

# TOB Projesi

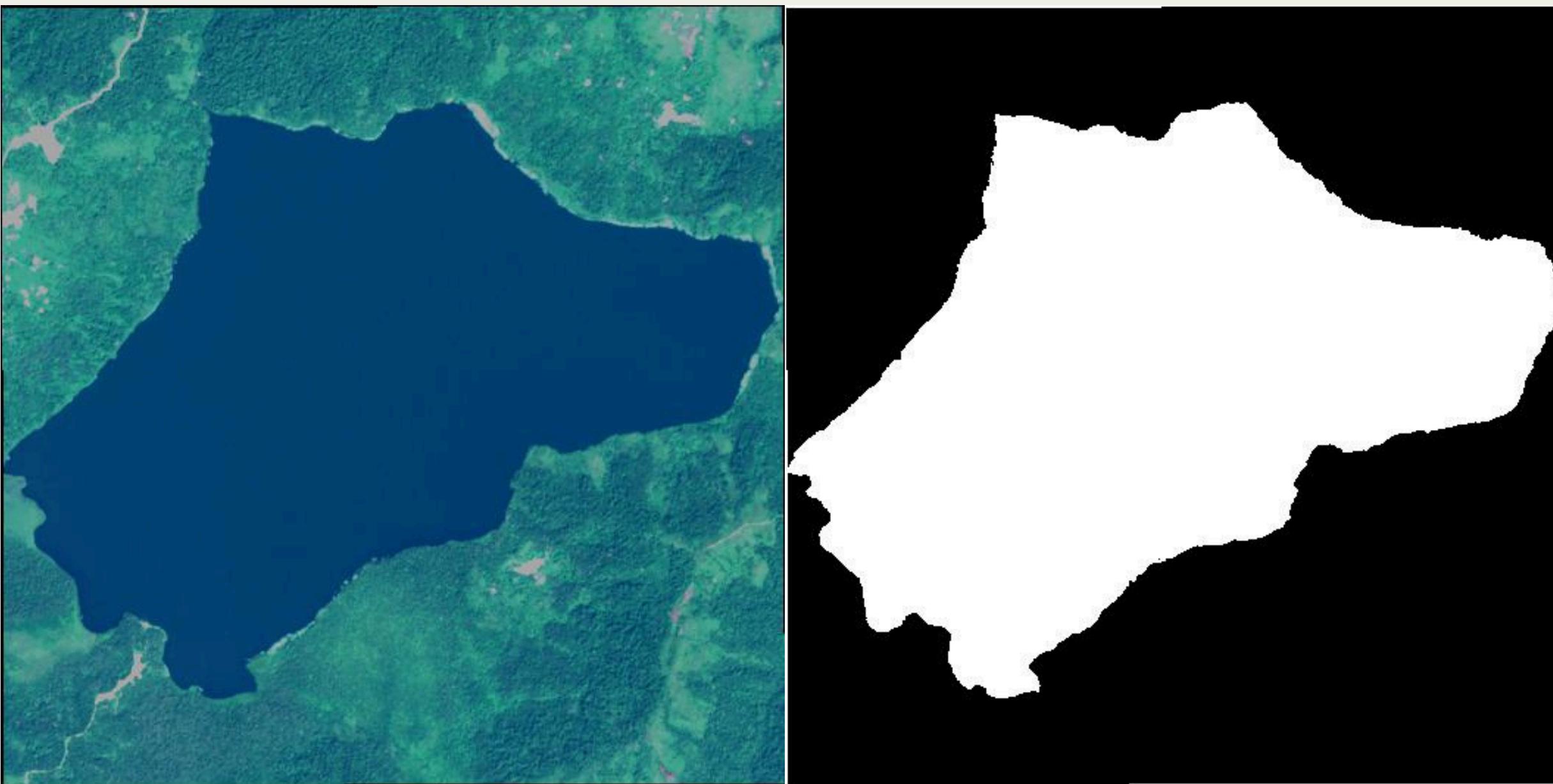
---

**Uydu Görüntüleri ve Yapay Zeka ile Su Yüzey Alanı Tespiti**

ahmetnurieroglu@gmail.com  
[github.com/anurieroglu](https://github.com/anurieroglu)

# Uydu Görüntüleri ve Yapay Zeka ile Su Yüzey Alanı Tespiti

---



## Mevcut Durum

---

Günümüzde su yüzey alanı ve su kalitesi ölçümleri genellikle yerinde gözlem veya otomatik ölçüm cihazları ile yapılmaktadır.

Bu yöntemler düzenli uygulanamamaktadır çünkü:

- Altyapı yetersizliği
- Personel eksikliği
- Bütçe kısıtları

Yapay zekâ destekli uydu görüntüleri kullanarak:

- Otomatik
- Düşük maliyetli
- Daha sık ve düzenli ölçümler yapmak mümkün hale gelecektir.

# Kullanılan Veri Seti

---

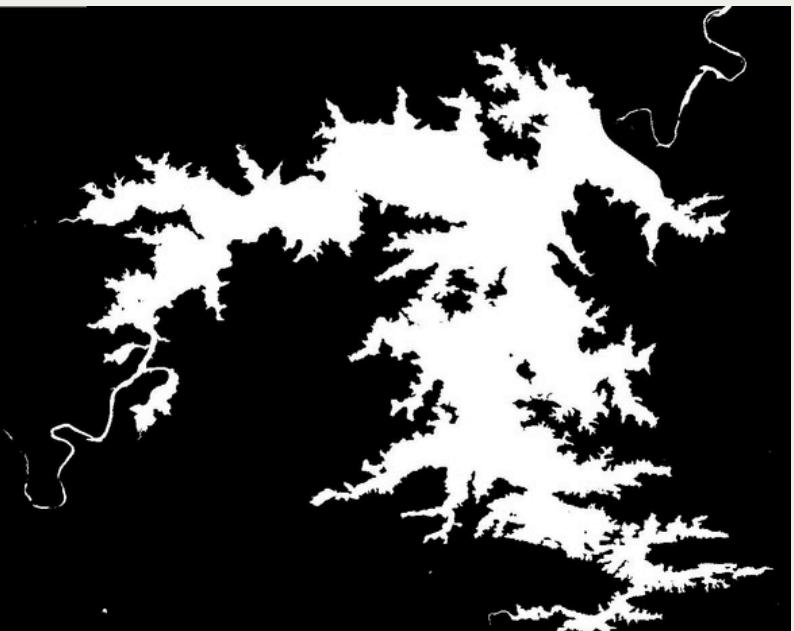
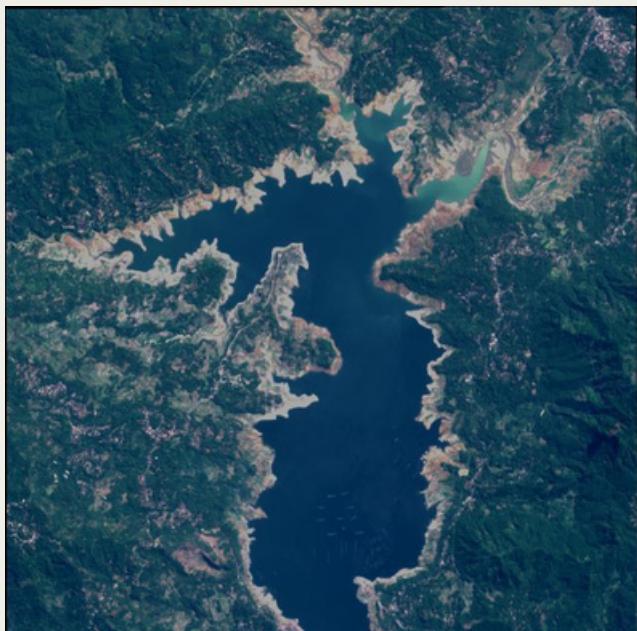
Proje boyunca eğitim ve test işlemleri için aşağıdaki veri seti kullanılmıştır:

## Satellite Images of Water Bodies

- İçeriğinde farklı kıtalardan göl ve baraj görüntüleri bulunur
- Maskeleme (ground truth) etiketleri ile birlikte sağlanmıştır
- Segmentasyon modellerinin eğitiminde bu veri seti kullanıldı

Görsel:

- Datasetten örnek bir uydu görüntüsü ve karşılık gelen su maskesi

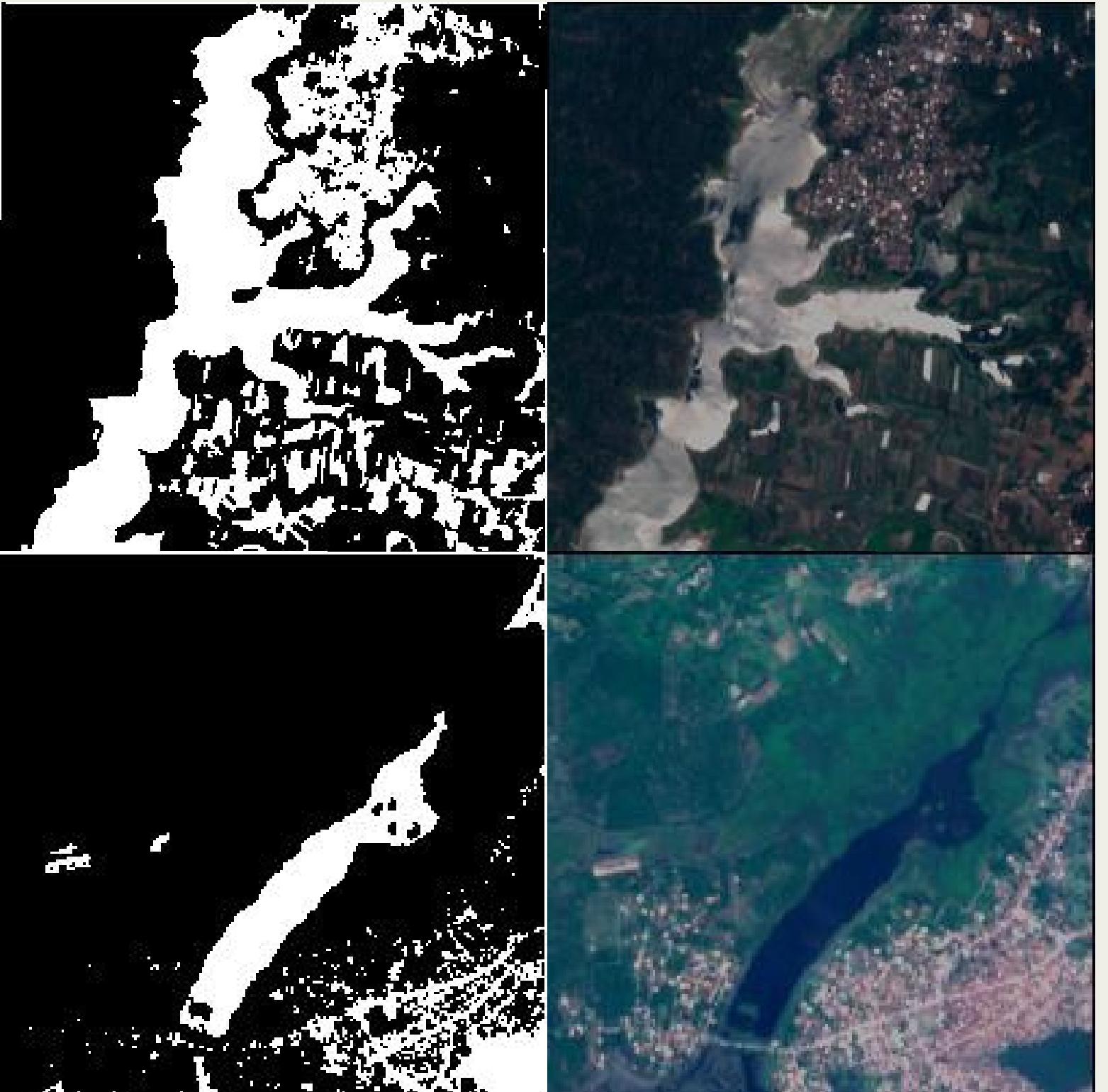


## Yanlış Maskelenme Nedenleri

Veri setinde, bazı görüntülerde yerleşim yerleri, yollar veya farklı yüzey türleri yanlışlıkla su kütleleri olarak maskelenmiştir. Bu durumun başlıca nedenleri şunlardır:

1. Spektral Benzerlikler:
2. Yüksek Eşik Değeri Kullanımı:
3. Atmosferik Koşullar:
4. Otomatik Maskeleme Yöntemi:

Sonuç olarak, NWDI tabanlı maskeleme yöntemi su tespiti için etkili olsa da tek başına yeterli değildir. Daha yüksek doğruluk elde edebilmek için ek indeksler (örn. NDVI, MNDWI) veya makine öğrenmesi tabanlı yöntemler ile desteklenmesi önerilmektedir.



# Kullanılan Yöntemler (Modeller)

---

Projede farklı yapay zekâ tabanlı segmentasyon modelleri denendi:

## 1. SegFormer (B3 ve B5)

- Transformer tabanlı gelişmiş segmentasyon modeli
- Uydu görüntülerinde yüksek doğruluk

## 2. ResNet50 + Unet

- CNN tabanlı klasik yaklaşım
- Daha düşük karmaşıklık, hızlı eğitim

## 3. ResNet101 + Unet++

- Daha derin mimari, daha fazla parametre
- İnce detayları daha iyi yakalama

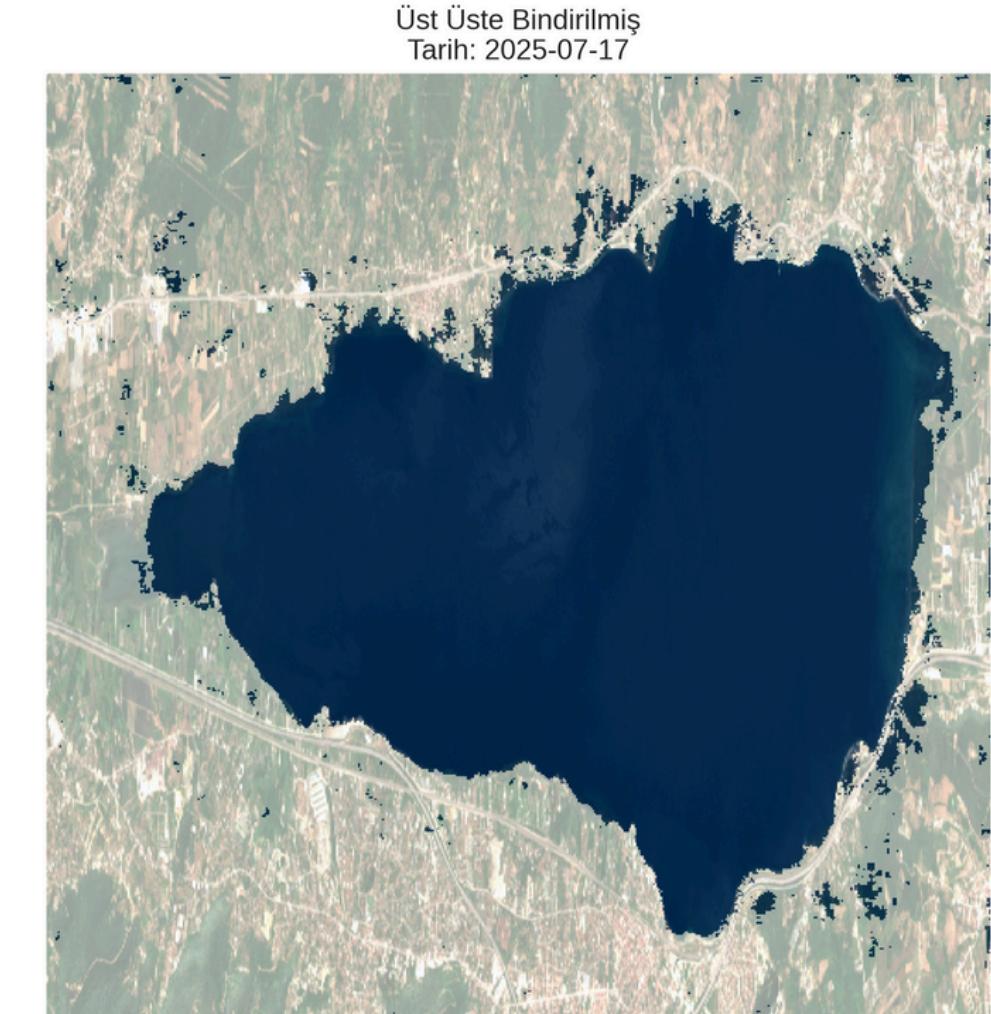
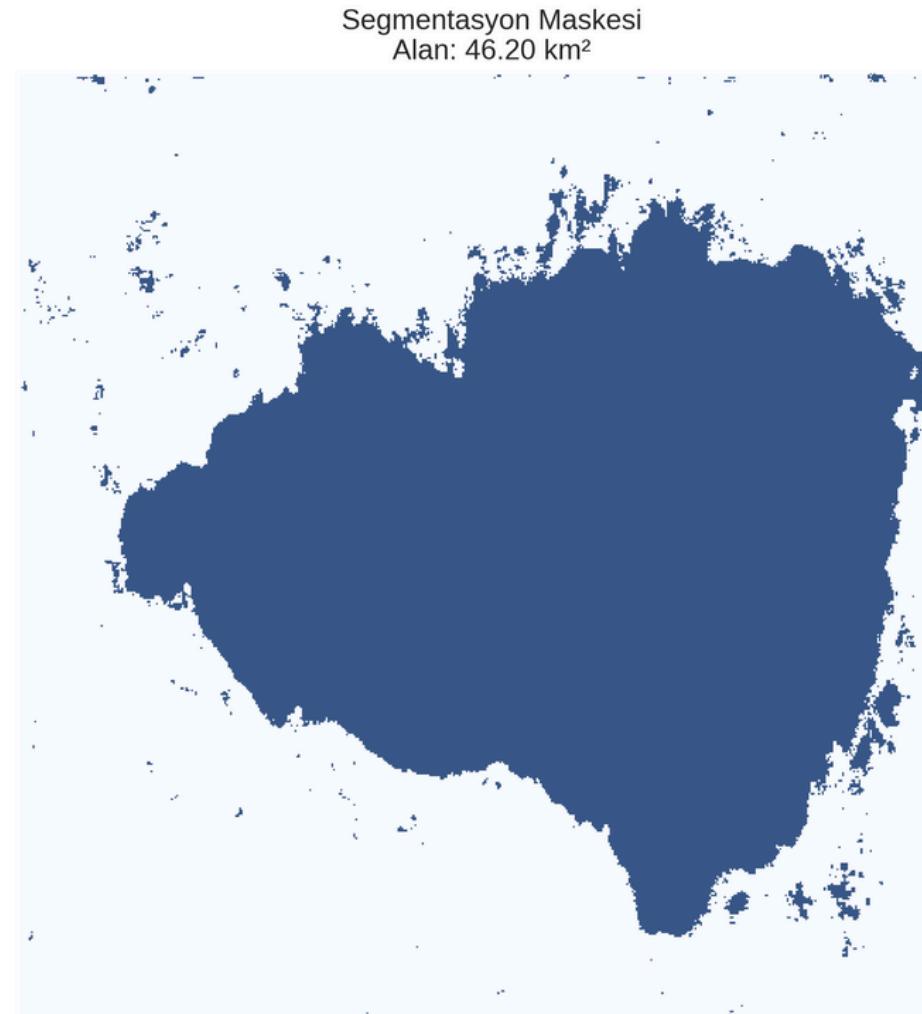
## 4. SAM (Segment Anything Model)

- Meta AI tarafından geliştirildi
- Zero-shot segmentasyon yeteneği
- Uydu görüntülerinde göl/su segmentasyonu yapılabildi

# ResNet50 + Unet

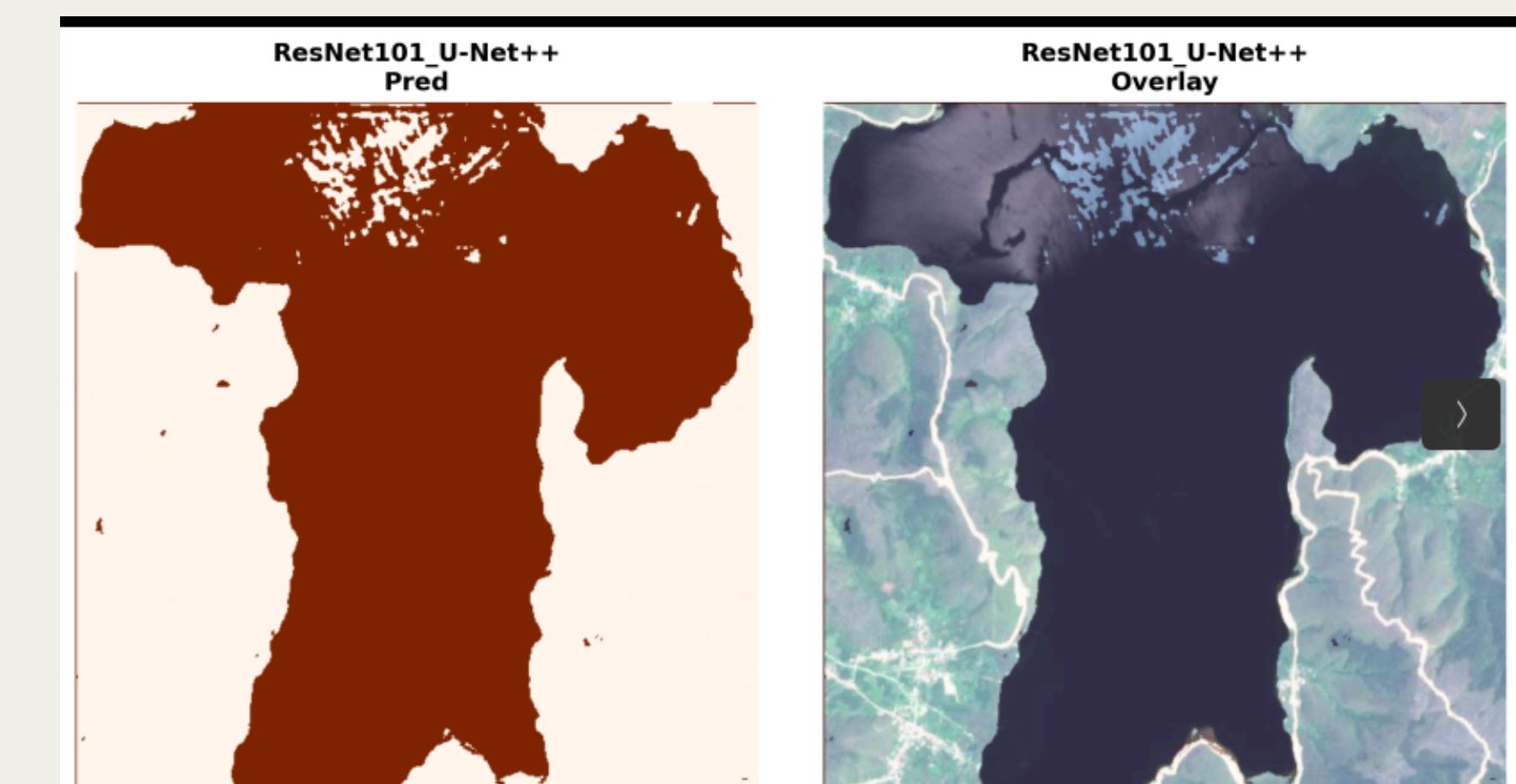
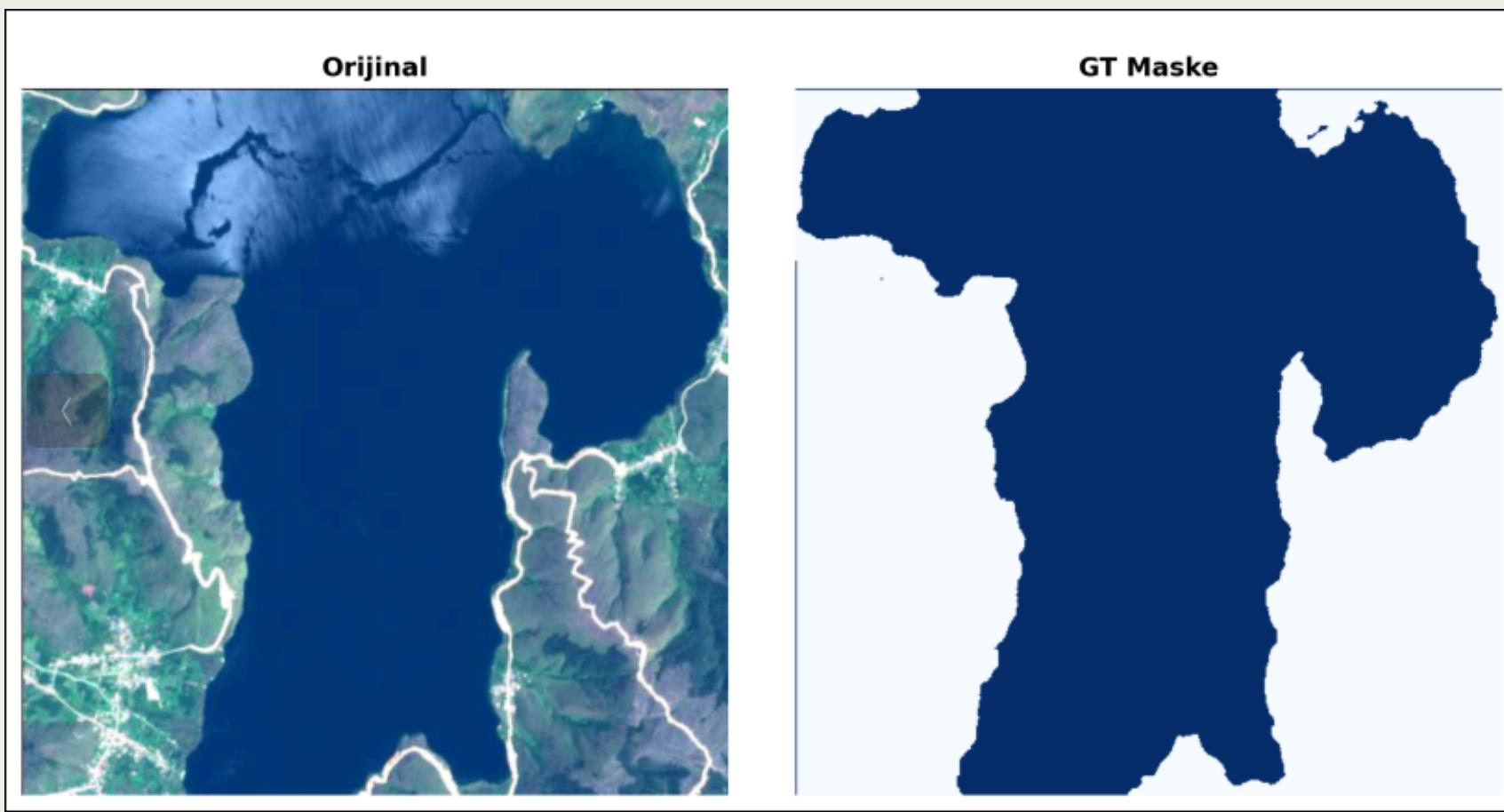
---

- Temel bir CNN (Convolutional Neural Network) mimarisi
- Encoder: ResNet50
- Decoder: Klasik Unet yapısı
- Eğitim süresi kısa, kaynak tüketimi az
- **Avantaj:** Basitlik ve hız
- **Dezavantaj:** Detaylarda hassasiyet düşüklüğü



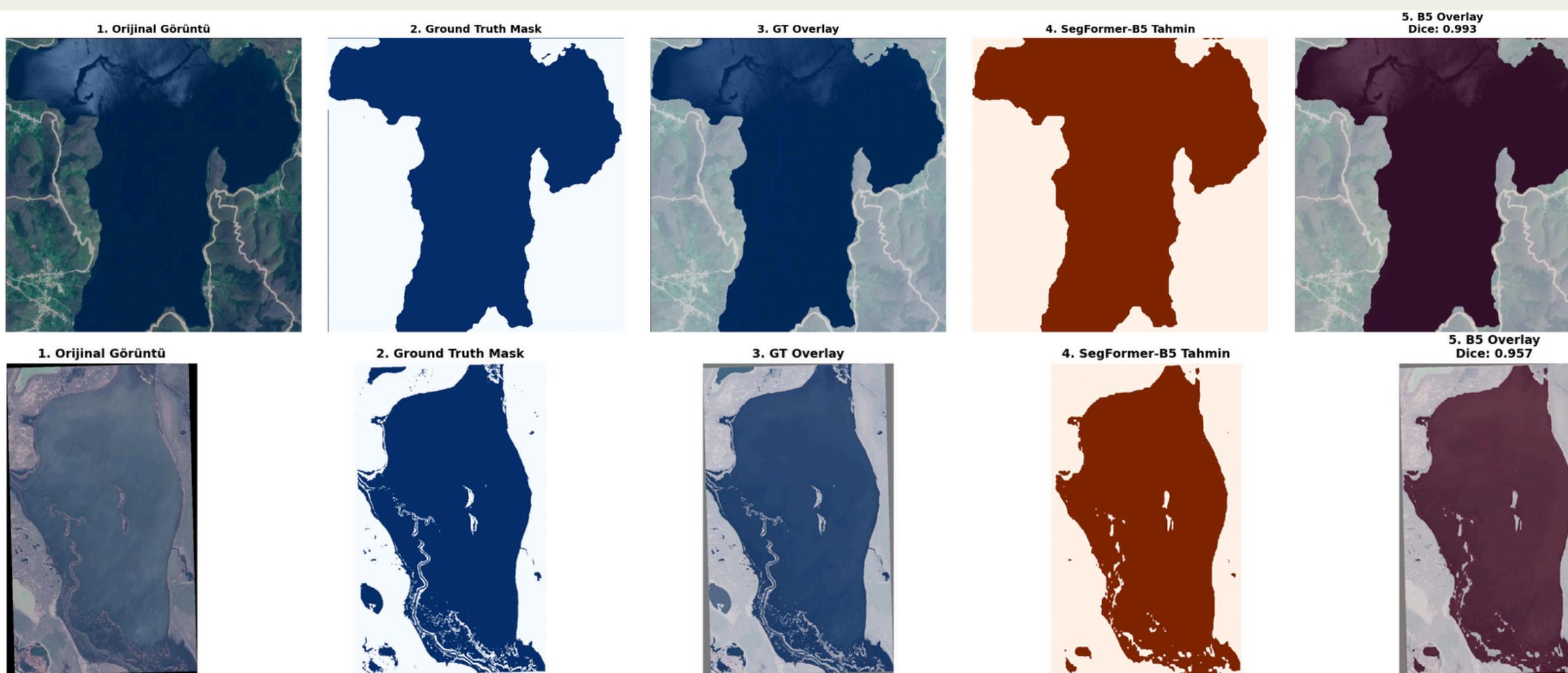
# ResNet101 + Unet++

- Daha derin encoder mimarisi (ResNet101)
- Unet++ ile detayların daha iyi korunması
- Özellikle küçük su kütlelerinde başarımı arttı
- **Avantaj:** Daha doğru kenar tespiti
- **Dezavantaj:** Eğitim süresi daha uzun



# SegFormer-B3/B5

- Transformer tabanlı encoder
- CNN yerine self-attention ile uzamsal bilgiyi daha iyi temsil eder
- Düşük boyutlu versiyon (B3) ile yüksek doğruluk ve hızlı tahmin
- Özellikle büyük su yüzeylerinde genel performans çok iyi idi
- Karmaşıklık artsa da, en yüksek Dice ve IoU skorları bu modelle elde edildi
- **Avantaj:** Yüksek doğruluk
- **Dezavantaj:** Yavaş tahmin süresi ve yüksek kaynak ihtiyacı



# Segment Anything Model (SAM)

- Meta AI tarafından geliştirilen güçlü zero-shot segmentasyon modeli
- Hiç eğitim yapmadan test edildi
- Transfer öğrenme kabiliyetiyle yeni veri setlerine uygulanabilir
- **Avantaj:** Veri gerektirmeden tahmin
- **Dezavantaj:** Hassas segmentasyonda düşüş



# Model Karşılaştırma Tablosu

---

Aşağıda modellerin başarım karşılaştırması verilmiştir:

Model	Loss	Backbone	Param (M)	IoU ↑	Dice ↑	Prec ↑	Rec ↑	Boundary F1 ↑	AP ↑	ECE ↓
ResNet50+UNet	BCE + Dice	UNet	29,803,074	0.7499	0.8571	0.8916	0.8252	0.4818	0.8839	0.0258
ResNet101+UNet+	BCE + Dice	UNet++	56,465,986	0.7594	0.8633	0.8809	0.8463	0.4923	0.9077	0.0190
SegFormer-B3	BCE + Dice	Nvidia/mit-b3	13,677,762	0.7597	0.8634	0.9012	0.8287	0.4894	0.7212	0.0245
SegFormer-B5	Focal + Dice	Nvidia/mit-b5	84,594,113	0.9095	0.9198	0.9653	0.9393	0.7442	0.9693	0.6024

# Karşılaştığım Hatalar ve Çözümlerim

---

## Zorluk:

Bazı .tif uzantılı uydu görselleri Rasterio veya PyTorch tarafından okunamıyordu.

## Çözüm:

- Rasterio ile yeniden dönüştürme yaparak uyumlu hale getirdim
- GeoTIFF metadata'sını kontrol edip transform, crs, dtype gibi alanları normalize ettim

```
with rasterio.open("example.tif") as src:  
    img = src.read([1, 2, 3]) # RGB kanallar  
    transform = src.transform  
    crs = src.crs
```

# Karşılaştığım Hatalar ve Çözümlerim

---

## Zorluk:

Her model için aynı metrikleri üretmekte ve özellikle **boundary-F1** gibi özel metrikleri hesaba katmakta zorluk yaşadım.

## Çözüm:

- Tüm modeller için IoU, Dice, Precision, Recall hesaplayan bir benchmark script'i oluşturdum
- Skimage segmentation modülünden find\_boundaries kullanarak boundary-F1 hesaplaması ekledim

```
def iou(pred, target):  
    intersection = (pred & target).float().sum()  
    union = (pred | target).float().sum()  
    return (intersection + 1e-6) / (union + 1e-6)  
  
def dice(pred, target):  
    smooth = 1e-6  
    return (2 * (pred * target).sum() + smooth) / \  
        ((pred + target).sum() + smooth)
```

# Karşılaştıığım Hatalar ve Çözümlerim

---

## Zorluk:

SegFormer eğitiminde bazı görüntüler için eşleşen maske bulunamıyor uyarısı aldım.  
[UserWarning: 2 görüntü için maske bulunamadı ve atlandı.]

## Çözüm:

- os.listdir kullanımıyla veri eşleşmesini manuel kontrol ettim
- Maske olmayanları otomatik atlayacak şekilde kodu güncelledim

```
image_files = sorted(os.listdir("dataset/Images"))
mask_files = sorted(os.listdir("dataset/Masks"))

valid_pairs = [
    (img, msk) for img, msk in zip(image_files, mask_files)
    if os.path.exists(f"dataset/Masks/{msk}")
]

print(f"{len(valid_pairs)} eşleşen veri bulundu.")
```

# Karşılaştığım Hatalar ve Çözümlerim

---

## Zorluk:

Tahmin maskesi ile uydu görüntüsünü overlay etmekte sorun yaşadım, boyutlar uyuşmuyordu.

## Çözüm:

- Tahmin çıktısını resize ile orijinal görüntü boyutuna getirdim
- matplotlib ile 3 panel oluşturup mask, orijinal ve birleşik overlay görsellerini hazırladım

```
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(original_img.transpose(1, 2, 0))
axs[0].set_title("Orijinal")
axs[1].imshow(pred_mask, cmap='Blues')
axs[1].set_title("Tahmin Maskesi")
axs[2].imshow(original_img.transpose(1, 2, 0))
axs[2].imshow(pred_mask, cmap='Reds', alpha=0.4)
axs[2].set_title("Overlay")
plt.show()
```

# Karşılaştığım Hatalar ve Çözümlerim

---

## Zorluk:

Segmentasyon sonrası ikili maske oluşturmak için sabit bir `thresh=0.5` değeri veriyordum ama bazı örneklerde bu iyi çalışmayıordu.

## Çözüm:

- Otsu Thresholding algoritmasını entegre ettim
- Her görüntüye göre otomatik eşik değeri belirledim

```
from rasterio.transform import Affine

def calculate_area(binary_mask, transform: Affine):
    pixel_area_m2 = abs(transform.a) * abs(transform.e)
    total_area_m2 = binary_mask.sum() * pixel_area_m2
    return total_area_m2 / 1e6 # km2
```

# Karşılaştığım Hatalar ve Çözümlerim

---

## Zorluk:

SegFormer-B5 ve Unet++ gibi modeller çok büyük olduğundan eğitimi uzun sürdü, GPU RAM yetmedi.

## Çözüm:

- Batch size'ı 4–8 aralığında optimize ettim
- GPU otomatik seçim fonksiyonu ile uygun GPU'yu tercih ettim (nvidia-smi bazlı)

```
def get_free_gpu():  
    smi_output = subprocess.check_output("nvidia-smi --query-gpu=memory.free --format=csv"  
    free_memories = [int(x.split()[0]) for x in smi_output.strip().split('\n')[1:]]  
    return free_memories.index(max(free_memories))  
  
selected_gpu = get_free_gpu()  
os.environ["CUDA_VISIBLE_DEVICES"] = str(selected_gpu)
```

# Karşılaştığım Hatalar ve Çözümlerim

---

## Zorluk:

- Earth Engine üzerinde büyük göllerin Sentinel-2 görüntülerini indirirken “Request too large” (istek çok büyük) uyarısı aldım.
- Bu sorun özellikle Beyşehir Gölü gibi geniş alanları kapsayan polygonlarda ortaya çıktı.
- Uydu görüntüsünün çözünürlüğü (scale) çok küçük kaldığında, hedef alanın piksel sayısı sınırı aşılıyordu.
- Scale değerini ayarlamak bile yeterli olmayınca bölge çok büyük kaldı ve Export.image.toDrive() başarısız oldu.

## Çözüm:

- İndirilen görsellerin scale parametresini  $10 \rightarrow 30$  metre gibi değerlere yükselttim.
- Bu sayede daha düşük çözünürlüklü ama indirilebilir boyutta görüntüler elde ettim.
- Ayrıca gerekirse polygonları alt parçalara ayırarak daha küçük bölgeler halinde indirdim.
- Göl polygon’unu kare grid'lere ayırarak, her biri için ayrı Export işlemi gerçekleştirdim.
- Earth Engine üzerindeki piksel sınırlamasını bu şekilde aştım.

# Karşılaştıığım Hatalar ve Çözümlerim

```
import ee
ee.Initialize()

# Beyşehir Gölü'nün yaklaşık sınırlarını temsil eden örnek bir polygon
beysehir_polygon = ee.Geometry.Polygon([
    [31.2, 37.7], [31.2, 37.9], [31.6, 37.9], [31.6, 37.7], [31.2, 37.7]]
])

# Grid (parça) boyutu derece cinsindendir.
# dx: enine (longitude) uzunluk, dy: boyuna (latitude) yükseklik
dx = 0.1
dy = 0.1

def create_grid(region, dx, dy):
    """
    Verilen bir polygon'u dx ve dy boyutlarında dikdörtgenlere böler.
    """
    bounds = region.bounds().coordinates().getInfo()[0]
    xmin, ymin = bounds[0]
    xmax, ymax = bounds[2]

    grid_list = []
    x = xmin
    while x < xmax:
        y = ymin
        while y < ymax:
            grid = ee.Geometry.Rectangle([x, y, min(x + dx, xmax), min(y + dy, ymax)])
            grid_list.append(grid)
            y += dy
        x += dx
    return grid_list

# Polygon'u grid'lere böl
tiles = create_grid(beysehir_polygon, dx, dy)

# Sentinel-2 uydu görüntüsü koleksiyonu alınır
collection = ee.ImageCollection("COPERNICUS/S2_SR") \
    .filterDate("2022-06-01", "2022-06-30") \
    .filterBounds(beysehir_polygon) \
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10)) \
    .median()

# Her tile için görüntüyü ayrı ayrı dışa aktar
for i, tile in enumerate(tiles):
    task = ee.batch.Export.image.toDrive({
        'image': collection,
        'description': f'beysehir_tile_{i}', # İndirilecek dosya adı
        'region': tile,
        'scale': 10, # 10 metre çözünürlük
        'maxPixels': 1e13 # Earth Engine sınırını yükseltme
    })
    task.start()
    print(f"Başlatıldı: Tile {i}")
```

## Daha Fazla Ne Yapılabilir

---

- Daha fazla göl üzerinde testlerin genişletilmesi
- Su kalitesi parametrelerinin (klorofil-a, bulanıklık vb.) tahmini
- Parametre optimizasyonu (loss fonksiyonları, learning rate tuning)
- Modellerin ensemble (birleştirilmiş) şekilde kullanılması
- Raporlama ve karar destek sistemine entegrasyon

# Tesekkürler

---

AHMET NURI EROĞLU

23120205087