OSNOVI GEOINFORMATIKE

Tema: Semantička Segmentacija Vodenih Površina pomoću DeepLabV3 Arhitekture

---

1. Cilj Projekta

Cilj ovog projekta je razvoj i implementacija sistema za automatsku detekciju i ekstrakciju vodenih površina sa ortofoto snimaka korišćenjem metoda dubokog učenja. Za zadatak semantičke segmentacije, gde se svakom pikselu na slici dodeljuje određena klasa (u ovom slučaju "vodena površina" ili "pozadina"), korišćena je napredna DeepLabV3 arhitektura. Krajnji rezultat projekta nije samo rasterska maska generisana od strane modela, već i njena konverzija u vektorski format (GeoJSON), čime se omogućava dalja primena i analiza u standardnim GIS alatima.

---

2. Priprema i Predprocesiranje Podataka

Kvalitet ulaznih podataka je preduslov za uspešno treniranje modela. Proces pripreme se sastojao iz nekoliko ključnih faza.

2.1. Kreiranje Obučavajućeg Skupa

Kao osnova su korišćeni ortofoto snimci i odgovarajući vektorski podaci (npr. .shp fajlovi) koji precizno definišu granice vodenih površina. Pošto model radi sa rasterskim podacima (slikama), bilo je neophodno izvršiti **rasterizaciju** vektorskih podataka. Ovaj proces konvertuje poligone iz .shp fajla u binarne rasterske maske, gde pikseli unutar poligona imaju vrednost 1 (voda), a svi ostali vrednost 0 (pozadina).

2.2. Augmentacija i Normalizacija Podataka

Da bi se povećala robusnost modela i sprečilo prekomerno prilagođavanje (overfitting), primenjene su tehnike augmentacije podataka pomoću biblioteke Albumentations. Ove tehnike veštački proširuju trening skup primenom nasumičnih transformacija.

Ključni koraci predprocesiranja, enkapsulirani u transformacionu putanju, su:

- **Promena veličine:** Sve slike i maske se svode na fiksnu veličinu (npr. 512x512).
- **Geometrijske transformacije:** Slučajne rotacije i horizontalna/vertikalna okretanja.
- **Normalizacija:** Vrednosti piksela se skaliraju na standardni opseg, što je ključno za stabilizaciju procesa treniranja.

Važan deo koda: Definicija transformacija za trening skup

code Python
```
import albumentations as A
```

```
from albumentations.pytorch import ToTensorV2

train_transform = A.Compose(
    [
        A.Resize(512, 512),
        A.Rotate(limit=35, p=0.5),
        A.HorizontalFlip(p=0.5),
        A.VerticalFlip(p=0.5),
        A.Normalize(
            mean=[0.485, 0.456, 0.406],
            std=[0.229, 0.224, 0.225],
            max_pixel_value=255.0,
        ),
        ToTensorV2(),
    ],
)
```

---

3. Arhitektura Modela i Proces Treniranja

3.1. Arhitektura Modela – DeepLabV3

Za zadatak je odabrana DeepLabV3 arhitektura sa ResNet50 osnovom (backbone). Nije se pristupilo treniranju od nule, već je primenjen **transfer learning**. Korišćen je model prethodno istreniran na COCO skupu podataka, a zatim je izvršeno fino podešavanje (fine-tuning) za naš specifičan problem. To podrazumeva zamenu poslednjeg, klasifikacionog sloja modela novim slojem koji odgovara broju naših klasa (2 – voda i pozadina).

Važan deo koda: Funkcija za kreiranje modela

```
code Python
  import torch
import torchvision.models as models

def create_deeplabv3_model(num_classes=2):
    """
    Funkcija kreira DeepLabV3 model sa ResNet50 osnovom i
    prilagođava ga za naš broj klasa.
    """
    model = models.segmentation.deeplabv3_resnet50(
        weights=models.segmentation.DeepLabV3_ResNet50_Weights.DEFAULT
    )
    # Menjamo poslednji sloj
    model.classifier[4] = torch.nn.Conv2d(256, num_classes, kernel_size=(1, 1), stride=(1, 1))
    return model
```

3.2. Proces Treniranja

Model je treniran korišćenjem sledećih hiperparametara:

- **Optimizator:** Adam
- **Funkcija gubitka:** CrossEntropyLoss
- **Stopa učenja (Learning Rate):** 1e-4
- **Veličina grupe (Batch Size):** 12
- **Broj epoha:** 25

Tokom treniranja, javio se problem sa BatchNorm slojevima koji ne mogu da obrade grupu podataka veličine 1. Problem je rešen postavljanjem argumenta drop_last=True u DataLoader-u, čime se poslednja, nekompletna grupa podataka u epohi ignoriše.

---

4. Predikcija i Evaluacija Rezultata

Nakon treniranja, model je sačuvan kao .pth fajl i korišćen za predikciju na novim, velikim ortofoto snimcima.

4.1. Predikcija na Velikim Slikama

Pošto su ulazni snimci veći od dimenzija koje model prihvata (512x512), implementiran je **"sliding window"** pristup. Velika slika se deli na manje isečke (patches) koji se preklapaju. Model vrši predikciju na svakom isečku, a rezultati se zatim spajaju u finalnu masku pune rezolucije.

4.2. Vizuelizacija Rezultata

Za jasnu evaluaciju performansi, kreirana je funkcija koja generiše sliku sa četiri panela:

1. **Originalna Slika:** Ulazni ortofoto snimak.
2. **Stvarna Maska (Ground Truth):** Tačna, ručno kreirana maska.
3. **Predikcija Modela:** Crno-bela maska koju je generisao model.
4. **Anotirana Predikcija:** Originalna slika preklopljena sa predikcijom modela, gde su detektovane vodene površine obojene upečatljivom, polutransparentnom plavom bojom radi lakše vizuelne inspekcije.

Važan deo koda: Kreiranje anotirane slike (overlay)

```
code Python
   import cv2
import numpy as np

# 'large_image' je originalna slika, 'prediction_map' je maska (0/1)
# Kreiranje sloja u boji za anotaciju
overlay = np.zeros_like(large_image, dtype=np.uint8)
# Gde je predikcija 1 (voda), bojimo u plavo (BGR format)
overlay[prediction_map == 1] = [255, 144, 30] # Boja 'dodgerblue'
```

```
# Spajanje originalne slike i sloja sa anotacijom
alpha = 1.0  # Težina originala
beta = 0.6   # Težina anotacije (providnost)
annotated_image = cv2.addWeighted(large_image, alpha, overlay, beta, 0)
```

---

5. Postprocesiranje: Konverzija u Vektorski Format

Finalni i ključni korak projekta je prevođenje rezultata iz domena obrade slika u domen geoinformatike. Rasterska maska koju model generiše je samo slika; da bi bila korisna za prostornu analizu, mora se pretvoriti u vektore.

5.1. Vektorizacija

Procesom vektorizacije, grupe piksela koje predstavljaju vodene površine se konvertuju u **poligone**. Ovaj proces je izvršen pomoću biblioteka rasterio i geopandas. Svaki poligon je definisan nizom geografskih koordinata nasleđenih iz georeferencijskih informacija originalnog snimka.

5.2. Primena Rezultata (.geojson)

Rezultujući poligoni su sačuvani u **GeoJSON** formatu. Ovaj format je standard u GIS svetu i omogućava:

- **Učitavanje u GIS softver (QGIS, ArcGIS):** Za vizuelizaciju, preklapanje sa drugim prostornim podacima i dalju analizu.
- **Precizna merenja:** Izračunavanje tačne površine i obima svake detektovane vodene površine.
- **Prostorni upiti:** Odgovaranje na kompleksna pitanja poput "Kolika je ukupna površina vode u ovoj opštini?" ili "Koje se parcele graniče sa vodom?".

Važan deo koda: Funkcija za konverziju rastera u vektor

```
code Python
  import rasterio
import geopandas as gpd
from rasterio.features import shapes

def raster_to_vector(raster_path, output_vector_path):
    with rasterio.open(raster_path) as src:
        image = src.read(1)
        mask = image == 1 # Maska gde je vrednost 1 (voda)

        # Generisanje poligona iz maske
        results = (
            {'properties': {'raster_val': v}, 'geometry': s}
            for i, (s, v) in enumerate(shapes(image, mask=mask, transform=src.transform))
```

```
    )
    geometries = list(results)

    if geometries:
        gdf = gpd.GeoDataFrame.from_features(geometries)
        gdf.set_crs(src.crs, inplace=True)
        gdf.to_file(output_vector_path, driver='GeoJSON')
```

---

6. Zaključak

Kroz ovaj projekat uspešno je demonstriran kompletan proces primene dubokog učenja na problem iz geoinformatike. Od sirovih ortofoto snimaka, preko treniranja DeepLabV3 modela, do generisanja finalnog, analitički upotrebljivog GeoJSON fajla, pokazano je kako se moderna AI rešenja mogu integrisati sa tradicionalnim GIS alatima za efikasno rešavanje realnih problema.