Lokalizacja pojazdów na zdjęciach satelitarnych

Raport

Mateusz Bieńkowski

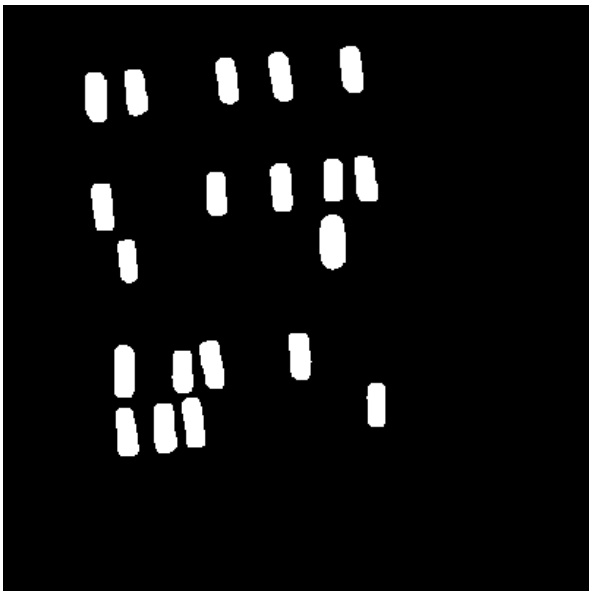
Opis problemu badawczego

Zajmowany prze zemnie problem polega na przygotowaniu rozwiązania, które na podstawie danych wejściowych przygotowuje maskę lokalizującą obszar zajmowany przez samochody. Oprócz przygotowania rozwiązania realizującego dane zadanie zostanie sprawdzone to jak dokładne wyniki uda się uzyskać przy pomocy **Sztucznych Sieci Neuronowych** uczonych na bardzo małym zbiorze (zbiór treningowy-18 obrazków, zbiór walidacyjny 4 obrazki oraz zbiór testowy 5 obrazków), gdzie przy takich problemach liczność takich zbiorów mierzy się w tysiącach a nawet w setkach tysięcy.

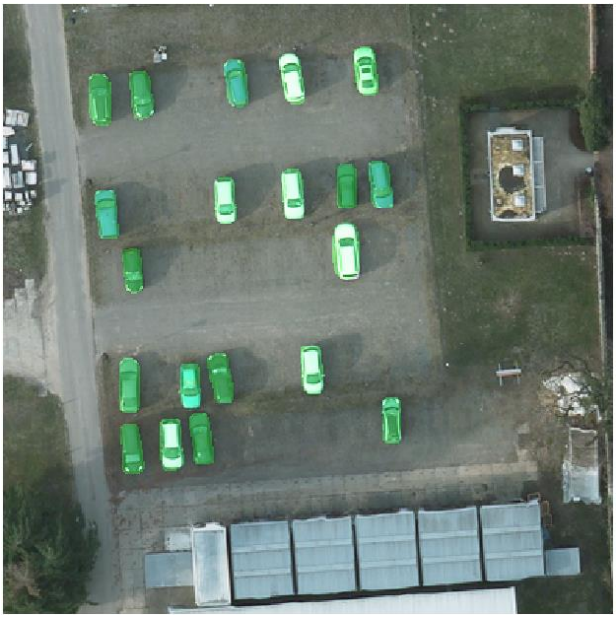
Cel projektu

Uzyskanie jak najlepszego rezultatu w lokalizacji pojazdów na zdjęciach satelitarnych.

Rysunek 1. Przykładowe wejście



Rysunek 2. Maska obszarów zajmowanych przez samochody (spodziewane wyjście sieci)

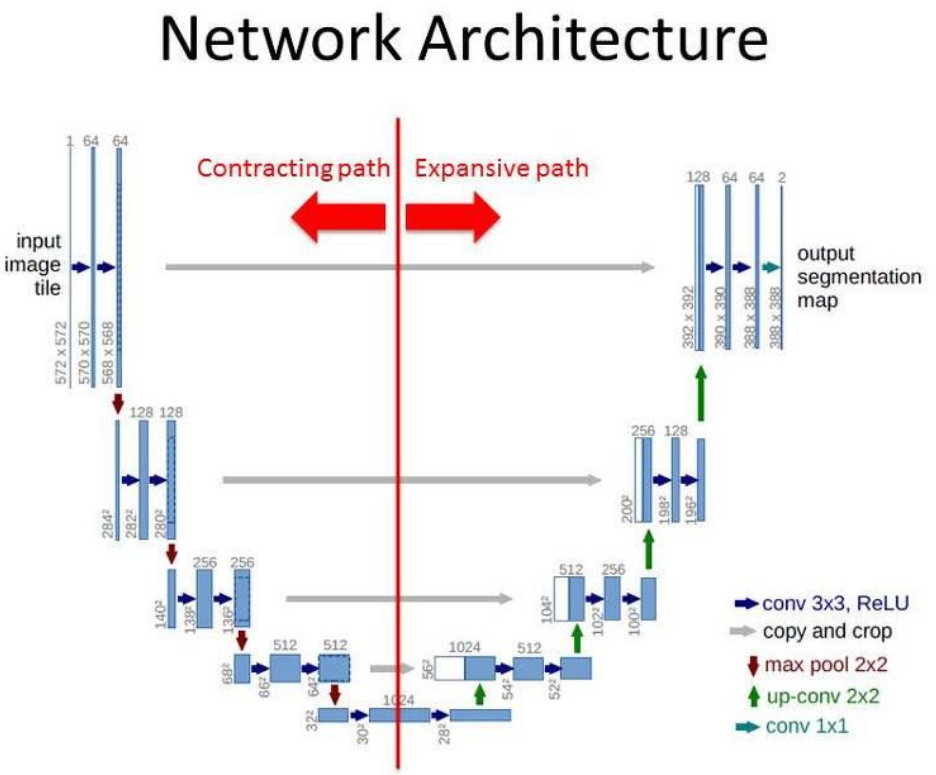


Rysunek 3. Wejście z nałożoną maską - oczekiwany wynik

Wykorzystane narzędzia i techniki

Rozwiązanie zostało zaimplementowane w języku ***Python*** do tworzenia sieci neuronowej została wykorzystana biblioteka ***Keras*** wykorzystująca w swej implementacji ***TensorFlow.***

Pierwotnie realizacja zadania miała być oparta o wykład dr inż. Artura Nowakowskiego prowadzony na przedmiocie „Analiza i rozpoznawanie danych obrazowych”. W tym podejściu wejściowy obraz miał być dzielony na małe fragmenty, które następnie miały zostać zaklasyfikowane, jako samochód lub jako tło (wszystko poza samochodem). Jednakże podczas zbierania i szukania informacji o podobnych problemach (generowania odpowiednich masek na obrazy wejściowe, problem taki zalicza się również do problemów segmentacji) natknąłem się na artykuł, w którym do takiego rodzaju problemu została użyta sieć neuronowa, o architekturze **U-Net**.



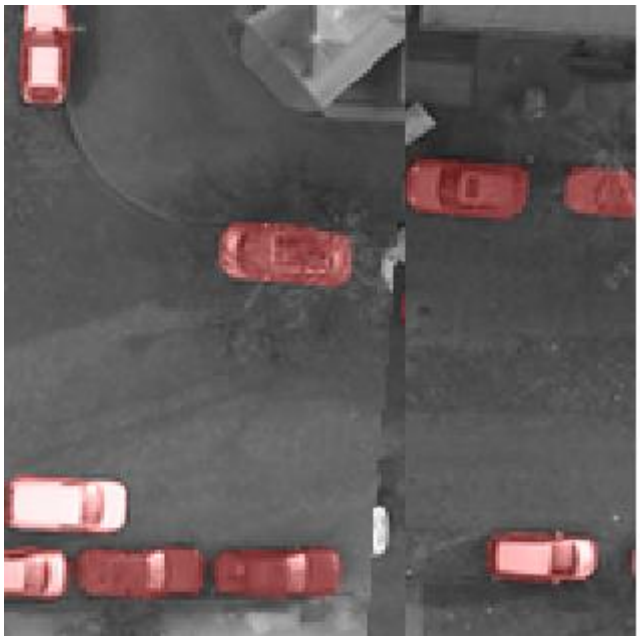
Rysunek 4. Schemat architektury U-Net

U-Net jest to splotowa sieć neuronowa opracowana głównie dla segmentacji obrazków biomedycznych na Uniwersytecie w Freiburgu w Niemczech. Architektura ta była zmodyfikowana pod kontem pracy ze zbiorami mało licznymi oraz aby dawała precyzyjne wyniki segmentacji, obie wspomniane cechy doskonale pasują do podjętego prze zemnie tematu, dlatego do jego realizacji została wykorzystana właśnie sieć neuronowa o takiej architekturze.

W celu zwielokrotnienia danych uczących w trakcie działania programu są one generowane poprzez różnego rodzaju modyfikacje oryginalnych danych. Modyfikacje wykorzystane do realizacji tego problemu to losowe obroty oryginalnego obrazu oraz losowe przesunięcia w różne kierunki o losową wartość. Faktem jest to, że taka modyfikacja nie wprowadzała nowych kształtów, które potencjalnie mogłyby być samochodem lub nie, co więcej mogą one zaburzyć rzeczywisty kształt auta, jeżeli takie auto stało na krawędzi obrazu.

Na poniższym rysunku widać artefakty powstałe w wyniku modyfikacji takie jak:

1. Kilku-pixelowe fragmenty oznaczone, jako auto, chociaż w cale auta nie przypominające widoczne w środkowej części obrazka oraz w prawym dolnym rogu (mające poświatę czerwoną)
2. Przez środek przechodzi nienaturalna prosta wynikająca z *zawinięcia* obrazka po jego przesunięciu lub obrotu, ewentualnie złożeniu dwóch tych operacji.



Rysunek 5 Fałszywe dane na zmodyfikowanym obrazie

Opis danych

Dane uczące składają się z par obrazka satelitarnego oraz ręcznie przygotowanej dla tego obrazka maski położeń pojazdów. Obrazki mają wymiary (n x n pikseli, gdzie n = 400).

Do realizacji tego problemu zostało przygotowanych 27 par obrazek-maska. Dane zostały podzielone na zbiór testowy o liczności 5, zbiór treningowy i walidacyjny o łącznej liczności 22 podzielone w stosunku 70% zbiór treningowy i 30% zbiór walidacyjny.

Instrukcja użytkowania aplikacji

Przed uruchomieniem aplikacji należy upewnić się czy w pliku **Params.py** są skonfigurowane odpowiednio dla nas ustawienia:

- **COMITET_MODELS** – tablica ścieżek do plików z wagami modeli w formacie „.h5”
- **COMITET_PREDICT** – (True/False) – informacja czy należy użyć komitetu przy predykcji
- **CREATE_THRESHOLD_TEST_FOR_COMITET** – czy wykonać test komitetu w zależność od progu
- **COMITET_THRESHOLDS** – tablica progów użytych w teście
- **THRESHOLD_TEST_FOR_COMITET_OUT_FILE –** plik wyjściowy testu w formacie csv
- **TEST_MODELS_IN_COMITET** – czy wykonać test dla modeli użytych w komitecie
- **IMAGES_DIR** – katalog z danymi treningowymi
- **TEST_DIR** – Katalog z danymi testowymi
- **MASK_DIR** – katalog z orginalnymi maskami (dla wszystkich zbiorów)
- **MODEL_CHECKPOINT** – plik, w którym zapisywany będą wagi modelu w trakcie nauki
- **MODEL_LOCATION** – plik z zapisanymi wagami modelu (możliwa wartość **None** w takim przypadku wagi będą ustawione losowo)
- **HISTORY_PATH** – miejsce zapisania historii uczenia (plik zapisany w formacie JSON)
- **IMG_WIDTH/IMG_HEIGHT** – szerokość/ wysokość przetwarzanych obrazków (400x400px)
- **BATCH_SIZE** – rozmiar batch
- **THRESHOLD –** domyślny próg wykorzystywany w komitecie

- **PREDICT_DATA_THRESHOLD** – próg wykorzystywany przy liczeniu jakości podczas generowania danych prezentacyjnych w testach np. gdy zaznaczone jest TEST_MODEL = True)
- **PRINT_MODEL** – (True/False) informacja czy model powinien być wypisany w konsoli
- **PLOT_MODEL** - (True/False) informacja czy model ma być zapisany do pliku „model.png”
- **TRAIN_MODEL** – (True/False) informacja czy model ma być trenowany danymi z katalogu wskazanym przez **IMAGES_DIR**
- **TEST_MODEL** – (True/False) informacja o tym czy obrazy z katalogu test mają zostać użyte to weryfikacji modelu.
- **TEST_IMAGE_IDS** – tablica dwu znakowych napisów odpowiadającym numerom obrazkom testowym

Przykładowe uruchomienie:

1. W **Params.py,** jeżeli struktura katalogu nie została zmodyfikowana (jest taka, jaka została wysłana) to wystarczy ustawić zmienną **TEST_MODEL = True** i uruchomić skrypt **AutoFinder.py**
2. Jeżeli struktura plików się zmieniła należy zaktualizować zmienne odpowiadające ścieżką tak, aby odpowiadały aktualnej strukturze katalogu
3. Jeżeli chcemy pokazać wyniki dla komitetu należy ustawić zmienną **COMITET_PREDICT**=**True** a następnie uruchomić skrypt **AutoFinder.py**

Wybrane wyniki

Poniżej zostały przedstawione wyniki uzyskane na zbiorze testowym przez komitet złożony z czterech niezależnie uczonych modeli sieci o takiej samej architekturze przedstawionej we wcześniejszej sekcji.

Thresholds	Image_23	Image_24	Image_25	Image_26	Image_27	AVG	SD
0.30	0.92	0.90	0.89	0.93	0.82	0.89	0.043
0.40	0.91	0.90	0.89	0.94	0.82	0.89	0.046
0.50	0.90	0.90	0.88	0.94	0.81	0.89	0.049
0.60	0.88	0.88	0.87	0.94	0.79	0.87	0.052
średnie	0.90	0.89	0.88	0.94	0.81	0.88	

Tabela 1. Jakość segmentacji komitetu na danych testowych dla poszczególnych progów

Tabela przedstawia dokładność, jaką uzyskał komitet dla poszczególnych danych testowych przy określonym progu. Wartości powyżej progu zostawały uznawane za część auta natomiast, poniżej jako część „tła”.

Z tabeli można odczytać, że najlepsze rezultaty zostały uzyskane dla progów poniżej **0.6,** dla których średnia wartość wyniosła **89%** idealnego dopasowania.

Najgorsza średnia dla wszystkich progów została otrzymana na obrazku „**Image_27”** i wyniosła **81%,** natomiast najlepsza dla obrazka „**Image_26**” i wyniosła **94%**

Rysunek 6. Najsłabiej rozwiązywany obrazek

Rysunek 7. Najlepiej rozwiązywany obrazek

Poniżej zostały przedstawione dane dotyczące modeli użytych w komitecie. Przy klasyfikacji został użyty próg **0.4.**

Model	Image_23	Image_24	Image_25	Image_26	Image_27	AVG	SD
model_2_weights_1.h5	0.89	0.86	0.82	0.89	0.78	0.85	0.047
model_3_weights_1.h5	0.87	0.87	0.87	0.91	0.81	0.87	0.033
model_4_weights_1.h5	0.87	0.85	0.84	0.93	0.78	0.85	0.057
model_5_weights_1.h5	0.88	0.85	0.87	0.90	0.81	0.86	0.036
średnie	0.87	0.86	0.85	0.91	0.79	0.86	

Tabela 2. Jakość segmentacji modeli wchodzących w skład komitetu na danych testowych

Z tabeli można wywnioskować, że poszczególne modele prezentują podobny poziom nauczenia, dodatkowo wszystkie obrazy poza „**Image_27**” są segmentowane na podobnym poziomie (tzn. nie ma znacznych różnic, pomiędzy jakością segmentacji jednego obrazu od drugiego) wyjątek stanowi wspomniany obraz „**Image_27**”, z którym każdy model nie radzi sobie dobrze.

Podsumowując zastosowanie komitetu poprawiło średnią jakości segmentacji w porównaniu z najlepszym indywidualnym modelem o 2 punkty procentowe. W odniesieniu do poszczególnych obrazów testowych wyniki komitetu są w 100% przypadkach lepsze niż w indywidualnych modelach.

Sukcesy i niepowodzenia

Sieć najlepiej sobie radzi z obrazami, na których widać całe auta:

Gdzie niegdzie widać błędy gołym okiem, lecz wynik jest satysfakcjonujący jak na naukę zbiorem danych o tak małej liczności.

Można zauważyć, że sieć nie nauczyła się jeszcze radzić z samochodami, które nie są w pełni widoczne:

Na *rysunku 7*. Auto po lewej nie jest całkowicie widoczne prawdopodobnie przez nachodzące na nie gałęzie drzewa (na tym rysunku nie są one dobrze widoczne, ale po przyjrzeniu jesteśmy w stanie je dostrzec)

Rysunek 8

Rysunek 8 przedstawia auto, przez które częściowo przechodzi jakaś przeszkoda

Rysunek 9

Różnego rodzaju artefakty klasyfikowane, jako auto, choć w rzeczywistości nim nie są.