Mikołaj Czachorowski 22.01.2025

Sprawozdanie Przetwarzanie Obrazów i Widzenie Komputerowe

Wstęp

Celem projektu było stworzenie systemu segmentacji obiektów, który automatycznie identyfikuje i klasyfikuje różne rodzaje odpadów na podstawie zdjęć. Projekt wykorzystuje zaawansowane techniki głębokiego uczenia, w tym modele oparte na architekturze ResUNet oraz biblioteki takie jak PyTorch, ONNX i Streamlit. Sprawozdanie opisuje główne założenia projektu, wykorzystane technologie, napotkane wyzwania oraz wyniki jakościowe modelu na przygotowanym zbiorze danych.

Dane

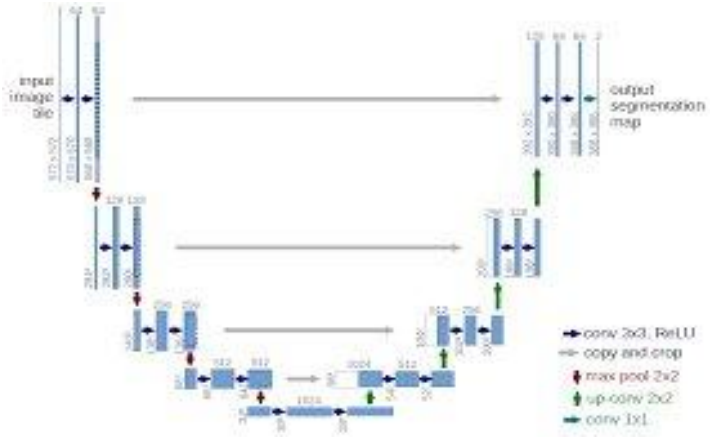
W celu wyuczenia modelu zostały użyte dane ze zbioru: link. Ten zbiór zawiera 7 212 obrazów z adnotacjami w formacie segmentacji instancji. Obrazy przedstawiają śmieci, zdalnie sterowane pojazdy (ROV) oraz różnorodną florę i faunę morską. Adnotacje w zbiorze to maski bitmapowe, które wskazują piksele należące do poszczególnych obiektów. Obrazy pochodzą z datasetu J-EDI, opracowanego przez japońską agencję JAMSTEC, obejmującego nagrania z ROV-ów operujących głównie w Morzu Japońskim od 1982 roku. TrashCan dostępny jest w dwóch wersjach: TrashCan-Material i TrashCan-Instance, różniących się konfiguracją klas obiektów. Do uczenia mojego modelu użyłem wersji TrashCan-Material, w której występuje 16 klas obiektów.

Przetwarzanie danych

Do przetwarzania danych w projekcie ograniczyłem się do dwóch głównych kroków. Pierwszym z nich było przycięcie obrazów do rozmiaru 256x256 pikseli. Drugim krokiem było wygenerowanie masek na podstawie dostępnych anotacji, które stanowiły wymagane wyjście modelu. Maski te zawierały informacje o poszczególnych obiektach w obrazie, umożliwiając skuteczną segmentację na poziomie pikseli.

Model

Wybrałem model ResUNet - autoenkoder z residual connections, mający wiele zastosowań.



Moja implementacja modelu ResUNet

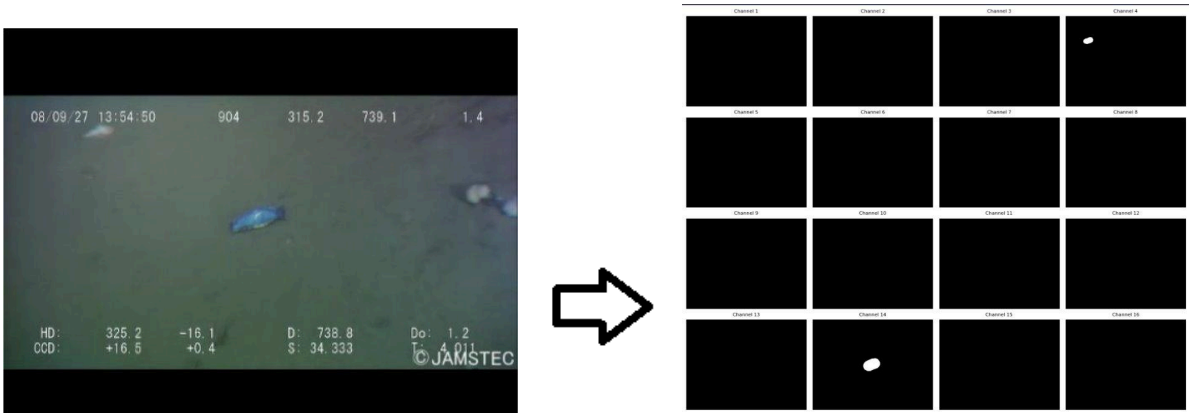
Moja implementacja modelu ResUNet bazuje na strukturze autoenkodera z połączeniami skip, co pozwala na skuteczną segmentację obrazów. Model składa się z pięciu warstw enkodera, w których używane są bloki konwolucyjne z dwoma warstwami Conv2d, a następnie z warstw dekodera, gdzie po każdej operacji upconvolution następuje połączenie z odpowiednią warstwą enkodera, co umożliwia zachowanie ważnych szczegółów przestrzennych. Takie podejście zostało opisane w literaturze, np. w pracy Aloma i współpracowników (2019), gdzie autorzy zaprezentowali zastosowanie sieci ResUNet w medycznej segmentacji obrazów. Do trenowania modelu zastosowano funkcję strat DiceLoss, która jest szeroko stosowana w segmentacji obrazów (Zhao et al., 2020). Do optymalizacji użyto algorytmu AdamW, który wykazuje lepszą wydajność w porównaniu do tradycyjnego Adam w zadaniach związanych z trenowaniem głębokich sieci neuronowych (Llugsi et al., 2021).

Model ResUNet fine-tunowany

W celu porównania wyników, postanowiłem wytrenować na danych model ResUNet prtrenowany na zbiorze danych ImageNet, wykorzystując model jako bazowy enkoder w architekturze UNet. Model został fine-tunowany na zadaniu segmentacji, gdzie w pierwszym etapie zablokowano aktualizację wag enkodera, co pozwoliło na szybsze dostosowanie modelu do specyfiki danych Takie podejście umożliwia wykorzystanie już nabytych przez model cech z ImageNet, co może przyczynić się do lepszych wyników w porównaniu do trenowania od podstaw.

Dane wejściowe i wyjściowe

W obu modelach na wejściu i wyjściu model dostaje tensor o kształcie BATCH X CHANNELS X HEIGHT X WIDTH. Na wejściu dostaje batch obrazów które posiadają 3 kanały (rgb), wysokość 256 i szerokość 256. Natomiast na wyjściu batch obrazów-masek, które posiadają 17 kanałów (dlatego, że jest 16 klas i 1 dodatkowy kanał odpowiedzialny za tło).



Trening i obserwacje

Ze względu na złożoność modelu (16 klas) oraz ograniczone zasoby obliczeniowe, nie udało się osiągnąć wystarczającej efektywności w pełnej segmentacji obiektów. Niemniej jednak, model po fine-tuningu sprawdził się dobrze w zadaniu wykrywania obiektów, choć nie udało się uzyskać poprawnych wyników w klasyfikacji poszczególnych klas. Mimo tego, model skutecznie oddziela obiekty od tła, co stanowi istotny krok w procesie segmentacji. W celu poprawy wyników klasyfikacji, można by zastosować dodatkowy klasyfikator, który przypisywałby odpowiednią klasę do wykrytego obiektu.

Interfejs

Przygotowałam interfejs z streamlit posiadający następujące funkcjonalności:

- **Przeglądanie obrazów**: Nawigacja między obrazami za pomocą przycisków "Previous" i "Next".
- **Generowanie segmentacji**: Tworzenie nakładki segmentacyjnej na obrazie, z wyróżnieniem obiektów, za pomocą modelu wczytanego za pomocą onnx.
- **Podgląd wyników**: Wyświetlanie oryginalnego obrazu oraz segmentacji w formie kolorowego overlayu.
- **Pobieranie przetworzonego obrazu**: Możliwość pobrania segmentowanego obrazu w formacie PNG.

Przykładowy wygląd interfesju

Trashcan Segmentation

Original Image (Index: 0)

Original Image

Previous Image

Next Image

Generate Segmentation

Segmented Overlay

Overlay with Segmentation

Download Segmented Image

Wyniki

Poniżej przedstawiłem wizualizację przykładowych wyników z aplikacji.