

---

# **ANALIZA OBRAZÓW I WIDEO - PROJEKT**

## **Detekcja aut i wyznaczanie obszarów parkingowych na zdjęciach lotniczych**

---

**Agata Skibińska, Kornel Romański**  
{238314, 238127}@student.pwr.edu.pl

### **1 Opis problemu**

W projekcie podjęte zostały zagadnienia związane z problemem detekcji obiektów na zdjęciach lotniczych, które podzielić można na dwa etapy:

- detekcja i oznaczenie aut na wejściowym zdjęciu wykonanym z powietrza,
- oznaczenie, z wykorzystaniem przeprowadzonej detekcji aut, obszarów, na których znajdują się miejsca parkingowe.

W projekcie założono, że:

- detekcja dotyczyła będzie głównie aut osobowych,
- detekcja przeprowadzana będzie na zdjęciach wykonanych w taki sposób, że oś kamery ustawiona jest względem fotografowanego obszaru pod kątem zblżonym do kąta prostego,
- rozpatrywane będą zdjęcia wykonane z takiej wysokości i w takiej rozdzielczości, aby auta zajmowały obszar conajmniej 40x40px.

Celem projektu była implementacja i przebadanie zaobserwowanych w literaturze rozwiązań podjętych problemów, gdzie rozwiązania w etapie pierwszym zainspirowane były głównie pracą [2], natomiast w etapie drugim wypróbowane zostało autorskie podejście, będące połączeniem metod występujących w [1] i [3]. Wytworzony kod miał umożliwić przetwarzanie zbiorów zdjęć w celu wytrenowania wybranych modeli, a następnie zdefiniowanie potoku przetwarzania pojedynczego obrazu, tak aby wykonać na nim detekcję aut lub obszarów parkingowych i zapisać zdjęcie z naniemionymi oznaczeniami.

### **2 Wykorzystane zbiory danych**

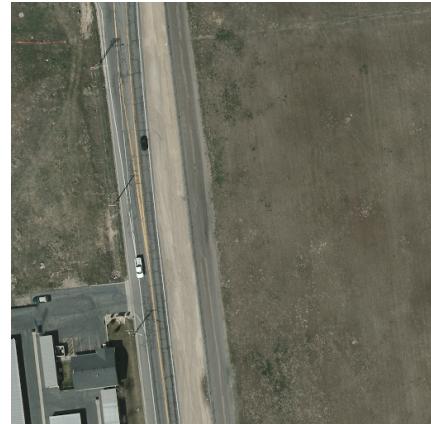
W projekcie wykorzystane zostały dwa zbiory danych:

- **AerialCars** (154 zdjęcia, 3787 aut) - jest to zbiór charakteryzujący się wysoką jakością zdjęć oraz dużym zagęszczeniem aut, ponieważ zdjęcia wykonywane były nad skrzyżowaniami w mieście,
- **VEDAI** (668 zdjęcia, 1265 aut) - zbiór charakteryzuje niższa jakość zdjęć oraz niższe zagęszczenie aut - zdjęcia wykonywane były na obszarach poza miejskich.

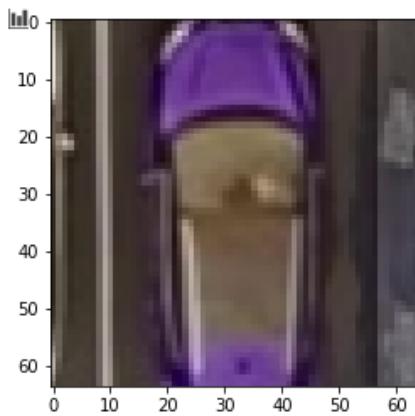
Zbiory dobrano w taki sposób, aby metody móc porównać na zdjęciach o różnej jakości oraz różnym zagęszczeniu aut i innych obiektów. Dodatkowo w drugim etapie wykorzystane zostały zdjęcia lotnicze pozyskane z ortofotomap Wrocławia (z lat 2018, 2017, 2015). Na tych zdjęciach przeprowadzono eksperymenty związane z detekcją parkingów oraz możliwością generalizacji wiedzy przez metody z etapu pierwszego.



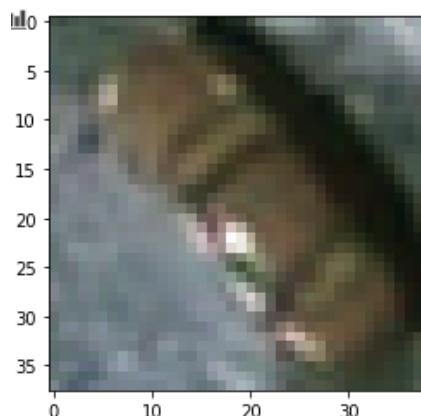
Rysunek 1: Przykładowe zdjęcie ze zbioru *AerialCars*



Rysunek 2: Przykładowe zdjęcie ze zbioru *VEDAI*



Rysunek 3: Przykładowe auto wycięte ze zdjęcia w zbiorze *AerialCars* (na osiach przedstawiono liczbę pikseli)



Rysunek 4: Przykładowe auto wycięte ze zdjęcia w zbiorze *VEDAI* (na osiach przedstawiono liczbę pikseli)

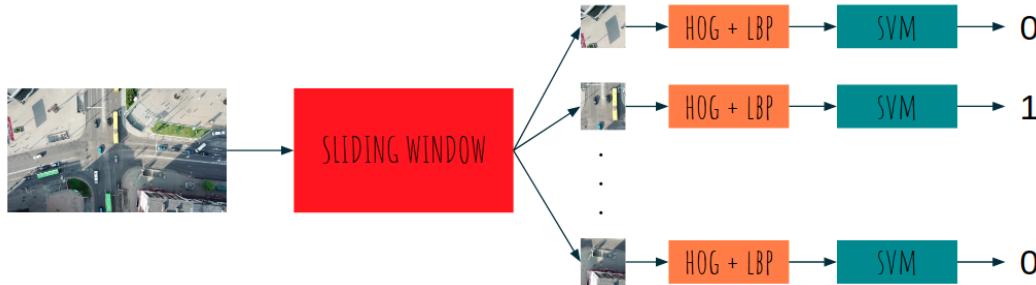
### 3 Przyjęte metody

#### 3.1 Etap pierwszy - detekcja aut

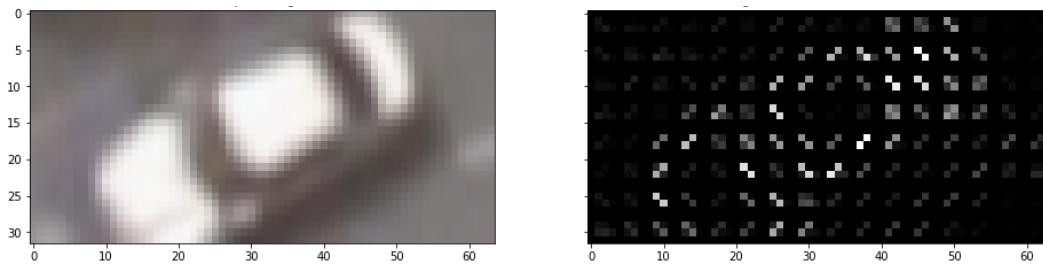
Wszystkie podejścia w etapie pierwszym bazowały na metodzie okna przesuwnego, czyli okna o zdefiniowanych wymiarach (mniejszych niż rozmiar zdjęcia), które przesuwa się po zdjęciu w obydwu osiach o zadany krok, wyznaczając w ten sposób obszary kandydujące, na których może zostać oznaczone auto, o czym decyduje wybrany klasyfikator.

##### 3.1.1 Podejście klasyczne

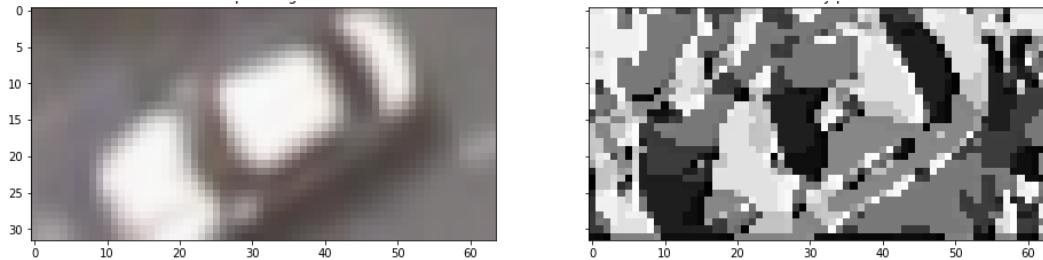
W podejściu klasycznym jako klasyfikator zastosowany został model *SVM*, który na wejściu przyjmował wektorową reprezentację obszarów kandydujących otrzymaną za pomocą deskryptorów *Histogram of Oriented Gradients (HOG)* oraz *Local Binary Patterns (LBP)*. Deskryptor *HOG* miał być odpowiedzialny za reprezentowanie kształtów występujących na obrazie, natomiast deskryptor *LBP* reprezentować miał tekstury obrazu. Informacje zawarte w wektorach zostały zwizualizowane na rysunkach 6 i 7. Ze względu na duży rozmiar przygotowanych wektorów do klasyfikacji wybrany został właśnie model *SVM*, który wykazuje dobre działanie przy dużych rozmiarach próbek wejściowych. Schemat podejścia klasycznego przedstawiono na rysunku 5.



Rysunek 5: Schemat podejścia klasycznego dla detekcji aut



Rysunek 6: Zdjęcie po prawej przedstawia wizualizację działania deskryptora *HOG* dla zdjęcie z lewej strony



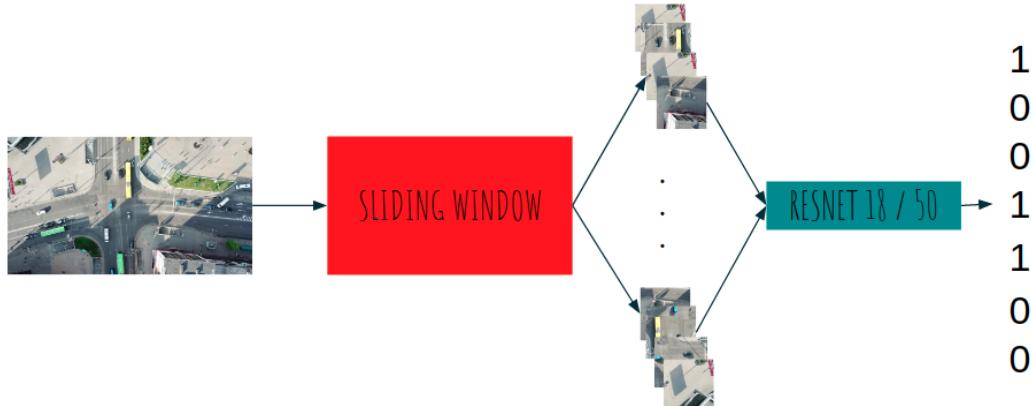
Rysunek 7: Zdjęcie po prawej przedstawia wizualizację działania deskryptora *LBP* dla zdjęcie z lewej strony

### 3.1.2 Podejście neuronowe

Podejście neuronowe odróżnia się tym, że jako klasyfikator wykorzystane zostały sieci neuronowe: *ResNet-18* oraz *ResNet-50*, czyli odpowiednik o bardziej rozbudowanej architekturze. Zdecydowano się na modele typu *ResNet*, ponieważ wykazują rozsądny kompromis pomiędzy jakością klasyfikacji, a liczbą parametrów. W przypadku podejścia neuronowego po wybraniu obszaru kandydującego, wycięty obraz zawsze był normalizowany, a dodatkowo w trakcie eksperymentów wykorzystano inne operacje: rozmycie Gaussa oraz wyrównanie histogramów. Schemat podejścia przedstawiono na rysunku 8

### 3.1.3 Dodatkowe przetwarzanie wyników detekcji

We wstępnych testach zauważono, że wysoka wartość kroku w oknie przesuwnym znacząco osłabia wyniki detekcji, dlatego starano się zachować jego możliwie niską wartość (tak, aby za bardzo nie zwiększyć czasu detekcji), jednak wtedy problemem okazywały się nakładające się na siebie oznaczenia dotyczące tego samego auta. Dlatego w obydwu podejściach dodatkowo zastosowano popularne dla tego problemu rozwiązanie, czyli metodę *Non-maximum Suppression (NMS)* - operację, która polega na usuwaniu oznaczeń z niższą predykowaną wartością prawdopodobieństwa nachodzących



Rysunek 8: Schemat podejścia neuronowego dla detekcji aut

na oznaczenia o wyższym predykowanym prawdopodobieństwie, jeżeli stopień nachodzenia na siebie oznaczeń przekracza zadany próg.

### 3.2 Etap drugi - wyznaczanie obszarów parkingowych

W etapie drugim badano podejście, w którym założono, że jeżeli na zdjęciach wykonanych w różnym czasie auta wykrywane są w tych samych miejscach, to w miejscach tych możemy założyć występowanie parkingów. Algorytm składa się z poniższych kroków:

- przygotowanie zdjęć tego samego obszaru wykonanych w różnym czasie,
- detekcja aut na przygotowanych zdjęciach,
- klasteryzacja wykrytych obiektów (w tym celu wykorzystana została metoda *DBSCAN*, głównie dlatego, że nie wymaga zdefiniowania liczby klastrów),
- wyznaczenie części wspólnych obszarów wyznaczonych na wszystkich zdjęciach przez znalezione klastry,
- oznaczenie otrzymanych w ten sposób obszarów jako parkingi.

Schemat podejścia przedstawiony został na rysunku 9

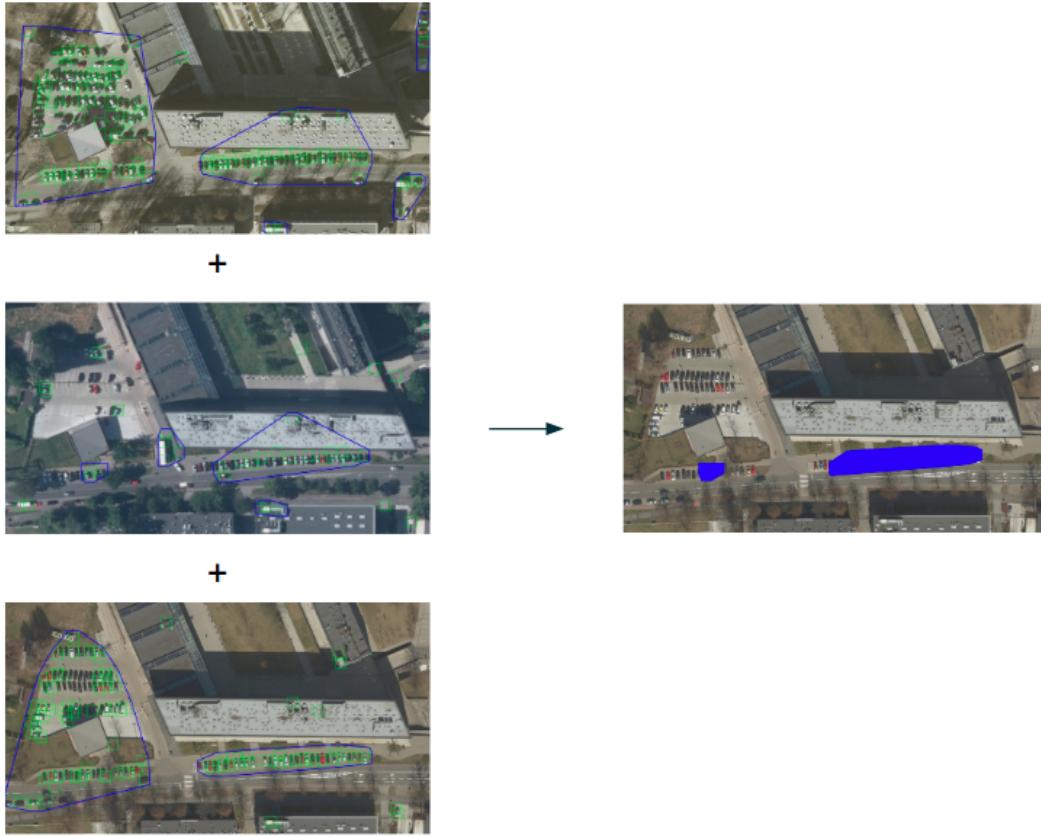
## 4 Eksperymenty - detekcja aut

### 4.1 Parametry okna przesuwnego

W trakcie eksperymentów dla każdego zbioru ręcznie dopierano rozmiar okna przesuwnego, tak aby odpowiadał rozmiarom aut na zdjęciach w zbiorze. Dla zbioru *AerialCars* wybrano rozmiar okna 80x80px, a dla zbioru *VEDAI* 50x50px. Dla obydwu zbiorów przyjęto rozmiar kroku 20px.

### 4.2 Przygotowanie danych treningowych dla klasyfikatora

W celu wytrenowania klasyfikatorów konieczne było przygotowanie danych uczących. W tym celu dla każdego zbioru ze zdjęć (z wyłączeniem zdjęć przeznaczonych do walidacji działania detekcji) wycinano auta, z miejsc zgodnych z rzeczywistymi oznaczeniami, zachowując rozmiary okien takie same jak rozmiar okna przesuwnego i w ten sposób otrzymywano przypadki pozytywne. Liczba przypadków pozytywnych była dodatkowo zwiększała przez ich obracanie o 90, 180 i 270 stopni. Przypadki negatywne (okna, na których nie znajdują się auta) wycinano ze zdjęć w sposób losowy, ale upewniając się, że w wyciętych oknach nie znajduje się żadne auto. Wycinano tyle przypadków negatywnych, aby zrównoważyć liczbę przypadków pozytywnych. Następnie przygotowane w ten sposób obrazy były przetwarzane zgodnie z wybranym dla podejścia potokiem (obliczanie deskryptorów dla podejścia klasycznego, normalizacja lub jedna z dodatkowych operacji,



Rysunek 9: Schemat podejścia zproponowanego w etapie drugim, jednocześnie przedstawiający jeden z jego słabych punktów, czyli wpływ błędów detekcji na pojedynczych zdjęciach

a następnie normalizacja dla podejścia neuronowego). Ten sam potok przetwarzania zachowany był dla okien wycinanych w trakcie detekcji.

#### 4.3 Przyjęte sposoby ewaluacji

Opisany w poprzedniej sekcji zbiór dzielono dodatkowo na zbiór treningowy i testowy, aby przetestować działanie samych klasyfikatorów, co robiono za pomocą miary *Accuracy*. Natomiast walidacja działania całego procesu detekcji polegała na tym, że:

- dla każdego rzeczywistego (*ground truth*) oznaczenia sprawdzano, czy któreś z predykowanych oznaczeń pokrywa ponad połowę jego powierzchni, jeżeli tak to taką predykcję liczono jako prawidłową pozytywną predykcję (*true positive*), ale stosowano ograniczenie, że dla każdego rzeczywistego oznaczenia jako *true positive* wybrać można było tylko jedną predykcję (tę pokrywającą jego największą powierzchnię)
- pozostałe predykowane oznaczenia zliczano jako błędne pozytywne predykcje (*false positive*),
- rzeczywiste oznaczenia, dla których nie otrzymano odpowiednio pokrywających je predykcji, liczono jak przypadki błędnej negatywnej predykcji (*false negative*),
- wykorzystując w ten sposób zliczone przypadki *true positive*, *false positive* i *false negative* obliczane były standardowe metryki stosowane dla problemów klasyfikacji: *Precision*, *Recall* i *F1Score*, które są powszechnie stosowane również do oceny detekcji, ponieważ pozwalają ocenić jaki odsetek pożądanych obiektów udało się wykryć oraz jak wiele błędów zostało popełnionych w trakcie detekcji. Jako najważniejszą miarę uznana była miara *F1Score*, ponieważ przedstawia ona balans pomiędzy dwiema pozostałymi miarami.

#### 4.4 Badane parametry

Pomiędzy różnymi eksperymentami modyfikowane były parametry:

- *Threshold* - graniczna wartość predykowanego prawdopodobieństwa, od której klasyfikator uznawał obecność auta w wyciętym oknie,
- *NMS overlap* - graniczna wartość pokrycia oznaczeń w metodzie *Non-maximum Suppresion*, od której usuwano oznaczenie z niższą wartością prawdopodobieństwa.

Dodatkowo w podejściu neuronowym badano różne potoki przetwarzania:

- *Norm* - normalizacja obrazu,
- *GaussBlur, Norm* - wygładzenie Gaussa (z domyślnymi parametrami w implementacji w bibliotece *OpenCV*), a następnie normalizacja,
- *EqualHist, Norm* - wyrównanie histogramów, a następnie normalizacja.

#### 4.5 Wyniki dla podejścia klasycznego

##### 4.5.1 Wyniki klasyfikacji

	precision	recall	f1-score	support
0	0.99	0.98	0.99	469
1	0.98	0.99	0.99	475
accuracy			0.99	944
macro avg	0.99	0.99	0.99	944
weighted avg	0.99	0.99	0.99	944

Rysunek 10: Wyniki działania klasyfikatora *SVM* na zbiorze *AerialCars*

	precision	recall	f1-score	support
0	0.99	0.99	0.99	961
1	0.99	0.99	0.99	923
accuracy			0.99	1884
macro avg	0.99	0.99	0.99	1884
weighted avg	0.99	0.99	0.99	1884

Rysunek 11: Wyniki działania klasyfikatora *SVM* na zbiorze *VEDAI*

Powyzsze wyniki pokazują, że dla obydwu zbiorów testy samych klasyfikatorów *SVM* pokazują ich niemal bezbłędne działanie.

##### 4.5.2 Wyniki detekcji

Wyniki detekcji przedstawione na rysunkach 12 i 13 pokazują, że zbiór *VEDAI* okazał się trudniejszy mimo przypuszczeniom, że niższe zagęszczanie obiektów powinno ułatwić detekcję. Można też zauważyc, że zwiększanie parametru *Threshold* poprawia wyniki miary *F1Score*, natomiast zbadane zmiany parametru *NMS overlap* nie mają znaczącego wpływu na wyniki. Obserwując miary *Precision* i *Recall* można powiedzieć, że wykryta została znacząca część aut, jednak popełniono przy tym wiele błędów typu *false positive*.

Zbiór	NMS overlap	Threshold	Precision	Recall	F1Score
AerialCars	0,4	0,7	0,289	0,805	0,425
		0,8	0,329	0,752	0,458
		0,9	0,389	0,664	<b>0,49</b>
VEDAI	0,4	0,7	0,102	0,459	0,166
		0,8	0,128	0,447	0,199
		0,9	0,163	0,435	<b>0,237</b>

Rysunek 12: Wyniki detekcji dla podejścia klasycznego, manipulacja parametrem *Threshold*

Zbiór	Threshold	NMS overlap	Precision	Recall	F1Score
AerialCars	0,9	0,3	0,391	0,637	0,485
		0,4	0,389	0,664	<b>0,49</b>
		0,5	0,389	0,664	<b>0,49</b>
VEDAI	0,9	0,3	0,164	0,435	<b>0,239</b>
		0,4	0,163	0,0435	0,237
		0,5	0,163	0,435	0,237

Rysunek 13: Wyniki detekcji dla podejścia klasycznego, manipulacja parametrem *NMS overlap*

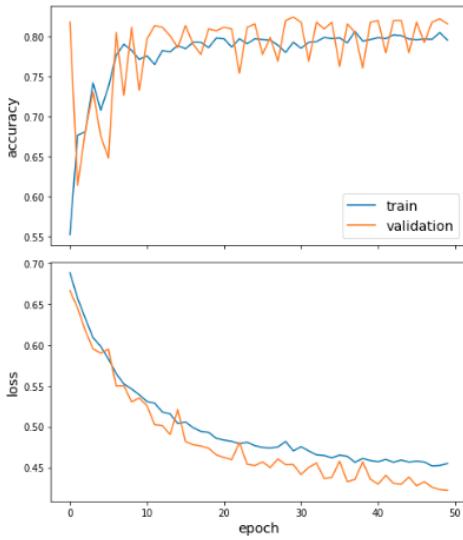
## 4.6 Wyniki dla podejścia neuronowego

### 4.6.1 Wyniki klasyfikacji

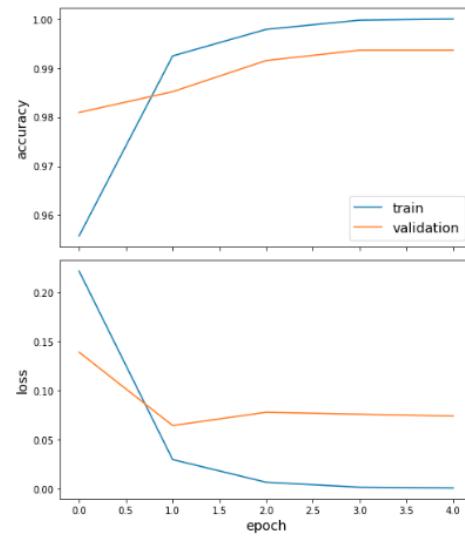
Na rysunkach 14, 15, 16 i 17 przedstawione zostały przebiegi uczenia modeli głębokich, w trakcie których, na podstawie przygotowanych zbiorów, trenowano ostatnią warstwę, zachowując przygotowane przez autorów modeli wagę we wcześniejszych warstwach. Analizując wykresy dla obydwu zbiorów można zaobserwować, że modele na zbiorze walidacyjnym szybko osiągają wysoką wartość *Accuracy*, jednak w przypadku modelu *ResNet-50*, jest ona nieco wyższa.

### 4.6.2 Wyniki detekcji

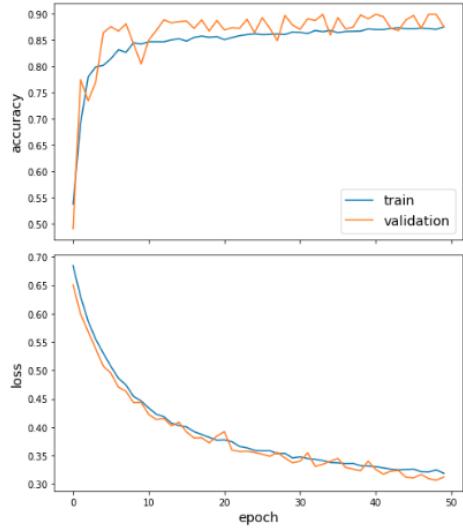
Omawianie wyników detekcji w podejściu głębokim należy rozpocząć od rysunku 18, gdzie widać dużą przewagę modelu *ResNet-50*, który został wybrany do kolejnych eksperymentów, przedstawionych na rysunku 19 i 20, które pokazały ponownie pozytywny wpływ zwiększenia parametru *Threshold* oraz zaskakujący negatywny wpływ stosowania dodatkowych operacji w potoku przetwarzania obrazów. Natomiast zmiana parametru *NMS overlap* dała tym razem poprawę wyników w przypadku zbioru *VEDAI*. Ogólnie zauważać można, że podejście głębokie cechują bardzo wysokie wartości miary *Recall*, jednak z powodu niskiej precyzji oznaczeń, ostatecznie miara *F1Score* nie jest do końca zadowalająca.



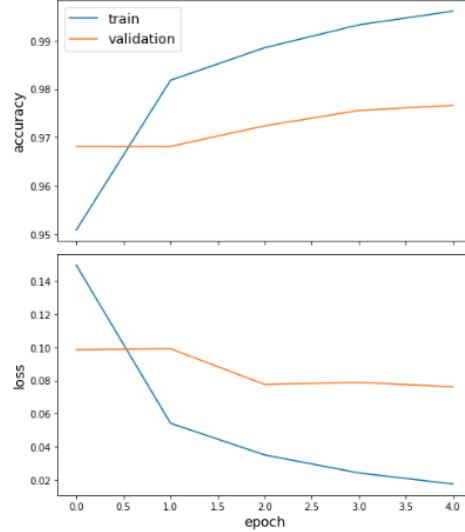
Rysunek 14: Przebieg uczenia modelu ResNet-18 na zbiorze *AerialCars*



Rysunek 15: Przebieg uczenia modelu ResNet-50 na zbiorze *AerialCars*



Rysunek 16: Przebieg uczenia modelu ResNet-18 na zbiorze *VEDAI*



Rysunek 17: Przebieg uczenia modelu ResNet-50 na zbiorze *VEDAI*

Zbiór	Preprocessing	NMS overlap	Model	Threshold	Precision	Recall	F1Score
Aerial Cars	Norm	0,4	Resnet-18	0,7	0,069	0,504	0,122
				0,8	0,092	0,398	<b>0,15</b>
				0,9	0,11	0,221	0,147
	Norm	0,4	Resnet-50	0,7	0,254	0,903	0,397
				0,8	0,235	0,867	0,37
				0,9	0,275	0,912	<b>0,423</b>
			Resnet-18	0,7	0,037	0,459	0,069
				0,8	0,049	0,353	0,087
				0,9	0,071	0,224	<b>0,108</b>
VEDAI	Norm	0,4	Resnet-50	0,7	0,081	0,847	0,148
				0,8	0,094	0,835	0,169
				0,9	0,117	0,824	<b>0,205</b>

Rysunek 18: Wyniki detekcji dla podejścia neuronowego, manipulacja parametrem *Threshold*

Zbiór	Threshold	NMS overlap	Preprocessing	Precision	Recall	F1Score
AerialCars	0,9	0,4	GaussBlur, Norm	0,257	0,876	0,398
			EqualHist, Norm	0,219	0,841	0,348
			Norm	0,275	0,912	<b>0,423</b>
VEDAI	0,9	0,4	GaussBlur, Norm	0,089	0,706	0,157
			EqualHist, Norm	0,073	0,835	0,135
			Norm	0,117	0,824	<b>0,205</b>

Rysunek 19: Wyniki detekcji dla podejścia neuronowego, dodatkowe operacje przetwarzania

Zbiór	Preprocessing	Threshold	NMS overlap	Precision	Recall	F1Score
Aerial Cars	Norm	0,9	0,3	0,287	0,69	0,405
			0,4	0,275	0,912	<b>0,423</b>
			0,5	0,257	0,867	0,396
VEDAI	Norm	0,9	0,3	0,13	0,729	<b>0,221</b>
			0,4	0,117	0,824	0,205
			0,5	0,117	0,824	0,205

Rysunek 20: Wyniki detekcji dla podejścia neuronowego, manipulacja parametrem *NMS overlap*

#### 4.6.3 Zestawienie metod

Zestawiając ze sobą najlepsze wyniki z podejścia klasycznego i podejścia neuronowego zauważymo, że mimo wysokiej wartości miary *Recall* w podejściu neuronowym, to najważniejsza w eksperymentach miara *F1Score* jest dla obydwu zbiorów wyższa w przypadku podejścia klasycznego. Jednak czas działania podejścia klasycznego może okazać problematyczny i pod tym względem dużo lepiej wykazuje się podejście neuronowe, szczególnie gdy sieć neuronowa uruchomiona zostanie na procesorze graficznym.

Zbiór	Miara	Podejście klasyczne	Podejście neuronowe
Aerial Cars	Precision	<b>0,389</b>	0,275
	Recall	0,664	<b>0,912</b>
	F1Score	<b>0,49</b>	0,423
VEDAI	Precision	<b>0,164</b>	0,13
	Recall	0,435	<b>0,729</b>
	F1Score	<b>0,239</b>	0,221

Rysunek 21: Zestawienie najlepszych wyników uzyskanych w obydwu podejściach

Wartość kroku	Podejście klasyczne CPU	Podejście neuronowe CPU	Podejście neuronowe GPU (Google Colab)
16px	304s	180s	<b>31s</b>
32px	70s	37s	<b>5s</b>

Rysunek 22: Zestawienie czasów wykonania detekcji dla zdjęcia o rozmiarze 1900x1200px i wymiarach okna przesuwnego 128x128px

## 5 Eksperyenty - wyznaczanie parkingów

### 5.1 Generalizacja wiedzy z podejścia pierwszego

Zgodnie z przedstawionym wcześniej schematem działania metody, pierwszym krokiem jest przeprowadzenie detekcji na wybranych zdjęciach, do czego miały być wykorzystane modele i podejścia przygotowane w etapie pierwszym. Najlepsze warianty zostały przetestowane na 5 przygotowanych zdjęciach z ortofotomapy 2018 Wrocławia, na których dokładnie oznaczono wszystkie auta. Przykład takiego zdjęcia przedstawiono na rysunku 23. Do detekcji wykorzystano modele wytrenowane na zbiorze *AerialCars*, ponieważ zawarte w nim zdjęcia bardziej przypominały te z ortofotomapy. Zachowano również parametry okna przesuwnego wykorzystywane za zbiorze *AerialCars*. W eksperymetach dodatkowo manipulowano parametrami *Threshold* i *NMS overlap*, ale również sprawdzono czy model głęboki nie przeucza się za bardzo do danych ze zbioru *AerialCars*, dlatego przetestowano jego wariant wytrenowany na mniejszej liczbie epok.

Jak pokazują wyniki na rysunkach 24 i 25 model *SVM* nie poradził sobie z generalizacją wiedzy na nowym zbiorze danych, natomiast model głęboki zrobił to zaskakującą dobrze, dodatkowo pokazując poprawę po zmniejszeniu liczby epok.



Rysunek 23: Przykładowe zdjęcie z przygotowanymi oznaczeniami aut

Zbiór	Threshold	NMS overlap	Precision	Recall	F1Score
Ortofotomapa 2018	0,9	0,3	0,486	0,132	0,208
		0,4	0,474	0,14	<b>0,216</b>
		0,5	0,474	0,14	<b>0,216</b>
	0,95	0,3	0,5	0,08	0,138
		0,4	0,485	0,085	0,145
		0,5	0,485	0,085	0,145
	0,98	0,4	0,568	0,054	0,099

Rysunek 24: Wyniki detekcji aut na ortofotomapie 2018 z zastosowaniem podejścia klasycznego z modelem SVM wytrenowanym na zbiorze *AerialCars*

Zbiór	Epochs	Threshold	NMS overlap	Precision	Recall	F1Score
Ortofotomapa 2018	1	0,9	0,3	0,521	0,505	0,513
			0,4	0,475	0,598	<b>0,53</b>
			0,5	0,475	0,598	<b>0,53</b>
		0,95	0,4	0,488	0,567	0,525
		0,98	0,4	0,521	0,505	0,513
	5	0,9	0,4	0,43	0,622	0,508
		0,95	0,3	0,505	0,526	<b>0,515</b>
			0,4	0,447	0,604	<b>0,514</b>
			0,5	0,447	0,604	<b>0,514</b>
		0,98	0,4	0,471	0,518	0,493

Rysunek 25: Wyniki detekcji aut na ortofotomapie 2018 z zastosowaniem podejścia neuronowego z modelem *ResNet-50* wytrenowanym na zbiorze *AerialCars* i z zastosowaniem jedynie normalizacji zdjęć

## 5.2 Przygotowanie danych

Do kolejnych kroków w etapie drugim przygotowano 20 zestawów zawierających po 3 zdjęcia dokładnie tych samych miejsc z ortofotomap z różnych lat. Na zdjęciach z roku 2018 oznaczono miejsca parkingowe, tak jak na rysunku 26, co w dalszej części będzie potrzebne do walidacji działania metody.



Rysunek 26: Oznaczenia parkingów na ortofotomapie Wrocławia z 2018 roku

## 5.3 Badane parametry

Do odnajdywania klastrów zbudowanych ze znalezionych aut wykorzystano algorytm *DBSCAN*, w którym badano następujące parametry:

- *Epsilon* - maksymalna odległość pomiędzy obiekty, dla której algorytm rozpatruje obiekty jako sąsiadujące ze sobą,
- *Min Samples* - minimalna liczba sąsiadujących obiektów potrzebna do utworzenia klastra.

## 5.4 Ewaluacja

Do liczbowego wykazania jakości działania metody wykorzystano miarę *Intersection over Union* (*IoU*), która liczona była dla każdego zdjęcia pomiędzy całkowitym obszarem oznaczonym przez autorów jako parkingi oraz całkowitym obszarem oznaczonym jako parkingi przez algorytm.

## 5.5 Wyniki

Komentując wyniki z rysunku 27 można powiedzieć, że metoda nie zadziałała. Jednak przeprowadzenie eksperymentów pozwoliło pokazać wrażliwość podejścia głębokie na zmiany w jakości zdjęć lotniczych, ponieważ oglądając przykłady wykonywanych klasteryzacji na rysunkach 28 i 29 jako przyczyny niepowodzenia można wskazać:

- niską jakość detekcji na zdjęciach o niższej rozdzielczości,
- problemy z detekcją w obszarach zacienionych,
- fakt, że jedna z ortofotomap wykonana była w porze roku, gdy na drzewach znajdowały się liście zasłaniające ważne obszary,
- słabość testowanego założenia na przygotowanych próbkach - na ruchliwych ulicach auto stoją w korkach na wszystkich trzech zdjęciach, co powoduje oznaczeniem ulicy jako parking.

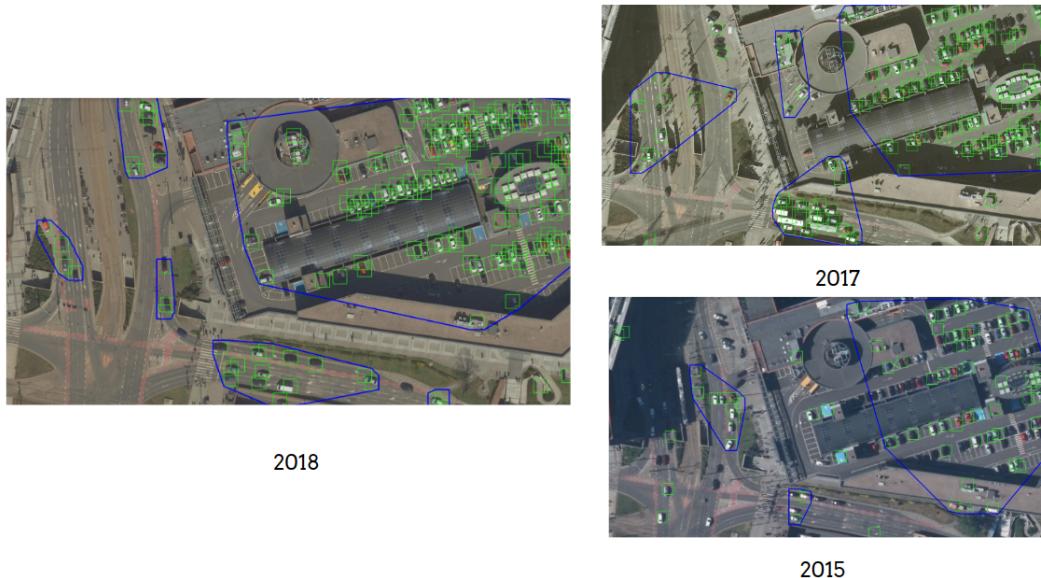
Jednak pomimo wskazanych problemów wizualna ocena wyników potwierdza, że model głęboki bardzo dobrze zgeneralizował wiedzę na ortofotomapię z 2018 roku i pozwolił na niej uzyskać względnie dobre wyniki detekcji.

Threshold	NMS overlay	Epsilon	Min Samples	Mean IoU	Max IoU
0,9	0,4	100	3	0,0157	0,1016
		150		0,0334	0,1687
		200		0,0649	0,2269
		250		0,1143	0,4925
		300	1	0,1497	0,5961
		300	3	<b>0,1455</b>	<b>0,5981</b>
		350	5	0,1239	0,5148
		350	3	0,1238	0,5147

Rysunek 27: Wyniki działania metody wyznaczania parkingów zaproponowanej w etapie drugim



Rysunek 28: Przykładowe wyniki klasteryzacji w etapie drugim



Rysunek 29: Przykładowe wyniki klasteryzacji w etapie drugim



Rysunek 30: Wyniki detekcji na ortofotomapie 2018

## 6 Wnioski

### 6.1 Etap pierwszy

- Zastosowanie okna przesuwnego okazało się trudnym wyzwaniem, ponieważ ciężko było przełożyć dobre wyniki samych klasyfikatorów na jakość detekcji na rzeczywistych zdjęciach.
- Na testowanych zbiorach lepsza okazała się metoda klasyczna, ponieważ zachowała lepszy balans między odsetkiem wykrytych aut i odsetkiem błędnych detekcji.
- Metoda neuronowa znajdowała większość aut, jednak popełniała przy tym znaczącą liczbę błędów typu *false positive*.
- Modele zwracały bardzo wysokie predykowane prawdopodobieństwa również dla błędnych predykcji, dlatego ważne było wprowadzenie do projektu możliwości operowania parametrem *Threshold*.
- Biorąc pod uwagę popełniane błędy równie ważne okazało się zastosowanie metody *Non-maximum Suppresion* z parametrem *NMS overlap*, który zazwyczaj najlepsze wyniki dawał przy wartości 0.4.
- Wbrew przypuszczeniom stosowanie dodatkowych operacji w przetwarzaniu wycinanych obrazów nie przyniosło zmniejszenia popełnianych błędów, a wręcz przeciwnie znaczaco osłabiało wyniki.
- Podejścia lepiej sprawdziły się na zbiorze *AerialCars*, który cechowało wyższe zagęszczenie obiektów, ale jednocześnie wyższa jakość zdjęć, stąd można przypuszczać, że jakość zdjęcia jest kluczowa dla badanych podejść.
- Dużym obciążeniem stosowania okna przesuwnego, szczególnie w podejściu klasycznym, jest czas wykonywania detekcji.
- Szukanie poprawy wyników należało zacząć od rozszerzenia augmentacji danych treningowych dla klasyfikatora oraz dokładnego zbadania wpływu parametrów okna przesuwnego na jakość detekcji - miejsca te mogły być najsłabszymi punktami projektu.
- Na ten moment opracowane podejścia nie mogą konkurować z rozwiązaniami podjętych problemów prezentowanymi w literaturze.

### 6.2 Etap drugi

- Metoda zaproponowana w etapie drugim nie sprawdziła się z powodów wymienionych w sekcji 5.4.
- Mimo słabych wyników przeprowadzenie drugiego etapu wniosło dużą wartość do projektu, ponieważ był testem generalizacji wiedzy modeli opracowanych w poprzednim etapie i pokazało, że model *SVM* nie był w stanie zgeneralizować wiedzy, a model głęboki dobrze zgeneralizował wiedzę na ortofotamapie najbardziej zbliżonej jakością do zbioru *AerialCars* jednocześnie pokazując swoje słabe punkty (powody obniżenia jakości detekcji na pozostałych ortofotomapach), jak wrażliwość na zmianę rozdzielczości lub tonacji barw zdjęcia, czy problemy z miejscami bardziej zaciennionymi.
- Samo zaproponowane rozwiązanie wydaje się mieć potencjał, jednak wymagałoby przygotowania dużo lepszego modelu detekcji aut oraz przygotowania większej liczby zdjęć tych samych miejsc wykonanych w krótszych odstępach czasowych (np. co 30 minut) - przygotowanie takiego zbioru najprawdopodobniej wymagałoby zmianę rozpatrywanych zdjęć z lotniczych na satelitarne.

### 6.3 Wnioski ogólne

Przeprowadzony projekt był pierwszym podejściem autorów do przetwarzania obrazów, dlatego pozwolił im zapoznać się z podstawowymi technikami i bibliotekami jak np. *OpenCV* oraz z zagadnieniami specyficznymi dla problemu detekcji. Mimo ostatecznie nie do końca zadowalających wyników, cały przeprowadzony proces dał ich znaczącą poprawę od pierwszych prototypów do ostatecznej formy.

## 7 Opis repozytorium

### 7.1 Komponenty obce

W projekcie jako komponenty obce, oprócz bibliotek zawartych w pliku *requirements.txt* oraz pretrainedowych modeli głębowych, wykorzystano i przystosowano skrypty do ewaluacji wyników detekcji oraz przeprowadzenia metody *Non-maximum Supresion*. Reszta kodu jest pracą własną autorów.

### 7.2 Struktura plików

Główne katalogi w repozytorium to:

- *src* - kod źródłowy projektu, zawiera klasy i funkcje pozwalające m.in. wczytywać i przetwarzać dane, trenować modele, definiować potoki przetwarzania obrazów i detektory bazujące na oknie przesuwnym,
- *data* - folder na dane zawierający również skrypty do ich pobrania,
- *experiments/datasets\_preparation* - zawiera notebooki pozwalające przygotować dane do eksperymentów,
- *experiments/svm* - zawiera notebooki pozwalające przeprowadzić eksperymenty dla podejścia klasycznego w etapie pierwszym,
- *experiments/resnet* - zawiera notebooki pozwalające przeprowadzić eksperymenty dla podejścia neuronowego w etapie pierwszym,
- *experiments/parkings* - zawiera notebook pozwalający przeprowadzić eksperymenty dla etapu drugiego.

### 7.3 Uruchomienie badań

Aby uruchomić badania należy:

- zainstalować biblioteki z pliku *requirements.txt*,
- pobrać dane wykorzystując skrypty w folderze *data*,
- przetworzyć wybrane dane dla wybranego podejścia uruchamiając odpowiedni notebook w folderze *experiments/datasets\_preparation*,
- uruchomić odpowiedni notebook dla wybranego zbioru danych i wybranego podejścia z folderu *experiments/svm*, *experiments/resnet* lub *experiments/parkings*.

W repozytorium zbiory danych przyjęły robocze nazwy: *AerialCars* - aerial-cars-dataset, *VEDAI* - vehicles, zdjęcia z ortofotomapy 2018 - orto, zestawy zdjęć z ortofotomap - orto\_parkings.

## Literatura

- [1] D. Kamenetsky and J. Sherrah. Aerial car detection and urban understanding. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2015.
- [2] W. Shao, W. Yang, G. Liu, and J. Liu. Car detection from high-resolution aerial imagery using multiple features. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 4379–4382. IEEE, 2012.
- [3] S. Zambanini, A.-M. Loghin, N. Pfeifer, E. M. Soley, and R. Sablatnig. Detection of parking cars in stereo satellite images. *Remote Sensing*, 12(13):2170, 2020.