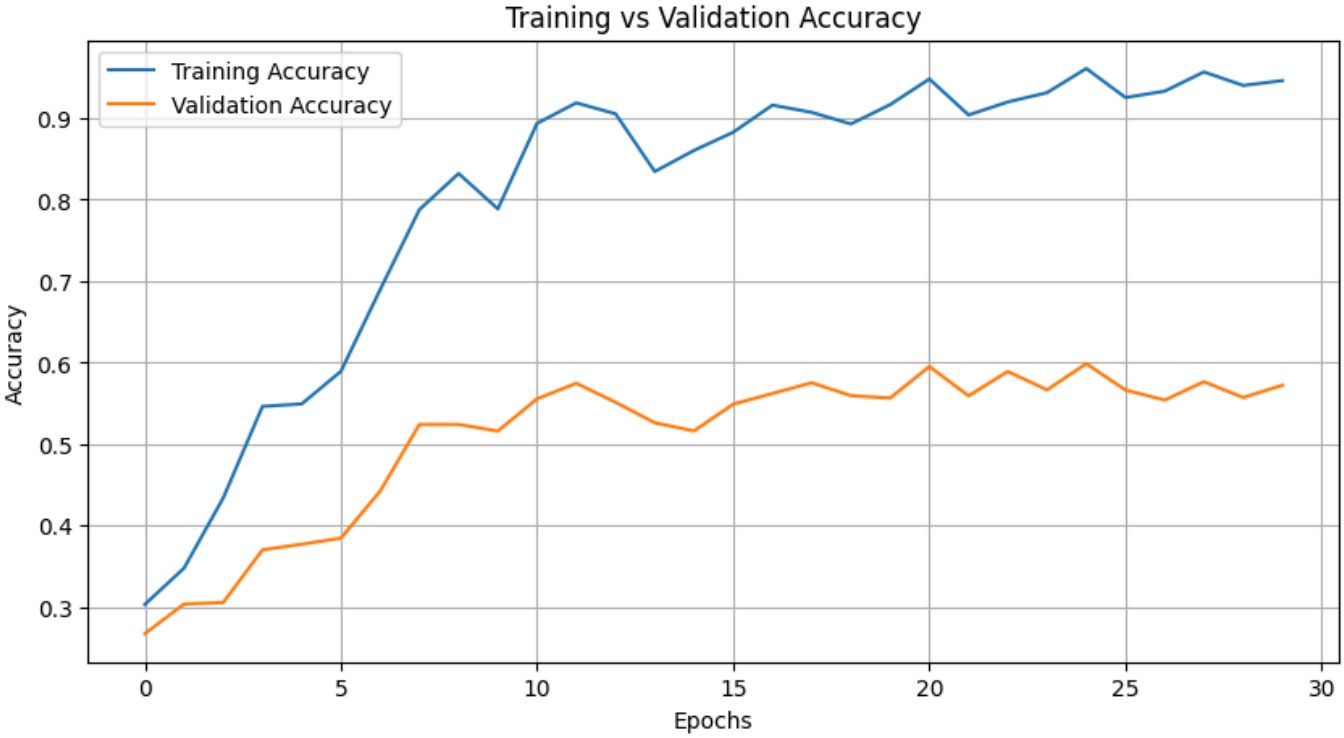
```
        val_losses.append(val_loss)
        train_acc = evaluate_accuracy(model, train_loader, device)
        val_acc = evaluate_accuracy(model, val_loader, device)

        train_accuracies.append(train_acc)
        val_accuracies.append(val_acc)

        print(f"Epoch {epoch+1}/{num_epochs} | Train Loss: {train_loss:.4f} | Val Loss: {val_loss:.4f} | Train

        if val_loss < best_val_loss:
            best_val_loss = val_loss
            torch.save({
                'epoch': epoch,
                'model_state_dict': model.state_dict(),
                'optimizer_state_dict': optimizer.state_dict(),
                'val_loss': val_loss
            }, save_path)
            print(f"Saved Best Model at Epoch {epoch+1}")
```

```
Epoch 1/30 | Train Loss: 0.9181 | Val Loss: 0.7398 | Train Acc: 0.3033 | Val Acc: 0.2674
Saved Best Model at Epoch 1
Epoch 2/30 | Train Loss: 0.6193 | Val Loss: 0.4993 | Train Acc: 0.3479 | Val Acc: 0.3038
Saved Best Model at Epoch 2
Epoch 3/30 | Train Loss: 0.4083 | Val Loss: 0.3078 | Train Acc: 0.4341 | Val Acc: 0.3058
Saved Best Model at Epoch 3
Epoch 4/30 | Train Loss: 0.2628 | Val Loss: 0.2027 | Train Acc: 0.5464 | Val Acc: 0.3704
Saved Best Model at Epoch 4
Epoch 5/30 | Train Loss: 0.1879 | Val Loss: 0.1506 | Train Acc: 0.5494 | Val Acc: 0.3772
Saved Best Model at Epoch 5
Epoch 6/30 | Train Loss: 0.1468 | Val Loss: 0.1193 | Train Acc: 0.5894 | Val Acc: 0.3847
Saved Best Model at Epoch 6
Epoch 7/30 | Train Loss: 0.1192 | Val Loss: 0.0998 | Train Acc: 0.6896 | Val Acc: 0.4424
Saved Best Model at Epoch 7
Epoch 8/30 | Train Loss: 0.1015 | Val Loss: 0.0782 | Train Acc: 0.7878 | Val Acc: 0.5242
Saved Best Model at Epoch 8
Epoch 9/30 | Train Loss: 0.0863 | Val Loss: 0.0692 | Train Acc: 0.8321 | Val Acc: 0.5242
Saved Best Model at Epoch 9
Epoch 10/30 | Train Loss: 0.0731 | Val Loss: 0.0624 | Train Acc: 0.7887 | Val Acc: 0.5161
Saved Best Model at Epoch 10
Epoch 11/30 | Train Loss: 0.0628 | Val Loss: 0.0579 | Train Acc: 0.8937 | Val Acc: 0.5557
Saved Best Model at Epoch 11
Epoch 12/30 | Train Loss: 0.0552 | Val Loss: 0.0558 | Train Acc: 0.9188 | Val Acc: 0.5747
Saved Best Model at Epoch 12
Epoch 13/30 | Train Loss: 0.0523 | Val Loss: 0.0517 | Train Acc: 0.9054 | Val Acc: 0.5514
Saved Best Model at Epoch 13
Epoch 14/30 | Train Loss: 0.0450 | Val Loss: 0.0500 | Train Acc: 0.8345 | Val Acc: 0.5263
Saved Best Model at Epoch 14
Epoch 15/30 | Train Loss: 0.0415 | Val Loss: 0.0478 | Train Acc: 0.8604 | Val Acc: 0.5162
Saved Best Model at Epoch 15
Epoch 16/30 | Train Loss: 0.0366 | Val Loss: 0.0432 | Train Acc: 0.8828 | Val Acc: 0.5491
Saved Best Model at Epoch 16
Epoch 17/30 | Train Loss: 0.0325 | Val Loss: 0.0473 | Train Acc: 0.9160 | Val Acc: 0.5622
Epoch 18/30 | Train Loss: 0.0301 | Val Loss: 0.0420 | Train Acc: 0.9071 | Val Acc: 0.5754
Saved Best Model at Epoch 18
Epoch 19/30 | Train Loss: 0.0288 | Val Loss: 0.0393 | Train Acc: 0.8929 | Val Acc: 0.5595
Saved Best Model at Epoch 19
Epoch 20/30 | Train Loss: 0.0261 | Val Loss: 0.0407 | Train Acc: 0.9166 | Val Acc: 0.5566
Epoch 21/30 | Train Loss: 0.0251 | Val Loss: 0.0409 | Train Acc: 0.9481 | Val Acc: 0.5953
Epoch 22/30 | Train Loss: 0.0231 | Val Loss: 0.0392 | Train Acc: 0.9040 | Val Acc: 0.5593
Saved Best Model at Epoch 22
Epoch 23/30 | Train Loss: 0.0219 | Val Loss: 0.0409 | Train Acc: 0.9199 | Val Acc: 0.5892
Epoch 24/30 | Train Loss: 0.0202 | Val Loss: 0.0391 | Train Acc: 0.9313 | Val Acc: 0.5666
```

```
Saved Best Model at Epoch 24
Epoch 25/30 | Train Loss: 0.0194 | Val Loss: 0.0399 | Train Acc: 0.9610 | Val Acc: 0.5988
Epoch 26/30 | Train Loss: 0.0186 | Val Loss: 0.0378 | Train Acc: 0.9253 | Val Acc: 0.5665
Saved Best Model at Epoch 26
Epoch 27/30 | Train Loss: 0.0173 | Val Loss: 0.0390 | Train Acc: 0.9332 | Val Acc: 0.5541
Epoch 28/30 | Train Loss: 0.0163 | Val Loss: 0.0387 | Train Acc: 0.9568 | Val Acc: 0.5766
Epoch 29/30 | Train Loss: 0.0152 | Val Loss: 0.0393 | Train Acc: 0.9402 | Val Acc: 0.5573
Epoch 30/30 | Train Loss: 0.0147 | Val Loss: 0.0411 | Train Acc: 0.9461 | Val Acc: 0.5723
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10,5))
plt.plot(train_losses, label='Training Loss')
plt.plot(val_losses, label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training vs Validation Loss')
plt.legend()
plt.grid()
plt.show()
plt.figure(figsize=(10,5))
plt.plot(train_accuracies, label='Training Accuracy')
plt.plot(val_accuracies, label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.grid()
plt.show()
```






```
checkpoint = torch.load("best_model_checkpoint.pth")
model.load_state_dict(checkpoint['model_state_dict'])
model.eval()
```

```
Unet(
  (encoder): ResNetEncoder(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
```

```
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
        (layer1): Sequential(
          (0): BasicBlock(
            (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
          (1): BasicBlock(
            (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
          (2): BasicBlock(
            (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
        )
        (layer2): Sequential(
          (0): BasicBlock(
            (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (downsample): Sequential(
              (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
              (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            )
          )
          (1): BasicBlock(
            (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
          (2): BasicBlock(
            (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (relu): ReLU(inplace=True)
            (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
          (3): BasicBlock(
            (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
            (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
from torchmetrics.classification import MulticlassAccuracy, MulticlassJaccardIndex

def final_evaluate_metrics(model, dataloader, device, num_classes=4):
    model.eval()

    acc_metric = MulticlassAccuracy(num_classes=num_classes, average='macro').to(device)
    iou_metric = MulticlassJaccardIndex(num_classes=num_classes, average='macro').to(device)
```

```
        with torch.no_grad():
            for images, masks in dataloader:
                images = images.to(device)
                masks = masks.to(device)

                outputs = model(images)
                preds = torch.argmax(outputs, dim=1)

                acc_metric.update(preds, masks)
                iou_metric.update(preds, masks)

        accuracy = acc_metric.compute().item()
        miou = iou_metric.compute().item()

        print(f" Final Validation Accuracy: {accuracy:.4f}")
        print(f" Final Validation Mean IoU: {miou:.4f}")

    final_evaluate_metrics(model, val_loader, device)
```

```
 Final Validation Accuracy: 0.5665
 Final Validation Mean IoU: 0.5165
```

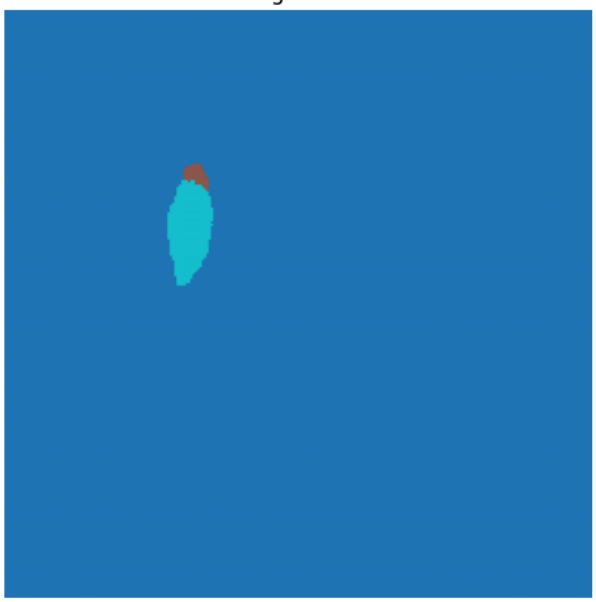
Start coding or generate with AI.

```
from torchmetrics import ConfusionMatrix
from torchmetrics.classification import MulticlassAccuracy, MulticlassPrecision, MulticlassRecall

acc_metric = MulticlassAccuracy(num_classes=4, average='macro').to(device)
prec_metric = MulticlassPrecision(num_classes=4, average='macro').to(device)
rec_metric  = MulticlassRecall(num_classes=4, average='macro').to(device)

confmat = ConfusionMatrix(num_classes=4, task="multiclass").to(device)

with torch.no_grad():
    for imgs, masks in val_loader:
        imgs, masks = imgs.to(device), masks.to(device)
        preds = torch.argmax(model(imgs), dim=1)

        acc_metric.update(preds, masks)
        prec_metric.update(preds, masks)
        rec_metric.update(preds, masks)
        confmat.update(preds.flatten(), masks.flatten())
accuracy = acc_metric.compute().item()
precision = prec_metric.compute().item()
recall = rec_metric.compute().item()
cm = confmat.compute().cpu().numpy()

print(f"Mean Accuracy: {accuracy:.4f}")
print(f"Mean Precision: {precision:.4f}")
print(f"Mean Recall: {recall:.4f}")
print("Confusion Matrix:\n", cm)
```

```
Mean Accuracy: 0.5665
Mean Precision: 0.6116
Mean Recall: 0.5665
Confusion Matrix:
 [[1279592    1076       0    4136]
```

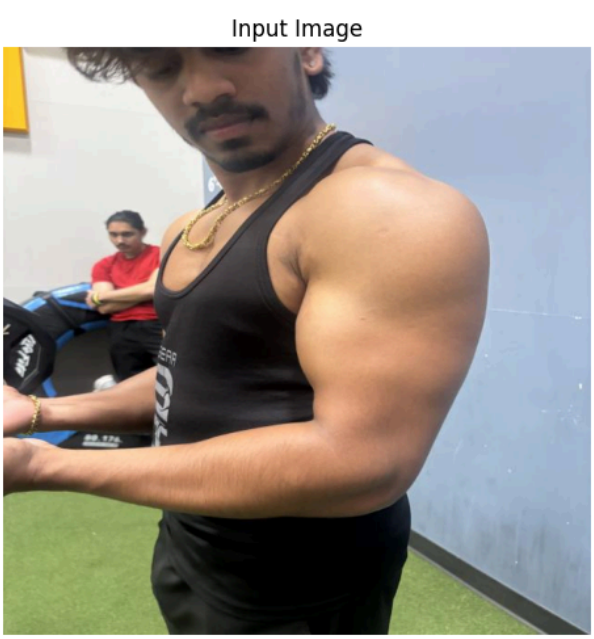
```
        [   3313   12449       0       0]
        [    275       0       0       0]
        [   5111      23       0    4745]]
```

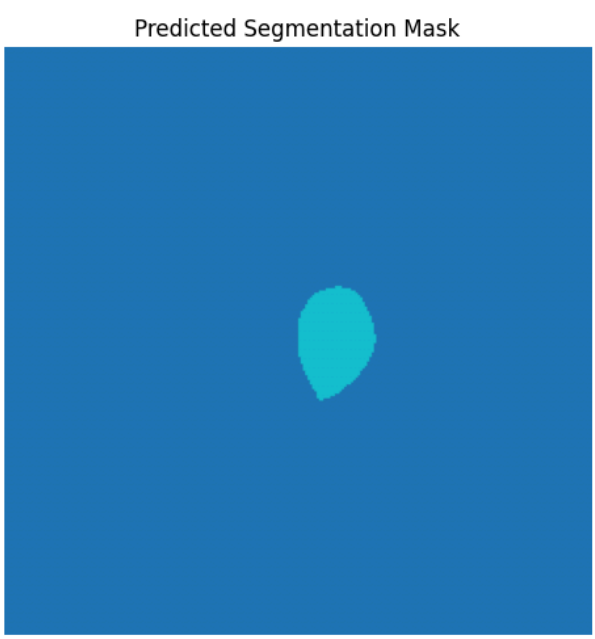
```
import cv2
import numpy as np
import torch
import matplotlib.pyplot as plt
from albumentations.pytorch import ToTensorV2
import albumentations as A

single_image_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=(0.485, 0.456, 0.406),
                std=(0.229, 0.224, 0.225)),
    ToTensorV2()
])

def predict_single_image(model, image_path, device):
    model.eval()
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    augmented = single_image_transform(image=image)
    input_tensor = augmented['image'].unsqueeze(0).to(device)
    with torch.no_grad():
        output = model(input_tensor)
        pred_mask = torch.argmax(output, dim=1).squeeze(0).cpu().numpy()


    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(image)
    plt.title("Input Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(pred_mask, cmap='tab10')
    plt.title("Predicted Segmentation Mask")
    plt.axis("off")

    plt.tight_layout()
    plt.show()

predict_single_image(model, "/content/Jim_sem-3/test/images/IMG_0954_jpg.rf.d764f05dd55ef5e315b3cdf029b7906
```




```
import cv2
import numpy as np
import torch
import matplotlib.pyplot as plt
from albumentations.pytorch import ToTensorV2
import albumentations as A

single_image_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=(0.485, 0.456, 0.406),
                std=(0.229, 0.224, 0.225)),
    ToTensorV2()
])

def predict_single_image(model, image_path, device):
    model.eval()
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    augmented = single_image_transform(image=image)
    input_tensor = augmented['image'].unsqueeze(0).to(device)
    with torch.no_grad():
        output = model(input_tensor)
        pred_mask = torch.argmax(output, dim=1).squeeze(0).cpu().numpy()


    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(image)
    plt.title("Input Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(pred_mask, cmap='tab10')
    plt.title("Predicted Segmentation Mask")
    plt.axis("off")
```

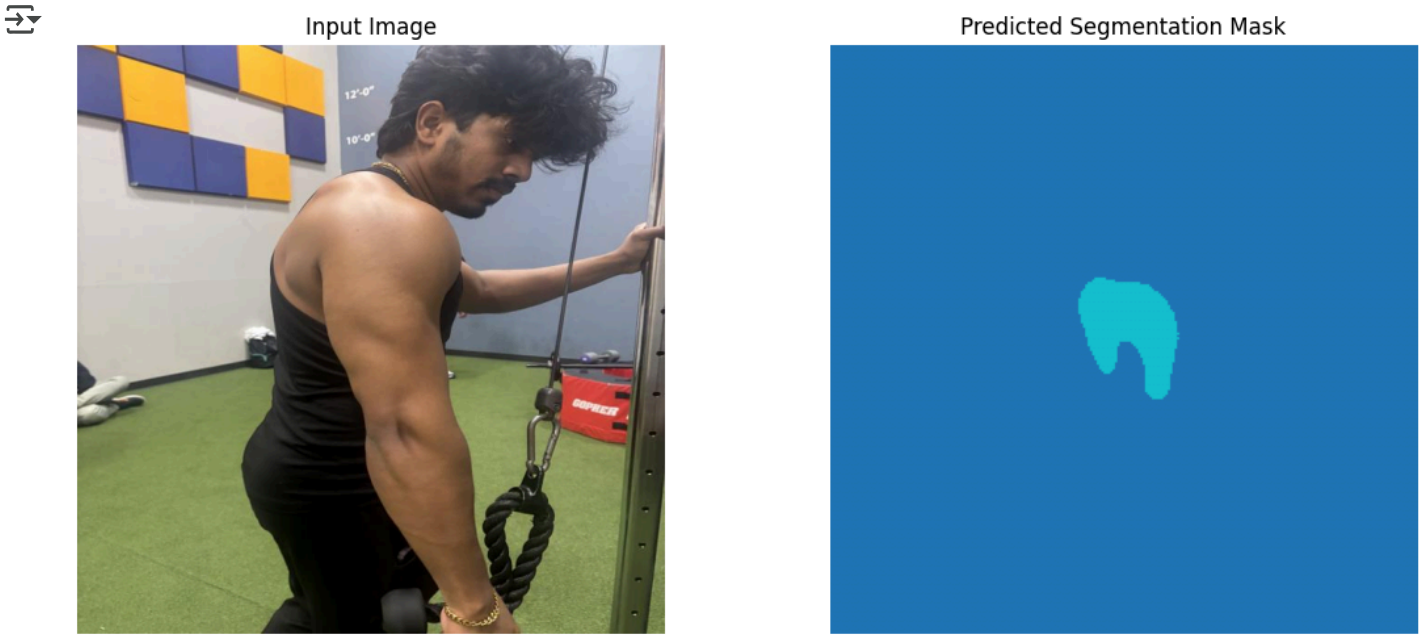
```
    plt.tight_layout()
    plt.show()

predict_single_image(model, "/content/Jim_sem-3/test/images/IMG_0954_jpg.rf.d764f05dd55ef5e315b3cdf029b7906
```






```
import cv2
import numpy as np
import torch
import matplotlib.pyplot as plt
from albumentations.pytorch import ToTensorV2
import albumentations as A

single_image_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=(0.485, 0.456, 0.406),
                std=(0.229, 0.224, 0.225)),
    ToTensorV2()
])

def predict_single_image(model, image_path, device):
    model.eval()
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    augmented = single_image_transform(image=image)
    input_tensor = augmented['image'].unsqueeze(0).to(device)
    with torch.no_grad():
        output = model(input_tensor)
        pred_mask = torch.argmax(output, dim=1).squeeze(0).cpu().numpy()


    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(image)
```

```
    plt.title("Input Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(pred_mask, cmap='tab10')
    plt.title("Predicted Segmentation Mask")
    plt.axis("off")

    plt.tight_layout()
    plt.show()

predict_single_image(model, "/content/Jim_sem-3/train/images/IMG_0963_jpg.rf.33fab83a3b64b4a117f3a1c2cb6469
```




```
Start coding or generate with AI.


import itertools
import pandas as pd
import torch
import torch.nn as nn
from torch.optim import Adam, SGD
from torchmetrics.classification import MulticlassJaccardIndex


learning_rates = [1e-3, 5e-4, 1e-5]
batch_sizes    = [4, 8, 16]
optimizers     = ['adam', 'sgd']

iou_metric = MulticlassJaccardIndex(num_classes=4, average='macro').to(device)

results = []

for lr, bs, opt_name in itertools.product(learning_rates, batch_sizes, optimizers):
    print(f"\n  LR={lr}, BS={bs}, OPT={opt_name.upper()}")

    train_loader = DataLoader(train_dataset, batch_size=bs, shuffle=True)
```

```
        val_loader   = DataLoader(val_dataset,   batch_size=bs, shuffle=False)

        model = smp.Unet(
            encoder_name="resnet34",
            encoder_weights="imagenet",
            in_channels=3,
            classes=4
        ).to(device)

        if opt_name == 'adam':
            optimizer = Adam(model.parameters(), lr=lr)
        else:
            optimizer = SGD(model.parameters(), lr=lr, momentum=0.9)

        loss_fn = nn.CrossEntropyLoss()

        for epoch in range(2):
            model.train()
            for imgs, masks in train_loader:
                imgs, masks = imgs.to(device), masks.to(device)
                preds = model(imgs)
                loss  = loss_fn(preds, masks)
                optimizer.zero_grad(); loss.backward(); optimizer.step()

        model.eval()
        val_loss = 0.0
        iou_metric.reset()
        with torch.no_grad():
            for imgs, masks in val_loader:
                imgs, masks = imgs.to(device), masks.to(device)
                preds = model(imgs)
                val_loss += loss_fn(preds, masks).item()
                iou_metric.update(torch.argmax(preds, dim=1), masks)
        val_loss /= len(val_loader)
        val_iou   = iou_metric.compute().item()

        print(f"   → Val Loss: {val_loss:.4f}, Val mIoU: {val_iou:.4f}")

        results.append({
            'lr': lr,
            'batch_size': bs,
            'optimizer': opt_name,
            'val_loss': val_loss,
            'val_mIoU': val_iou
        })

    df = pd.DataFrame(results)
    df = df.sort_values('val_mIoU', ascending=False).reset_index(drop=True)
    print("\n Top 3 configs by mIoU:")
    print(df.head(3))

    best = df.iloc[0]
    print(f"\n Best config → LR={best.lr}, BS={best.batch_size}, OPT={best.optimizer.upper()}, mIoU={best.val_m
```

```
    LR=0.001, BS=4, OPT=ADAM
     → Val Loss: 0.0874, Val mIoU: 0.3006

    LR=0.001, BS=4, OPT=SGD
```

```
 → Val Loss: 0.1065, Val mIoU: 0.2451

LR=0.001, BS=8, OPT=ADAM
 → Val Loss: 0.0895, Val mIoU: 0.2451

LR=0.001, BS=8, OPT=SGD
 → Val Loss: 0.1331, Val mIoU: 0.2451

LR=0.001, BS=16, OPT=ADAM
 → Val Loss: 0.0917, Val mIoU: 0.2451

LR=0.001, BS=16, OPT=SGD
 → Val Loss: 0.1632, Val mIoU: 0.2451

LR=0.0005, BS=4, OPT=ADAM
 → Val Loss: 0.0761, Val mIoU: 0.2451

LR=0.0005, BS=4, OPT=SGD
 → Val Loss: 0.1386, Val mIoU: 0.2451

LR=0.0005, BS=8, OPT=ADAM
 → Val Loss: 0.1154, Val mIoU: 0.2451

LR=0.0005, BS=8, OPT=SGD
 → Val Loss: 0.1894, Val mIoU: 0.2450

LR=0.0005, BS=16, OPT=ADAM
 → Val Loss: 0.1628, Val mIoU: 0.3000

LR=0.0005, BS=16, OPT=SGD
 → Val Loss: 0.2746, Val mIoU: 0.2449

LR=1e-05, BS=4, OPT=ADAM
 → Val Loss: 1.0547, Val mIoU: 0.1943

LR=1e-05, BS=4, OPT=SGD
 → Val Loss: 1.3151, Val mIoU: 0.1000

LR=1e-05, BS=8, OPT=ADAM
 → Val Loss: 1.5187, Val mIoU: 0.0906

LR=1e-05, BS=8, OPT=SGD
 → Val Loss: 1.5767, Val mIoU: 0.0592

LR=1e-05, BS=16, OPT=ADAM
```

```
from google.colab import files
files.download('best_model_checkpoint.pth')
```

```
print(f"Epoch {epoch+1}/{num_epochs} - Train Loss: {train_loss:.4f} - Val Loss: {val_loss:.4f}")
```

```
Epoch 10/10 - Train Loss: 0.0831 - Val Loss: 0.0831
```

```
torch.save({
    'epoch': num_epochs,
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
    'loss': val_losses[-1],
```

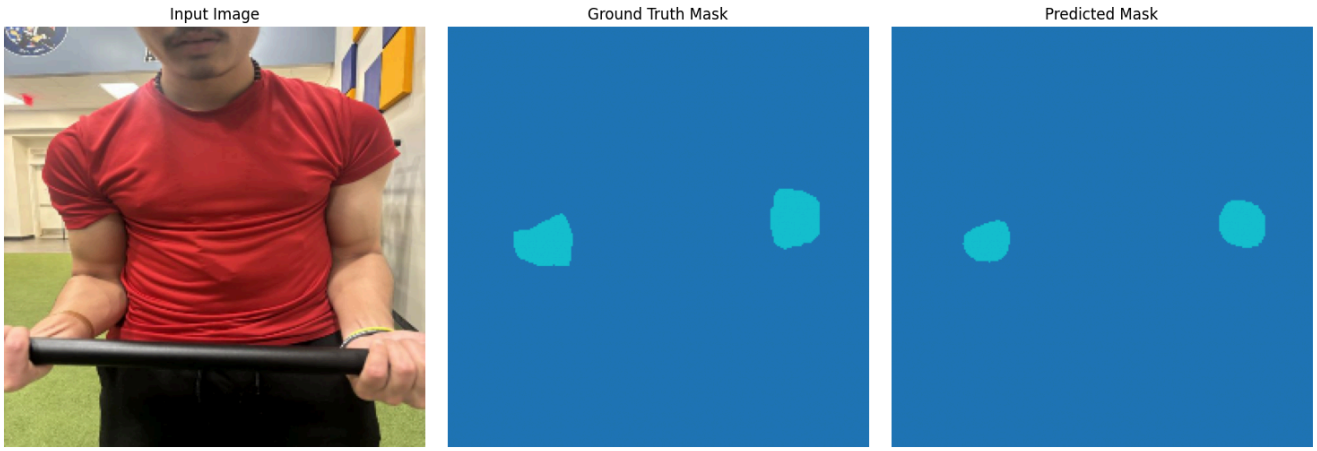
```
}, "unet_checkpoint.pth")
print("Full training checkpoint saved as unet_checkpoint.pth")
```

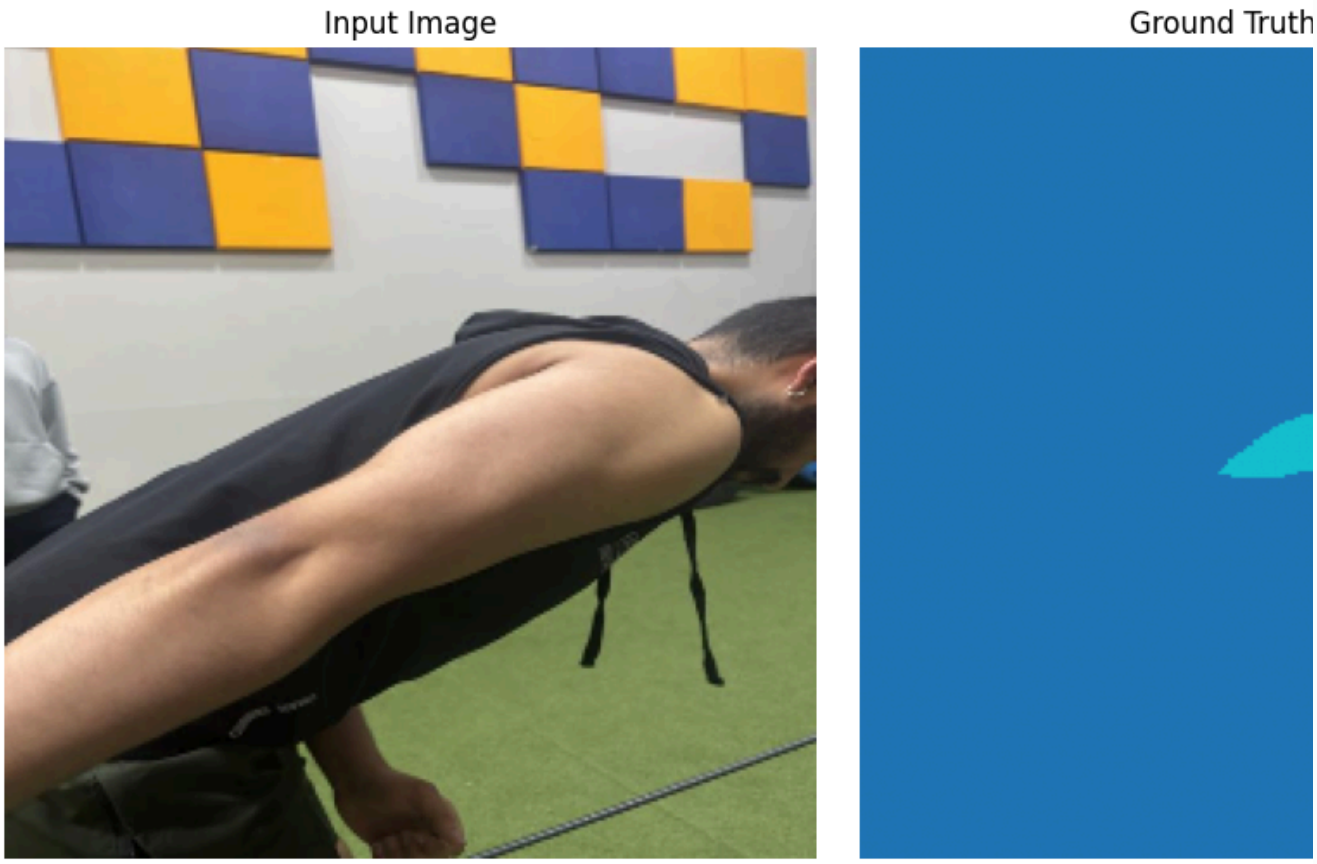
```
Full training checkpoint saved as unet_checkpoint.pth
```

```
import matplotlib.pyplot as plt
import torch
import numpy as np

def denormalize(tensor):
    # Undo ImageNet normalization
    mean = torch.tensor([0.485, 0.456, 0.406]).view(3, 1, 1)
    std = torch.tensor([0.229, 0.224, 0.225]).view(3, 1, 1)
    return tensor * std + mean

def visualize_prediction(model, dataloader, device):
    model.eval()
    with torch.no_grad():
        for images, masks in dataloader:
            images = images.to(device)
            masks = masks.to(device)

            outputs = model(images)
            preds = torch.argmax(outputs, dim=1)

            for i in range(images.shape[0]):
                image = denormalize(images[i].cpu()).permute(1, 2, 0).numpy()
                true_mask = masks[i].cpu().numpy()
                pred_mask = preds[i].cpu().numpy()

                fig, axs = plt.subplots(1, 3, figsize=(15, 5))

                axs[0].imshow(image)
                axs[0].set_title("Input Image")
                axs[0].axis("off")

                axs[1].imshow(true_mask, cmap='tab10')
                axs[1].set_title("Ground Truth Mask")
                axs[1].axis("off")

                axs[2].imshow(pred_mask, cmap='tab10')
                axs[2].set_title("Predicted Mask")
                axs[2].axis("off")

                plt.tight_layout()
                plt.show()

            break
```

Start coding or generate with AI.

```
visualize_prediction(model, val_loader, device)
```

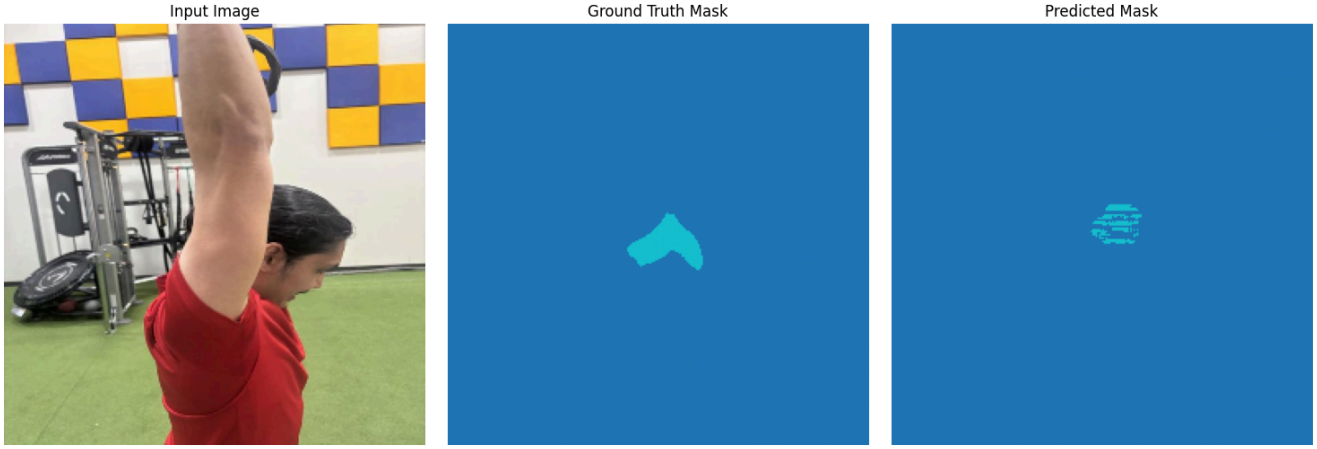

Input Image
Ground Truth Mask
Predicted Mask


Input Image
Ground Truth

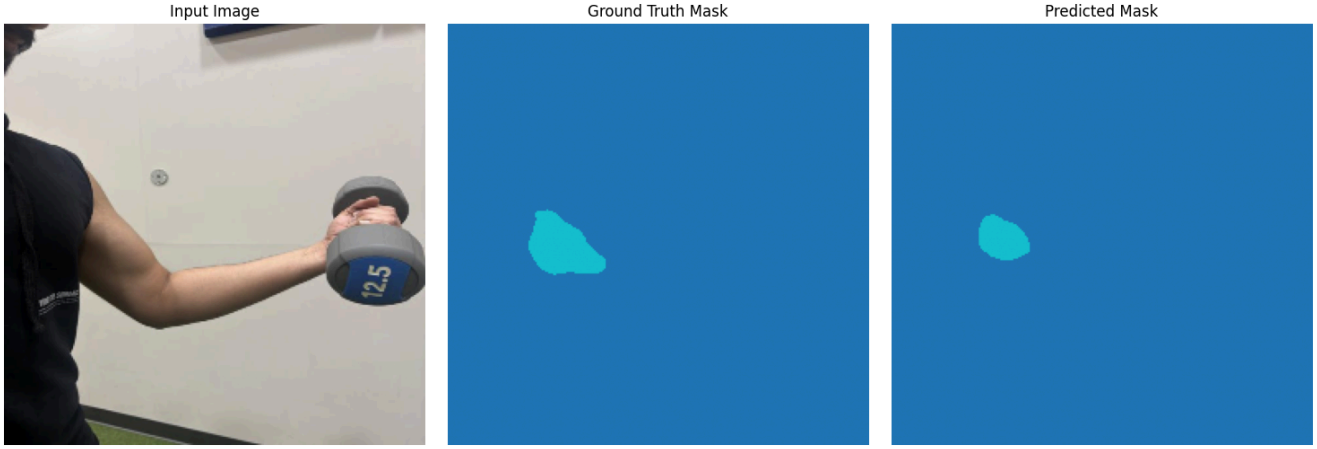

Input Image
Ground Truth Mask
Predicted Mask


Input Image
Ground Truth Mask
Predicted Mask

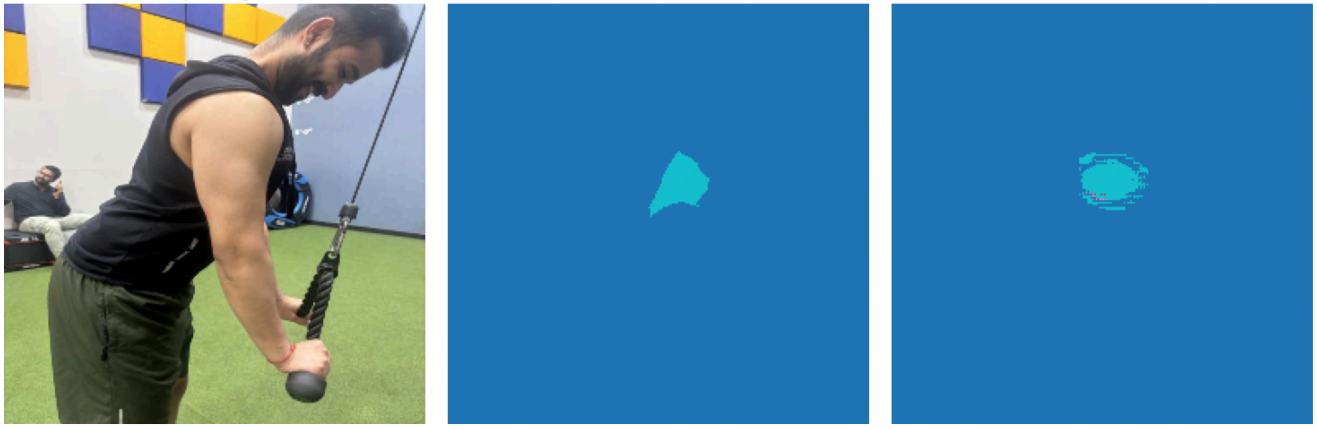
Input Image | Ground Truth Mask | Predicted Mask



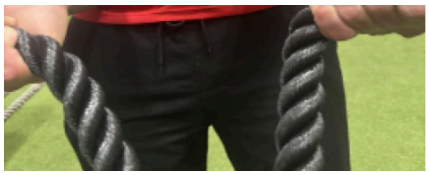
Input Image | Ground Truth Mask | Predicted Mask



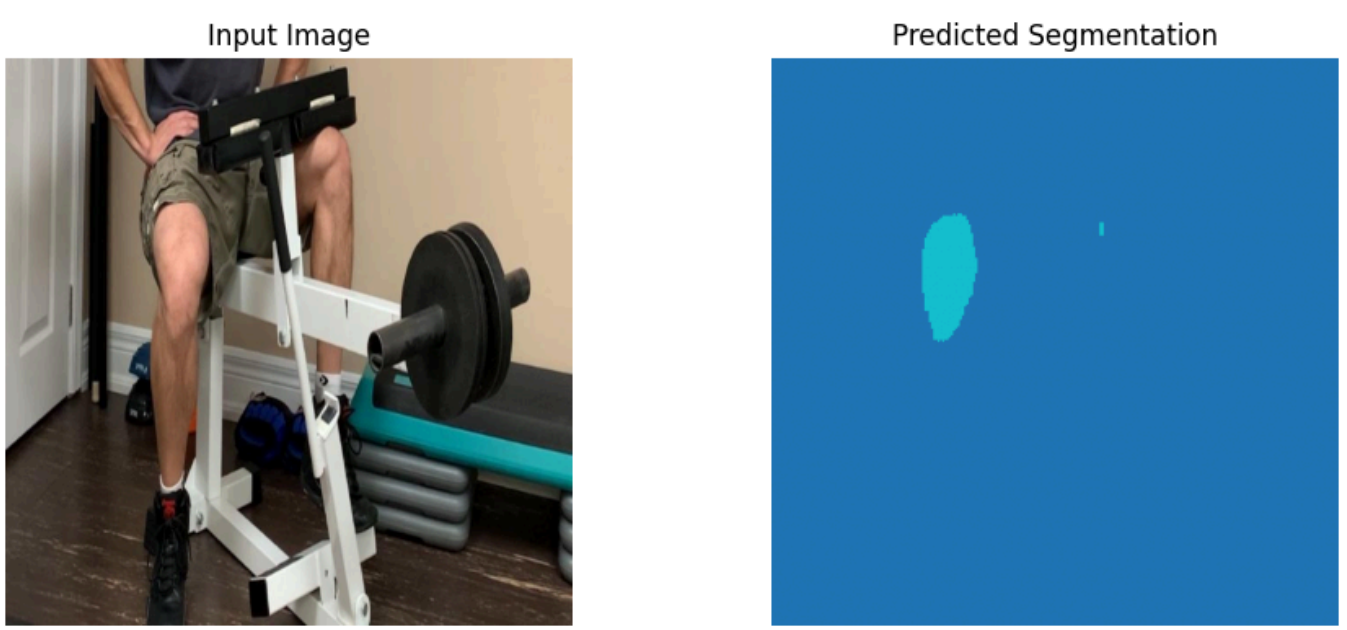
Input Image | Ground Truth Mask | Predicted Mask



Input Image | Ground Truth Mask | Predicted Mask



```
import cv2
import numpy as np
import torch
import matplotlib.pyplot as plt
from albumentations.pytorch import ToTensorV2

single_image_transform = A.Compose([
    A.Resize(256, 256),
    A.Normalize(mean=(0.485, 0.456, 0.406),
                std=(0.229, 0.224, 0.225)),
    ToTensorV2()
])

def predict_single_image(model, image_path, device):
    model.eval()

    # Load and preprocess image
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    transformed = single_image_transform(image=image)
    input_tensor = transformed['image'].unsqueeze(0).to(device)  # Add batch dim

    # Prediction
    with torch.no_grad():
        output = model(input_tensor)
        pred_mask = torch.argmax(output, dim=1).squeeze(0).cpu().numpy()

    # Denormalized image for display
    img_display = image.astype(np.uint8)

    # Visualization
    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    plt.imshow(img_display)
    plt.title("Input Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    plt.imshow(pred_mask, cmap='tab10')
    plt.title("Predicted Segmentation")
    plt.axis("off")

    plt.tight_layout()
    plt.show()



predict_single_image(model, "/content/Jim_sem-3/test/images/calf9_jpeg.rf.8253fa10f98dee9ed2647bdda454626d.
```




```
import torch.nn.functional as F

def evaluate(model, dataloader):
    model.eval()
    total_loss = 0
    criterion = nn.CrossEntropyLoss()

    with torch.no_grad():
        for images, masks in dataloader:
            images, masks = images.to(device), masks.to(device)
            outputs = model(images)
            loss = criterion(outputs, masks)
            total_loss += loss.item()

    avg_loss = total_loss / len(dataloader)
    return avg_loss

device = "cuda" if torch.cuda.is_available() else "cpu"
train_loss = evaluate(model, train_loader)
val_loss = evaluate(model, val_loader)

print(f"Train Avg Loss: {train_loss:.4f}")
print(f"Val Avg Loss: {val_loss:.4f}")
```

```
Train Avg Loss: 1.3971
Val Avg Loss: 1.3968
```

Start coding or generate with AI.

Start coding or generate with AI.

```
%matplotlib inline
import matplotlib.pyplot as plt
```

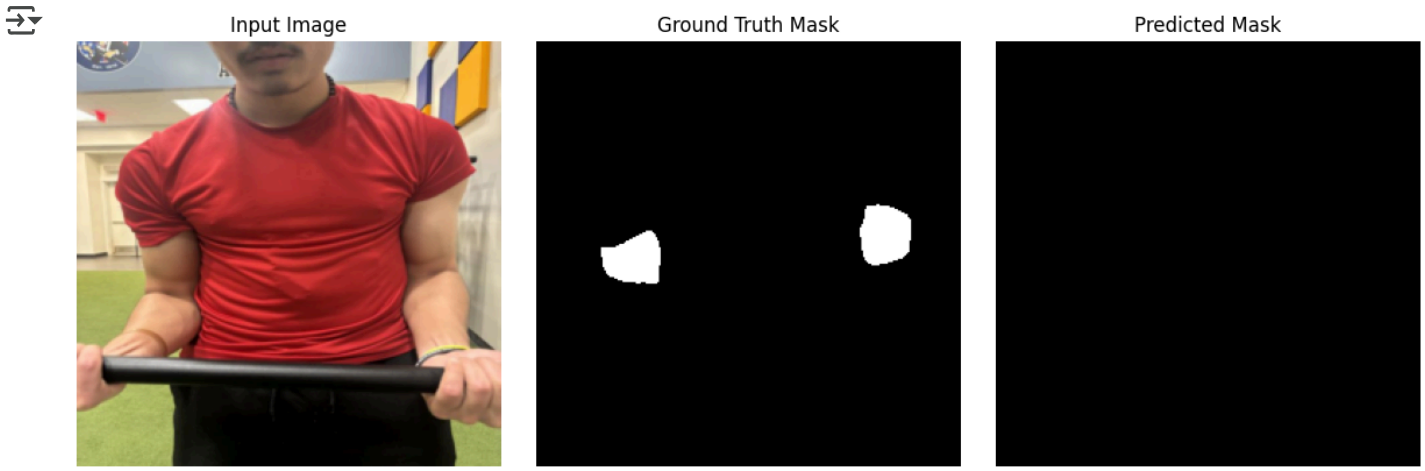
```
Start coding or generate with AI.
```

```
print(len(val_loader.dataset))
```

```
20
```

```
import matplotlib.pyplot as plt
import torch

def visualize_prediction(model, dataloader, device):
    model.eval()

    with torch.no_grad():
        for batch in dataloader:

            if isinstance(batch, (list, tuple)):
                images, masks = batch
            elif isinstance(batch, dict):
                images = batch['image']
                masks = batch['mask']
            else:
                raise TypeError("Unexpected batch format!")

            images = images.to(device)
            masks = masks.to(device)
            outputs = model(images)
            preds = torch.argmax(outputs, dim=1)
            img = images[0].cpu().permute(1, 2, 0).numpy()  # [C,H,W] -> [H,W,C]
            mask = masks[0].cpu().numpy()
            pred = preds[0].cpu().numpy()


            plt.figure(figsize=(12, 4))
            plt.subplot(1, 3, 1)
            plt.imshow(img)
            plt.title("Input Image")
            plt.axis("off")

            plt.subplot(1, 3, 2)
            plt.imshow(mask, cmap='gray')
            plt.title("Ground Truth Mask")
            plt.axis("off")

            plt.subplot(1, 3, 3)
            plt.imshow(pred, cmap='gray')
            plt.title("Predicted Mask")
            plt.axis("off")

            plt.tight_layout()
            plt.show()
            print("Unique values in predicted mask:", torch.unique(preds[0]))
```

```
            break

visualize_prediction(model, val_loader, device)
```

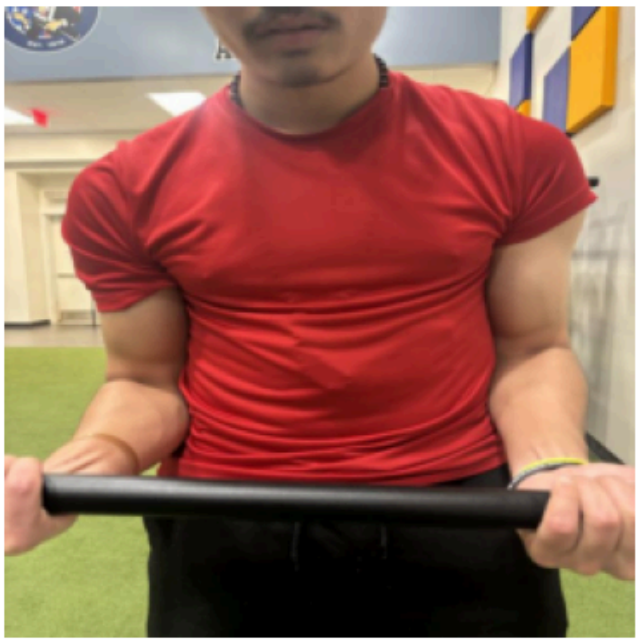


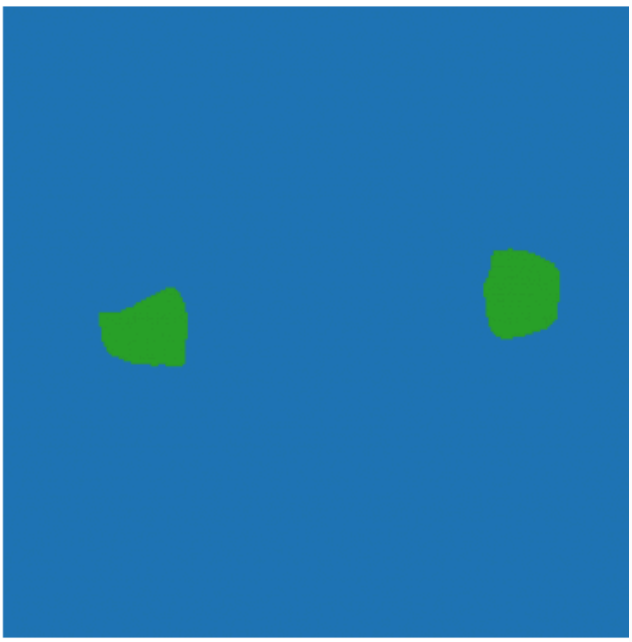

```
Unique values in predicted mask: tensor([3])
```

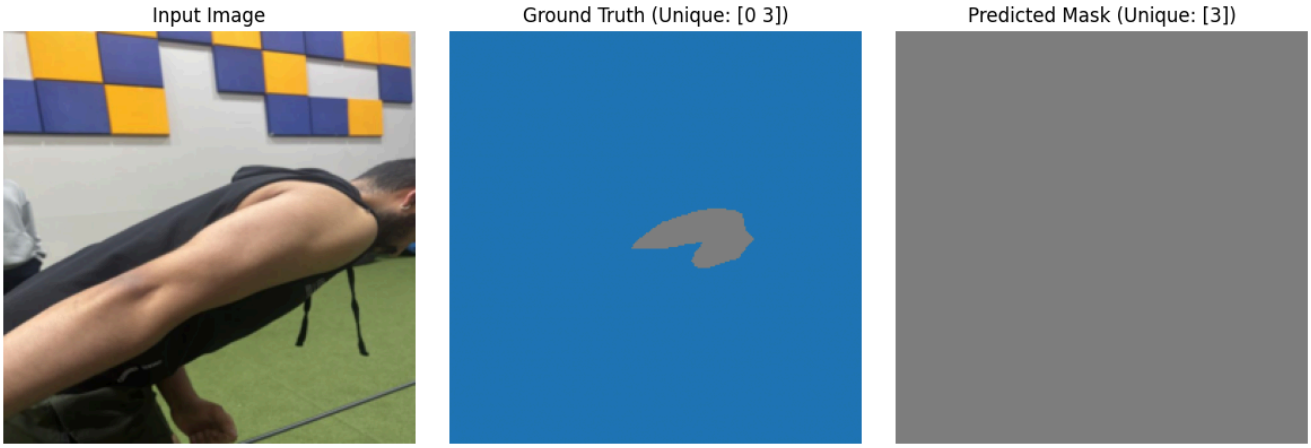
Start coding or generate with AI.

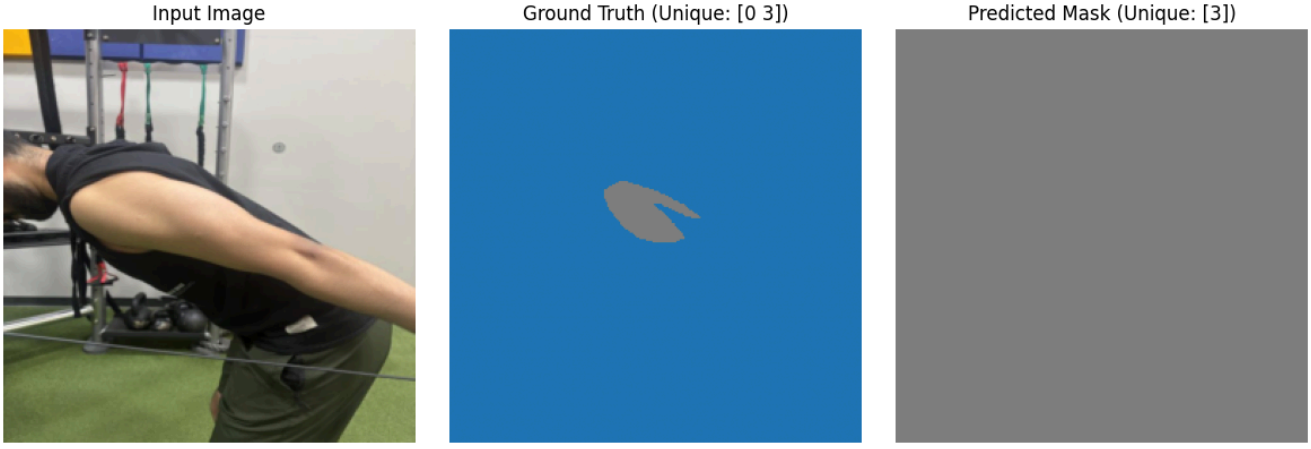
```
import torch
import matplotlib.pyplot as plt
import numpy as np

def denormalize(img_tensor):
    img = img_tensor.cpu().numpy().transpose(1, 2, 0)
    img = np.clip(img, 0, 1)
    return img

def visualize_prediction(model, dataloader, device):
    model.eval()
    cmap = plt.get_cmap('tab10')

    with torch.no_grad():
        for batch in dataloader:
            images, masks = batch
            images = images.to(device)
            masks = masks.to(device)

            outputs = model(images)
            preds = torch.argmax(outputs, dim=1)

            batch_size = images.size(0)

            for i in range(batch_size):
                image_np = denormalize(images[i])
                mask_np = masks[i].cpu().numpy()
                pred_np = preds[i].cpu().numpy()

                fig, axs = plt.subplots(1, 3, figsize=(12, 4))
                axs[0].imshow(image_np)
                axs[0].set_title('Input Image')
```

```
                axs[0].axis('off')

                axs[1].imshow(mask_np, cmap='tab10', vmin=0, vmax=4)
                axs[1].set_title(f'Ground Truth (Unique: {np.unique(mask_np)})')
                axs[1].axis('off')

                axs[2].imshow(pred_np, cmap='tab10', vmin=0, vmax=4)
                axs[2].set_title(f'Predicted Mask (Unique: {np.unique(pred_np)})')
                axs[2].axis('off')

                plt.tight_layout()
                plt.show()

            break




visualize_prediction(model, val_loader, device)
```


Input Image

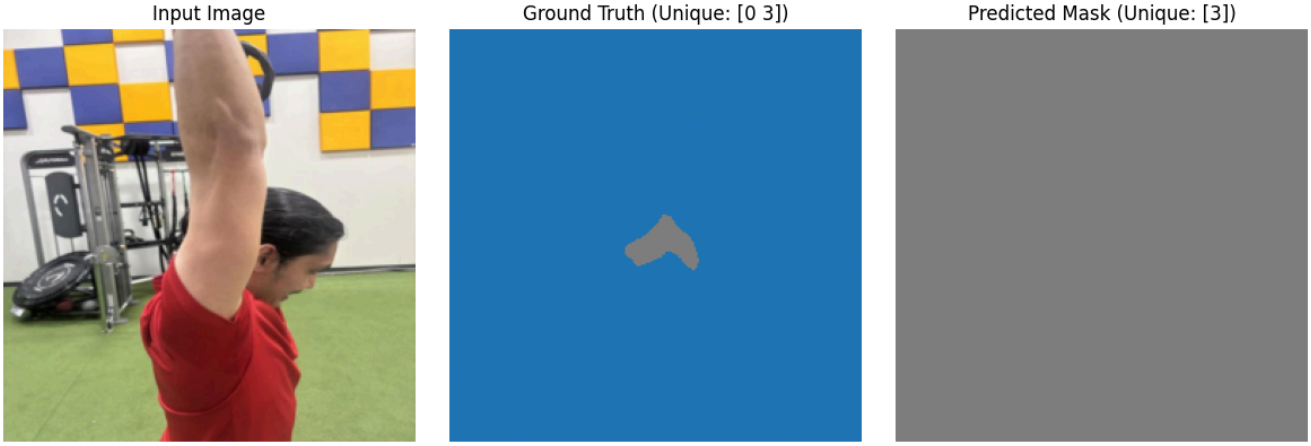

Ground Truth (Unique: [0 1])


Input Image
Ground Truth (Unique: [0 3])
Predicted Mask (Unique: [3])

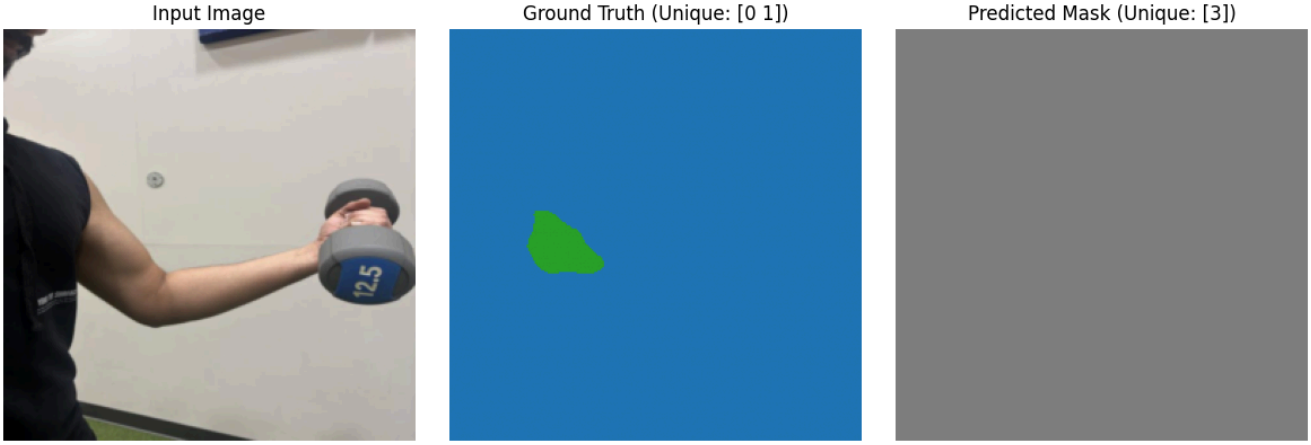

Input Image
Ground Truth (Unique: [0 3])
Predicted Mask (Unique: [3])


Input Image
Ground Truth (Unique: [0 3])
Predicted Mask (Unique: [3])

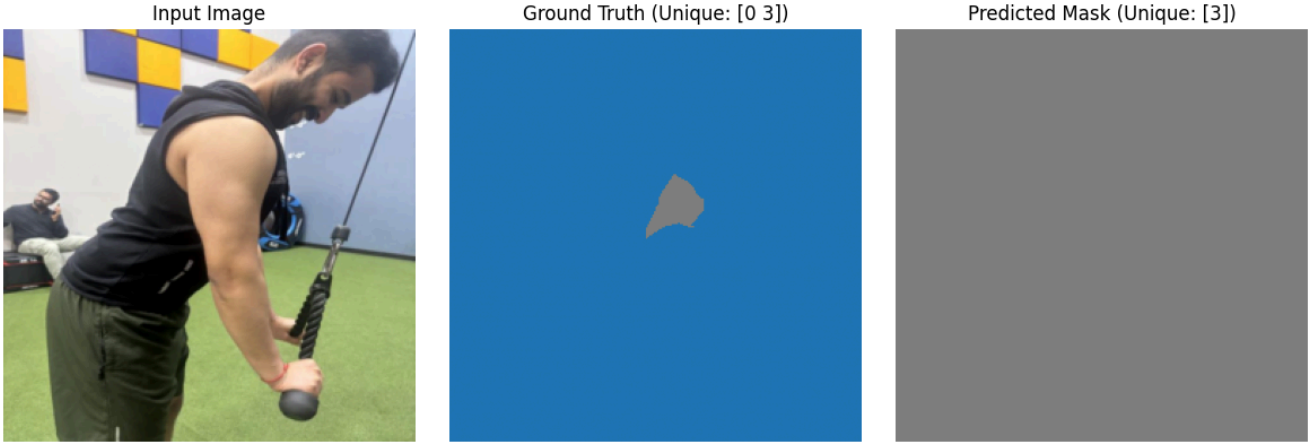
Input Image | Ground Truth (Unique: [0 3]) | Predicted Mask (Unique: [3])



Input Image | Ground Truth (Unique: [0 1]) | Predicted Mask (Unique: [3])



Input Image | Ground Truth (Unique: [0 3]) | Predicted Mask (Unique: [3])



Input Image | Ground Truth (Unique: [0 1]) | Predicted Mask (Unique: [3])