```
!pip install -q --upgrade pip
!pip install -q torch torchvision --extra-index-url
https://download.pytorch.org/whl/cu116
!pip install -q segmentation-models-pytorch==0.3.0 timm efficientnet-
pytorch
!pip install -q albumentations==1.3.1
!pip install -q onnx onnxruntime onnx-tf
!pip install -q tensorflowjs

from google.colab import drive
drive.mount('/content/drive', force_remount=True)

print("Dependencies installed and Drive mounted.")
```

```
Mounted at /content/drive
Dependencies installed and Drive mounted.
```

```
zip_path = '/content/drive/MyDrive/DriSegs.zip'

!unzip -q {zip_path} -d /content/DriSegs

!ls -R /content/DriSegs | head -40
```

```
/content/DriSegs:
DriSegs
__MACOSX

/content/DriSegs/DriSegs:
meta.json
test
train
val

/content/DriSegs/DriSegs/test:
ann
img

/content/DriSegs/DriSegs/test/ann:
cac32276-a70feba7.jpg.json
cb879996-ecab29b0.jpg.json
cbb0e311-1082d7ff.jpg.json
cc236fd9-90d374e4.jpg.json
cc388c9e-b41eace6.jpg.json
ccd0ec33-3386cee5.jpg.json
cd09a73f-5f6b9212.jpg.json
cd0c8cf0-d78ca88f.jpg.json
cd59c8c9-bc632097.jpg.json
cd9b086a-12d43303.jpg.json
cdcc30a8-a4132413.jpg.json
cdcd1c99-bab60519.jpg.json
cddf367c-8c38cdcf.jpg.json
```

```
ce20f7fa-bafb9098.jpg.json
ce32682a-7e0156ec.jpg.json
ce786ed1-7e196be2.jpg.json
ced813cc-480d1c0c.jpg.json
cf1038c6-38badb5e.jpg.json
cf3fa252-64b32820.jpg.json
cf7f5c97-7e06dad1.jpg.json
cf917d34-acba0c6f.jpg.json
cf9c863a-9c278f78.jpg.json
cfba56eb-39479ec3.jpg.json
cfcbcf90-7f9c6a78.jpg.json
cffe8287-a7e4d3f5.jpg.json
```

```
import os, json, math
from pathlib import Path
import numpy as np
import cv2
from glob import glob
from tqdm.auto import tqdm

BASE_DIR = '/content/DriSegs/DriSegs'
OUT_MASK_ROOT = os.path.join(BASE_DIR, 'masks')
os.makedirs(OUT_MASK_ROOT, exist_ok=True)

splits = ['train', 'val', 'test']

def try_get_polygons_from_json(j):
    polys = []

    def accept_label(lbl):
        txt = str(lbl).lower()
        keywords = ['drivable', 'drivable_area', 'driveable',
'directly-drivable', 'alternatively-drivable', 'direct',
'alternative']
        for k in keywords:
            if k in txt:
                return True
        return False

    if isinstance(j, dict):
        if 'labels' in j and isinstance(j['labels'], list):
            for obj in j['labels']:
                cat = obj.get('category') or obj.get('category_name')
or obj.get('label') or ''
                if accept_label(cat):
                    if 'poly2d' in obj and isinstance(obj['poly2d'],
list):
                        for polyobj in obj['poly2d']:
                            pts = polyobj.get('vertices') or
polyobj.get('poly') or polyobj.get('points')
```

```
                            if pts:
                                polys.append(pts)
                    elif 'poly' in obj and isinstance(obj['poly'],
list):
                        polys.append(obj['poly'])
                    elif 'polyline' in obj and
isinstance(obj['polyline'], list):
                        polys.append(obj['polyline'])
                    elif 'box2d' in obj and isinstance(obj['box2d'],
dict):
                        b = obj['box2d']
                        p = [(b['x1'], b['y1']), (b['x2'], b['y1']),
(b['x2'], b['y2']), (b['x1'], b['y2'])]
                        polys.append(p)

        if 'drivable_area' in j:
            da = j['drivable_area']
            if isinstance(da, list):
                for item in da:
                    if isinstance(item, dict) and 'polygon' in item:
                        polys.append(item['polygon'])
                    elif isinstance(item, list):
                        polys.append(item)
            elif isinstance(da, dict):
                if 'polygons' in da and isinstance(da['polygons'],
list):
                    for p in da['polygons']:
                        polys.append(p)

        for key in ('objects', 'annotations', 'features'):
            if key in j and isinstance(j[key], list):
                for obj in j[key]:
                    cat = obj.get('category') or obj.get('label') or
obj.get('type') or ''
                    if accept_label(cat):
                        if 'polygon' in obj and
isinstance(obj['polygon'], list):
                            polys.append(obj['polygon'])
                        if 'segmentation' in obj and
isinstance(obj['segmentation'], list):
                            seg = obj['segmentation']
                            if seg and isinstance(seg[0], list):
                                for s in seg:
                                    coords = [(s[i], s[i+1]) for i in
range(0, len(s), 2)]
                                    polys.append(coords)
                            else:
                                coords = [(seg[i], seg[i+1]) for i in
range(0, len(seg), 2)]
```

```
                                polys.append(coords)

    def find_polys_recursive(o):
        found = []
        if isinstance(o, dict):
            for v in o.values():
                found += find_polys_recursive(v)
        elif isinstance(o, list):
            if len(o) >= 3 and all(isinstance(pt, (list, tuple)) and
len(pt) >= 2 and all(isinstance(x, (int,float)) for x in pt[:2]) for
pt in o):
                found.append(o)
            else:
                for item in o:
                    found += find_polys_recursive(item)
        return found

    if not polys:
        polys = find_polys_recursive(j)

    norm_polys = []
    for p in polys:
        try:
            pts = []
            for pt in p:
                if isinstance(pt, dict):
                    x = pt.get('x') or pt.get('cx') or pt.get('X') or
0
                    y = pt.get('y') or pt.get('cy') or pt.get('Y') or
0
                elif isinstance(pt, (list, tuple)) and len(pt) >= 2:
                    x, y = pt[0], pt[1]
                else:
                    continue
                pts.append((float(x), float(y)))
            if len(pts) >= 3:
                norm_polys.append(pts)
        except Exception:
            continue

    return norm_polys

def rasterize_polygons(polygons, image_width, image_height):
    mask = np.zeros((image_height, image_width), dtype=np.uint8)
    for poly in polygons:
        coords = np.array(poly, dtype=np.float32)
        if coords.max() <= 1.01:
            coords[:,0] *= image_width
            coords[:,1] *= image_height
        pts = np.round(coords).astype(np.int32)
        pts[:,0] = np.clip(pts[:,0], 0, image_width-1)
        pts[:,1] = np.clip(pts[:,1], 0, image_height-1)
        if pts.shape[0] >= 3:
            cv2.fillPoly(mask, [pts], 255)
    return mask

summary = {}
for split in splits:
    split_img_dir = os.path.join(BASE_DIR, split, 'img')
    split_ann_dir = os.path.join(BASE_DIR, split, 'ann')
    split_out_ann_img = os.path.join(split_ann_dir, 'img')
    os.makedirs(split_out_ann_img, exist_ok=True)
    count = 0
    failed = 0
    for ann_file in tqdm(sorted(os.listdir(split_ann_dir)),
desc=f"Processing {split}"):
        if not ann_file.lower().endswith('.json'):
            continue
        ann_path = os.path.join(split_ann_dir, ann_file)
        try:
            with open(ann_path, 'r') as f:
                j = json.load(f)
        except Exception:
            failed += 1
            continue

        base = ann_file
        if base.endswith('.json'):
            base = base[:-5]
        img_name = base
        img_path_jpg = os.path.join(split_img_dir, img_name)
        if not os.path.exists(img_path_jpg):
            if not img_name.lower().endswith('.jpg') and not
img_name.lower().endswith('.png'):
                if os.path.exists(os.path.join(split_img_dir, img_name
+ '.jpg')):
                    img_path = os.path.join(split_img_dir, img_name +
'.jpg')
                elif os.path.exists(os.path.join(split_img_dir,
img_name + '.png')):
                    img_path = os.path.join(split_img_dir, img_name +
'.png')
                else:
                    candidates = [p for p in os.listdir(split_img_dir)
if os.path.splitext(p)[0] == os.path.splitext(img_name)[0]]
                    if candidates:
                        img_path = os.path.join(split_img_dir,
candidates[0])
                    else:
                        img_path = None
            else:
                img_path = None
        else:
            img_path = img_path_jpg

        if img_path is None or not os.path.exists(img_path):
            failed += 1
            continue

        im = cv2.imread(img_path)
        if im is None:
            failed += 1
            continue
        h, w = im.shape[:2]

        polys = try_get_polygons_from_json(j)
        if not polys:
            found_any = []
            def search_for_drivable(o):
                if isinstance(o, dict):
                    for k,v in o.items():
                        if 'drivable' in str(k).lower():
                            pts =
try_get_polygons_from_json({'labels':[v]})
                            if pts:
                                found_any.extend(pts)
                        search_for_drivable(v)
                elif isinstance(o, list):
                    for e in o:
                        search_for_drivable(e)
            search_for_drivable(j)
            if found_any:
                polys = found_any

        if not polys:
            failed += 1
            continue

        mask = rasterize_polygons(polys, w, h)

        mask_name = os.path.splitext(os.path.basename(img_path))[0] +
'.png'
        out_mask_path1 = os.path.join(split_out_ann_img, mask_name)
        out_mask_path2 = os.path.join(OUT_MASK_ROOT, mask_name)
        cv2.imwrite(out_mask_path1, mask)
        cv2.imwrite(out_mask_path2, mask)
        count += 1

    summary[split] = {'converted': count, 'failed': failed}
```

```
print("Conversion summary:", summary)
print("Masks saved in per-split ann/img/ and consolidated:",
OUT_MASK_ROOT)

vis_dir = os.path.join(BASE_DIR, 'conversion_preview')
os.makedirs(vis_dir, exist_ok=True)
for split in splits:
    split_img_dir = os.path.join(BASE_DIR, split, 'img')
    split_mask_dir = os.path.join(BASE_DIR, split, 'ann', 'img')
    saved = 0
    for imgfile in sorted(os.listdir(split_img_dir)):
        base = os.path.splitext(imgfile)[0]
        maskfile = os.path.join(split_mask_dir, base + '.png')
        imgpath = os.path.join(split_img_dir, imgfile)
        if os.path.exists(maskfile):
            img = cv2.imread(imgpath)
            mask = cv2.imread(maskfile, cv2.IMREAD_GRAYSCALE)
            overlay = img.copy()
            overlay[mask==255] = [0,255,0]
            preview = cv2.addWeighted(img, 0.6, overlay, 0.4, 0)
            cv2.imwrite(os.path.join(vis_dir,
f'{split}_{base}_preview.png'), preview)
            saved += 1
        if saved >= 3:
            break

print("Saved preview images in", vis_dir)
```

{"model_id":"e9eb1026f6174aa6b4aef24e9e9b8199","version_major":2,"version_minor":0}

{"model_id":"a47248e98c2b487f91a35ac96f3746e8","version_major":2,"version_minor":0}

{"model_id":"edeb113b9db24fa2b6d82cb9b034484e","version_major":2,"version_minor":0}

```
Conversion summary: {'train': {'converted': 666, 'failed': 33}, 'val':
{'converted': 107, 'failed': 2}, 'test': {'converted': 0, 'failed':
199}}
Masks saved in per-split ann/img/ and consolidated:
/content/DriSegs/DriSegs/masks
Saved preview images in /content/DriSegs/DriSegs/conversion_preview
```

```
import os, random, json
from pathlib import Path
import cv2
import numpy as np
import torch
import torch.nn as nn
```

```
from torch.utils.data import Dataset, DataLoader
from torch.cuda.amp import autocast, GradScaler
import albumentations as A
from albumentations.pytorch import ToTensorV2
from tqdm.auto import tqdm
import segmentation_models_pytorch as smp
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

class Config:
    DRIVE_BASE = '/content/DriSegs/DriSegs'
    IMAGE_DIR = f'{DRIVE_BASE}/train/img'
    MASK_DIR  = f'{DRIVE_BASE}/train/ann/img'
    OUTPUT_DIR = f'{DRIVE_BASE}/outputs'

    ENCODER = 'timm-efficientnet-b0'
    ENCODER_WEIGHTS = 'imagenet'
    INPUT_SIZE = (640, 640)
    NUM_CLASSES = 1
    BATCH_SIZE = 4
    NUM_EPOCHS = 20
    LEARNING_RATE = 1e-4
    WEIGHT_DECAY = 1e-5
    EARLY_STOPPING_PATIENCE = 10
    USE_AMP = True

    VAL_SPLIT = 0.15
    TEST_SPLIT = 0.1
    RANDOM_SEED = 42
    DEVICE = torch.device('cuda' if torch.cuda.is_available() else
'cpu')

    EXPORT_ONNX = True
    EXPORT_TFLITE = False
    NUM_WORKERS = 2 if torch.cuda.is_available() else 0

    @classmethod
    def create_dirs(cls):
        os.makedirs(cls.OUTPUT_DIR, exist_ok=True)
        os.makedirs(f'{cls.OUTPUT_DIR}/checkpoints', exist_ok=True)
        os.makedirs(f'{cls.OUTPUT_DIR}/logs', exist_ok=True)
        os.makedirs(f'{cls.OUTPUT_DIR}/exports', exist_ok=True)
        os.makedirs(f'{cls.OUTPUT_DIR}/visualizations', exist_ok=True)

Config.create_dirs()
torch.manual_seed(Config.RANDOM_SEED)
random.seed(Config.RANDOM_SEED)
np.random.seed(Config.RANDOM_SEED)

print("Device:", Config.DEVICE)
print("Dataset base:", Config.DRIVE_BASE)
print("Encoder:", Config.ENCODER)
```

```
Device: cuda
Dataset base: /content/DriSegs/DriSegs
Encoder: timm-efficientnet-b0
```

```
class DrivableAreaDataset(torch.utils.data.Dataset):
    def __init__(self, image_paths, mask_paths, transform=None):
        assert len(image_paths) == len(mask_paths), "Images and masks
must be equal length"
        self.image_paths = image_paths
        self.mask_paths = mask_paths
        self.transform = transform

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

    def __getitem__(self, idx):
        img_p, m_p = self.image_paths[idx], self.mask_paths[idx]

        image = cv2.imread(img_p)
        if image is None:
            raise RuntimeError(f"Image not found: {img_p}")
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        mask = cv2.imread(m_p, cv2.IMREAD_GRAYSCALE)
        if mask is None:
            raise RuntimeError(f"Mask not found: {m_p}")
        mask = (mask > 127).astype("uint8")

        if self.transform:
            data = self.transform(image=image, mask=mask)
            image, mask = data["image"], data["mask"]
        else:
            image = torch.from_numpy(image.transpose(2, 0, 1)).float()
/ 255.0
            mask = torch.from_numpy(mask).unsqueeze(0).float()

        if isinstance(mask, np.ndarray):
            mask = torch.fromnumpy(mask)
        if mask.ndim == 2:
            mask = mask.unsqueeze(0)
        return image.float(), mask.float()

def get_training_augmentation():
    return A.Compose([
        A.Resize(*Config.INPUT_SIZE),
        A.HorizontalFlip(p=0.5),
        A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.15,
rotate_limit=15, p=0.5),
        A.RandomBrightnessContrast(brightness_limit=0.3,
contrast_limit=0.3, p=0.7),
        A.CLAHE(p=0.2),
        A.OneOf([
            A.MotionBlur(blur_limit=5),
            A.GaussNoise(var_limit=(10, 50)),
            A.GaussianBlur(3)
        ], p=0.3),
        A.Normalize(mean=(0.485, 0.456, 0.406),
                    std=(0.229, 0.224, 0.225)),
        ToTensorV2()
    ], additional_targets={"mask": "mask"})

def get_validation_augmentation():
    return A.Compose([
        A.Resize(*Config.INPUT_SIZE),
        A.Normalize(mean=(0.485, 0.456, 0.406),
                    std=(0.229, 0.224, 0.225)),
        ToTensorV2()
    ], additional_targets={"mask": "mask"})

class EfficientNetUNet(nn.Module):
    def __init__(self, encoder_name=Config.ENCODER,
                 encoder_weights=Config.ENCODER_WEIGHTS,
                 num_classes=Config.NUM_CLASSES):
        super().__init__()
        self.model = smp.Unet(
            encoder_name=encoder_name,
            encoder_weights=encoder_weights,
            in_channels=3,
            classes=num_classes,
            activation=None
        )

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

class DiceLoss(nn.Module):
    def __init__(self, smooth=1.0):
        super().__init__()
        self.smooth = smooth
    def forward(self, pred, target):
        pred = torch.sigmoid(pred)
        pred = pred.view(-1)
        target = target.view(-1)
        inter = (pred * target).sum()
        dice = (2.0 * inter + self.smooth) / (pred.sum() +
target.sum() + self.smooth)
        return 1.0 - dice
```

```
class CombinedLoss(nn.Module):
    def __init__(self, bce_weight=0.5, dice_weight=0.5):
        super().__init__()
        self.bce = nn.BCEWithLogitsLoss()
        self.dice = DiceLoss()
        self.bce_weight = bce_weight
        self.dice_weight = dice_weight
    def forward(self, pred, target):
        return self.bce_weight*self.bce(pred,target) +
self.dice_weight*self.dice(pred,target)

def calculate_metrics(pred, target, threshold=0.5):
    pred_prob = torch.sigmoid(pred)
    pred_bin = (pred_prob > threshold).float()
    p, t = pred_bin.view(-1), target.view(-1).float()
    inter = (p*t).sum().item()
    union = p.sum().item() + t.sum().item() - inter
    iou = (inter+1e-7)/(union+1e-7)
    tp, fp, fn = inter, p.sum().item()-inter, t.sum().item()-inter
    tn = len(p)-tp-fp-fn
    precision = (tp+1e-7)/(tp+fp+1e-7)
    recall = (tp+1e-7)/(tp+fn+1e-7)
    f1 = 2*(precision*recall)/(precision+recall+1e-7)
    acc = (tp+tn)/len(p)
    return {"iou":float(iou),"precision":float(precision),

"recall":float(recall),"f1":float(f1),"accuracy":float(acc)}

def prepare_data_from_splits(base_dir):
    def find_pairs(split):
        img_dir = os.path.join(base_dir, split, "img")
        ann_dir = os.path.join(base_dir, split, "ann")
        ann_img_dir = os.path.join(ann_dir, "img")
        imgs = sorted([os.path.join(img_dir, f)
                       for f in os.listdir(img_dir)
                       if
f.lower().endswith((".jpg",".jpeg",".png"))])
        masks = []
        for p in imgs:
            base = os.path.splitext(os.path.basename(p))[0]
            found = None
            for d in (ann_img_dir, ann_dir):
                if os.path.isdir(d):
                    for ext in (".png",".jpg",".jpeg"):
                        f = os.path.join(d, base+ext)
                        if os.path.exists(f):
                            found = f; break
                if found: break
            masks.append(found)
        valid = [(i,m) for i,m in zip(imgs,masks) if m is not None]
        imgs_ok, masks_ok = zip(*valid) if valid else ([],[])
        print(f"{split}: {len(imgs_ok)} valid pairs (missing 
{len(imgs)-len(imgs_ok)})")
        return list(imgs_ok), list(masks_ok)
    train_imgs, train_masks = find_pairs("train")
    val_imgs, val_masks     = find_pairs("val")
    test_imgs, test_masks   = find_pairs("test")
    return train_imgs, train_masks, val_imgs, val_masks, test_imgs, 
test_masks

train_imgs, train_masks, val_imgs, val_masks, test_imgs, test_masks = 
prepare_data_from_splits(Config.DRIVE_BASE)
```

```
train: 666 valid pairs (missing 33)
val: 107 valid pairs (missing 2)
test: 0 valid pairs (missing 199)
```

```
train_dataset = DrivableAreaDataset(train_imgs, train_masks, 
transform=get_training_augmentation())
val_dataset   = DrivableAreaDataset(val_imgs, val_masks, 
transform=get_validation_augmentation())
test_dataset  = DrivableAreaDataset(test_imgs, test_masks, 
transform=get_validation_augmentation())

train_loader = DataLoader(train_dataset, batch_size=Config.BATCH_SIZE, 
shuffle=True,
                          num_workers=Config.NUM_WORKERS, 
pin_memory=True)
val_loader   = DataLoader(val_dataset, batch_size=Config.BATCH_SIZE, 
shuffle=False,
                          num_workers=Config.NUM_WORKERS, 
pin_memory=True)
test_loader  = DataLoader(test_dataset, batch_size=Config.BATCH_SIZE, 
shuffle=False,
                          num_workers=Config.NUM_WORKERS, 
pin_memory=True)

model = EfficientNetUNet().to(Config.DEVICE)
print("Model created — params:", sum(p.numel() for p in 
model.parameters()))
criterion = CombinedLoss()
optimizer = torch.optim.AdamW(model.parameters(), 
lr=Config.LEARNING_RATE, weight_decay=Config.WEIGHT_DECAY)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 
mode="min", factor=0.5, patience=5, verbose=True)
```

```
/usr/local/lib/python3.12/dist-packages/albumentations/augmentations/
blur/transforms.py:184: UserWarning: blur_limit and sigma_limit 
minimum value can not be both equal to 0. blur_limit minimum value 
changed to 3.
```

```
  warnings.warn(
Downloading:
"https://github.com/rwightman/pytorch-image-models/releases/download/
v0.1-weights/tf_efficientnet_b0_aa-827b6e33.pth" to
/root/.cache/torch/hub/checkpoints/tf_efficientnet_b0_aa-827b6e33.pth
100%|██████████| 20.4M/20.4M [00:02<00:00, 7.49MB/s]

Model created — params: 6251469

/usr/local/lib/python3.12/dist-packages/torch/optim/
lr_scheduler.py:60: UserWarning: The verbose parameter is deprecated.
Please use get_last_lr() to access the learning rate.
  warnings.warn(
```

```python
import torch, math
print("Train samples:", len(train_imgs), "Val samples:",
len(val_imgs))
batch = next(iter(train_loader))
images, masks = batch
print("images:", images.shape, "masks:", masks.shape)
images = images.to(Config.DEVICE)
masks = masks.to(Config.DEVICE)

model.eval()
with torch.no_grad():
    out = model(images)
print("output shape:", out.shape)

criterion = CombinedLoss()
loss = criterion(out, masks)
metrics = calculate_metrics(out, masks)
print(f"sample loss: {loss.item():.4f}")
print("sample metrics:", metrics)

assert images.shape[1] == 3, "Images should have 3 channels"
assert out.shape[1] == Config.NUM_CLASSES, "Output channel mismatch"
print("SMOKE TEST PASSED")
```

```
Train samples: 666 Val samples: 107
images: torch.Size([4, 3, 640, 640]) masks: torch.Size([4, 1, 640,
640])
output shape: torch.Size([4, 1, 640, 640])
sample loss: 0.7560
sample metrics: {'iou': 0.051153543470375325, 'precision':
0.12919536221827263, 'recall': 0.0780714970555217, 'f1':
0.09732834771495784, 'accuracy': 0.7483251953125}
SMOKE TEST PASSED
```

```python
class Trainer:
    def __init__(self, model, train_loader, val_loader, criterion,
optimizer):
        self.model = model
        self.train_loader = train_loader
        self.val_loader = val_loader
        self.criterion = criterion
        self.optimizer = optimizer
        self.scaler = GradScaler() if Config.USE_AMP and
Config.DEVICE.type=="cuda" else None
        self.best_val_iou, self.patience = 0.0, 0

    def train_epoch(self):
        self.model.train()
        losses=[]
        for imgs,masks in tqdm(self.train_loader,desc="train"):
            imgs,masks=imgs.to(Config.DEVICE),masks.to(Config.DEVICE)
            self.optimizer.zero_grad()
            if self.scaler:
                with autocast():
                    out=self.model(imgs)
                    loss=self.criterion(out,masks)
                self.scaler.scale(loss).backward()
                self.scaler.step(self.optimizer)
                self.scaler.update()
            else:
                out=self.model(imgs);loss=self.criterion(out,masks)
                loss.backward();self.optimizer.step()
            losses.append(loss.item())
        return np.mean(losses)

    def validate_epoch(self):
        self.model.eval();losses=[];metrics=[]
        with torch.no_grad():
            for imgs,masks in tqdm(self.val_loader,desc="val"):

imgs,masks=imgs.to(Config.DEVICE),masks.to(Config.DEVICE)
                out=self.model(imgs)
                loss=self.criterion(out,masks)
                losses.append(loss.item())
                metrics.append(calculate_metrics(out,masks))
        avg={k:float(np.mean([m[k] for m in metrics])) for k in
metrics[0]} if metrics else {}
        return np.mean(losses),avg

    def train(self,epochs,scheduler=None):
        hist={"train_loss":[],"val_loss":[],"val_iou":[],"val_f1":[]}
        for ep in range(epochs):
            print(f"Epoch {ep+1}/{epochs}")
            tr=self.train_epoch()
            vl,vm=self.validate_epoch()
            hist["train_loss"].append(tr);hist["val_loss"].append(vl)
```

```
hist["val_iou"].append(vm.get("iou",0));hist["val_f1"].append(vm.get("
f1",0))
            print(f"Train {tr:.4f}  Val {vl:.4f}  IoU
{vm.get('iou',0):.4f}")
            if scheduler: scheduler.step(vl)
            if vm.get("iou",0)>self.best_val_iou:
                self.best_val_iou=vm["iou"];self.patience=0

torch.save({"model_state_dict":self.model.state_dict(),
                            "val_iou":vm["iou"]},

f"{Config.OUTPUT_DIR}/checkpoints/best_model.pth")
                print("Saved best model.")
            else:
                self.patience+=1
                if self.patience>=Config.EARLY_STOPPING_PATIENCE:
                    print("Early stopping.");break
        return hist

trainer=Trainer(model,train_loader,val_loader,criterion,optimizer)
history=trainer.train(Config.NUM_EPOCHS,scheduler=scheduler)
```

```
Epoch 1/20

/tmp/ipython-input-2775631090.py:8: FutureWarning:
`torch.cuda.amp.GradScaler(args...)` is deprecated. Please use
`torch.amp.GradScaler('cuda', args...)` instead.
  self.scaler = GradScaler() if Config.USE_AMP and
Config.DEVICE.type=="cuda" else None
```

{"model_id":"e3e5b3f0d98449e2be8f63238d1eecbc","version_major":2,"version_minor":0}

```
/tmp/ipython-input-2775631090.py:18: FutureWarning:
`torch.cuda.amp.autocast(args...)` is deprecated. Please use
`torch.amp.autocast('cuda', args...)` instead.
  with autocast():
```

{"model_id":"d864e3e648694d409730d9b8f1577b11","version_major":2,"version_minor":0}

```
Train 0.4992  Val 0.2949  IoU 0.6809
Saved best model.
Epoch 2/20
```

{"model_id":"4f95188f8b2b48168814af1d115cbaf9","version_major":2,"version_minor":0}

{"model_id":"45e4960e10eb4dd1bb3991323d1c5acd","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
   Exception ignored in:  <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>
 Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
  ^    self._shutdown_workers()^^^^
^^  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
^^    ^if w.is_alive():
^ ^
      File "/usr/lib/python3.12/multiprocessing/process.py", line 160,
in is_alive
      assert self._parent_pid == os.getpid(), 'can only test a child
process'^^^
^   ^^  ^^ ^ ^ ^ ^
^  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
^    ^assert self._parent_pid == os.getpid(), 'can only test a child
process'^
 ^ ^ ^ ^ ^^  ^^ ^^ ^  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^AssertionError^: ^can only test a child process^
^^Exception ignored in: <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>^
^^Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
^^    ^self._shutdown_workers()^
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
^    ^^
if w.is_alive():AssertionError
:  can only test a child process
   Exception ignored in:  <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>
^^Traceback (most recent call last):
```

```
^  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    ^self._shutdown_workers()^
^^^  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
^^^    ^if w.is_alive():

  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
      assert self._parent_pid == os.getpid(), 'can only test a child
process'
        ^ ^ ^  ^^^  ^^ ^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
^    ^assert self._parent_pid == os.getpid(), 'can only test a child
process'^^
 ^ ^ ^^  ^ ^^ ^^^^ ^^ ^ ^ ^^^^^^^^^^^^^^^^
^AssertionError: can only test a child process
^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process

Train 0.2903  Val 0.2144  IoU 0.7253
Saved best model.
Epoch 3/20
```

{"model_id":"383336c7761e47d38427d4f2a1923a4c","version_major":2,"version_minor":0}

{"model_id":"529a7015b9f945f991e969dde1d3d46f","version_major":2,"version_minor":0}

```
Train 0.2288  Val 0.1758  IoU 0.7523
Saved best model.
Epoch 4/20
```

{"model_id":"be98621dd865495b965274e6592d36c8","version_major":2,"version_minor":0}

{"model_id":"445065e5629a49e4a842ed9729189150","version_major":2,"version_minor":0}

```
Train 0.2019  Val 0.1609  IoU 0.7644
Saved best model.
Epoch 5/20
```

{"model_id":"f3dae4d2e1734ae3871bab28656ae555","version_major":2,"version_minor":0}

```
{"model_id":"050ff5829bfd428bb0e066bf74ad0c14","version_major":2,"version_minor":0}

Train 0.1824  Val 0.1488  IoU 0.7711
Saved best model.
Epoch 6/20

{"model_id":"8d8316b143424a18aaa8295ad8e4d071","version_major":2,"version_minor":0}

{"model_id":"da5ae195ed9a41d89cc2cbc5649942f8","version_major":2,"version_minor":0}

Train 0.1719  Val 0.1509  IoU 0.7635
Epoch 7/20

{"model_id":"01e3fcde8e5844e2bd6bdbe808d6f4ad","version_major":2,"version_minor":0}

{"model_id":"261223500c964bcea3f2cbe072cfd2b5","version_major":2,"version_minor":0}

Train 0.1632  Val 0.1444  IoU 0.7724
Saved best model.
Epoch 8/20

{"model_id":"06adcd0f97fe47f6ba22b69d4affca8e","version_major":2,"version_minor":0}

{"model_id":"d15a33e04dd04be69bf06f2508435851","version_major":2,"version_minor":0}

Train 0.1566  Val 0.1392  IoU 0.7794
Saved best model.
Epoch 9/20

{"model_id":"19e1feae54bc4e18b2478ea1d9d28859","version_major":2,"version_minor":0}

{"model_id":"9ac05b644e9e41f4bbd6f0e40c212c15","version_major":2,"version_minor":0}

Train 0.1517  Val 0.1471  IoU 0.7681
Epoch 10/20

{"model_id":"2037d3ec43fd4e118b24a06b3ae1817f","version_major":2,"version_minor":0}

{"model_id":"a63013e96e8b41d486550e7d6e0ad4ce","version_major":2,"version_minor":0}
```

```
Train 0.1443  Val 0.1342  IoU 0.7836
Saved best model.
Epoch 11/20

{"model_id":"38a8075db739420691ce9c0866e8ee32","version_major":2,"vers
ion_minor":0}

{"model_id":"cd4ea6bf543145ab95c398cc886f8cce","version_major":2,"vers
ion_minor":0}

Train 0.1384  Val 0.1301  IoU 0.7914
Saved best model.
Epoch 12/20

{"model_id":"899d848d1000452bbb2793c4f4a54cd6","version_major":2,"vers
ion_minor":0}

{"model_id":"271b77ef0ade4e8ebb0978c8af8aa086","version_major":2,"vers
ion_minor":0}

Train 0.1319  Val 0.1373  IoU 0.7802
Epoch 13/20

{"model_id":"9cfd69b0009645c28cc0be5562c6370c","version_major":2,"vers
ion_minor":0}

{"model_id":"d168606923854e878a06555e6cc43110","version_major":2,"vers
ion_minor":0}

Train 0.1282  Val 0.1362  IoU 0.7807
Epoch 14/20

{"model_id":"20f08a59cc4f4d9ca2af9b64bf50f889","version_major":2,"vers
ion_minor":0}

{"model_id":"3e7ae865ff75448c80ad1b89522d8484","version_major":2,"vers
ion_minor":0}

Train 0.1290  Val 0.1339  IoU 0.7847
Epoch 15/20

{"model_id":"88d39f66f56a4ed784620b3bf267447e","version_major":2,"vers
ion_minor":0}

{"model_id":"0183955dffa845d2836b772463459b52","version_major":2,"vers
ion_minor":0}

Train 0.1218  Val 0.1322  IoU 0.7850
Epoch 16/20

{"model_id":"887b5e3625c4484192c06af3d7c7b706","version_major":2,"vers
ion_minor":0}
```

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1423, in _shutdown_workers
    self._pin_memory_thread.join()
  File "/usr/lib/python3.12/threading.py", line 1146, in join
    raise RuntimeError("cannot join current thread")
RuntimeError: cannot join current thread
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^Exception ignored in:
^<function _MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>

AssertionErrorTraceback (most recent call last):
:   File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
can only test a child process
Exception ignored in: Traceback (most recent call last):
<function _MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>
self._shutdown_workers()  File "/usr/local/lib/python3.12/dist-
packages/torch/utils/data/dataloader.py", line 1477, in __del__

  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
```

```
if w.is_alive():
     if w.is_alive():
            ^ ^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
^
      File "/usr/lib/python3.12/multiprocessing/process.py", line 160,
in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'assert self._parent_pid == os.getpid(), 'can only test a child
process'

                  ^ ^ ^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

AssertionErrorAssertionError: can only test a child process
: Exception ignored in: <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>can only test
a child process

Traceback (most recent call last):
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__

    Traceback (most recent call last):
self._shutdown_workers()  File "/usr/local/lib/python3.12/dist-
packages/torch/utils/data/dataloader.py", line 1477, in __del__

  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
self._shutdown_workers()
if w.is_alive():  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers

     if w.is_alive():
          ^^ ^ ^^^^^^^^^^^^^^^^
^  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    ^^assert self._parent_pid == os.getpid(), 'can only test a child
process'^^

   File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
       assert self._parent_pid == os.getpid(), 'can only test a child
process'
        ^ ^  ^^ ^ ^
```

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

AssertionError: AssertionErrorcan only test a child process
Exception ignored in: : <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>can only test
a child process
Traceback (most recent call last):

  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    Exception ignored in: self._shutdown_workers()<function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>

Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    if w.is_alive():
      self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
      if w.is_alive():
^^ ^^ ^^ ^  ^ ^ ^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
^    assert self._parent_pid == os.getpid(), 'can only test a child
process'^
  ^   ^^ ^ ^ ^assert self._parent_pid == os.getpid(), 'can only test a
child process'
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive

      ^ ^ ^ ^^ ^ ^^^ ^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^AssertionError: ^can only test a child process^^
^Exception ignored in: <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>
^Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    ^self._shutdown_workers()^
^  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
```

```
AssertionError    if w.is_alive():: can only test a child process

<function _MultiProcessingDataLoaderIter.__del__ at
0x7e32f5185080>Exception ignored in:
 Traceback (most recent call last):
    File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
        self._shutdown_workers()^
^  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
^    ^if w.is_alive():^
^  ^^ ^ ^^  ^
^  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    ^^assert self._parent_pid == os.getpid(), 'can only test a child
process'^
 ^^^  ^^ ^ ^^
   File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
      assert self._parent_pid == os.getpid(), 'can only test a child
process'
  ^ ^  ^ ^ ^ ^ ^ ^ ^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError^: can only test a child process^
^^^Exception ignored in: <function
_MultiProcessingDataLoaderIter.__del__ at 0x7e32f5185080>^^
^^Traceback (most recent call last):

  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    AssertionError: self._shutdown_workers()
can only test a child process  File "/usr/local/lib/python3.12/dist-
packages/torch/utils/data/dataloader.py", line 1460, in
_shutdown_workers

    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"9a0a7806961547679073704340fed34bc","version_major":2,"vers
ion_minor":0}

```
Train 0.1189  Val 0.1340  IoU 0.7812
Epoch 17/20

{"model_id":"698d6d77b4464d1bbf19b8c800475cb0","version_major":2,"vers
ion_minor":0}

{"model_id":"9f16915fc1b946f59d747c0c679ed1ca","version_major":2,"vers
ion_minor":0}

Train 0.1123  Val 0.1322  IoU 0.7859
Epoch 18/20

{"model_id":"6e7699a56c5e47d898fe8d97e95820e7","version_major":2,"vers
ion_minor":0}

{"model_id":"f526cfe0ec2f4c63972699e65672d6b4","version_major":2,"vers
ion_minor":0}

Train 0.1157  Val 0.1309  IoU 0.7886
Epoch 19/20

{"model_id":"3f52b581105945fa83f2d3e6f2c535bb","version_major":2,"vers
ion_minor":0}

Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
```

```
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
```

```
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x7e32f5185080>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1477, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1460, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"006ff79c13084e00b5daf2a02593a7fd","version_major":2,"vers
ion_minor":0}

```
Train 0.1049  Val 0.1297  IoU 0.7873
Epoch 20/20
```

{"model_id":"5a59ac683ca24a93a93ed1d5d2cdc0af","version_major":2,"vers
ion_minor":0}

{"model_id":"4d467388cdd14eda9b80305bb2b87a67","version_major":2,"vers
ion_minor":0}

```
Train 0.1044  Val 0.1425  IoU 0.7683
```

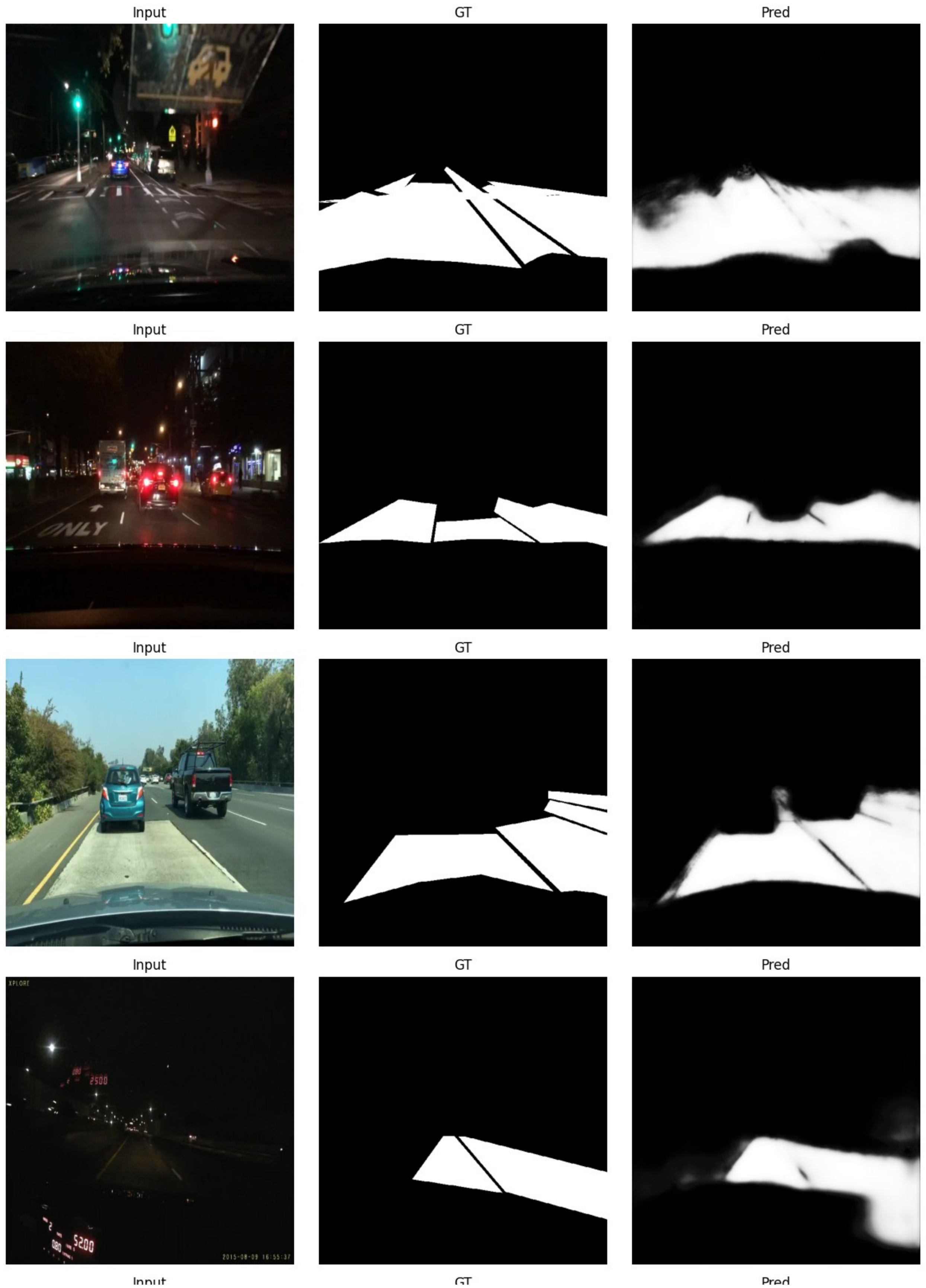
```python
ckpt=os.path.join(Config.OUTPUT_DIR,"checkpoints","best_model.pth")
if os.path.exists(ckpt):
    sd=torch.load(ckpt,map_location=Config.DEVICE)
    model.load_state_dict(sd["model_state_dict"])
    print(f"Loaded best model (val IoU={sd.get('val_iou',0):.4f})")
else: print("No checkpoint found.")

def evaluate_model(model,loader,save_vis=True,n_vis=6):
    model.eval();metrics_all=[];vis=[]
    with torch.no_grad():
        for imgs,masks in tqdm(loader,desc="test"):
            imgs,masks=imgs.to(Config.DEVICE),masks.to(Config.DEVICE)
            out=model(imgs)
            metrics_all.append(calculate_metrics(out,masks))
            if save_vis and len(vis)<n_vis:
                p=torch.sigmoid(out[0]).cpu().numpy()[0]
                img=imgs[0].cpu().permute(1,2,0).numpy()
                img=np.clip(img* np.array([0.229,0.224,0.225])
+np.array([0.485,0.456,0.406]),0,1)
                vis.append((img,masks[0].cpu().numpy()[0],p))
    avg={k:float(np.mean([m[k] for m in metrics_all])) for k in
metrics_all[0]}
    print("Test metrics:",avg)
    if vis:
        fig,axs=plt.subplots(len(vis),3,figsize=(12,4*len(vis)))
        for i,(img,m,pred) in enumerate(vis):

axs[i,0].imshow(img);axs[i,0].set_title("Input");axs[i,1].imshow(m,cma
p="gray")
```

```
axs[i,1].set_title("GT");axs[i,2].imshow(pred,cmap="gray");axs[i,2].se
t_title("Pred")
            [ax.axis("off") for ax in axs[i]]
        plt.tight_layout();plt.show()
    return avg

test_metrics=evaluate_model(model,val_loader,save_vis=True)
```

```
/tmp/ipython-input-2843455294.py: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.
  sd=torch.load(ckpt,map_location=Config.DEVICE)

Loaded best model (val IoU=0.7914)
```

{"model_id":"03460c7a5b5e4576b9f5e2c6d6ecbbd2","version_major":2,"vers
ion_minor":0}

```
Test metrics: {'iou': 0.7914311782306109, 'precision':
0.8543736328978936, 'recall': 0.9145017156009648, 'f1':
0.8819891070484843, 'accuracy': 0.9578896228178048}
```


Input
GT
Pred
Input
GT
Pred
ONLY
Input
GT
Pred
Input
GT
Pred
XPLORE
2500
5200
2015-08-09 16:55:37
Input
GT
Pred

```
def
export_model(model,export_dir,input_shape=(1,3,*Config.INPUT_SIZE)):
    os.makedirs(export_dir,exist_ok=True)
    model.eval()
    dummy=torch.randn(input_shape).to(Config.DEVICE)
    onnx_path=os.path.join(export_dir,"model.onnx")
    torch.onnx.export(model,dummy,onnx_path,export_params=True,
                      opset_version=11,do_constant_folding=True,
                      input_names=["input"],output_names=["output"],
                      dynamic_axes={"input":{0:"batch"},"output":
{0:"batch"}})
    print("Saved ONNX:",onnx_path)
    import onnx
    onnx.checker.check_model(onnx.load(onnx_path))
    print("ONNX verified.")
export_model(model,os.path.join(Config.OUTPUT_DIR,"exports"))
```

```
/usr/local/lib/python3.12/dist-packages/segmentation_models_pytorch/
base/model.py:16: TracerWarning: Converting a tensor to a Python
boolean might cause the trace to be incorrect. We can't record the
data flow of Python values, so this value will be treated as a
constant in the future. This means that the trace might not generalize
to other inputs!
  if h % output_stride != 0 or w % output_stride != 0:

Saved ONNX: /content/DriSegs/DriSegs/outputs/exports/model.onnx
ONNX verified.
```

```
def predict_single_image(model,image_path,threshold=0.5):
    model.eval()
    img=cv2.imread(image_path);img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    h,w=img.shape[:2]
    aug=get_validation_augmentation();augmented=aug(image=img)
    inp=augmented["image"].unsqueeze(0).to(Config.DEVICE)
    with torch.no_grad():
        out=model(inp);pred=torch.sigmoid(out).cpu().numpy()[0,0]
    pred=cv2.resize(pred,
(w,h));binary=(pred>threshold).astype("uint8")
    return pred,binary

def visualize_prediction(img_path,model):
    img=cv2.imread(img_path);img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    pred,pbin=predict_single_image(model,img_path)
    overlay=img.copy();overlay[pbin==1]=[0,255,0]
    blend=cv2.addWeighted(img,0.6,overlay,0.4,0)
    fig,ax=plt.subplots(1,3,figsize=(15,5))

ax[0].imshow(img);ax[0].set_title("Original");ax[1].imshow(pred,cmap="
hot")
```

```
ax[1].set_title("Probability");ax[2].imshow(blend);ax[2].set_title("Ov
erlay")
    [a.axis("off") for a in ax];plt.show()

ckpt = os.path.join(Config.OUTPUT_DIR, "checkpoints",
"best_model.pth")
if os.path.exists(ckpt):
    sd = torch.load(ckpt, map_location=Config.DEVICE)
    model.load_state_dict(sd["model_state_dict"])
    print("Loaded best model (val_iou
{:.4f})".format(sd.get("val_iou",0)))
else:
    print("No best_model.pth found — fine-tuning from current
weights.")

for p in model.parameters():
    p.requires_grad = True

enc_params = []
dec_params = []
for name, p in model.named_parameters():
    if "encoder" in name:
        enc_params.append(p)
    else:
        dec_params.append(p)

opt = torch.optim.AdamW([
    {"params": enc_params, "lr": 1e-5},
    {"params": dec_params, "lr": 1e-4}
], weight_decay=Config.WEIGHT_DECAY)

sched = torch.optim.lr_scheduler.ReduceLROnPlateau(opt, mode="min",
factor=0.5, patience=3, verbose=True)

from copy import deepcopy
best_iou = 0.0
ft_epochs = 12
criterion = CombinedLoss()
for ep in range(ft_epochs):
    print(f"FT Epoch {ep+1}/{ft_epochs}")
    model.train()
    tr_losses=[]
    for imgs, masks in tqdm(train_loader, desc="ft-train"):
        imgs, masks = imgs.to(Config.DEVICE), masks.to(Config.DEVICE)
        opt.zero_grad()
        out = model(imgs)
        loss = criterion(out, masks)
        loss.backward()
        opt.step()
        tr_losses.append(loss.item())
    model.eval()
    val_losses=[]; metrics=[]
    with torch.no_grad():
        for imgs, masks in tqdm(val_loader, desc="ft-val"):
            imgs, masks = imgs.to(Config.DEVICE),
masks.to(Config.DEVICE)
            out = model(imgs)
            val_losses.append(criterion(out, masks).item())
            metrics.append(calculate_metrics(out, masks))
    val_loss = float(np.mean(val_losses))
    val_iou = float(np.mean([m["iou"] for m in metrics]))
    print(f"FT epoch {ep+1} train_loss {np.mean(tr_losses):.4f}
val_loss {val_loss:.4f} val_iou {val_iou:.4f}")
    sched.step(val_loss)
    if val_iou > best_iou:
        best_iou = val_iou
        torch.save({
            "epoch": ep,
            "model_state_dict": model.state_dict(),
            "optimizer_state_dict": opt.state_dict(),
            "scheduler_state_dict": sched.state_dict(),
            "val_iou": best_iou
        }, os.path.join(Config.OUTPUT_DIR, "checkpoints",
f"ft_best_{int(best_iou*10000)}.pth"))
        torch.save({"model_state_dict": model.state_dict(), "val_iou":
best_iou}, os.path.join(Config.OUTPUT_DIR, "checkpoints",
"best_model.pth"))
        print("Saved fine-tuned best model.")
```

```
Loaded best model (val_iou 0.7914)
FT Epoch 1/12
```

```
/tmp/ipython-input-734528679.py: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.
  sd = torch.load(ckpt, map_location=Config.DEVICE)
```

```
{"model_id":"cc2f642b1cce4c829db8484a7eb8926d","version_major":2,"vers
ion_minor":0}

{"model_id":"5e2e0e75c8784a4494bc8183c00b6a37","version_major":2,"vers
ion_minor":0}

FT epoch 1 train_loss 0.1322 val_loss 0.1298 val_iou 0.7885
Saved fine-tuned best model.
FT Epoch 2/12

{"model_id":"2588d867d52d4baf81d67b017bf3cbdd","version_major":2,"vers
ion_minor":0}

{"model_id":"c3ea9b90f69f47d1be3c7acd48a4c575","version_major":2,"vers
ion_minor":0}

FT epoch 2 train_loss 0.1280 val_loss 0.1301 val_iou 0.7905
Saved fine-tuned best model.
FT Epoch 3/12

{"model_id":"e71e9a44416a43a89938e79af32654c8","version_major":2,"vers
ion_minor":0}

{"model_id":"80370bc093104abfb746450f40ac003b","version_major":2,"vers
ion_minor":0}

FT epoch 3 train_loss 0.1249 val_loss 0.1299 val_iou 0.7914
Saved fine-tuned best model.
FT Epoch 4/12

{"model_id":"79213b713dd54225b818fbd2c5e7647f","version_major":2,"vers
ion_minor":0}

{"model_id":"650dd88b42d144e3b77d10e6f96143e3","version_major":2,"vers
ion_minor":0}

FT epoch 4 train_loss 0.1243 val_loss 0.1324 val_iou 0.7867
FT Epoch 5/12

{"model_id":"53cff074e60c4715908f53dd94b7d7f0","version_major":2,"vers
ion_minor":0}

{"model_id":"63fce2d7daae4a3196e0c23a928430fa","version_major":2,"vers
ion_minor":0}

FT epoch 5 train_loss 0.1209 val_loss 0.1333 val_iou 0.7829
FT Epoch 6/12

{"model_id":"7fdf936a85f14a3596a226f6f0b9d0eb","version_major":2,"vers
ion_minor":0}

{"model_id":"6fda0a7ee1404b64a6baa64174163469","version_major":2,"vers
ion_minor":0}
```

```
FT epoch 6 train_loss 0.1199 val_loss 0.1288 val_iou 0.7901
FT Epoch 7/12

{"model_id":"8269a20e175c47dfbf200640259ef2dc","version_major":2,"vers
ion_minor":0}

{"model_id":"92eb7bfc746d4da1b2f948306e260318","version_major":2,"vers
ion_minor":0}

FT epoch 7 train_loss 0.1211 val_loss 0.1302 val_iou 0.7905
FT Epoch 8/12

{"model_id":"733ff9cc7ea94d6481c6fd7ac161bb14","version_major":2,"vers
ion_minor":0}

{"model_id":"f2d681e79d624480aa3b5ee4baa959cd","version_major":2,"vers
ion_minor":0}

FT epoch 8 train_loss 0.1183 val_loss 0.1293 val_iou 0.7880
FT Epoch 9/12

{"model_id":"da2251accfd6460db9af110f51c4a496","version_major":2,"vers
ion_minor":0}

{"model_id":"8a0bd1d45721493f84fde072964edd1f","version_major":2,"vers
ion_minor":0}

FT epoch 9 train_loss 0.1181 val_loss 0.1289 val_iou 0.7898
FT Epoch 10/12

{"model_id":"03144d8e14b4422794f169737f402e00","version_major":2,"vers
ion_minor":0}

{"model_id":"70b00d99c55846498d51984cf2a7f20e","version_major":2,"vers
ion_minor":0}

FT epoch 10 train_loss 0.1129 val_loss 0.1272 val_iou 0.7924
Saved fine-tuned best model.
FT Epoch 11/12

{"model_id":"8e697caaa7dd42bca4ab001e08fb3924","version_major":2,"vers
ion_minor":0}

{"model_id":"a7e442b56db54ebda50199485860ad2b","version_major":2,"vers
ion_minor":0}

FT epoch 11 train_loss 0.1129 val_loss 0.1282 val_iou 0.7899
FT Epoch 12/12

{"model_id":"c9aa9a31eabf4cfeb6eb98c5f988b2f1","version_major":2,"vers
ion_minor":0}
```

{"model_id":"fbdaecfe13ea46de928fd3b52f436275","version_major":2,"version_minor":0}

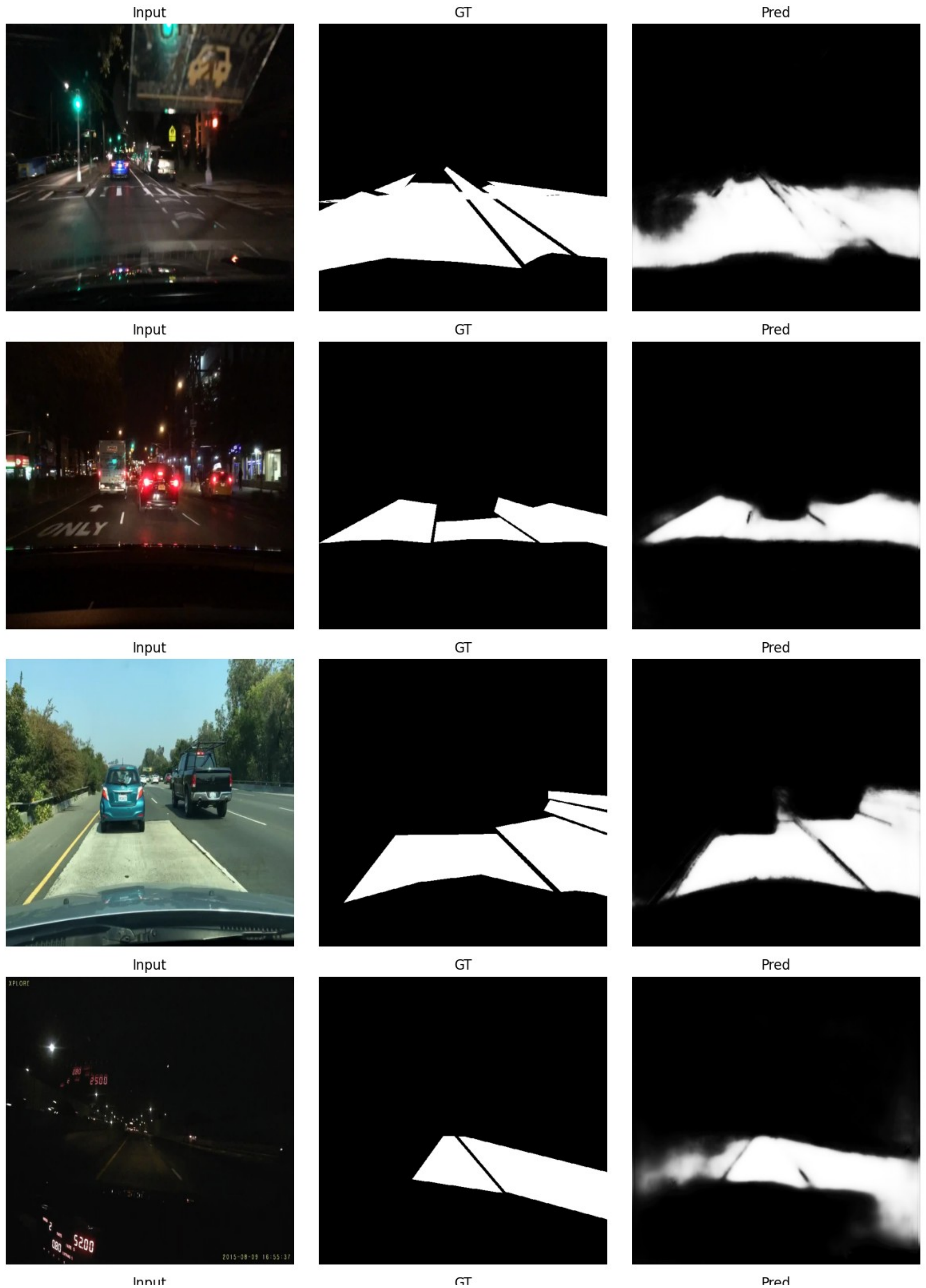
```
FT epoch 12 train_loss 0.1140 val_loss 0.1279 val_iou 0.7900
```

```python
export_model(model, os.path.join(Config.OUTPUT_DIR, "exports_final"))
final_metrics = evaluate_model(model, val_loader, save_vis=True)
print("Final validation metrics:", final_metrics)
```

```
Saved ONNX: /content/DriSegs/DriSegs/outputs/exports_final/model.onnx
ONNX verified.
```

{"model_id":"400a37c84ffc4e4883cc2090abc4b41f","version_major":2,"version_minor":0}

```
Test metrics: {'iou': 0.7899907163170153, 'precision':
0.862306554578868, 'recall': 0.9041228908757553, 'f1':
0.8813133486529486, 'accuracy': 0.957988409348476}
```


Input
GT
Pred
Input
GT
Pred
Input
GT
Pred
Input
GT
Pred
Input
GT
Pred

```
Final validation metrics: {'iou': 0.7899907163170153, 'precision':
0.862306554578868, 'recall': 0.9041228908757553, 'f1':
0.8813133486529486, 'accuracy': 0.957988409348476}
```

Evaluation Metrics

```python
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import torch

ckpt_path = os.path.join(Config.OUTPUT_DIR, "checkpoints",
"best_model.pth")
assert os.path.exists(ckpt_path), f"Best model not found at
{ckpt_path}"

sd = torch.load(ckpt_path, map_location=Config.DEVICE)
model.load_state_dict(sd["model_state_dict"])
model.eval()
print(f"Loaded best model (val_iou={sd.get('val_iou', 0):.4f})")

print("Evaluating model to get final metrics...")
final_metrics = evaluate_model(model, val_loader, save_vis=False)

df = pd.DataFrame(list(final_metrics.items()), columns=["Metric",
"Score"])
display(df)

fig, ax = plt.subplots(figsize=(8,5))
names = list(final_metrics.keys())
values = [final_metrics[k] for k in names]
x = np.arange(len(names))

bars = ax.bar(x, values, color='skyblue')
ax.set_xticks(x)
ax.set_xticklabels(names, fontsize=12)
ax.set_ylim(0, 1.05)
ax.set_ylabel("Score", fontsize=12)
ax.set_title("Best Model Validation Metrics", fontsize=14)

for rect, val in zip(bars, values):
    ax.text(rect.get_x() + rect.get_width() / 2, val + 0.02,
f"{val:.3f}",
            ha='center', va='bottom', fontsize=10)

plt.tight_layout()

out_dir = os.path.join(Config.OUTPUT_DIR, "visualizations")
os.makedirs(out_dir, exist_ok=True)
```

```
out_path = os.path.join(out_dir, "best_model_metrics.png")
plt.savefig(out_path, dpi=150, bbox_inches='tight')
plt.show()

print("Saved metric plot to:", out_path)
```

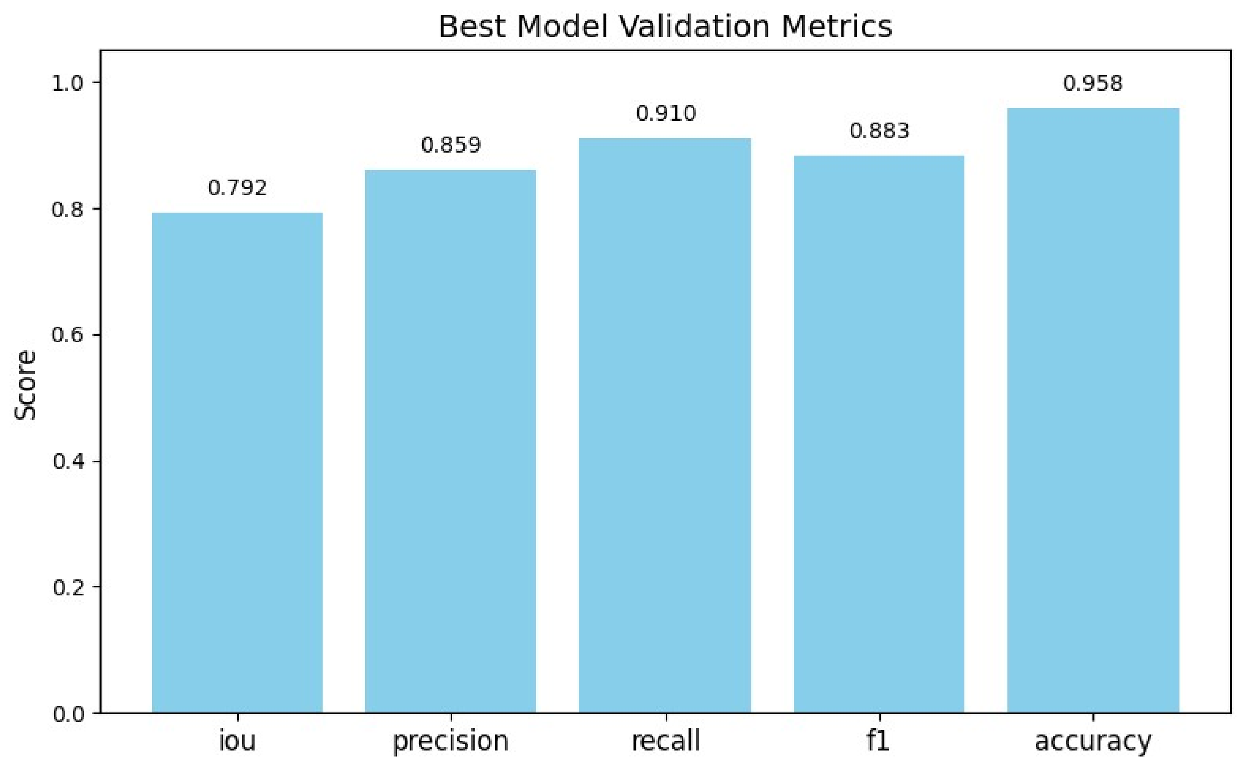
```
/tmp/ipython-input-1443203780.py:10: FutureWarning: You are using
`torch.load` with `weights_only=False` (the current default value),
which uses the default pickle module implicitly. It is possible to
construct malicious pickle data which will execute arbitrary code
during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
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.
  sd = torch.load(ckpt_path, map_location=Config.DEVICE)
```

```
Loaded best model (val_iou=0.7924)
Evaluating model to get final metrics...
```

{"model_id":"a40dcc52a92640b6967caedb9c4788b4","version_major":2,"vers
ion_minor":0}

```
Test metrics: {'iou': 0.7923787054774954, 'precision':
0.859225317324885, 'recall': 0.9104222931113116, 'f1':
0.8827395079313572, 'accuracy': 0.95823945975598}
```

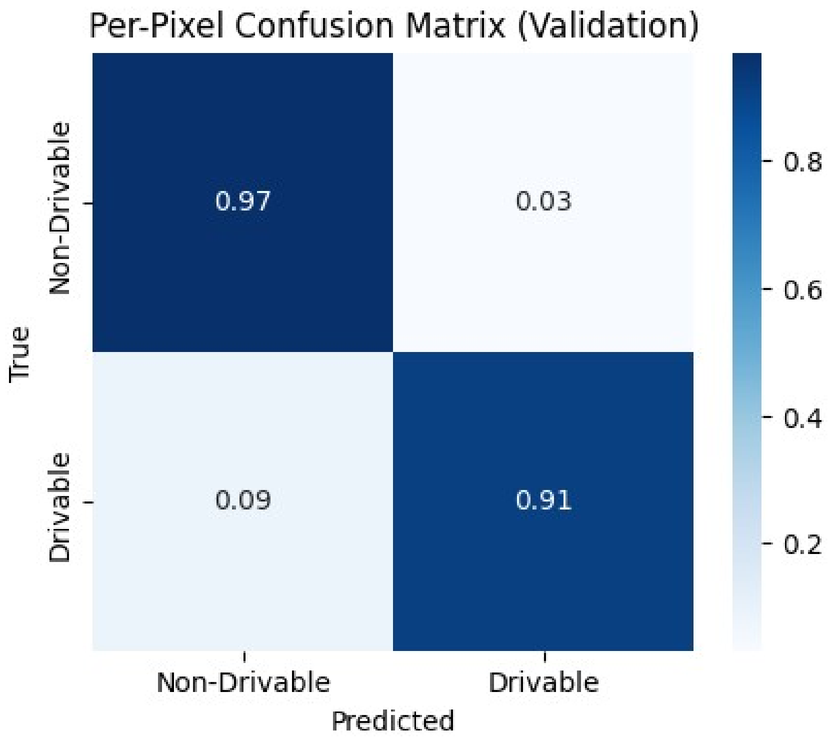
{"summary":"{\n \"name\": \"df\",\n \"rows\": 5,\n \"fields\": [\n
{\n   \"column\": \"Metric\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 5,\n
\"samples\": [\n   \"precision\",\n   \"accuracy\",\n
\"recall\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Score\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.06156644019453867,\n   \"min\": 0.7923787054774954,\n
\"max\": 0.95823945975598,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   0.859225317324885,\n
0.95823945975598,\n   0.9104222931113116\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df"}



```
Saved metric plot to:
/content/DriSegs/DriSegs/outputs/visualizations/best_model_metrics.png
```

Confusion Matrix

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

y_true_all, y_pred_all = [], []
model.eval()
with torch.no_grad():
    for imgs, masks in val_loader:
        imgs, masks = imgs.to(Config.DEVICE), masks.to(Config.DEVICE)
        out = torch.sigmoid(model(imgs))
        preds = (out > 0.5).float()
        y_true_all.extend(masks.cpu().numpy().ravel())
        y_pred_all.extend(preds.cpu().numpy().ravel())

cm = confusion_matrix(y_true_all, y_pred_all)
cm_norm = cm / cm.sum(axis=1, keepdims=True)

plt.figure(figsize=(5,4))
sns.heatmap(cm_norm, annot=True, fmt=".2f", cmap='Blues',
            xticklabels=['Non-Drivable','Drivable'],
            yticklabels=['Non-Drivable','Drivable'])
plt.title("Per-Pixel Confusion Matrix (Validation)")
plt.xlabel("Predicted"); plt.ylabel("True")
path_cm =
os.path.join(Config.OUTPUT_DIR,"visualizations","confusion_matrix.png"
)
plt.savefig(path_cm, dpi=150, bbox_inches='tight')
plt.show()
print("Saved:", path_cm)
```



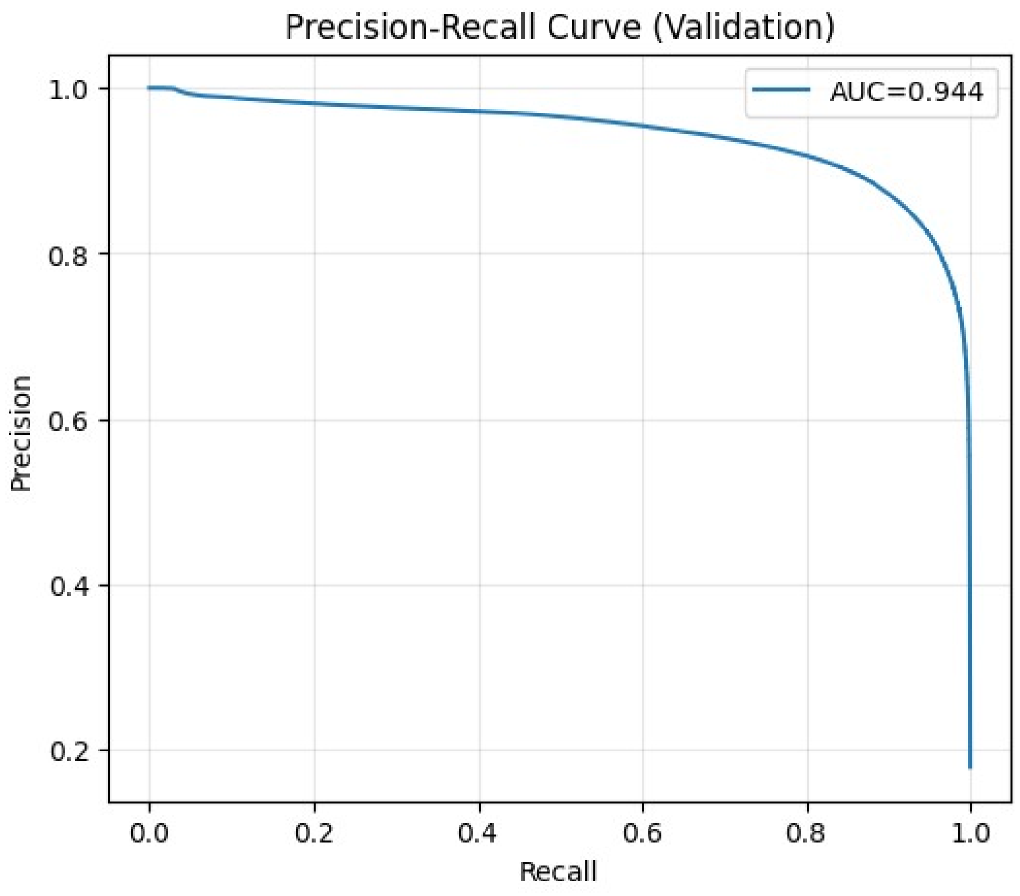
```
Saved:
/content/DriSegs/DriSegs/outputs/visualizations/confusion_matrix.png
```

Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve, auc

probs, targets = [], []
model.eval()
with torch.no_grad():
    for imgs, masks in val_loader:
        imgs, masks = imgs.to(Config.DEVICE), masks.to(Config.DEVICE)
        out = torch.sigmoid(model(imgs))
        probs.append(out.cpu().numpy().ravel())
        targets.append(masks.cpu().numpy().ravel())
probs = np.concatenate(probs)
targets = np.concatenate(targets)
```

```
prec, rec, thr = precision_recall_curve(targets, probs)
auc_pr = auc(rec, prec)
plt.figure(figsize=(6,5))
plt.plot(rec, prec, label=f"AUC={auc_pr:.3f}")
plt.xlabel("Recall"); plt.ylabel("Precision")
plt.title("Precision-Recall Curve (Validation)")
plt.legend(); plt.grid(True, alpha=0.3)
path_pr =
os.path.join(Config.OUTPUT_DIR,"visualizations","precision_recall_curv
e.png")
plt.savefig(path_pr, dpi=150, bbox_inches='tight')
plt.show()
print("Saved:", path_pr)
```

```
/tmp/ipython-input-1567701126.py:22: UserWarning: Creating legend with
loc="best" can be slow with large amounts of data.
  plt.savefig(path_pr, dpi=150, bbox_inches='tight')
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151
: UserWarning: Creating legend with loc="best" can be slow with large
amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```

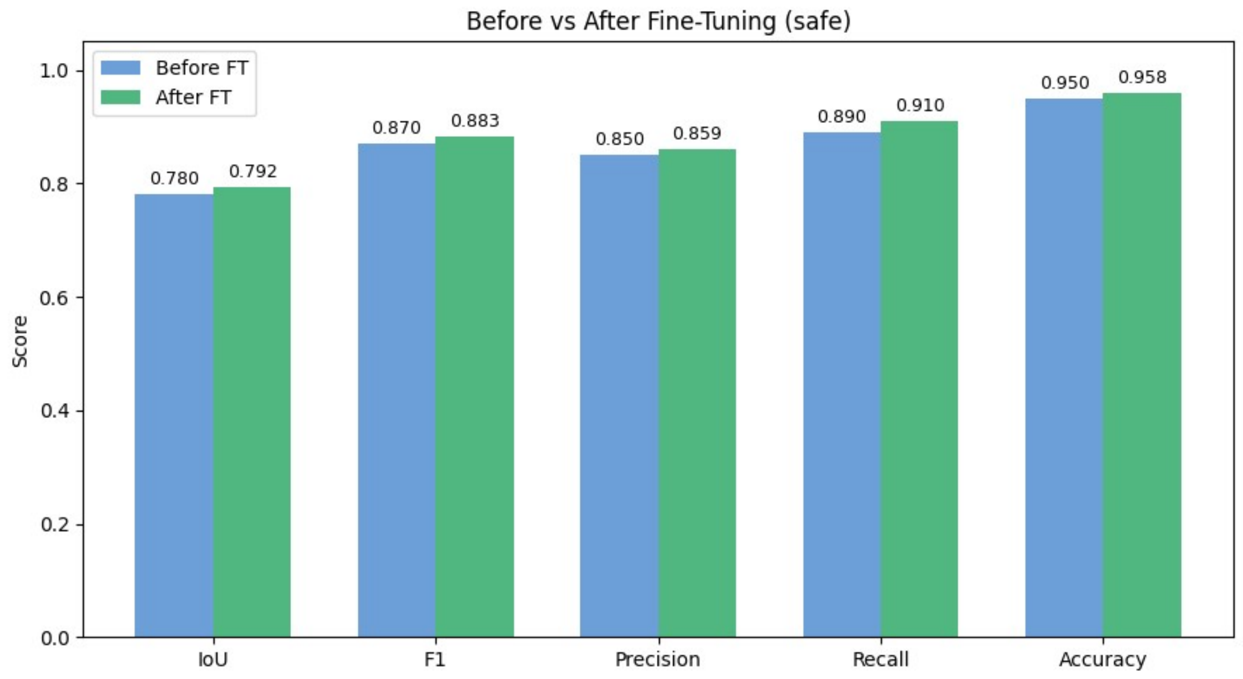


```
Saved:
/content/DriSegs/DriSegs/outputs/visualizations/precision_recall_curve
.png
```

Fine Tuning Report

```
import os, numpy as np, matplotlib.pyplot as plt, pandas as pd

baseline = {'IoU':0.78, 'F1':0.87, 'Precision':0.85, 'Recall':0.89,
'Accuracy':0.95}

print("finetune (raw) keys:", list(finetune.keys()))

def canonical(k):
    return k.strip().lower().replace('-', '_')

baseline_norm = {canonical(k): (k, v) for k,v in baseline.items()}
finetune_norm  = {canonical(k): v for k,v in finetune.items()}

names = [baseline_norm[norm][0] for norm in baseline_norm.keys()]
x = np.arange(len(names))
width = 0.35

before_vals = []
after_vals = []
missing = []
for norm_key in baseline_norm.keys():
    before_vals.append(baseline_norm[norm_key][1])
    if norm_key in finetune_norm:
        after_vals.append(finetune_norm[norm_key])
    else:
        after_vals.append(np.nan)
        missing.append(baseline_norm[norm_key][0])

print("Normalized baseline keys:", list(baseline_norm.keys()))
print("Normalized finetune keys:", list(finetune_norm.keys()))
if missing:
    print("Warning: the following baseline metrics were NOT found in
finetune results and will be plotted as NaN:", missing)

plt.figure(figsize=(9,5))
plt.bar(x - width/2, before_vals, width, label='Before FT',
color='#6c9fd8')
plt.bar(x + width/2, after_vals, width, label='After FT',
color='#4fb77f')
plt.xticks(x, names)
plt.ylim(0,1.05)
plt.ylabel("Score")
plt.title("Before vs After Fine-Tuning (safe)")
plt.legend()
plt.tight_layout()

for i,(b,a) in enumerate(zip(before_vals, after_vals)):
    plt.text(x[i]-width/2, (b if not np.isnan(b) else 0)+0.01,
f"{b:.3f}", ha='center', va='bottom', fontsize=9)
    if not np.isnan(a):
        plt.text(x[i]+width/2, a+0.01, f"{a:.3f}", ha='center',
va='bottom', fontsize=9)
    else:
        plt.text(x[i]+width/2, 0.01, "n/a", ha='center', va='bottom',
fontsize=8, color='red')

out_path = os.path.join(Config.OUTPUT_DIR, "visualizations",
"before_after_finetune_safe.png")
os.makedirs(os.path.dirname(out_path), exist_ok=True)
plt.savefig(out_path, dpi=150, bbox_inches='tight')
plt.show()

print("Saved safe before/after plot to:", out_path)
```

```
finetune (raw) keys: ['iou', 'precision', 'recall', 'f1', 'accuracy']
Normalized baseline keys: ['iou', 'f1', 'precision', 'recall',
'accuracy']
Normalized finetune keys: ['iou', 'precision', 'recall', 'f1',
'accuracy']
```

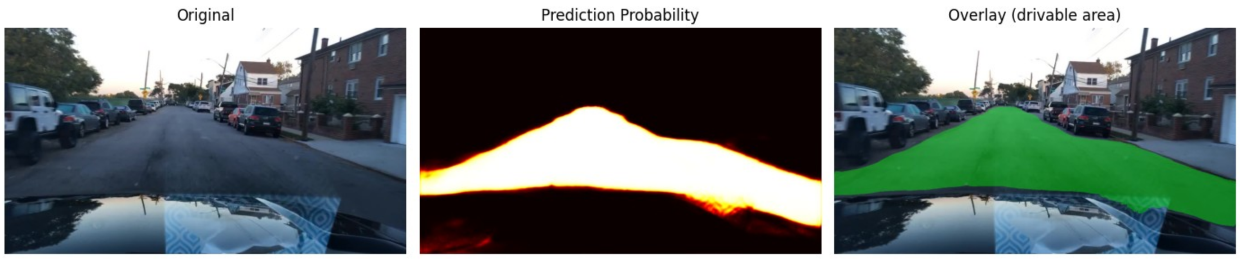


```
Saved safe before/after plot to:
/content/DriSegs/DriSegs/outputs/visualizations/before_after_finetune_
safe.png
```

Sample Model Output 1

```
from google.colab import files
import os
import cv2
import matplotlib.pyplot as plt
import torch
import numpy as np

print("Select an image to upload (jpg / png).")
uploaded = files.upload()
if not uploaded:
    print("No file uploaded — canceling.")
else:
    fname = list(uploaded.keys())[0]
    local_path = os.path.join('/content', fname)
    print("Saved upload to:", local_path)

    try:
        model
        print("Model already in memory.")
    except NameError:
        ckpt = os.path.join(Config.OUTPUT_DIR, "checkpoints",
"best_model.pth")
        if os.path.exists(ckpt):
            sd = torch.load(ckpt, map_location=Config.DEVICE)
            model = EfficientNetUNet().to(Config.DEVICE)
            model.load_state_dict(sd["model_state_dict"])
            model.eval()
            print(f"Loaded model from checkpoint
(val_iou={sd.get('val_iou',0):.4f}).")
        else:
            raise FileNotFoundError(f"Model not in memory and no
checkpoint found at {ckpt} — run training or load a checkpoint
first.")

    def run_and_visualize(image_path, model, threshold=0.5,
save_path='./uploaded_prediction.png'):
        img_bgr = cv2.imread(image_path)
        if img_bgr is None:
            raise RuntimeError("Could not read uploaded image.")
        img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
        h, w = img_rgb.shape[:2]

        aug = get_validation_augmentation()
        aug_out = aug(image=img_rgb)
        inp = aug_out['image'].unsqueeze(0).to(Config.DEVICE)

        with torch.no_grad():
            out = model(inp)
            prob = torch.sigmoid(out).cpu().numpy()[0,0]

        prob_resized = cv2.resize(prob, (w, h))
        binary = (prob_resized > threshold).astype(np.uint8)

        overlay = img_rgb.copy()
        overlay[binary == 1] = [0, 255, 0]
        blended = (0.6 * img_rgb + 0.4 * overlay).astype(np.uint8)

        plt.figure(figsize=(14,6))
        plt.subplot(1,3,1)
        plt.imshow(img_rgb)
        plt.title("Original")
        plt.axis('off')

        plt.subplot(1,3,2)
        plt.imshow(prob_resized, cmap='hot')
        plt.title("Prediction Probability")
        plt.axis('off')

        plt.subplot(1,3,3)
        plt.imshow(blended)
        plt.title("Overlay (drivable area)")
        plt.axis('off')

        plt.tight_layout()
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        print("Saved prediction to:", save_path)
        return prob_resized, binary, blended

    prob, binary_mask, blended = run_and_visualize(local_path, model)
```

```
Select an image to upload (jpg / png).

<IPython.core.display.HTML object>

Saving b7db31d1-fa13a84a.jpg to b7db31d1-fa13a84a.jpg
Saved upload to: /content/b7db31d1-fa13a84a.jpg
Model already in memory.
Saved prediction to: ./uploaded_prediction.png
```

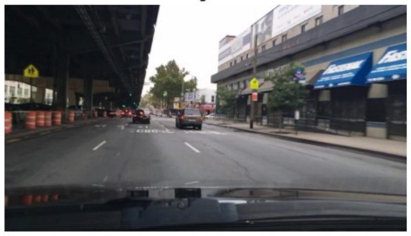


Sample Model Output 2

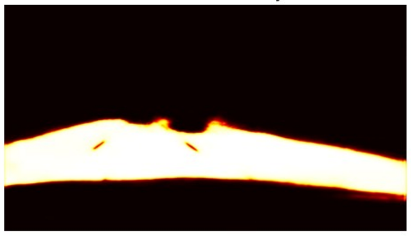
```
from google.colab import files
import os
import cv2
import matplotlib.pyplot as plt
import torch
import numpy as np

print("Select an image to upload (jpg / png).")
uploaded = files.upload()
if not uploaded:
    print("No file uploaded — canceling.")
else:
    fname = list(uploaded.keys())[0]
    local_path = os.path.join('/content', fname)
    print("Saved upload to:", local_path)

    try:
        model
        print("Model already in memory.")
    except NameError:
        ckpt = os.path.join(Config.OUTPUT_DIR, "checkpoints",
"best_model.pth")
        if os.path.exists(ckpt):
            sd = torch.load(ckpt, map_location=Config.DEVICE)
            model = EfficientNetUNet().to(Config.DEVICE)
            model.load_state_dict(sd["model_state_dict"])
            model.eval()
            print(f"Loaded model from checkpoint
(val_iou={sd.get('val_iou',0):.4f}).")
        else:
            raise FileNotFoundError(f"Model not in memory and no
checkpoint found at {ckpt} — run training or load a checkpoint
first.")

    def run_and_visualize(image_path, model, threshold=0.5,
save_path='./uploaded_prediction.png'):
        img_bgr = cv2.imread(image_path)
        if img_bgr is None:
            raise RuntimeError("Could not read uploaded image.")
        img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
        h, w = img_rgb.shape[:2]

        aug = get_validation_augmentation()
        aug_out = aug(image=img_rgb)
        inp = aug_out['image'].unsqueeze(0).to(Config.DEVICE)

        with torch.no_grad():
            out = model(inp)
            prob = torch.sigmoid(out).cpu().numpy()[0,0]

        prob_resized = cv2.resize(prob, (w, h))
        binary = (prob_resized > threshold).astype(np.uint8)

        overlay = img_rgb.copy()
        overlay[binary == 1] = [0, 255, 0]
        blended = (0.6 * img_rgb + 0.4 * overlay).astype(np.uint8)

        plt.figure(figsize=(14,6))
        plt.subplot(1,3,1)
        plt.imshow(img_rgb)
        plt.title("Original")
        plt.axis('off')

        plt.subplot(1,3,2)
        plt.imshow(prob_resized, cmap='hot')
        plt.title("Prediction Probability")
        plt.axis('off')

        plt.subplot(1,3,3)
        plt.imshow(blended)
        plt.title("Overlay (drivable area)")
        plt.axis('off')

        plt.tight_layout()
        plt.savefig(save_path, dpi=150, bbox_inches='tight')
        print("Saved prediction to:", save_path)
        return prob_resized, binary, blended

    prob, binary_mask, blended = run_and_visualize(local_path, model)
```

```
Select an image to upload (jpg / png).

<IPython.core.display.HTML object>

Saving b25fb716-78d8d49b.jpg to b25fb716-78d8d49b.jpg
Saved upload to: /content/b25fb716-78d8d49b.jpg
Model already in memory.
Saved prediction to: ./uploaded_prediction.png
```

Original



Prediction Probability



Overlay (drivable area)