Technical Report: Multispectral Habitat Segmentation of Coastal Imagery

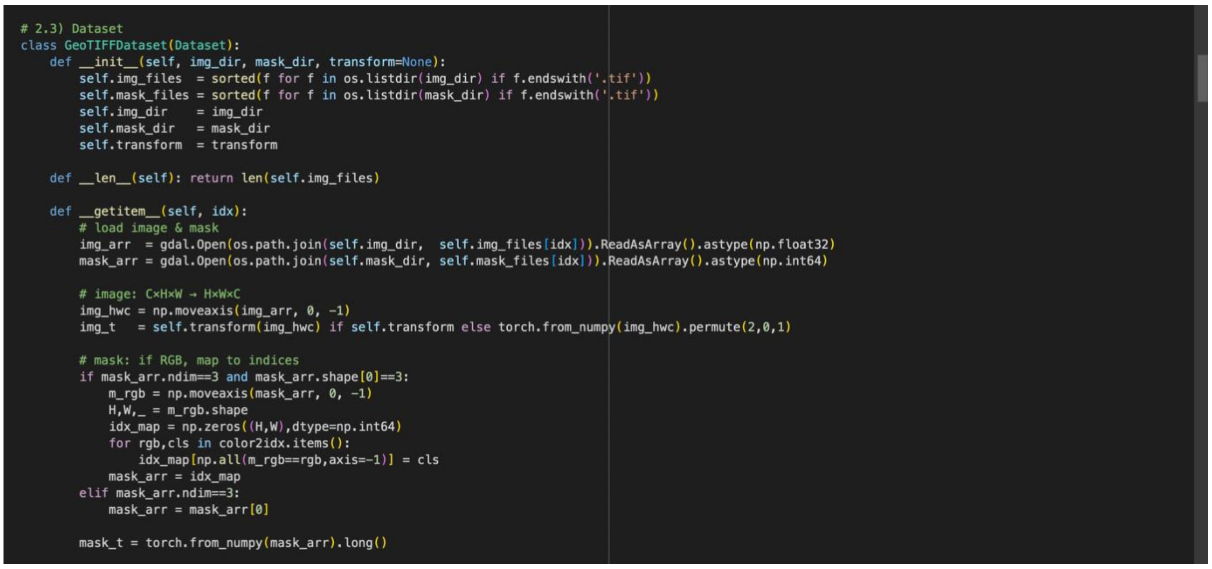
1. Introduction

We present a workflow for pixel-wise segmentation of multispectral satellite imagery into seven habitat classes:

- 0 Background
- 1 Seagrass
- 2 Coral
- 3 Macroalgae
- 4 Sand
- 5 Land
- 6 Ocean

Our goals were to:

1. Build a robust training pipeline using PyTorch
2. Handle 8-band GeoTIFFs with correct normalization and mask mapping
3. Compensate for class imbalance (especially the rare "Ocean" class)
4. Achieve an "Improved Labeled Habitat Map" (see Fig. 1)



```python
# 2.3) Dataset
class GeoTIFFDataset(Dataset):
    def __init__(self, img_dir, mask_dir, transform=None):
        self.img_files  = sorted(f for f in os.listdir(img_dir) if f.endswith('.tif'))
        self.mask_files = sorted(f for f in os.listdir(mask_dir) if f.endswith('.tif'))
        self.img_dir    = img_dir
        self.mask_dir   = mask_dir
        self.transform  = transform

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

    def __getitem__(self, idx):
        # load image & mask
        img_arr  = gdal.Open(os.path.join(self.img_dir,  self.img_files[idx])).ReadAsArray().astype(np.float32)
        mask_arr = gdal.Open(os.path.join(self.mask_dir, self.mask_files[idx])).ReadAsArray().astype(np.int64)

        # image: C×H×W → H×W×C
        img_hwc = np.moveaxis(img_arr, 0, -1)
        img_t   = self.transform(img_hwc) if self.transform else torch.from_numpy(img_hwc).permute(2,0,1)

        # mask: if RGB, map to indices
        if mask_arr.ndim==3 and mask_arr.shape[0]==3:
            m_rgb = np.moveaxis(mask_arr, 0, -1)
            H,W,_ = m_rgb.shape
            idx_map = np.zeros((H,W),dtype=np.int64)
            for rgb,cls in color2idx.items():
                idx_map[np.all(m_rgb==rgb,axis=-1)] = cls
            mask_arr = idx_map
        elif mask_arr.ndim==3:
            mask_arr = mask_arr[0]

        mask_t = torch.from_numpy(mask_arr).long()
```

2.3 Transforms

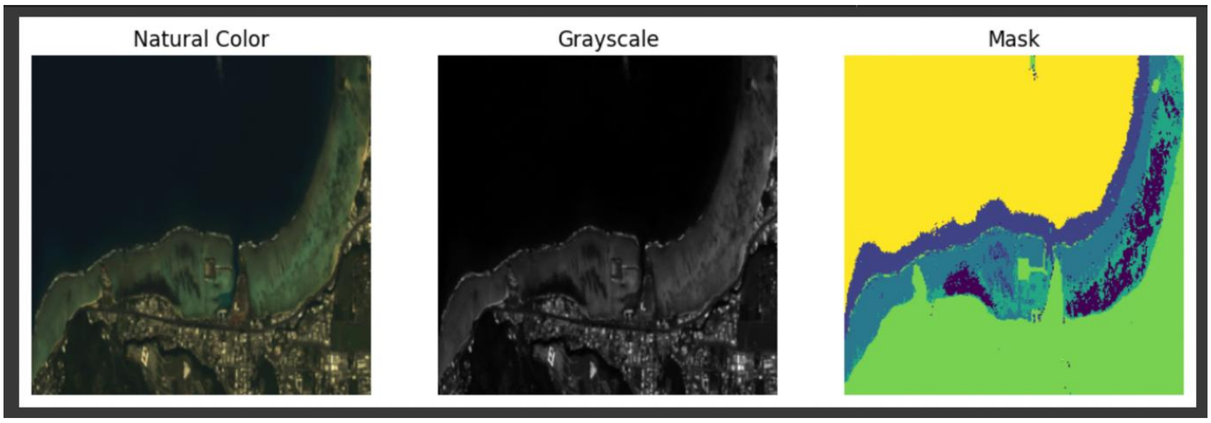
```
from torchvision.transforms import Resize, InterpolationMode

# Normalize each band to [0,1]
def multispectral_transform(img):
    mn, mx = img.min(), img.max()
    return torch.from_numpy(((img-mn)/(mx-mn+1e-8)).astype(np.float32))\
                .permute(2,0,1)

resize_img  = Resize((256,256), InterpolationMode.BILINEAR)
resize_mask = Resize((256,256), InterpolationMode.NEAREST)
```

3. Sample Visualization

Below we show one random tile in natural color, grayscale, and its ground-truth mask:



4. Addressing Class Imbalance

The raw pixel counts were highly skewed: I found that simple inverse-frequency weights under-penalized “Ocean.”
Solution**:** Median-Frequency Balancing to boost rare classes:

```
def calculate_mfb_weights(dataloader, num_classes=7, device=device):
    counts = torch.zeros(num_classes)
    for _, masks in dataloader:
        counts += torch.bincount(masks.view(-1), minlength=num_classes)
    freqs = counts / counts.sum()
    med   = torch.median(freqs[freqs>0])
    w     = med / (freqs + 1e-8)
    return (w / w.max()).to(device)

class_weights = calculate_mfb_weights(trainloader)
print("Median-freq weights:", class_weights.tolist())
```

5. Model Architectures & Training

5.1 Custom CNN

A lightweight encoder–decoder:

```
class CustomCNN(nn.Module):
    def __init__(self, in_ch=8, n_cls=7):
        super().__init__()
        self.enc = nn.Sequential(
            CNNBlock(in_ch,64), nn.MaxPool2d(2),
            CNNBlock(64,128),   nn.MaxPool2d(2),
            CNNBlock(128,256)
        )
        self.dec = nn.Sequential(
            nn.ConvTranspose2d(256,128,2,2), CNNBlock(128,128),
            nn.ConvTranspose2d(128,64, 2,2), CNNBlock(64,  64),
            nn.Conv2d(64,n_cls,1)
        )
    def forward(self,x): return self.dec(self.enc(x))
```

5.2 DeepLabV3+ Backbone

For stronger features, we also support DeepLabV3+:

```
model = models.segmentation.deeplabv3_resnet50(
    pretrained=True, num_classes=7, aux_loss=None
).to(device)
# Patch first conv from 3→8 bands:
old = model.backbone.conv1
new = nn.Conv2d(8, old.out_channels, old.kernel_size,
                old.stride, old.padding, bias=False)
with torch.no_grad():
    new.weight[:,:3]  = old.weight
    new.weight[:,3:]  = old.weight.mean(dim=1,keepdim=True)
model.backbone.conv1 = new
```

5.3 Training Loop (CPU/GPU)

```
optimizer = torch.optim.Adam(model.parameters(), lr=5e-4)
criterion = nn.CrossEntropyLoss(weight=class_weights)
scaler    = GradScaler()          # omit on CPU

for epoch in range(1, num_epochs+1):
    # train…
    # validate…
    # save best checkpoint…
```

Convergence**:**

- Train loss **fell from ~2.0 to ~0.6 over 50 epochs**
- Train accuracy **rose to ~85%**
- Val accuracy **stabilized ~76%**

6. Single-Tile Inference

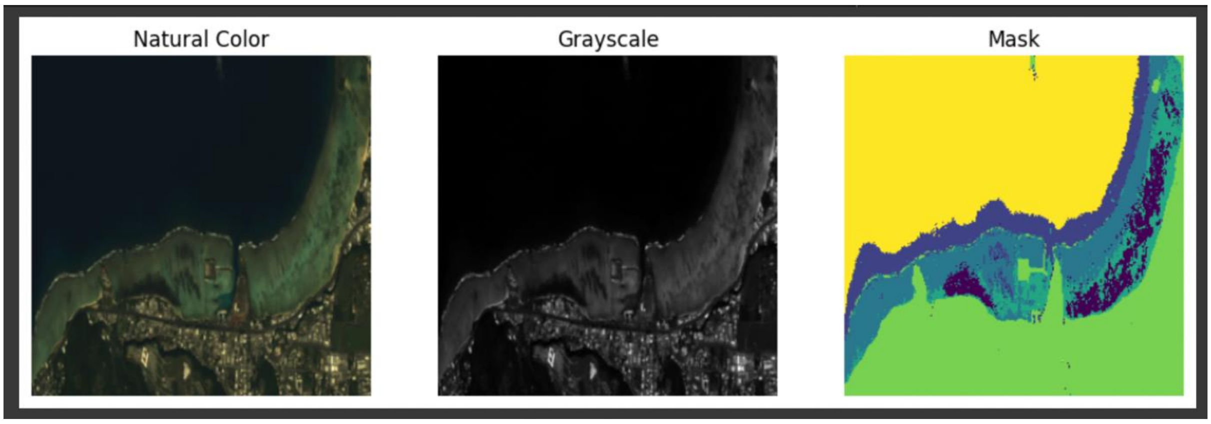
- We automatically locate and display a tile containing all seven classes:

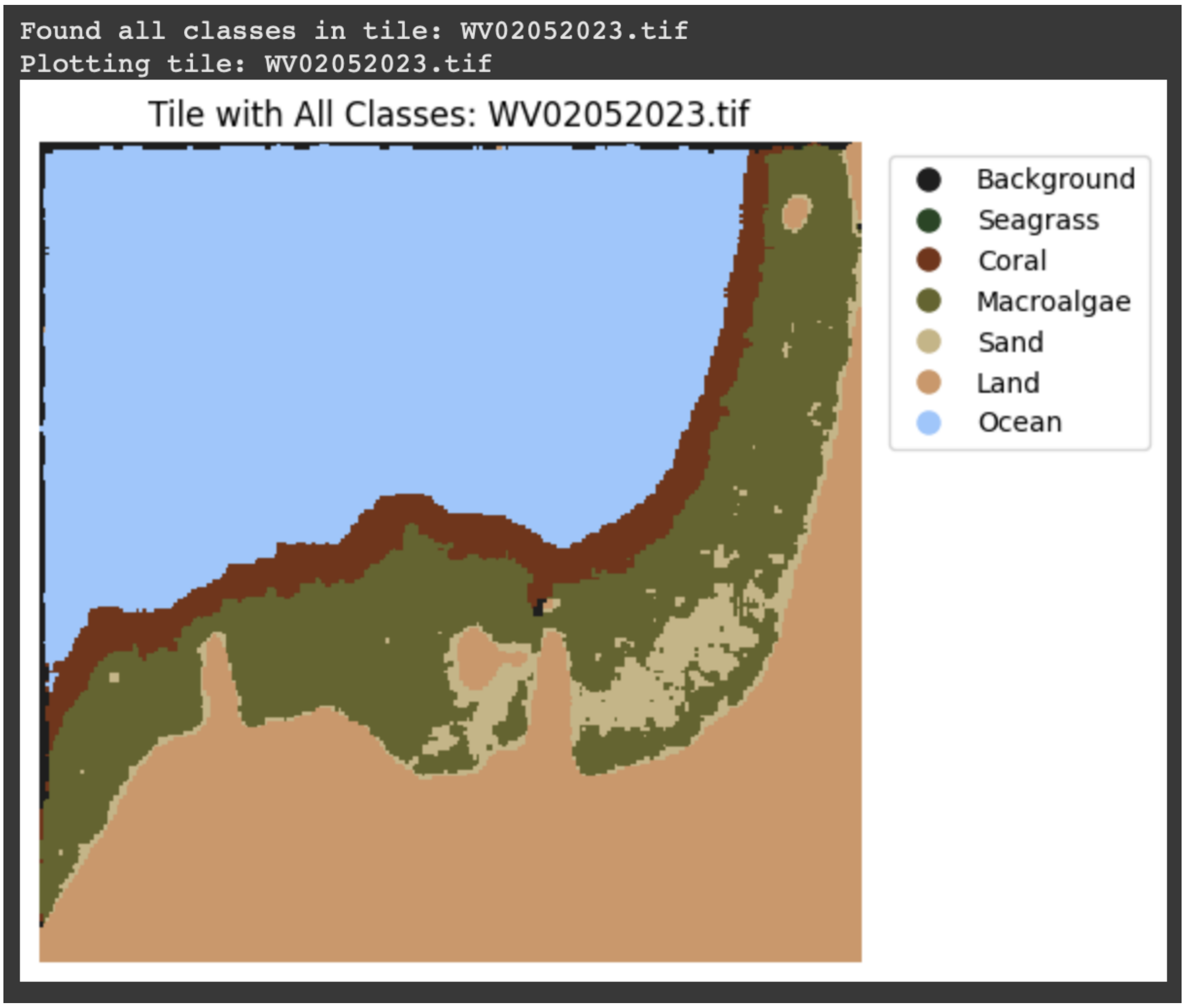
```
def find_tile_with_all_classes(image_dir, model):
    for f in sorted(os.listdir(image_dir)):
        pred, frac = infer_and_get_fraction(f)
        if set(np.unique(pred)) == set(range(7)):
            plot(pred, title=f)
            break
```

8. Conclusion

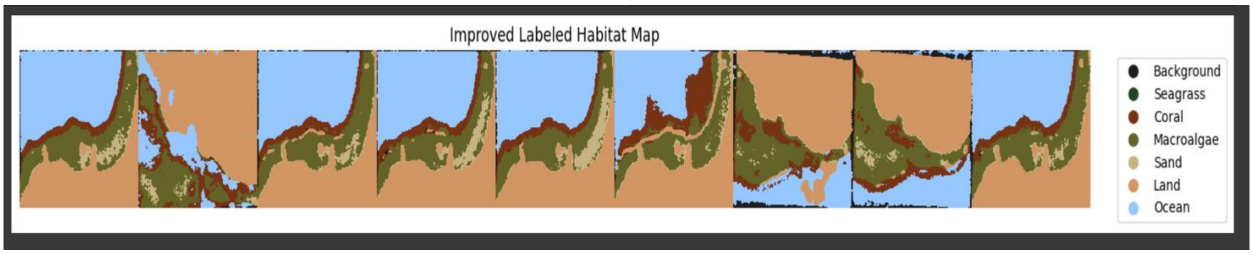
- End-to-end pipeline: from GeoTIFF I/O → multispectral normalization → mask mapping → model training → inference → visualization.
- Class-imbalance handled via median-frequency weights, preventing under-prediction of rare classes like Ocean
- Modular architecture: custom CNN or pretrained DeepLabV3+ backbone.

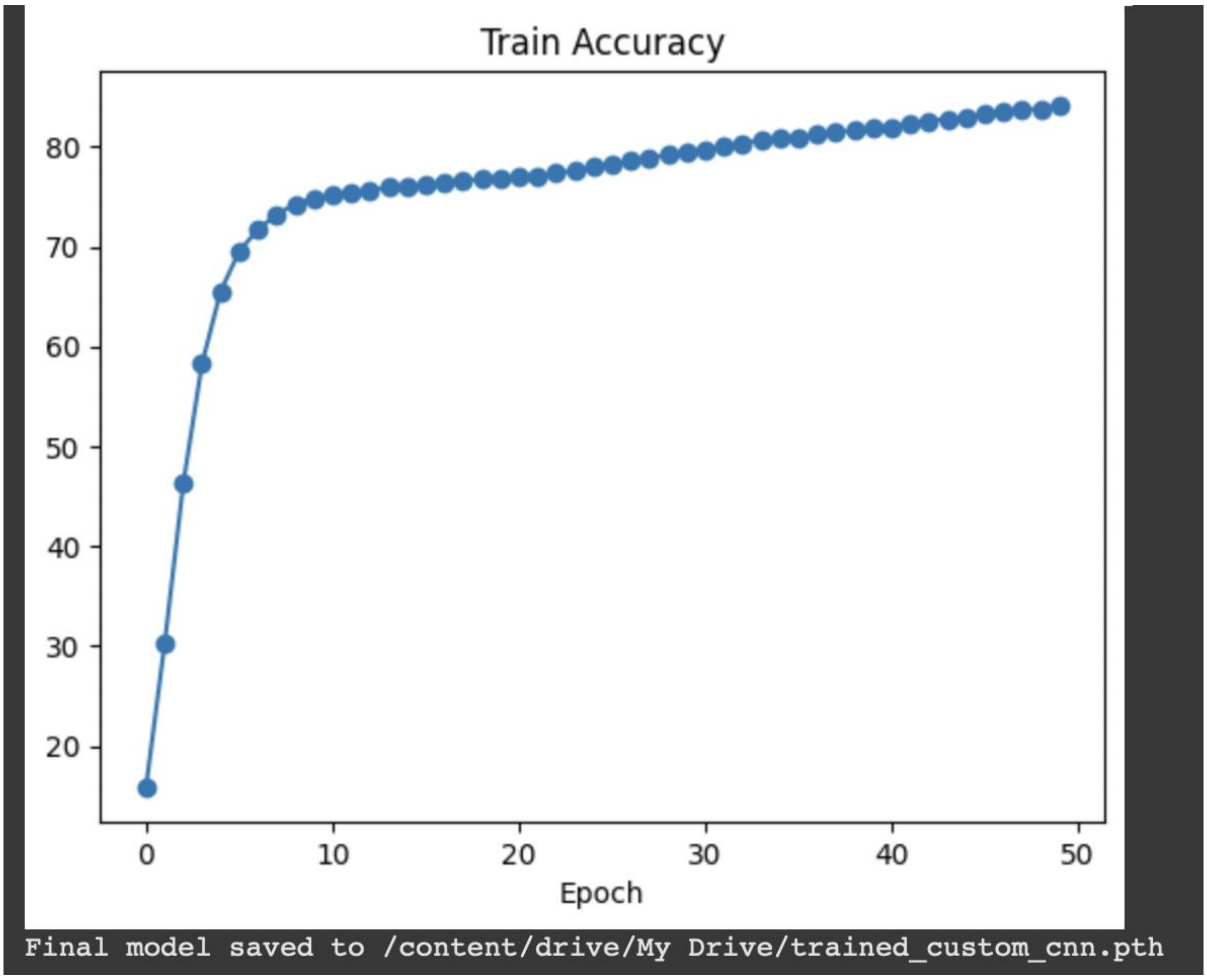
RESULTS





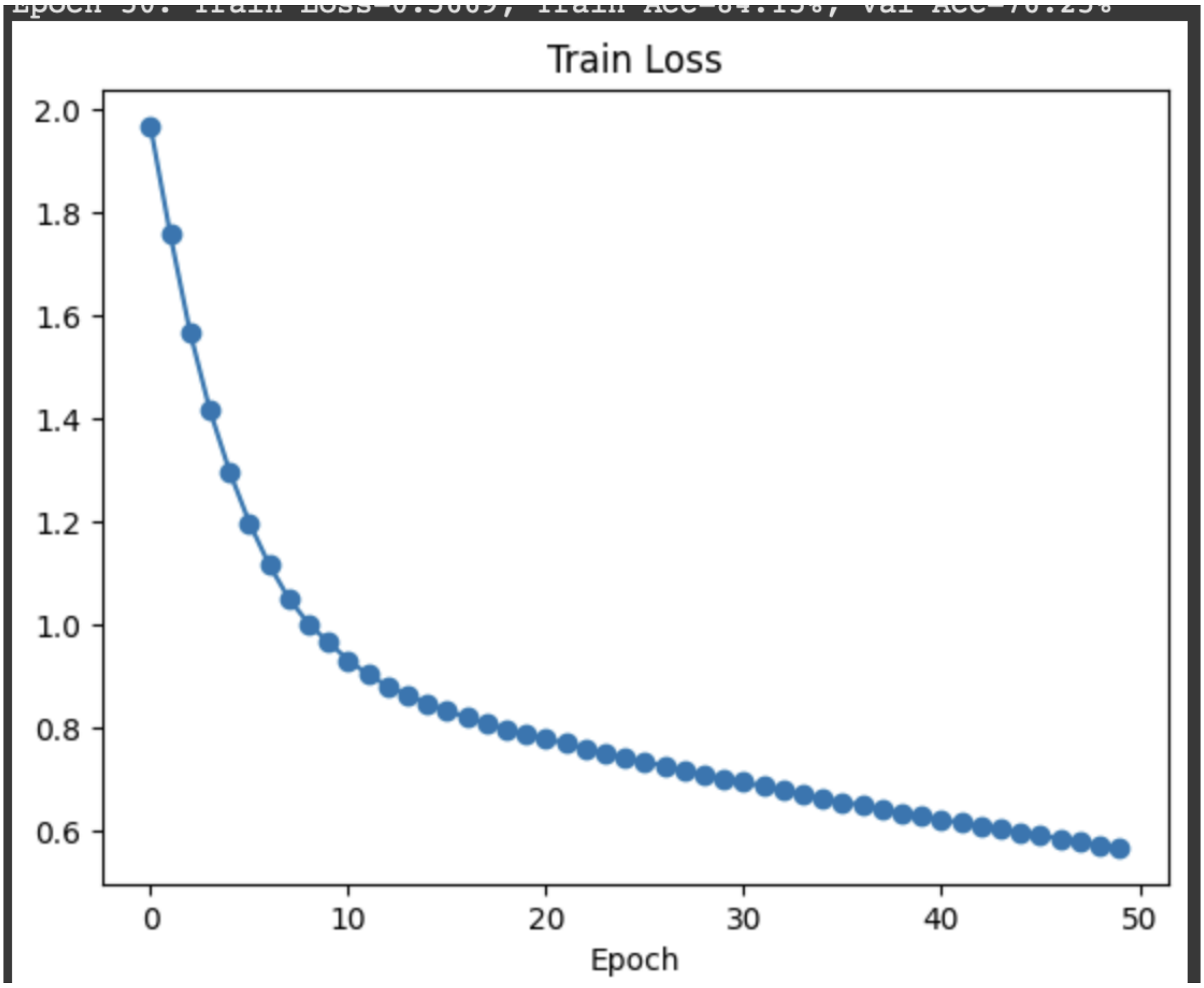
```
Found all classes in tile: WV02052023.tif
Plotting tile: WV02052023.tif
```





```
Final model saved to /content/drive/My Drive/trained_custom_cnn.pth
```

```
Epoch 50: Train Loss=0.5669, Train Acc=84.15%, Val Acc=70.25%
```


Train Loss
Epoch