

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Sterowania i Elektroniki Przemysłowej

Praca dyplomowa inżynierska

na kierunku Automatyka i Robotyka Stosowana

Rozpoznawanie małych i licznych obiektów na zdjęciach
z wykorzystaniem głębokich sieci neuronowych

Michał Wojcieszek

numer albumu 319184

promotor
dr inż. Witold Czajewski

Warszawa 2025

Rozpoznawanie małych i licznych obiektów na zdjęciach z wykorzystaniem głębokich sieci neuronowych

Streszczenie

W ostatnich latach widoczny jest dynamiczny rozwój w dziedzinie rozpoznawania obiektów na zdjęciach przez głębokie sieci neuronowe. Najnowsze modele uzyskują wysokie wyniki w różnorodnych testach i znalazły wiele praktycznych zastosowań. Pomimo tego, rozpoznawanie małych i licznych obiektów nadal pozostaje trudnym zadaniem, a efektywność sieci CNN w tym aspekcie jest niezadowalająca. Niniejsza praca przybliża przyczyny takiego stanu rzeczy, możliwe rozwiązania, w szczególności algorytm Slicing Aided Hyper Inference (SAHI) oraz opisuje zastosowania detekcji niewielkich obiektów. SAHI, pomimo wysokiej skuteczności, charakteryzuje się długim czasem analizy zdjęcia. W pracy przedstawiono nowatorską modyfikację tego algorytmu nazwaną Filtrowaniem Krawędzi (FK), która stara się rozwiązać problem SAHI, poprzez pominięcie fragmentów obrazu niezawierających konturów. W celu jej zbadania w praktycznej aplikacji, dwie najnowocześniejsze sieci neuronowe, YOLO11n oraz YOLO11l, zostały wytrenowane na zbiorze zdjęć SODA-D, skierowanym na rozpoznawanie małych obiektów w sytuacjach drogowych. Szeroko zbadano wpływ różnych zmiennych na działanie FK. Przeprowadzone eksperymenty wykazały wysoką skuteczność modyfikacji, która, w zależności od wybranych parametrów, umożliwiła skrócenie średniego czasu przetwarzania zdjęcia nawet 1,74 lub 2,74 raza. Wiązało się to z nieznacznym spadkiem wskaźnika mAP50-95 dla małych obiektów równym odpowiednio 0,0007 i 0,0293. Pomimo istotnego przyspieszenia, czas inferencji pozostał stosunkowo wysoki i wynosił blisko jedną sekundę. FK może jednak znaleźć wiele praktycznych zastosowań, w szczególności w analizie zdjęć wysokiej rozdzielczości.

Słowa kluczowe: widzenie komputerowe, rozpoznawanie małych obiektów, SAHI, YOLO11, Filtrowanie Krawędzi, SODA-D

Recognizing small and numerous objects in images using deep neural networks

Abstract

In recent years, a dynamic development in the field of object detection using deep convolutional neural networks has been observed. The latest models achieve high performance in various tests and have found countless practical applications. Nevertheless, the recognition of small and numerous objects remains a challenging task, and the efficiency of CNNs in this area is unsatisfactory. This engineering thesis explores the causes of this issue, possible solutions, with a particular focus on the Slicing Aided Hyper Inference algorithm (SAHI) and provides an overview of the applications of small object detection. SAHI, despite its high effectiveness, suffers from a long processing time. A novel modification of this algorithm, called Edge Filtering (EF), is presented in the thesis, aiming to address the limitations of SAHI by skipping image fragments that do not contain edges. In order to evaluate EF in a practical application, two state-of-the-art neural networks, YOLO11n and YOLO11l, were trained on the SODA-D dataset, designed for the detection of small objects in traffic scenarios. The influence of various variables on the performance of EF was thoroughly investigated. Conducted experiments demonstrated high effectiveness of the modification, which, depending on the selected parameters, enabled a reduction in average image processing time by as much as 1.74 or 2.74 times. This improvement came at a cost of a slight decrease in the mAP50-95 metric for small objects, with values of 0.0007 and 0.0293, respectively. Despite the significant reduction, the inference time remained relatively high, at about one second. However, EF may find other applications, particularly in the analysis of high-resolution images.

Keywords: computer vision, small object detection, SAHI, YOLO11, Edge Filtering, SODA-D

Spis treści

1 Wprowadzenie	1
1.1 Cel i układ pracy	2
2 Stan wiedzy	3
2.1 Przegląd zastosowań rozpoznawania małych i licznych obiektów	3
2.1.1 Autonomiczne samochody	3
2.1.2 Drony	5
2.1.3 Liczenie kolonii bakterii	6
2.1.4 Analiza zdjęć satelitarnych	7
2.1.5 Monitoring	8
2.1.6 Rolnictwo	8
2.2 Analiza przyczyn trudności w rozpoznawaniu małych obiektów	9
2.2.1 Ekstrakcja wysokopoziomowych cech	9
2.2.2 Rozdzielcość wejściowa sieci	10
2.2.3 Ramki kotwiczne	11
2.2.4 Niewielkie wymiary obiektów	11
2.2.5 Wariacja skali	11
2.2.6 Zaszumienie danych	12
2.2.7 Zagęszczanie obiektów	13
2.2.8 Niska tolerancja na przesunięcie ramek w procesie uczenia	14
2.2.9 Brak zbiorów zdjęć nastawionych na rozpoznawanie małych obiektów	15
2.3 Przegląd istniejących rozwiązań	15
2.3.1 Slicing Aided Hyper Inference	15
2.3.2 Piramida cech	17
2.3.3 Wielopoziomowa ekstrakcja cech	17
2.3.4 Równoległe głowice detekcyjne	17
2.3.5 Super rozdzielcość	18
2.3.6 Selekcja rejonu detekcji	19
2.3.7 Modyfikacje procesu uczenia	19

3 Opis przeprowadzonych prac	21
3.1 Koncepcja Filtrowania Krawędzi	21
3.2 Realizacja Filtrowania Krawędzi	21
3.3 Specyfikacja sprzętowa	22
3.4 Specyfikacja programowa	22
3.5 Wybór modelu sieci	23
3.6 Wybór zbioru zdjęć	24
3.7 Przygotowanie zbioru zdjęć do uczenia sieci	27
3.8 Trening sieci	27
4 Analiza wyników badań	33
4.1 Terminy	33
4.1.1 Podział	33
4.1.2 Przyspieszenie	35
4.2 Metryki	35
4.3 Ewaluacje	36
4.4 Seria ewaluacji odniesienia	36
4.5 Ponowne zbadanie czułości modelu	38
4.6 Optymalizacja parametrów Filtrowania Krawędzi	38
4.7 Serie ewaluacji z wykorzystaniem Filtrowania Krawędzi	41
4.8 Ewaluacje przy stałym rozmiarze wycinków	43
5 Podsumowanie i wnioski	47
5.1 Dalsze możliwości rozwoju	48
Bibliografia	51
Wykaz skrótów i symboli	57
Spis rysunków	59
Spis tabel	61
Spis załączników	63

Rozdział 1

Wprowadzenie

Rozpoznawanie obiektów na zdjęciach przez programy komputerowe i urządzenia elektroniczne od wielu lat jest codziennym elementem naszego życia. Spotykamy się z nim w aplikacjach mobilnych czy serwisach internetowych, a pierwsze aparaty potrafiące rozpoznawać twarze na fotografiach powstały już w latach 2000. Początki zainteresowania środowiska naukowego tą dziedziną i próby wprowadzenia jej założeń w życie sięgają lat 60. XX wieku [45]. Prawdziwy przełom nastąpił, gdy w roku 1969 po raz pierwszy wykorzystano do tego celu sieci neuronowe z warstwą konwolucyjną [13]. Wykonanie operacji splotu obrazu z jądrem było innowacyjnym rozwiązaniem, które umożliwiło ogromny rozwój możliwości w zadaniach klasyfikacji oraz lokalizacji obiektów. Do dzisiaj jest ono wykorzystywane jako podstawa sieci neuronowych do analizy zdjęć.

Współczesną rewolucję w dziedzinie rozpoznawania obiektów było wykorzystanie metod głębokiego uczenia, które umożliwiły stworzenie sieci zdolnych do analizy zdjęć z niespotykaną do tej pory skutecznością [27]. Modele wykorzystujące obie wspomniane techniki to tzw. Głębokie Splotowe Sieci Neuronowe (CNN – ang. *Convolutional Neural Networks*). Obecnie odgrywają one kluczową rolę w praktycznych zastosowaniach widzenia komputerowego oraz są przedmiotem intensywnych badań naukowych. Umożliwiły imponujący rozwój w dziedzinie rozpoznawania obiektów na zdjęciach obserwowany w ciągu ostatnich lat. Świadczą o tym chociażby co raz to wyższe metryki uzyskiwane w ewaluacjach na zbiorach zdjęć takich jak MS COCO [32, 56] czy ImageNet [7, 27].

Rozpoznawanie małych obiektów (SOD – ang. *Small Object Detection*) jest poddziedziną widzenia komputerowego. Pomimo rosnącej skuteczności sieci CNN, detekcja małych i licznych obiektów, takich na rys. 1, stale charakteryzuje się znaczco gorszymi wynikami niż dla instancji średnich i dużych, często niesatisfakcjonującymi. Przykładowo nowoczesna sieć DyHead z ekstraktorem cech ResNet-101 testowana na zbiorze test-dev bazy MS COCO [32] uzyskuje wartość wskaźnika mAP50-95 dla małych obiektów równą 0,283 [5, 6]. Dla porównania wartość ta dla obiektów średnich i dużych wynosi odpowiednio 0,503 i 0,575. Z powodu niskiej skuteczności, oraz z powodu licznych zastosowań SOD, istotny jest rozwój metod optymalizujących rozpoznawanie małych obiektów. Mogą one w przyszłości odegrać znaczącą rolę w dziedzinach zajmujących się autonomicznymi samochodami, dronami czy analizą zdjęć satelitarnych.

1.1 Cel i układ pracy

Celem teoretycznym niniejszej pracy inżynierskiej było przedstawienie:

- praktycznych zastosowań rozpoznawania małych i licznych obiektów,
- problemów, które powodują wysoki poziom trudności tego zadania,
- możliwych technik, które starają się te problemy rozwiązywać, w szczególności algorytmu Slicing Aided Hyper Inference (SAHI) [1, 2].

Celami praktycznymi były implementacja oraz zbadanie autorskiej modyfikacji algorytmu SAHI zwanej Filtrowaniem Krawędzi (FK), która umożliwia skrócenie czasu analizy zdjęcia dzięki pominięciu fragmentów obrazu niezawierających konturów. Istotne było, aby modyfikacja ta nie wpływała negatywnie na jakość detekcji, czyli nie pomijała rejonów zawierających instancje rozpoznawanych klas. Aby to ocenić, zostały obliczone metryki sieci wytrenowanej na wybranym zbiorze zdjęć małych obiektów. Cztery najważniejsze cele przeprowadzonych badań to:

- określenie, jakie jest możliwe przyspieszenie algorytmu SAHI dzięki zastosowaniu FK,
- ocena utraty jakości spowodowanej badanym algorytmem,
- optymalizacja parametrów Filtrowania Krawędzi,
- zebranie danych, które umożliwiają ocenę praktycznego zastosowania FK.

Rozdział *Stan wiedzy* jest podzielony na trzy podrozdziały. Opisują one kolejno zastosowania SOD, problemy, które mu towarzyszą, oraz techniki stosowane, aby owe problemy rozwiązać. Następny rozdział, zatytułowany *Opis przeprowadzonych prac*, przybliża, w jaki sposób zostało zrealizowane Filtrowanie Krawędzi i wymienia wykorzystane do tego narzędzia. Ponadto przedstawia on proces treningu sieci neuronowych, od wyboru modelu i zbioru zdjęć, po uzyskane rezultaty. Rozdział *Analiza wyników badań* jest poświęcony przeprowadzonym ewaluacjom, które służyły ocenie Filtrowania Krawędzi.



Rysunek 1. Zdjęcie zawierające małe i liczne obiekty. Źródło: własne.

Rozdział 2

Stan wiedzy

„Małe obiekty” można definiować poprzez ich powierzchnie w pikselach, względną powierzchnię zajmowaną na obrazie lub szerokość. Najczęściej spotykana jest definicja, mówiąca, że małe obiekty to takie, których powierzchnia jest mniejsza bądź równa 1024 pikselom. Zakładając kwadratową ramkę obiektu, oznacza to szerokość wynoszącą 32 piksele. Definicja ta jest wykorzystywana, chociażby w domyślnych ustawieniach popularnej biblioteki języka Python pycocotools [9]. Jest ona również przyjęta w niniejszej pracy.

2.1 Przegląd zastosowań rozpoznawania małych i licznych obiektów

Rozpoznawanie małych i licznych obiektów znajduje zastosowanie w wielu praktycznych zadaniach. W niniejszej pracy zostały opisane wybrane z nich. Przedstawione przykłady nie wyczerpują wszystkich zastosowań SOD, aczkolwiek obrazują one ich mnogość oraz różnorodność. Podobnie do samych sieci neuronowych, część z nich znajduje wykorzystanie w praktyce już dzisiaj, podczas gdy inne to na razie rozważania teoretyczne. W każdym jednak przypadku zdaje się, iż optymalizacja działania sieci pod kątem rozpoznawania małych obiektów może przyczynić się do poprawy efektywności, bezpieczeństwa lub automatyzacji w obrębie danej dziedziny.

2.1.1 Autonomiczne samochody

Autonomiczne samochody to zagadnienie, które w ostatnich latach przyciąga znaczną uwagę zarówno środowiska naukowego, jak i koncernów motoryzacyjnych. Producenci samochodów już wprowadzają do sprzedaży pojazdy posiadające możliwość przejęcia kontroli nad swoim sterowaniem zgodnie z drugim lub trzecim poziomem autonomii [20, 49]. Przeszkodą w implementacji tego rozwiązania w praktyce są często regulacje prawne [44]. Przykładowo, polskie prawo dopuściło możliwość warunkowego testowania autonomicznych samochodów w ruchu drogowym dopiero w 2024 r. [42] Przyczyną takiego stanu rzeczy są często obawy związane z bezpieczeństwem [15, 53]. Skuteczne rozpoznawanie małych obiektów może przyczynić się do poprawy tego aspektu i w efekcie wspomóc rozwój całej dziedziny.

Rozdział 2. Stan wiedzy

W środowisku drogowym nieustannie występuje wiele klas obiektów, które powinny być rozpoznane z większej odległości:

- piesi,
- przejścia dla pieszych,
- znaki drogowe,
- samochody,
- światła,
- barierki przydrożne, ostrzegające o zakręcie,
- fotoradary,
- progi zwalniające.

Przykład znajduje się na rys. 2.

Samochód, dzięki wcześniemu rozpoznaniu takiego obiektu, ma szansę na niego lepiej zareagować np. zwalniając, gdy zbliża się do przejścia dla pieszych lub niebezpiecznego zakrętu. Może to realnie poprawić bezpieczeństwo użytkowania takich pojazdów, które mogą w przyszłości zminimalizować liczbę wypadków na drogach [53].

W ciągu ostatnich kilku lat powstało wiele prac naukowych badających zagadnienie widzenia komputerowego w kontekście autonomicznych pojazdów i SOD [16, 24]. Często wykorzystują one sieć YOLO, ze względu na jej zdolność przetwarzania obrazu w czasie rzeczywistym przy zachowaniu wysokiej liczby klatek na sekundę [34, 35, 50].

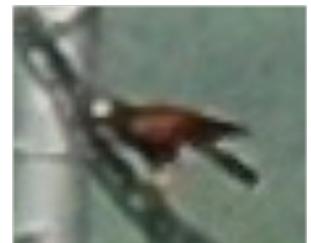


Rysunek 2. Zdjęcie sytuacji drogowej wraz ze zbliżeniem na oddalone znaki. Opracowano na podstawie: [5].

2.1.2 Drony

W ostatnich latach wyraźnie wzrasta wykorzystanie dronów w różnych dziedzinach od rolnictwa, przez szukanie defektów na liniach energetycznych, po zastosowania wojskowe. W każdym z wymienionych przykładów mogą występować małe instancje klas. Dzieje się tak szczególnie wtedy, gdy dron jest wyposażony w kamerę szerokokątną, która optycznie zmniejsza postrzegany rozmiar fotografowanych obiektów, a jednocześnie musi zachować od nich bezpieczny dystans, na przykład od linii energetycznych.

W takich sytuacjach dronom stawia się wysokie wymagania i oczekuje od nich niezawodności. Jest to istotne, zwłaszcza gdy urządzenie ma pracować autonomicznie bez nadzoru operatora. Jednym z możliwych zagrożeń są ataki ze strony agresywnych ptaków takich jak na rys. 3.



Rysunek 3. Zdjęcie zawierające drapieżnego ptaka na słupie energetycznym. Opracowano na podstawie: [12].

W Internecie są dostępne liczne nagrania z takich incydentów [17, 40, 48]. Niektóre modele, posiadają sensory umożliwiające wykrywanie przeszkód na drodze drona, ale są one ograniczone do obiektów znajdujących się jedynie w niewielkiej odległości [37]. Nie jest to odpowiednie rozwiązanie do wykrywania ptaków poruszających się z dużą prędkością. Oznacza to konieczność rozpoznawania zbliżających się zwierząt innymi metodami. Jedną z możliwości jest wykorzystanie LIDARu, ale nie jest on w stanie rozróżnić gatunków agresywnych. Ponadto jest to drogie rozwiązanie i zwiększające masę urządzenia.

Lepszym podejściem jest rozpoznawanie ptaków przez sieci neuronowe na ujęciach z kamery wysokiej rozdzielczości, będącej standardowym wyposażeniem wielu istniejących modeli. Taka sieć może rozróżnić gatunki ptaków, aby określić, czy stanowią one zagrożenie. Zwierzę znajdujące się w odległości kilkudziesięciu lub kilkuset metrów jest niewielkim obiektem, a jego wczesne rozpoznanie pozwala na szybszą reakcję.

O znaczeniu zagadnienia świadczy fakt powstania specjalnie dedykowanej mu bazy zdjęć [12]. Został nawet zorganizowany konkurs, w którym ponad 200 uczestników starało się różnymi metodami osiągnąć najlepszą skuteczność rozpoznawania ptaków w tymże zbiorze [18].

Problemem może być również analogiczna sytuacja, w której to dron wlatuje w przestrzeń, w której panuje zakaz lotów, np. nad lotniskiem lub obiektem infrastruktury krytycznej. Odpowiedni system wizyjny mógłby go wtedy automatycznie rozpoznać. Pojawia się tutaj problem rozróżnienia drona od ptaka [43]. Zbiór [46] powstał jako baza do treningu sieci właśnie pod tym kątem.

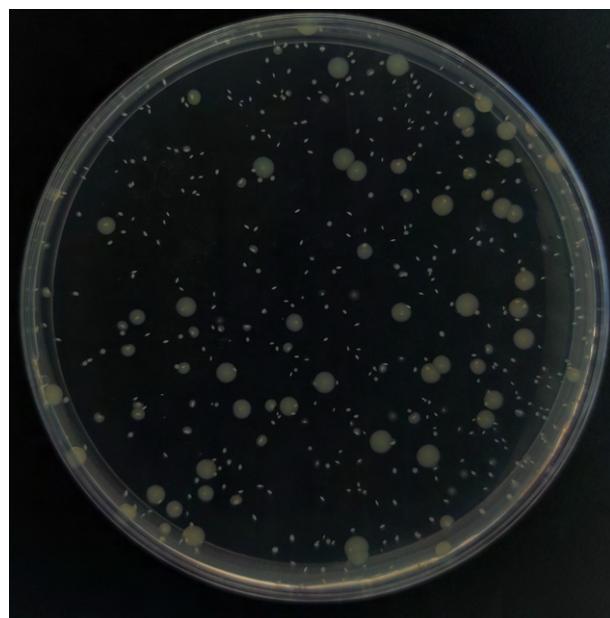
2.1.3 Liczenie kolonii bakterii

Liczenie kolonii bakterii (BCC – ang. *bacterial colony counting*) stanowi istotne zadanie w wielu dziedzinach [36, 59]. Są to przykładowo:

- mikrobiologia kliniczna,
- kontrola żywności,
- wakcynologia,
- badanie jakości wody.

Zapotrzebowanie na BCC stale rośnie, podczas gdy jest to czasochłonna czynność typowo wykonywana ręcznie [36], a błędy w tej dziedzinie mogą mieć katastrofalne skutki np. nierozpoznanie skażenia w zbiorniku wodnym lub błędna diagnoza pacjenta. Zastosowanie sieci CNN umożliwia automatyzację tego zadania. Było to obiektem wielu badań [11]. W pracy [59] udało się skrócić czas BCC z 4 minut do wartości poniżej sekundy przy zachowaniu skuteczności na poziomie 97,4%.

Niestety kolonie bakterii na zdjęciu obejmującym całą szalkę Petriego stanowią niewielkie, gęsto położone obiekty, co ilustruje rys. 4. Znacząco zwiększa to poziom trudności tego zadania. Zastosowanie sieci CNN i metod rozpoznawania małych obiektów może zwiększyć skuteczność procesu BCC i zmniejszyć liczbę błędnych wyników, ograniczając negatywne skutki pomyłek.



Rysunek 4. Zdjęcie kolonii bakterii *E. coli* na szalce Petriego, które rozwinęły się w 48 godzin. Źródło: [59].

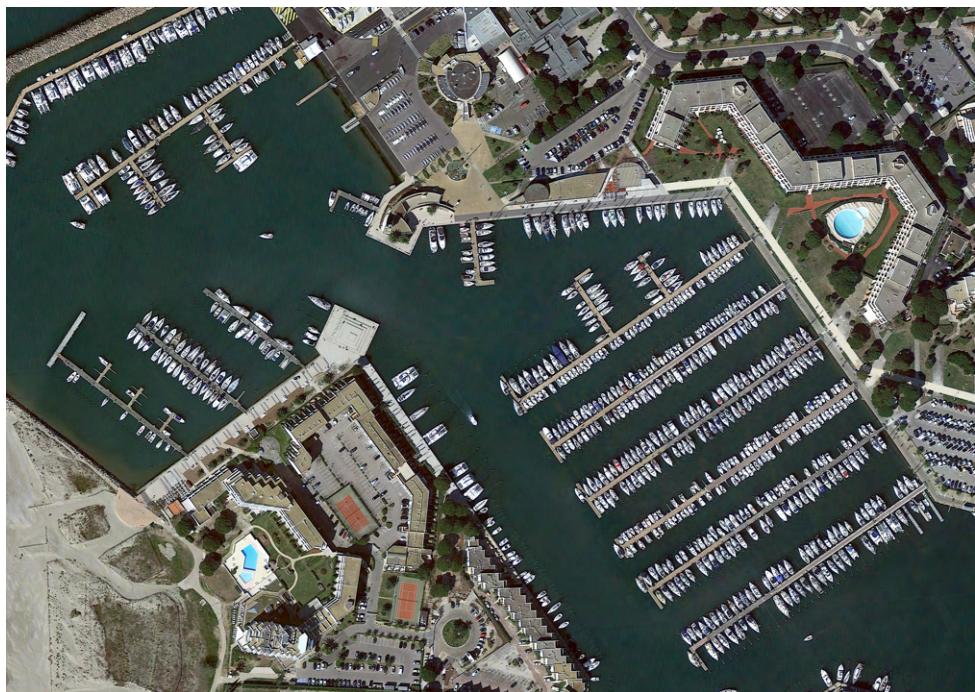
2.1.4 Analiza zdjęć satelitarnych

We współczesnym świecie na stałe zagościło wykorzystanie zdjęć satelitarnych. Znalazły one zastosowanie w wielu dziedzinach takich jak:

- urbanistyka,
- zarządzanie zasobami naturalnymi,
- kartografia,
- zarządzanie kryzysowe,
- badania ekosystemu,
- wojskowość.

O ile niektóre z wymienionych przykładów wykorzystują ocenę powierzchni całych obszarów, o tyle równie często spotyka się konieczność precyzyjnej lokalizacji i klasyfikacji pojedynczych instancji np. w zastosowaniach militarnych. Natura wykonywania fotografii z ogromnej wysokości sprawia, że ujęte na nich obiekty mają niewielkie rozmiary. Pomimo stale rosnących możliwości urządzeń zamontowanych w satelitach, problemem pozostaje niska rozdzielczość tychże obiektów.

W analizie zdjęć satelitarnych często wykorzystuje się OBB (ang. *Oriented Bounding Box*). Jest to wariacja klasycznej detekcji, w której ramki detekcji są zorientowane, tak aby oddać kierunek, w którym jest zwrócony obiekt. Jest to cecha takich zbiorów jak DOTA [8] czy SODA-A [5]. Zdjęcie na rys. 5 zawiera liczne łodzie o różnej orientacji.



Rysunek 5. Zdjęcie satelitarne zawierające wiele małych, licznych i różnie zorientowanych łodzi. Źródło: [8].

2.1.5 Monitoring

Rozpoznawanie osób na widoku kamer z monitoringu może się przyczynić do poprawy bezpieczeństwa w niektórych lokalizacjach np. poprzez automatyczne rozpoznawanie włamania. Kamery monitoringu często charakteryzują się szerokim kątem widzenia oraz niską rozdzielczością, jak zdjęcie na rys. 6. Obie te cechy utrudniają automatyczną detekcję. Na szczęście jakość obrazu nowych modeli ciągle rośnie.

Optymalizacja pod kątem rozpoznawania małych obiektów może umożliwić monitoring większych powierzchni, zwiększając bezpieczeństwo oraz skuteczność pojedynczej kamery. Takie rozwiązanie może znaleźć zastosowanie w lokacjach jak np. magazyny. Inną możliwością jest wykorzystanie SOD do rozpoznawania niebezpiecznych przedmiotów w przestrzeniach publicznych [26] lub wczesnego wykrywania pożaru [51].



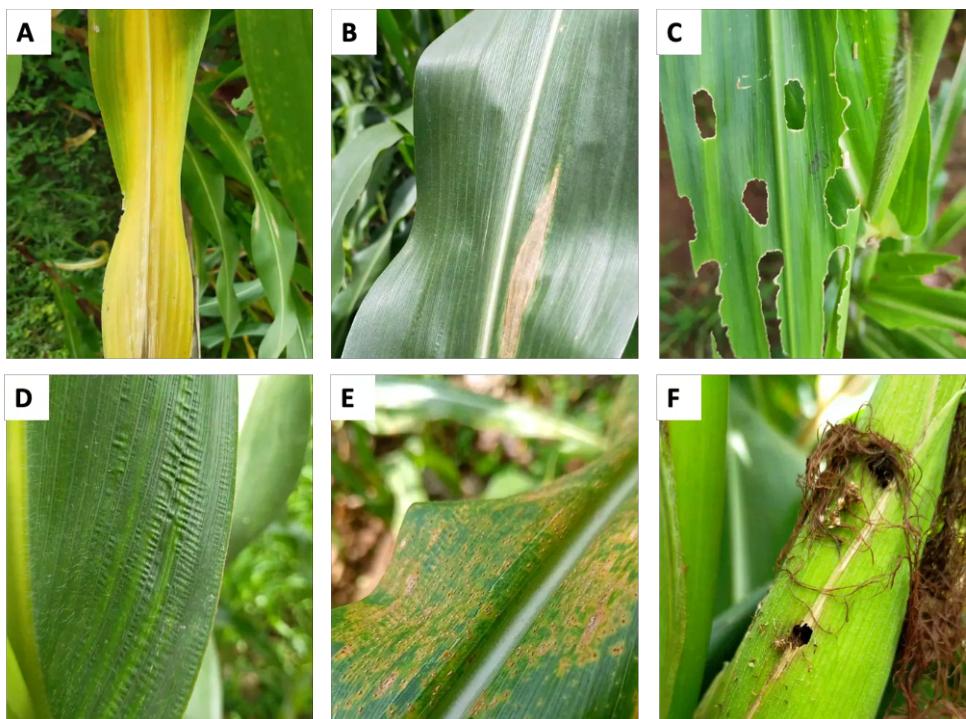
Rysunek 6. Zdjęcie przedstawiające scenę typową dla miejskiego monitoringu. Źródło: [62].

2.1.6 Rolnictwo

SOD znajduje również zastosowanie w nowoczesnym zautomatyzowanym rolnictwie. Sieci CNN do analizy obrazów umożliwiają ocenę stanu upraw, detekcję i klasyfikację roślin, owoców etc. Szczególne zainteresowanie w związku z rozpoznawaniem małych obiektów wiąże się z detekcją pasożytów, szkodników i chorób roślin. Ślady ich obecności mogą być początkowo niewielkie i trudne do zauważenia. Zautomatyzowane wczesne rozpoznanie umożliwia sprawne usunięcie zagrożenia.

2.2. Analiza przyczyn trudności w rozpoznawaniu małych obiektów

W 2020 r. ukazała się baza zdjęć Pest24 [52] specjalnie dedykowana rozpoznawaniu szkodników spotykanych w uprawach. Rozróżnia ona 24 gatunki agrofagów. Rozpoznanie konkretnego gatunku umożliwia zastosowanie specyficznego środka chemicznego i uniknięcie nadużycia pestycydów [52], które w nadmiernej ilości mogą stwarzać zagrożenie dla bezpieczeństwa żywności. Przykładowe klasy chorób roślin przedstawia rys. 7.



Rysunek 7. Przykładowe szkodliwe efekty działania chorób roślin. Źródło: [3].

2.2 Analiza przyczyn trudności w rozpoznawaniu małych obiektów

2.2.1 Ekstrakcja wysokopoziomowych cech

Konwolucyjne sieci neuronowe do rozpoznawania obrazów, jak sama nazwa wskazuje, posiadają w sobie warstwy konwolucyjne, które wykonują operację splotu obrazu z danym jądrem. W efekcie tej operacji powstają mapy cech, takich jak np. krawędzie, tekstury. Mapy te posiadają znaczną ilość informacji zbędnych z punktu widzenia typowego rozpoznawania obiektów. Z tego powodu wykonuje się na nich operację łączenia (ang. *pooling*). Łączenie analizuje grupę sąsiadujących ze sobą pikseli (np. czterech w kwadracie 2×2) i przypisuje im jedną wartość. W ten sposób zmniejsza się rozdzielcość analizowanego obrazu, a wydobywane są najistotniejsze informacje, które umożliwiają detekcję całych obiektów. Proces ten, zwany ekstrakcją cech, może się powtarzać kilkukrotnie, wydobywając coraz bardziej wysokopoziomowe cechy, jednocześnie coraz bardziej zmniejszając rozdzielcość zdjęcia.

Opisana metoda jest korzystna dla klasycznego rozpoznawania obiektów, które posiadają znaczną powierzchnię, która zawiera wiele cech. W przypadku detekcji małych obiektów sytuacja jest zgoła inna. Proces zmniejszania rozdzielczości nieuchronnie powoduje utratę niskopoziomowych informacji, a małe obiekty są obecne właśnie na niskim poziomie cech.

2.2.2 Rozdzielcość wejściowa sieci

Przed wykonaniem właściwej analizy rozmiar obrazu jest zmieniany, tak aby był on zgodny z wejściem sieci. Jest to część etapu przetwarzania wstępnego (ang. *preprocessing*). Na ogół zdjęcie ma większe wymiary niż wejście sieci, a zmniejszenie rozdzielczości oznacza utratę informacji podobnie jak w przypadku ekstrakcji cech. Niestety oczywiste rozwiązanie w postaci zwiększenia rozdzielczości wejściowej wiąże się ze znacznym wzrostem wymaganej pamięci karty graficznej. Problem ten jest szczególnie dotkliwy w przypadku zdjęć wysokiej rozdzielczości. Postęp technologiczny sprawia, że coraz częściej mamy kontakt z takimi obrazami. Właśnie one zawierają najwięcej detali i mają największy potencjał, aby rozpoznawać na nich małe obiekty. Rozdzielcość wejściowa stanowi w tym przypadku wąskie gardło ograniczające możliwość detekcji tychże obiektów. Różnicę w zdolności rozpoznawania małych obiektów w zależności od rozdzielczości wejściowej przedstawia rys. 8.



Rysunek 8. Lewa strona: oryginalne zdjęcie, prawy górny róg i prawy dolny róg: efekty detekcji sieci YOLOv8 przy rozdzielcości wejściowej równej odpowiednio 640×640 i 2560×2560 . Źródło: własne.

2.2.3 Ramki kotwiczne

Starsze modele sieci, między innymi z rodzin YOLO i R-CNN, wykorzystywały tzw. ramki kotwiczne (ang. *anchor boxes*). W uproszczeniu idea polegała na podzieleniu obrazu na mniejsze fragmenty o predefiniowanych wymiarach i położeniu, a następnie ich klasyfikacji. W kolejnym kroku sieć, znając położenie tychże fragmentów, czyli właśnie ramek kotwicznych, oraz prawdopodobieństwo wystąpienia na nich danej klasy, generowała końcowe ramki obiektów. Jednakże jeżeli obiekty są mniejsze od ramki kotwicznej, sieć nie może dokładnie stwierdzić położenia obiektu. Była to jedna z przyczyn zaniechania stosowania tego rozwiązania w nowszych generacjach.

2.2.4 Niewielkie wymiary obiektów

Jest to prawdopodobnie najbardziej oczywista trudność w rozpoznawaniu małych i licznych obiektów. Wraz z oddaleniem się przedmiotu lub osoby od aparatu, zmniejsza się jego postrzegana wielkość oraz liczba pikseli, które go reprezentują – twarz człowieka na zdjęciu portretowym znacznie różni się od tej samej twarzy na dalszym planie zdjęcia krajobrazu. Niska liczba pikseli sprawia, że obiekt ma mniej cech charakterystycznych, które go definiują i pozwalają rozpoznać, jak w przypadku zdjęcia na rys. 9. Prowadzi to jednocześnie do braku jakiejkolwiek detekcji oraz przyporządkowania do złej klasy. Dodatkowo dla małych obiektów łatwiej o ich częściowe lub całkowite zasłonięcie.



Rysunek 9. Zdjęcie, na którym ze względu na niską rozdzielcość, obiekty są trudne do klasyfikacji.
Opracowano na podstawie: [58].

2.2.5 Wariacja skali

Innym problemem, występującym w rozpoznawaniu małych obiektów, jest wariacja skali. Wspomniana w poprzednim punkcie twarz, w każdym z dwóch przypadków powinna być zakwalifikowana do tej samej klasy; jak również w każdym przypadku pośrednim. Zdolności generalizacji sieci są niestety ograniczone przez skalę obiektu, czyli jej względny rozmiar w obrębie obrazu. Sieć wytrenowana na zdjęciach portretowych będzie mieć problem z rozpoznawaniem małych twarzy ludzi w tłumie i vice versa. Analogicznie może być z autami na rys. 10.



Rysunek 10. Zdjęcie zawierające pojazdy w różnych skalach. Samochód ciężarowy po lewej jest znacznie większy niż położony dalej biały van, który z kolei jest większy niż inne znajdujące się w oddali samochody. Źródło: [5].

2.2.6 Zaszumienie danych

Zdjęcia cyfrowe są podatne na powstawanie szumów o wielu różnych źródłach. Są to przykładowo:

- czułość matrycy (ISO),
- stratna kompresja danych, np. w formacie *JPG*,
- temperatura, która powoduje powstawanie tzw. szumu termicznego,
- kwantyzacja.

Nie bez znaczenia są również właściwości optyczne obiektywu. Szkła niższej jakości powodują powstawanie mniej ostrych zdjęć oraz są bardziej podatne na zjawiska takie jak np. aberracja chromatyczna. Ponadto gdy obiekty są znacznie oddalone, mogą do tego dojść również zjawiska atmosferyczne takie jak deszcz czy mgła.

Wszystkie wymienione fenomeny przyczyniają się do powstawania zniekształceń na poziomie pikseli, które są nieistotne w rozpoznawaniu dużych obiektów. Generalizacja i dostosowywanie się do zaszumionych danych są jednymi z cech charakterystycznych sieci neuronowych. W sieciach CNN do zmniejszania wpływu szumu przyczyniają się chociażby opisane wcześniej operacje splotu i łączenia, które pozwalają wydobyć cechy wysokopoziomowe jednocześnie omijając niskopoziomowe zakłócenia.

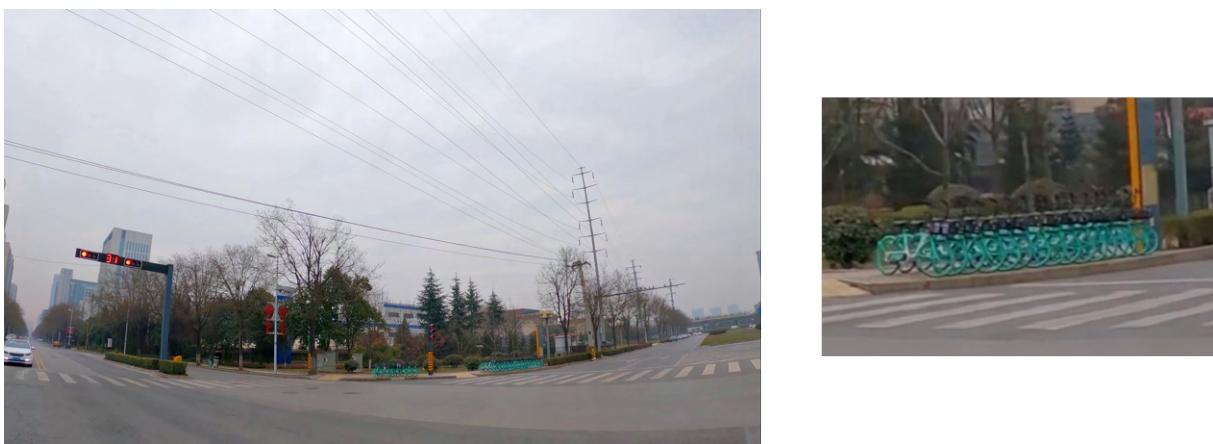
Niestety, gdy obiekty są małe i ich rozmiary są rzędu kilkunastu pikseli lub mniej, szum może wprowadzić stosunkowo duże zniekształcenie i niski kontrast między obiektem a tłem, jak ma to miejsce na rys. 11. Ma to wpływ jednocześnie na trening sieci oraz na jej działanie.



Rysunek 11. Zdjęcie wykonane z drona w niekorzystnych warunkach oraz zbliżenie na jego fragment zawierający pieszego o wymiarach 10×20 pikseli, który prawdopodobnie prowadzi rower. Zauważalny jest znaczny szum. Opracowano na podstawie: [62].

2.2.7 Zagęszczenie obiektów

Problemem jest również zagęszczenie małych obiektów, w wyniku którego wiele instancji klasy zostaje objętych jedną ramką. W wielu sytuacjach rozróżnienie poszczególnych obiektów może być trudne nawet dla człowieka, jak na rys. 12. Dla sieci CNN dodatkowy problem wynika z wykorzystania algorytmów takich jak *NMS*, *Confluence*, *NMM* czy *WBF*. Sieć na swoim wyjściu wskazuje zazwyczaj wiele ramek wokół każdego obiektu, z których każda obejmuje go w nieco inny sposób. Wspomniane algorytmy służą do wybrania jednej najlepszej ramki (*NMS*, *Confluence*) lub połączenia kilku ramek w jedną (*NMM*, *WBF*). W tym procesie ramki blisko położonych obiektów mogą nie zostać odpowiednio rozróżnione i wiele obiektów będzie objętych jedną ramką. Dzieje się tak szczególnie, gdy obiekty nakładają się na siebie.



Rysunek 12. Zdjęcie oraz zbliżenie na region o znacznym zagęszczeniu małych obiektów, które nakładają się na siebie. Opracowano na podstawie: [5].

2.2.8 Niska tolerancja na przesunięcie ramek w procesie uczenia

Do oceny błędu lokalizacji w procesie treningu sieci wykorzystuje się zwyczajowo parametr IoU (ang. *Intersection over Union*). Jego wartość jest składową błędu wykorzystywanego w procesie regresji wstecznej oraz ewaluacji. Mówiąc o nim, mówimy o stosunku części wspólnej faktycznej ramki obiektu z przewidzianą przez sieć do ich sumy. Definiuje się go przy użyciu poniższego wzoru:

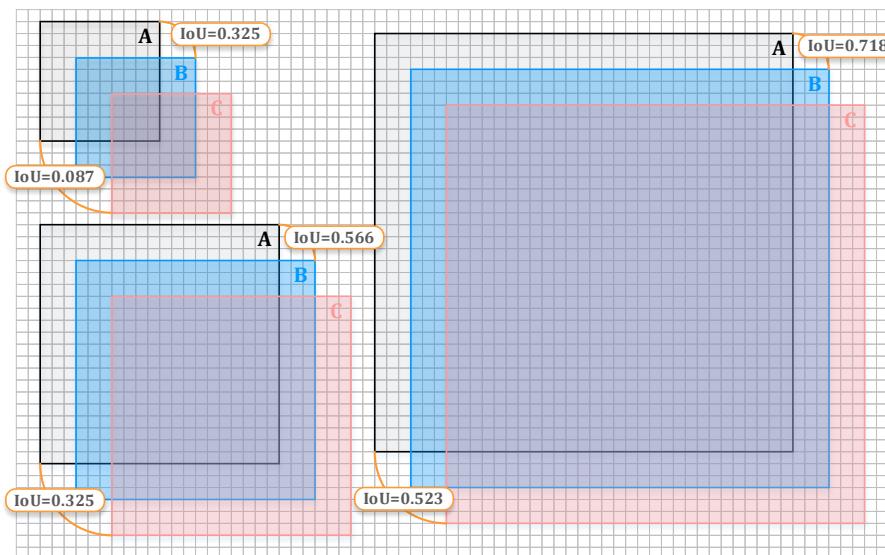
$$\text{IoU} = \frac{R_{rz} \cap R_p}{R_{rz} \cup R_p}, \quad R_{rz} - \text{ramka rzeczywista}, \quad R_p - \text{ramka przewidywana}$$

Niewielkie obiekty są bardziej wrażliwe na zmianę wartości IoU w wyniku przesunięcia przewidywanej ramki. Ilustruje to przykład na rys. 13. Szerokość jednej kratki odpowiada dwóm pikselom. A oznacza faktyczną ramkę obiektu, B przemieszoną o 6 pikseli, C przemieszoną o 12 pikseli. Przesunięcie o 6 pikseli sprawia, że IoU dla małego obiektu spada z wartości 1 do 0,325, podczas gdy dla dużego jest to 0,718. Przy większym przesunięciu różnica między obiektami jeszcze rośnie: 0,087 dla mniejszego obiektu, 0,523 dla większego.

Warto zaznaczyć, iż opisane zjawisko może się również dotyczyć niedokładnych adnotacji danych, ponieważ położenie ramki jest subiektywną decyzją osoby oznaczającej obiekt. Opisane wcześniej zjawiska, takie jak np. szum, utrudniają obiektywne określenie granic obiektu.

W praktyce oznacza to, że rzeczywiście niewielkie przesunięcie jest w trakcie treningu interpretowane jako znaczny błąd. Ponadto w trakcie ewaluacji sieci wykorzystuje się IoU do oceny poprawności detekcji – tylko ramka o IoU większym od danego progu (np. 0,5) jest klasyfikowana jako prawidłowa. Oba te zjawiska utrudniają proces uczenia sieci do rozpoznawania małych obiektów.

Ponadto błąd w położeniu ramki małego obiektu może również wynikać z rozpoznania przez sieć cienia obiektu jako jego części.



Rysunek 13. Zmiana wartości IoU w wyniku przesunięcia ramki dla obiektów różnych rozmiarów. Źródło: [5].

2.2.9 Brak zbiorów zdjęć nastawionych na rozpoznawanie małych obiektów

Trening głębokiej sieci neuronowej wymaga bazy danych zawierającej ogromną liczbę obrazów. Niestety dostępne powszechnie duże zbiory w znacznej większości składają się ze zdjęć niskiej rozdzielczości, na których małe obiekty stanowią jedynie niewielką część wszystkich instancji klas. Przez wiele lat relatywnie niska liczba odpowiednich zbiorów stanowiła zaporę dla środowiska naukowego w badaniach nad SOD. Dopiero w ostatnich latach zaczęły się pojawiać zbiory nastawione konkretne na rozwój tego zagadnienia. Nawet pośród nich występuje jednak problem zdjęć niskiej rozdzielczości nieadekwatnej do jakości współczesnych kamer i aparatów.

Istnieje kilka przyczyn takiego stanu. Po pierwsze SOD jest jedynie niszą całego zagadnienia rozpoznawania obiektów na zdjęciach. Po drugie opisane wcześniej zjawiska, takie jak zasumienie danych i zagęszczenie obiektów, utrudniają stworzenie dokładnych adnotacji. Jednocześnie zdjęcia małych obiektów posiadają więcej instancji, przez co ich opisywanie jest bardziej czasochłonne. Największe zbiory posiadają ich nawet ponad milion [5].

2.3 Przegląd istniejących rozwiązań

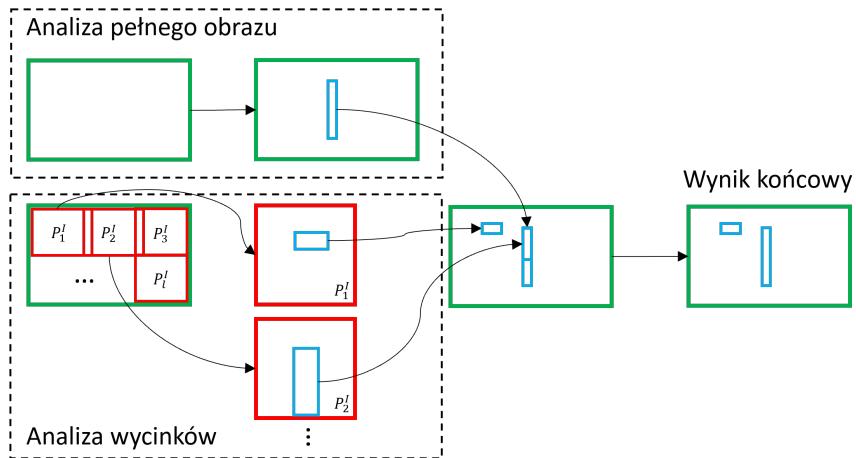
Architektury sieci neuronowych są niezwykle zróżnicowane i regularnie ukazują się ich nowe warianty, wprowadzające niespotykane wcześniej rozwiązania. Wiele z tych innowacji, pośrednio lub bezpośrednio, ma wpływ na rozpoznawanie małych i licznych obiektów; szczególnie że jest to powszechnie znany słaby punkt sieci CNN. Z tego powodu nie sposób wymienić wszystkich technik wykorzystywanych w tym celu. Niniejszy podrozdział stanowi przegląd wybranych metod, które pozwalają zobrazować różnorodność możliwych strategii.

2.3.1 Slicing Aided Hyper Inference

Jest to generyczny algorytm skierowany na SOD zwany w skrócie SAHI [1, 2]. Działa on w następujący sposób. Najpierw obraz jest dzielony na prostokątne fragmenty na kształt mozaiki, której elementy nakładają się na siebie. W niniejszej pracy będą one nazywane wycinkami (ang. *slices*). Na każdym z wycinków jest wykonywana oddzielna inferencja. Do tego wykonywana jest analiza pełnego obrazu. Na koniec wszystkie uzyskane detekcje są łączone ze sobą w finalny rezultat. Ilustruje to rys. 14.

Zastosowanie wycinków sprawia, że rośnie względna skala obiektów w obrazie bezpośrednio przetwarzanym przez sieć. Dzięki temu uzyskujemy mniejszą utratę informacji w wyniku działania warstw splotowych i łączących. Ponadto na etapie przetwarzania wstępного następuje mniejsza utrata informacji spowodowana dostosowaniem rozmiaru obrazu do rozdzielczości wejściowej sieci (lub żadna jeżeli rozmiar wycinka jest mniejszy bądź równy rozdzielczości wejściowej).

Istotne jest to, że wycinki częściowo nakładają się na siebie. Gdyby tak nie było, obiekt znajdujący się na granicy dwóch wycinków byłby „pocięty na pół”, tj. opisany dwiema stycznymi do siebie ramkami, z których każda obejmuje go w połowie, zamiast jedną obejmującą go w całości. Stopień, w jakim wycinki się nakładają, jest zmienny. Im większy, tym mniejsza szansa zajścia takiej sytuacji.



Rysunek 14. Schemat działania algorytmu SAHI. Obiekty rozpoznane w analizie pełnego obrazu są łączone z tymi rozpoznanymi na wycinkach, aby uzyskać efekt końcowy. Wycinki $P_1^I \dots P_l^I$ częściowo się na siebie nakładają. Opracowano na podstawie: [2]

Nadal jednak może się zdarzyć, że obiekt będzie większy niż pojedynczy wycinek. Zastosowanie inferencji na pełnym obrazie (FI – ang. *Full Inference*) umożliwia rozpoznanie takich obiektów. Dzięki temu nie tracimy zdolności rozpoznawania dużych obiektów. Jest to ważne, gdy na zdjęciu występują instancje klas w różnych skalach. Ten element odróżnia SAHI od metody przesuwającego się okna (ang. *sliding window approach*).

Algorytm został nazwany generycznym, ponieważ można go użyć, aby poprawić zdolność rozpoznawania niewielkich obiektów sieci CNN o dowolnej architekturze. Ponadto sieć nie musi być wytrenowana z myślą o rozpoznawaniu małych obiektów. Niemniej jednak taki proces uczenia lub douczenia zwiększa jej możliwości w połączeniu z SAHI. Douczanie sieci z wykorzystaniem wycinków zbioru treningowego to tzw. Fine Tuning [2].

Warto uwagi jest również fakt, iż SAHI nie wymaga większych zasobów pamięci obliczeniowej niż zwykła detekcja, w przeciwieństwie do podejścia polegającego na zwiększeniu rozdzielczości wejściowej sieci. Z drugiej strony jest on jednak wolniejszy.

Zwielokrotnienie czasu potrzebnego na analizę zdjęcia jest jego podstawową wadą. Wynika to z konieczności wykonania wielu inferencji zamiast jednej oraz połączenia ich wyników. Uniemożliwia to wykorzystanie go w zadaniach czasu rzeczywistego lub znacznie ogranicza jego możliwości.

Do tego, w wyniku nakładania się na siebie wycinków, w pewnych obszarach obrazu wykonywana jest redundantna analiza. Problem ten jest szerzej opisany w pracy [60], gdzie autorzy proponują również jego rozwiązanie zwane ASAHI (*Adaptive Slicing Aided Hyper Inference*).

Algorytm SAHI znalazł zastosowanie w wielu praktycznych aplikacjach [14, 30, 38, 47].

Ciekawe podejście zostało zaproponowane w artykule [33]. Zadanie polegało na śledzeniu ptaka w locie. Na początku algorytm wykonywał pełną analizę i po zlokalizowaniu zwierzęcia, przetwarzał jedynie wycinki, w których spodziewał się go znaleźć. W ten sposób liczba analizowanych wycinków została diametralnie zredukowana.

2.3.2 Piramida cech

Jednym z rozwiązań stosowanych w architekturze sieci CNN, które mają wpływ na zdolność rozpoznawania małych obiektów, jest piramida cech (ang. *Feature Pyramid*). Polega ona na przeskalowaniu obrazu wejściowego do różnych rozdzielczości i połączenia wyników detekcji uzyskanych dzięki nim. Istnieje kilka wariacji tej metody [31, 39, 57]. Przykładowo może się ona składać z obrazów o wymiarach 1024×1024 , 640×640 i 160×160 . Pierwotne zdjęcie jest przeskalowane do każdej z tych rozdzielczości, dzięki czemu powstają trzy niezależne obrazy wejściowe. Następnie, zakładając najprostszy wariant, sieć równolegle analizuje każdy z tych obrazów, jak gdyby były to trzy różne sieci i na koniec łączy uzyskane wyniki w ramach przetwarzania końcowego.

Dzięki temu model ma szansę przeanalizować to samo zdjęcie z perspektywy różnych rozdzielczości wejściowych. Neurony odpowiedzialne za obraz w wysokiej rozdzielczości mogą się wyspecjalizować w małych obiektach, ponieważ „dostrzegają” więcej szczegółów. Jednocześnie, te analizujące obraz o małych wymiarach będą mogły się skupić na dużych obiektach, które nadal są rozpoznawalne pomimo mniejszej rozdzielczości. Może to prowadzić do skuteczniejszego rozpoznawania małych obiektów i lepszej generalizacji dla zmiennej skali.

Wadą pozostaje fakt, że rozdzielczość zdjęcia jest zmniejszana. Nawet jeśli największy analizowany wariant obrazu wejściowego ma znaczne wymiary (jak na sieć neuronową), to nadal pewne informacje zostaną bezpowrotnie utracone.

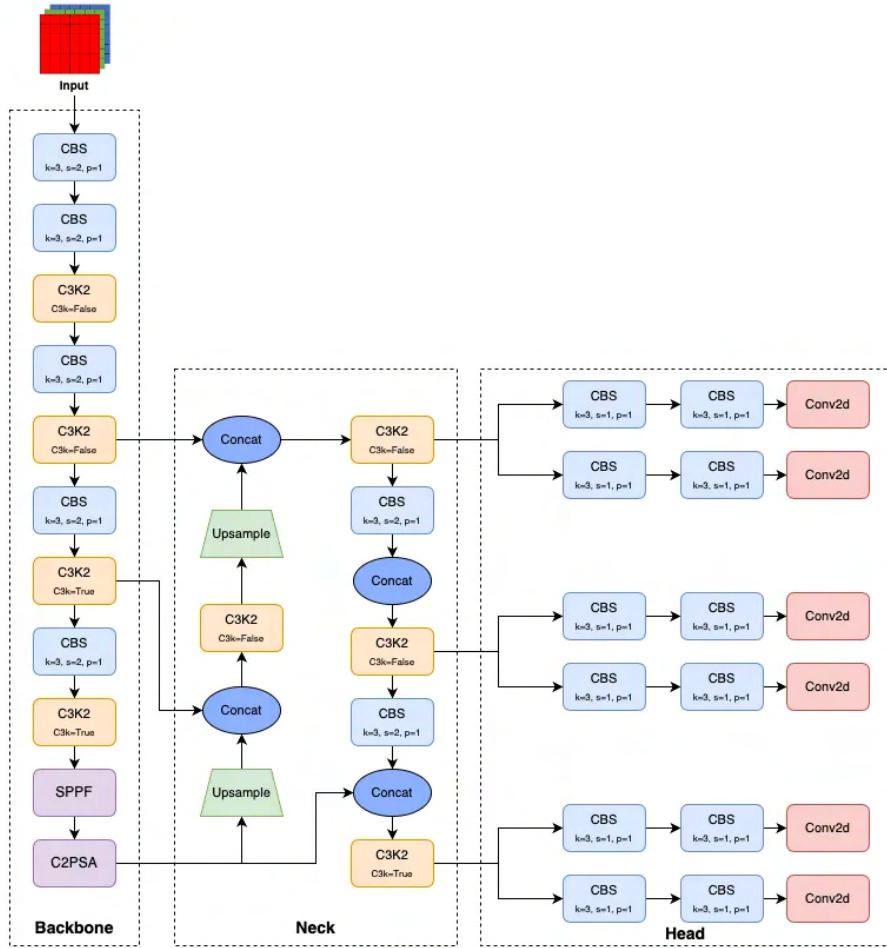
2.3.3 Wielopoziomowa ekstrakcja cech

Podobną ideę, znaną jako wielopoziomowa ekstrakcja cech, stosuje chociażby ujęta w części praktycznej pracy sieć YOLO11 [56]. Schemat jej architektury jest przedstawiony na rys. 15. Na etapie zwanym szyją (ang. *neck*) pobiera ona dane z ekstraktora cech (tutaj nazwanego *backbone*) na różnych poziomach. Dzięki temu niwelowany jest problem utraty informacji w kolejnych warstwach łączących. Jest tak, ponieważ dane służące do stwierdzenia detekcji nie zostały zaczerpnięte tylko z map cech najwyższego poziomu. Nieznaczną wadą tej metody jest skomplikowanie architektury sieci, która staje się nieliniowa.

2.3.4 Równoległe głowice detekcyjne

Inną techniką zaimplementowaną we wspomnianej sieci, która idzie w parze z wielopoziomową ekstrakcją cech, jest użycie kilku równoległych głowic detekcyjnych (ang. *detection heads*). Przykład ich wykorzystania w głowie sieci YOLO11 (ang. *head*) znajduje się na rys. 15. Każda przyjmuje na swoje wejście obraz w innej rozdzielczości (80×80 , 40×40 , 20×20) i specjalizuje się w rozpoznawaniu obiektów różnej skali. Dzięki temu rosną zdolności generalizacyjne sieci.

Wykorzystanie dodatkowej głowicy dedykowanej małym objetkom jest jednym z kierunków, jakie można objąć, aby optymalizować architekturę modelu w zadaniach SOD [19].



Rysunek 15. Schemat architektury sieci YOLOv11. Widoczne jest, jak część środkowa (*neck*) pobiera informacje z różnych warstw ekstraktora cech (*backbone*), które po przetworzeniu są interpretowane przez trzy głowice detekcyjne w części *head*. Źródło: [22].

2.3.5 Super rozdzielczość

Ta metoda polega na zastosowaniu innej sieci neuronowej w celu zwiększenia wymiarów obrazu, aby sztucznie odtworzyć detale małych obiektów utracone przez ograniczoną rozdzielczość. Jest to podejście alternatywne do próby wydobycia informacji z istniejące zdjęcia. Może ono się okazać lepsze w sytuacji, gdy obiekt jest niewyraźny lub częściowo przesłonięty. Sieć GAN (ang. *Generative Adversarial Network*) jest jednym z narzędzi, które naukowcy wykorzystywali w tym zadaniu, uzyskując obiecujące rezultaty [4, 41].

Ta strategia wydaje się szczególnie interesująca w kontekście gwałtownego rozwoju metod super rozdzielczości, który obserwujemy od kilku lat, chociażby w kontekście generowania grafiki w czasie rzeczywistym [54, 61].

2.3.6 Selekcja rejonu detekcji

Inną klasę metod reprezentuje sieć CDMNet [10]. Należy ona do technik zwanych z angielskiego „focus and detect”, co można przetłumaczyć jako „skupienie i rozpoznanie”. Polegają one na tym, że najpierw wybierane są pewne szczególne rejony zdjęcia, a następnie wykonywane są detekcje obiektów w tych rejonach. Dzięki temu mogą uzyskać większą skalę na analizowanym obrazie, podobnie do metody SAHI. Odbywa się to jednak mniejszym kosztem obliczeniowym, ponieważ przetwarzana jest tylko część zdjęcia.

W przypadku wspomnianej sieci CDMNet wybór obszarów jest podejmowany na podstawie zgrubnej mapy gęstości (ang. *coarse-grained density estimation*), generowanej przez inną sieć, która pozwala estymować prawdopodobieństwo wystąpienia obiektów w danym miejscu. Połączenie dwóch modeli w jednej architekturze utrudnia optymalizację sieci i komplikuje proces uczenia. Mimo to metoda ta wykazuje wysoką skuteczność w analizie zdjęć z dronów w środowisku miejskim.

Można dostrzec pewne podobieństwo tego podejścia do proponowanego w niniejszej pracy algorytmu SAHI z Filtrowaniem Krawędzi. Obie metody optymalizują rozpoznanie małych obiektów poprzez wykonywanie detekcji na wybranych fragmentach zdjęcia. Ponadto obie opierają się na założeniu, iż znaczną część fotografii można zignorować bez utraty skuteczności. Różnią się sposobem, w jaki wybierane są miejsca zainteresowania. Jedna z nich wskazuje obszary, na których chce się skupić, podczas gdy druga odrzuca wycinki, których nie będzie analizować. Istotne jest również to, że fragmenty zdjęcia w metodach *focus and detect*, w przeciwieństwie do algorytmu SAHI, mają zmienne wymiary. Zaletą FK zdaje się być jego prostota – zaznaczenie krawędzi na obrazie jest zadaniem wręcz trywialnym. Aby dokładniej porównać obie metody, należałoby przeprowadzić ich ocenę w praktycznym teście.

2.3.7 Modyfikacje procesu uczenia

Sposób, w jaki jest przeprowadzony trening sieci, ma wpływ na jej późniejsze zdolności w zakresie SOD. Rozwiązaniem problemu zmienności IoU może być jego zamiana na odległość między środkami ramki rzeczywistej i przewidywanej [55]. Takie podejście pozwala adekwatniej ocenić błąd lokalizacji małych obiektów w trakcie uczenia. W rezultacie cały proces przebiega efektywniej.

Inną możliwością jest podzielenie zdjęć zbioru treningowego na części [2]. W ten sposób obiekty są większe w skali obrazu i łatwiejsze do rozpoznawania przez sieć. Oczywistą wadą takiego podejścia jest to, że sieć będzie sobie radzić źle z detekcjami na pełnym zdjęciu. Z tego powodu obraz także musi zostać później podzielony na mniejsze części.

Augmentacja danych w procesie uczenia również nie jest bez znaczenia. Wprowadzenie wariancji skali do danych wejściowych może poprawić zdolności modelu do generalizacji dla obiektów o różnych wymiarach.

Rozdział 3

Opis przeprowadzonych prac

3.1 Koncepcja Filtrowania Krawędzi

Filtrowanie Krawędzi (FK) to proponowana w niniejszej pracy modyfikacja algorytmu SAHI. Polega ona na tym, że algorytm pomija wycinki nieposiadające krawędzi. Opiera się to na prostym założeniu, że jeżeli fragment obrazu nie posiada krawędzi, to nie będzie na nim żadnego obiektu. Jeśli tak jest, ograniczona zostanie liczba przetwarzanych wycinków, i co za tym idzie łączny czas wnioskowania, jednak bez utraty jakości.

Powodzenie działania algorytmu (przyspieszenie przetwarzania zdjęcia) zależy od tego, czy czas poświęcony na sprawdzenie, czy wycinki mają krawędzie, będzie krótszy niż czas zaoszczędzony dzięki pominiętym inferencjom. Oznacza to, że algorytm nie sprawdzi się do analizy zdjęć, które nie posiadają przestrzeni negatywnej. Jednakże w wielu praktycznych zastosowaniach obrazy posiadają takie obszary. Są to przykładowo:

- niebo,
- jezdnia,
- tło zdjęcia mikroskopowego,
- zbiorniki wodne na zdjęciach satelitarnych.

Można do tej listy dopisać również rozmyte tło, aczkolwiek praktyczne systemy częściej charakteryzują się wysoką głębią ostrości.

3.2 Realizacja Filtrowania Krawędzi

Do realizacji FK zostały wybrane dwie funkcje z biblioteki OpenCV [21]: Canny oraz countNonZero. Pierwsza z nich służy do wygenerowania obrazu zawierającego jedynie krawędzie, które są oznaczone przez białe piksele na czarnym tle. Uzyskane krawędzie są ostre i wyraźne, w przeciwieństwie do Filtru Sobela – alternatywnej rozważanej funkcji. Filtr ten również zwraca krawędzie obrazu, ale bardziej rozmyte i niewyraźne, co sprawia, że są mniej użyteczne do tego zadania. Druga z wybranych funkcji, countNonZero, zwraca liczbę nieczarnych pikseli obrazu. W tym przypadku oznacza ona powierzchnię zajmowaną przez krawędzie.

Jeżeli liczba białych pikseli (powierzchnia krawędzi) jest większa od danego progu, algorytm uznaje, że wycinek posiada krawędzie i dodaje go do listy przetwarzanych fragmentów. W przeciwnym wypadku wycinek jest ignorowany i program przechodzi do kolejnego. Przedstawia to poniższy pseudokod. Implementacja tego rozwiązania w kodzie SAHI wymaga dodania jedynie dwóch linijek. Usuwanie wycinków w tym procesie będzie zwane filtrowaniem.

```
for wycinek in wycinki
    krawedzie = Canny(wycinek)
    if countNonZero(krawedzie) > próg:
        dodaj wycinek do listy analizowanych wycinków
```

Funkcja Canny ma zmienne parametry, od których zależy wynik jej działania. Są one opisane w dalszej części pracy. Spodziewane jest, że jeżeli algorytm będzie zbyt czuły, tj. odrzuci zbyt wiele wycinków, jakość detekcji spadnie.

3.3 Specyfikacja sprzętowa

Do wykonania projektu wykorzystano komputer należący do Wydziału Elektrycznego Politechniki Warszawskiej działający na systemie Linux Ubuntu. Jest to jednostka o ogromnej mocy obliczeniowej wykorzystująca jedne z najwydajniejszych dostępnych obecnie podzespołów. Tab. 1 przedstawia dokładną specyfikację sprzętową.

Tabela 1. Specyfikacja sprzętowa jednostki obliczeniowej.

Podzespół	Model
Procesor (CPU)	AMD EPYC 7F72
Karta graficzna (GPU)	2 x NVIDIA RTX 5000 32 GB
Pamięć (RAM)	256 GB

3.4 Specyfikacja programowa

Dostęp do komputera odbywał się poprzez zdalne połączenie SSH za pośrednictwem wydziałowej sieci VPN. Dostęp do sieci VPN był możliwy dzięki programowi Cisco AnyConnect. Połączenie SSH było realizowane bezpośrednio poprzez wykorzystane IDE, tj. Visual Studio Code. Została do tego użyta wygodna funkcja Remote Server, która umożliwia pracę nad projektem znajdującym się na zdalnym serwerze, tak jakby był on dostępny lokalnie¹. Bazę zdjęć oraz adnotacji przesyłano na komputer przy użyciu programu WinSCP, wykorzystującego protokół sieciowy SFTP. Do uruchomienia treningu sieci oraz ewaluacji wykorzystano narzędzie Tmux. Dzięki niemu sesja terminala, w którym był uruchomiony kod, działała nawet po zerwaniu połączenia z serwerem. Cały kod powstały w ramach projektu był napisany w języku Python.

¹Autor chciałby podziękować Wojciechowi Sokołowskiemu za sugestię takiego rozwiązania.

Tab. 2 przedstawia wersje wspomnianych programów oraz systemu operacyjnego. Tab. 3 zawiera informacje o wykorzystanych bibliotekach oraz ich odsłonie.

Tabela 2. Specyfikacja programowa

Oprogramowanie	Wersja	Opis
Linux Ubuntu	24.04.1 LTS	System operacyjny
Jądro systemu	6.8.0-49-generic	Wersja komplikacji jądra
Python	3.12.3	Język programowania
Tmux	3.4	Narzędzie terminala systemu Linux
Visual Studio Code	1.96	IDE
WinSCP	6.3.6	Program do obsługi SFTP
Cisco AnyConnect	4.10	VPN

Tabela 3. Specyfikacja bibliotek

Biblioteka języka Python	Wersja
Ultralytics	8.3.39 ²
SAHI	0.11.19
OpenCV-Python	4.9.0.80
PyTorch	2.5.1
Torchvision	0.20.1
CUDA	12.4
cuDNN	9.1.0.70
NumPy	1.26.4
Pillow	11.0.0
Globox	2.4.7

3.5 Wybór modelu sieci

Do wykonania pracy została wybrana rodzina modeli YOLO11 [56] stworzona przez firmę Ultralytics. Jest to najnowsza generacja sieci YOLO (ang. *You Only Look Once*), która ukazała się 30 września 2024 r. Reprezentuje ona ostatnie osiągnięcia w rozwoju konwolucyjnych sieci neuronowych do analizy zdjęć. Posiada wiele cech wyróżniających ją od poprzednich generacji sieci YOLO oraz innych modeli, które były istotne dla wykonania pracy. Są to chociażby poprawiona ekstrakcja cech, istotna przy rozpoznawaniu małych obiektów czy zredukowana liczba parametrów, która umożliwia szybszy trening i skrócenie czasu inferencji [25]. Ponadto jest ona zintegrowana z przystępnią biblioteką Ultralytics napisaną w języku Python.

²Biblioteka SAHI była modyfikowana w ramach projektu.

Ważną cechą wszystkich generacji YOLO jest nastawienie na szybkość, co idzie w parze z celem pracy, jakim jest optymalizacja algorytmu SAHI pod kątem czasu przetwarzania.

W rodzinie YOLO11 znajduje się pięć modeli różniących się liczbą parametrów, szybkością działania oraz skutecznością. Wybrano dwa z nich: YOLO11n oraz YOLO11l. YOLO11n to najmniejszy oraz jednocześnie najszybszy dostępny model. Jego nazwa pochodzi od słowa *nano*. Został wybrany ze względu na najkrótszy czas wnioskowania, który jest istotny w zadaniach czasu rzeczywistego – takich jak prowadzenie autonomicznego samochodu. YOLO11l to przedostatni największy dostępny model. Litera *l* w nazwie pochodzi od słowa *large* (ang. *duży*). Charakteryzuje się lepszą skutecznością w rozpoznawaniu małych obiektów niż model nano [22]. Był to główny powód jego wyboru. Największy model, YOLO11x, został odrzucony ze względu na około dwukrotnie dłuższy czas inferencji przy jakości porównywalnej z modelem YOLO11l [22, 56].

Ponadto celem wyboru dwóch modeli sieci było sprawdzenie hipotezy, według której większy model uzyska większe przyspieszenie dzięki FK. Mogło się tak stać, ponieważ czas pojedynczej analizy jest wtedy większy, a zatem stanowi większą część całego czasu przetwarzania zdjęcia. Stąd spodziewane stosunkowo większe przyspieszenie.

3.6 Wybór zbioru zdjęć

Aby ocenić działanie algorytmu Filtrowania Krawędzi, konieczne było wybranie odpowiedniej bazy danych, która posłużyła do policzenia metryk. Ważne było kilka aspektów. Po pierwsze, co oczywiste, zbiór powinien zawierać liczne małe obiekty. Po drugie powinien wiązać się z zadaniem, którego przyspieszenie ma praktyczną wartość. Przykładowo w zadaniach czasu rzeczywistego skrócenie czasu przetwarzania pozwala uzyskać większą liczbę klatek na sekundę. Po trzecie zdjęcia powinny być wysokiej rozdzielczości. Dzięki temu można znacznie zmniejszyć rozmiar wycinka analizowanego przez algorytm SAHI. Jest to istotne, ponieważ gdy wycinki są małe, istnieje większa szansa na to, że będą też puste i, co za tym idzie, możliwe do odfiltrowania. Ponadto zdjęcia wysokiej rozdzielczości mogą zawrzeć w sobie mniejsze obiekty. Przykładowo, jeżeli założymy, że najmniejsza możliwa do rozpoznania przez człowieka instancja klasy ma wymiary 10×10 pikseli, to na zdjęciu o rozdzielczości 2000×2000 będzie ona stosunkowo mniejsza niż na zdjęciu o wymiarach 640×640 . Mówiąc kolokwialnie, większa rozdzielcość umożliwia większe „przybliżenie”. Do tego obserwowany rozwój kamer dąży w kierunku matryc o większej rozdzielczości i takie właśnie są coraz częściej wykorzystywane w zadaniach praktycznych.

Dodatkowym atutem są duże puste przestrzenie, które umożliwiają odfiltrowanie większej liczby wycinków i większe przyspieszenie algorytmu. Bez nich może się okazać, że czas potrzebny na sprawdzenie, czy na zdjęciu są krawędzie, będzie większy niż zaoszczędzony dzięki pominięciu wycinków.

W ramach projektu rozważano kilka różnych zbiorów zdjęć:

- TinyPerson [58],
- DOTA [8],
- SODA-A [5],
- SODA-D [5],
- VisDrone [62],
- xView [29],
- zdjęcia kolonii bakterii użyte w pracy [59].

Zbiory zawierające zdjęcia satelitarne, tj. DOTA, SODA-A, xView, zostały odrzucone. Wymienione w pkt 2.1.4 zastosowania zdjęć satelitarnych nie wymagają szybkiej inferencji – służą bardziej jako wstęp do analizy, na podstawie której są później podejmowane działania. Z tego powodu nawet stosunkowo długie czasy są tutaj akceptowalne, a w balansowaniu między szybkością a dokładnością, zdecydowanie ważniejsza jest dokładność. Niemniej potencjalną zaletą tychże zbiorów byłaby wysoka rozdzielcość zdjęć.

Zbiory VisDrone oraz TinyPerson nie zostały wybrane ze względu na niską rozdzielcość obrazów.

Autorzy pracy [59] badali możliwości rozpoznawania i liczenia kolonii bakterii przez sieci CNN. Na potrzeby eksperymentu stworzyli bazę zdjęć kolonii na szalkach Petriego. Niestety wiąże się z nią kilka mankamentów:

- zbiór zawiera tylko jedną klasę – błędne przypisanie do klasy jest jednym z wyzwań SOD,
- jest niezróżnicowany – wszystkie zdjęcia przedstawiają to samo ujęcie,
- podzbiór testowy nie posiada adnotacji – zadanie polega na zliczaniu kolonii, więc lokalizacja nie jest istotna i niemożliwe jest określenie precyzji sieci.

Z powyższych powodów on również został odrzucony.

Ostatecznie wybrano bazę SODA-D: Small Object Detection dAtaset – Driving. Zawiera ona 24 828 fotografii sytuacji drogowych, w których oznaczono 278 433 instancji dziewięciu klas obiektów ważnych z punktu widzenia kierowcy samochodu. Klasy są przedstawione na rys. 16. Zdjęcia zostały podzielone na podzbiór treningowy, walidacyjny i testowy. Zbiór przedstawia praktyczny problem rozpoznawania obiektów w środowisku drogowym istotny dla systemów wspomagania kierowcy i autonomicznych samochodów.

SODA-D jest bardzo zróżnicowany, co ma wpływ na zdolności generalizacyjne wytrenowanej na nim sieci. Zdjęcia pochodzą z ulic całego świata, były wykonywane w różnych warunkach pogodowych, o różnych porach dnia, na drogach różnej wielkości, w regionach o różnym stopniu urbanizacji. Przykładowe fotografie przedstawia rys. 17.

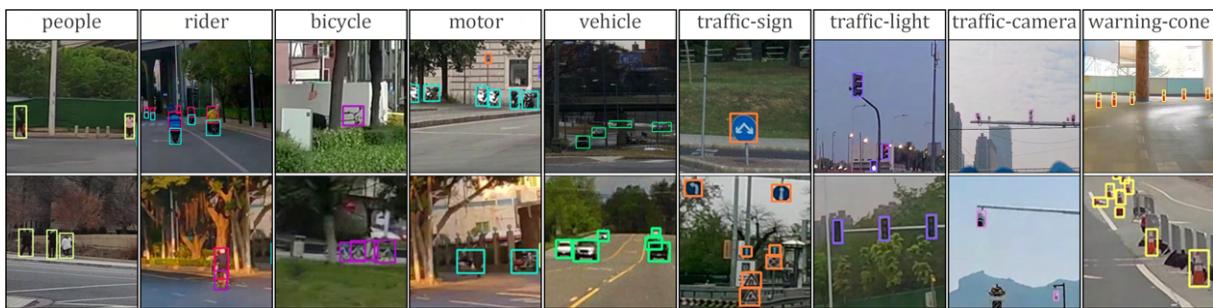
Jest to zbiór nastawiony na rozpoznawanie małych obiektów. Na tle innych podobnych zbiorów wyróżnia go również liczba ekstremalnie małych instancji, zdefiniowanych przez jego autorów jako obiekty o powierzchni mniejszej bądź równej 144 pikselom. Zawiera on ich aż 25 834. Instancje o powierzchni większej niż 2000 pikseli zostały oznaczone dodatkową klasą *ignore*. Dzięki temu obiekty te są pomijane w trakcie obliczania metryk oraz nie biorą udziału w procesie uczenia sieci.

Rozdział 3. Opis przeprowadzonych prac

Pozwala to w pełni skupić się na rozpoznawaniu małych obiektów. Ponadto klasa *ignore* została również wykorzystana, gdy obiekty są tak małe lub przysłonięte, że nie mogą zostać rozróżnione przez człowieka. Tym samym rozwiązyano niektóre z opisanych wcześniej problemów, takich jak niejednoznaczność adnotacji i niemożliwość lokalizacji obiektów przez ich zagęszczenie.

Zdjęcia charakteryzują się również wysoką średnią rozdzielczością, wynoszącą 3407×2470 , która umożliwia ustalenie niewielkiego rozmiaru wycinka algorytmu SAHI.

Inherentną cechą zdjęć drogowych jest występowanie dużych pustych powierzchni, głównie nieba i jezdni. Są one widoczne na przykładowych zdjęciach na rys. 17. Stwarza to możliwość odfiltrowania znacznej liczby wycinków i przedstawienie potencjału Filtrowania Krawędzi w praktycznej sytuacji.



Rysunek 16. Klasy zbioru SODA-D. Źródło: [5]



Rysunek 17. Przykładowe zdjęcia ze zbioru SODA-D, obrazujące jego różnorodność. Opracowano na podstawie: [5].

3.7 Przygotowanie zbioru zdjęć do uczenia sieci

Podstawowym zadaniem sieci miało być rozpoznawanie obiektów na wycinkach pełnego zdjęcia. Z tego powodu zbiory treningowy oraz walidacyjny bazy zdjęć podzielono na mniejsze wycinki o rozdzielczości 640×640 , bez ich wzajemnego nakładania. Przykładowy wycinek, będący wynikiem tej operacji, przedstawia rys. 18. Wykorzystano do tego moduł *slicing.py* biblioteki SAHI [1]. Dzięki temu rozwiązaniu sieć uczyła się na obrazach o rozdzielczości zbliżonej do wycinków, które później analizowała oraz, co nawet ważniejsze, posiadających obiekty w odpowiedniej skali.

Jest to rozwiązanie podobne do *Fine Tuning* zaproponowanego przez autorów SAHI [2]. Różnica polega na tym, że tam wycinki pełnych obrazów były rozszerzeniem zbioru treningowego, które miało sprawić, że sieć będzie skuteczna zarówno w analizie wycinków jak i pełnego obrazu. Tutaj wycinki stanowią całość zbioru treningowego. Praca [23] wykazała wyższość takiego rozwiązania w rozpoznawaniu małych obiektów. Tam również zastosowano dzielenie obrazu na wycinki o wymiarach 640×640 bez nakładania.

W wyniku „cięcia” liczba zdjęć w zbiorze treningowym wzrosła z 12 tys. do 138 tys., a w zbiorze walidacyjnym z 5 tys. do 57 tys. Poza tym powstały dwa towarzyszące im pliki *.json*, zawierające etykiety danych w formacie *COCO*. Indeksy klas „przesunięto” w taki sposób, aby najniższy indeks był równy „0”, a nie jak oryginalnie założyli twórcy zbioru „1”. Jest to konieczne do poprawnego działania sieci YOLO, która zakłada właśnie takie oznaczenia. Użyto do tego biblioteki *Globox – Object Detection Toolbox* [28]. Następnie wspomnianą bibliotekę wykorzystano do konwersji adnotacji z formatu *COCO* do formatu *YOLO*. Ze zbiorów zostały usunięte zdjęcia nieposiadające żadnych obiektów oraz odpowiadające im pliki *.txt*. Następnie usunięto zdjęcia, na których znajdowały się tylko instancje klasy *ignore*, wykorzystywanej wyłącznie do oceny działania modelu. W ten sposób finalna liczba obrazów w zbiorach treningowym i walidacyjnym wyniosła odpowiednio 42 292 oraz 17 398.

Rozważaną opcją było również umieszczenie w zbiorze treningowym części zdjęć stanowiących jedynie tło, czyli nieposiadających obiektów. W teorii mogłoby to zwiększyć różnorodność zbioru oraz zmniejszyć liczbę fałszywych pozytywów. Taki krok nie został jednak wykonany, ponieważ obiekty w wybranej bazie zdjęć są bardzo małe i nawet na wycinkach o rozmiarze 640×640 znajduje się znaczna ilość pustego tła.

3.8 Trening sieci

Obie wybrane sieci wytrenowano w identyczny sposób przez 100 epok. Podawaną na raz liczbę próbek (ang. Batch Size) ustawiono na 32. Oba te parametry zostały dobrane przez autora eksperymentalnie. Są to typowe wartości, tak jak rozdzielcość wejściowa, która była równa rozmiarowi pojedynczego wycinka w otrzymanym zbiorze treningowym, czyli 640×640 . Pozostałe parametry zostały dobrane przez skrypt uczący sieć YOLO i algorytm optymalizujący Stochastic Gradient Descent (SGD). Dokładne parametry treningu obu sieci przedstawia tab. 4.



Rysunek 18. Przykładowe zdjęcie oraz powstały z niego wycinek, zawierający znak, światła drogowe i osobę.
Opracowano na podstawie: [5].

Wartości osiągnięte na zbiorze walidacyjnym mogą nie być do końca zgodne z prawdą z powodu usunięcia klasy *ignore* w procesie przygotowania zbioru do treningu. Małe nakładające się na siebie instancje klas (wcześniej oznaczone klasą *ignore*) mogą być rozpoznane przez sieć, podczas gdy algorytm uczący zinterpretuje to jako fałszywy pozytyw. Może to mieć również wpływ na suboptymalny przebieg uczenia. Należy zaznaczyć, iż sam trening sieci w kontekście niniejszej pracy jest jedynie środkiem do celu i autor nie ma intencji optymalizacji procesu uczenia.

Przed ukończeniem treningu sieci YOLO11l problemem stało się przetrenowanie (ang. *overfitting*), co jest widoczne na rys. 19. Niestety większy model jest bardziej podatny na to zjawisko [22]. Wraz z postępem uczenia wartości funkcji celu uzyskane na zbiorze walidacyjnym zaczęły rosnąć, podczas gdy błędy zbioru treningowego nadal malały. Możliwym rozwiązaniem byłoby wykorzystanie dropout-u lub zmiana wartości batch size. Finalnie wybrano wagę, dla których sieć osiągnęła najwyższy wskaźnik mAP50 równy 0,614. Stało się to dla epoki 74. Sieć nano nie miała takiego problemu. Przebieg jej uczenia przedstawia rys. 20.

Widoczny na wykresach funkcji celu skok obserwowany przy epoce 90. jest efektem wyłączenia augmentacji danych poprzez mozaikę w ciągu ostatnich 10 epok. W teorii ma ono za zadanie stabilizować trening przed jego końcem. Wyraźna poprawa wyników od 90. epoki sugerowała, że być może całkowite wyłączenie tej augmentacji pozwoliłoby osiągnąć finalnie lepszy rezultat. Sprawdzono to, trenując sieci ponownie bez użycia mozaiki. W przypadku sieci YOLO11l, z powodu przetrenowania, ponownie wybrano epokę, dla której wartość mAP50 była najwyższa. Tym razem była to epoka 71. Okazało się, że poza czułością dla sieci nano i precyzją dla sieci large wszystkie metryki są wyższe lub równe, gdy augmentacja poprzez mozaikę była obecna. Z tego powodu modele te nie zostały wykorzystane.

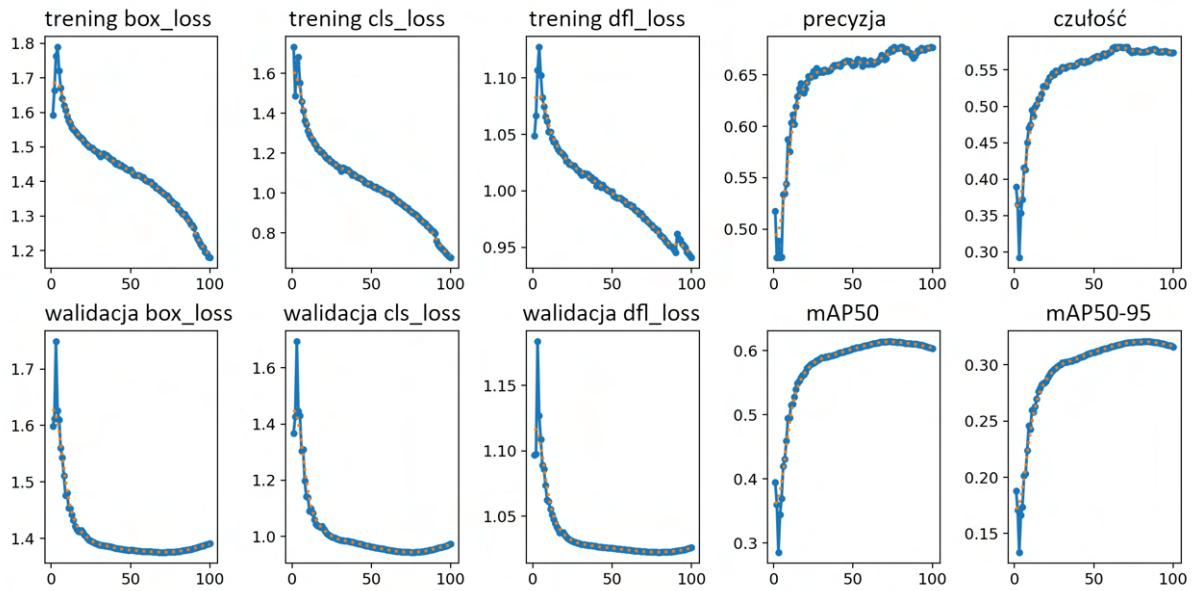
Finalne rezultaty uzyskane we wszystkich czterech treningach są zawarte w tab. 5. Czasy treningów sieci YOLO11n i YOLO11l wynosiły odpowiednio 5 i 13 godzin. Zgodnie z oczekiwaniami większy model uzyskał lepsze wyniki, ale efekty uczenia obu sieci są satysfakcyjne i umożliwiają analizę zastosowania algorytmu SAHI. Przykład detekcji wykonanych przy użyciu tego algorytmu i wytrenowanej sieci YOLO11n znajduje się na rys. 21.

Tabela 4. Parametry treningu sieci neuronowych.

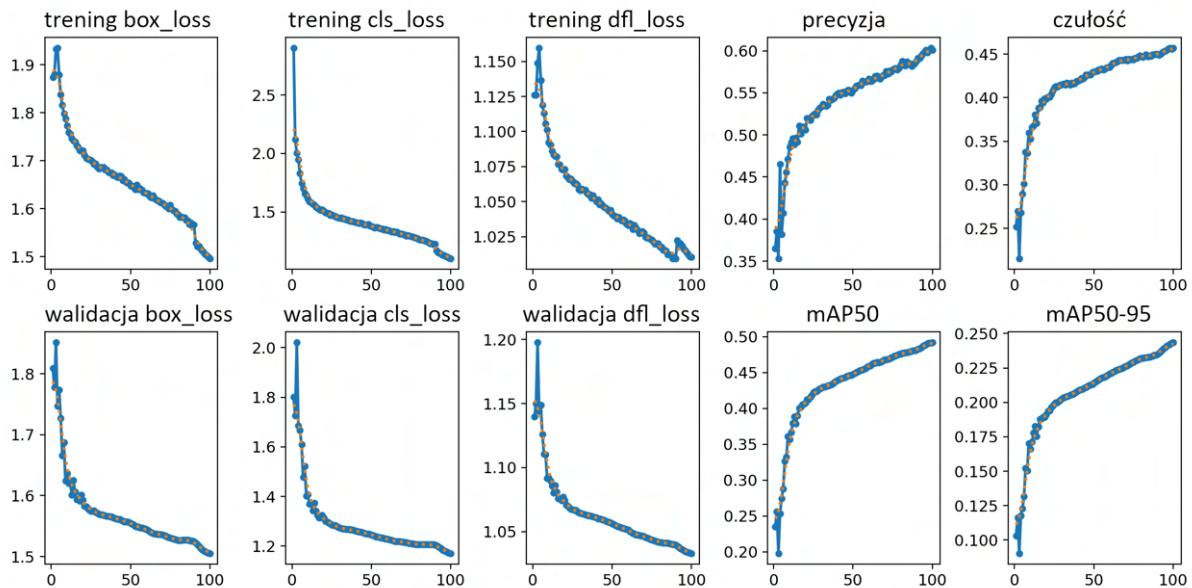
Parametr	Wartość
Liczba epok	100
Rozdzielcość wejściowa	640×640
Jednorazowa liczba próbek	32
Szybkość uczenia	0,01
Waga rozkładu	0,0005
Epoki rozgrzewkowe	3
Wagi błędów box/cls/dfl	7,5/0,5/1,5

Tabela 5. Metryki uzyskane przez sieci YOLO11n i YOLO11l. Dopisek „bez mozaiki” oznacza, że w treningu nie wykorzystano augmentacji danych poprzez mozaikę.

Metryka	YOLO11n	YOLO11n bez mozaiki	YOLO11l	YOLO11l bez mozaiki
czułość	0,457	0,464	0,574	0,568
precyzja	0,600	0,597	0,675	0,676
mAP50	0,492	0,491	0,614	0,606
mAP50-95	0,244	0,242	0,320	0,315
F1	0,51	0,51	0,62	0,61



Rysunek 19. Wartości błędów na zbiorze treningowym i walidacyjnym oraz metryki mAP50 i mAP50-95 w trakcie trwania treningu sieci large. Box_loss, cls_loss oraz dfl_loss to odpowiednio funkcje celu odpowiadające za poprawność lokalizacji, klasy oraz rozkładu położenia ramek.



Rysunek 20. Wartości funkcji celu dla zbioru treningowego i walidacyjnego oraz metryki obliczone na zbiorze walidacyjnym w trakcie uczenia sieci YOLO11n.



Rysunek 21. Przykładowe wyniki detekcji na dwóch zdjęciach ze zbioru SODA-D przy użyciu SAHI i wytrenowanej sieci YOLO11n. Małe obiekty są skutecznie rozpoznawane z dużą dokładnością.

Rozdział 4

Analiza wyników badań

4.1 Terminy

Na potrzeby niniejszej pracy zostały wprowadzone dwa terminy służące do scharakteryzowania działania algorytmu. Są to podział i przyspieszenie.

4.1.1 Podział

Twórcy algorytmu SAHI, przy ocenie jego działania, zakładali sztywne wymiary pojedynczego analizowanego wycinka równe 640×640 [2]. Niestety, takie podejście sprawia, że wycinek zajmuje różną część obrazu zależnie od rozdzielczości zdjęcia. Obrazując to na liczbach: wycinek o wymiarach 100×100 stanowi $\frac{1}{4}$ obrazu o wymiarach 200×200 . Jeżeli użyjemy go do analizy obrazu o rozmiarze 400×400 , będzie stanowił $\frac{1}{16}$. Jeśli oba zdjęcia były wykonane przez obiektywy o takim samym kącie widzenia, a różnią się jedynie rozdzielczością, spowoduje to zmianę względnego rozmiaru obiektów w obrębie wycinka. Jest to problem, gdy jeden model chcemy zastosować do urządzeń o różnej rozdzielczości. Sieć neuronowa ma zdolność generalizacji do obiektów w różnej skali. Jednak znaczna wariacja może mieć wpływ na pogorszenie detekcji, co zostało opisane w pkt 2.2.5.

Aby rozwiązać ten problem, w pracy wykorzystano wycinki o stałej proporcji ich powierzchni do powierzchni całego obrazu. Inaczej mówiąc, wycinek zawsze zajmuje taką samą część powierzchni zdjęcia. Przykładowo, jeżeli chcemy, aby pojedynczy analizowany fragment zajmował czterć pełnego obrazu to dla zdjęcia o wymiarach 100×100 przyjmie on rozmiar 50×50 , a dla obrazu 800×800 będzie to 400×400 . Dokładne wymiary wycinka są charakterystyczne dla danego zdjęcia. Stosunek powierzchni wycinka do powierzchni obrazu jest stały, w tym przypadku równy $\frac{1}{4}$. Jest to istotne w przypadku zbioru zdjęć SODA-D, ponieważ podzbiór testowy zawiera zdjęcia o różnych wymiarach.

Autor uważał takie podejście za bardziej praktyczne. Podobna idea jest eksplorowana przez autorów [60]. Takie rozwiązanie sprawia, że obiekty wewnętrz wycinka mają zbliżoną skalę. Nadal jednak będzie ona zmienna z powodu zmienności kąta widzenia kamer użytych do wykonania zdjęć, ale ma to mniejszy wpływ niż zmiana rozdzielczości. Ponadto pozwala to ocenić, jaki jest optymalny wzajemny rozmiar wycinka dla danego zastosowania, i przenieść ten rozmiar na inne urządzenie.

Rozdział 4. Analiza wyników badań

Podział służy do opisania, jaki jest względny rozmiar wycinka w stosunku do całego zdjęcia. Definiuje się go jako stosunek szerokości i wysokości obrazu do szerokości i wysokości wycinka. Wartości podziału nie muszą być liczbami całkowitymi, aczkolwiek tylko takie były badane w ramach pracy. Przykład podziału równego 3 przedstawia rys. 22.

$$\text{podział} = \frac{\text{szerokość obrazu}}{\text{szerokość wycinka}} = \frac{\text{wysokość obrazu}}{\text{wysokość wycinka}}$$

Powyższa definicja sprawia, że wycinek ma proporcje równe proporcjom obrazu wejściowego. Fakt ten może się wiązać z pewnym mankamentem. Na etapie przetwarzania wstępnego sieć nie tylko zmienia rozdzielcość obrazu, ale również jego proporcje. Trudno ocenić praktyczny wpływ tego faktu na działanie sieci. Jeżeli wynikają z tego jakieś negatywne konsekwencja dla działania algorytmu, to najprawdopodobniej są one większe dla obrazów o proporcjach dalszych od kwadratu np. formatu scope.

Należy zaznaczyć, że liczba wycinków przypadających na obraz niekoniecznie jest wtedy stała. Wynika to z tego, że obliczony teoretycznie rozmiar wycinka jest poddawany operacji *entier*, aby uzyskać liczbę całkowitą. W niektórych przypadkach może to prowadzić do drobnej różnicy w liczbie wycinków. Podobny problem jest opisany w [60].

Niemniej wykorzystanie stałych wymiarów również ma swoje uzasadnienie, ponieważ w taki sposób była trenowana sieć, a w zastosowaniu praktycznym rozdzielcość użytej kamery pozostaje stała. Do tego wykorzystując stały rozmiar wycinka, równy wejściu sieci, oszczędza się czas potrzebny w etapie przetwarzania wstępnego poświęcony na przeskalowanie obrazu. Jest to rozwiązanie sprawdzone w późniejszej części pracy.



Rysunek 22. Przykład podziału równego 3 przy zerowym nakładaniu. Każdy wycinek ma szerokość i wysokość równą odpowiednio $\frac{1}{3}$ szerokości i $\frac{1}{3}$ wysokości pełnego obrazu. Źródło: własne.

4.1.2 Przyspieszenie

Przyspieszenie definiuje się jako stosunek czasu przetwarzania zdjęcia¹ przez algorytm SAHI bez zastosowania Filtrowania do czasu przetwarzania zdjęcia przez tenże algorytm z Filtrowaniem. Mówiąc on, ile razy szybciej wykonywana jest analiza dzięki zastosowaniu filtrowania.

$$\text{przyspieszenie} = \frac{\text{czas przetwarzania bez Filtrowania}}{\text{czas przetwarzania z Filtrowaniem}}$$

4.2 Metryki

Do oceny jakości sieci użyto konwencji zaproponowanej w [5], która skupia się na skuteczności rozpoznawania małych obiektów. Definiuje się podział wszystkich instancji klas na podzbiory Mały i Normalny (ang. *Small* i *Normal*), w zależności od ich powierzchni. Podzbiór *Mały* dzieli się dalej na Ekstremalnie Mały, Relatywnie Mały i Generalnie Mały. Pozwala to lepiej ocenić wpływ badanych parametrów na skuteczność sieci w badanym zagadnieniu. Dokładne wartości powierzchni, według których instancje były przyporządkowywane do zbiorów, przedstawia tab. 6.

Tabela 6. Przedstawienie rozmiarów, według których jest oceniana skuteczność sieci. Powierzchnia odnosi się do liczby pikseli.

Podzbiór	Ekstremalnie Mały	Relatywnie Mały	Generalnie Mały	Normalny
Skrót	eS	rS	gS	N
Nazwa oryginalna	extremely Small	relatively Small	generally Small	Normal
Powierzchnia	(0, 144]	(144, 400]	(400, 1024]	(1024, 2000]

Za najważniejszą metrykę uznaje się AP, definiowane jako mAP50-95 dla obiektów o rozmiarze małym. AP50 i AP75 to odpowiednio mAP50 i mAP75 również dla rozmiaru małego. APeS, APrS, APgS i APN to mAP50-95 dla odpowiadających podzbiorów. Wartości są podawane w formacie procentowym, stąd są to liczby z zakresu od 0 do 100 i takie będą używane w niniejszej pracy od tego momentu.

Autor popełnił błąd przy implementacji powyższych definicji i we wszystkich ewaluacjach metryka AP75 była liczona dla wszystkich obiektów, a nie tylko dla małych. Z tego powodu jej wartości są zaniżone i nie można jej porównać z wartościami otrzymanymi we wspomnianym artykule. Nadal można jednak porównywać jej wartości między ewaluacjami wykonanymi w ramach niniejszej pracy. Dla uniknięcia niejasności metryka obliczana w ramach pracy będzie oznaczona jako AP75*.

¹Przez czas przetwarzania zdjęcia rozumiany jest łączny czas przetwarzania wstępnego, wszystkich inferencji i przetwarzania końcowego.

4.3 Ewaluacje

W pierwszej kolejności wykonano seria ewaluacji odniesienia, tj. samego algorytmu SAHI bez FK. Później kolejne serie wykonywano z zastosowaniem Filtrowania Krawędzi. Dokładne parametry opisano w dalszej części tekstu. Wartością modyfikowaną był podział, który zmieniał się od wartości 1 do 12 dla serii odniesienia i od 2 do 12 dla serii z FK – zastosowanie filtrowania do analizy podziału 1, czyli całego zdjęcia, nie ma większego sensu, ponieważ każde zdjęcie posiada krawędzie. Dokładne wyniki wszystkich serii znajdują się w zał. 1.

Do przetwarzania ramek końcowych został wybrany najczęściej spotykany algorytm NMS. Czułość IoU algorytmu SAHI była ustalona na 0,45, a czułość modelu na 0,05. Wcześniejsze eksperymenty autora na fragmencie badanego zbioru, zawierającym 50 zdjęć, wskazały te dwie wartości jako dające satysfakcyjne rezultaty. Nakładanie się wycinków zostało ustalone na 0,25, co jest typową wartością.

Mierzone były opisane metryki, jak również średni czas przetwarzania jednego zdjęcia oraz łączna liczba wycinków z rozróżnieniem na analizowane i odrzucane w wyniku filtrowania. Wszystkie wymienione wartości zapisywano do pliku .txt razem z hiperparametrami danej ewaluacji i czasem wykonania. Możliwym usprawnieniem byłoby zapisywanie wyników w formacie CSV, JSON lub XML.

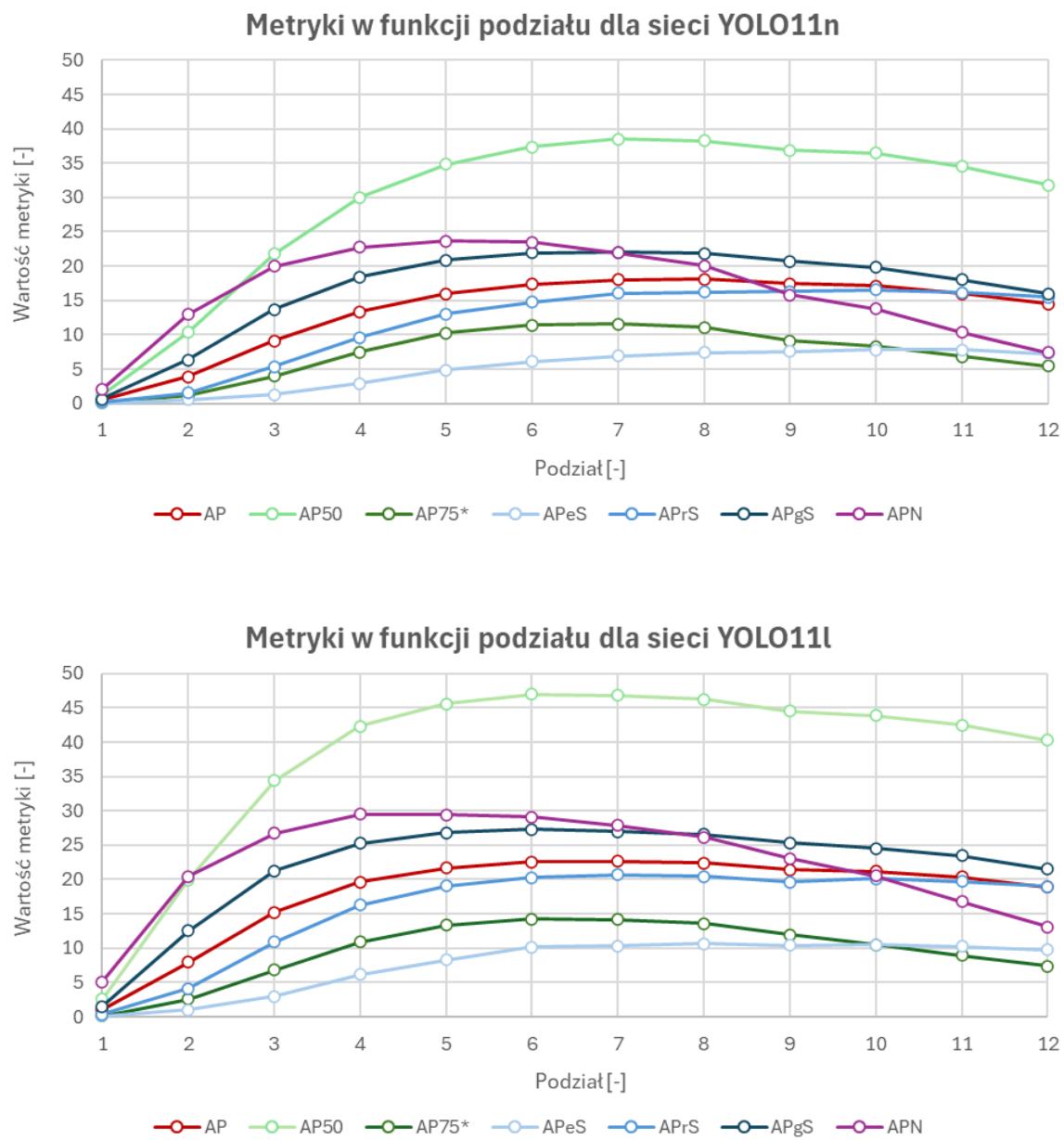
Czas trwania każdej z opisanych serii ewaluacji wynosił od około jednego do dwóch dni.

4.4 Seria ewaluacji odniesienia

Zebrane metryki dla sieci nano i large są przedstawione na rys. 23. Na wykresach można łatwo zaobserwować, jak zmieniają się wartości danych wskaźników wraz ze zmniejszaniem rozmiaru wycinka. Dla każdego analizowanego rozmiaru obiektów metryka osiąga maksimum dla innej wartości podziału. Im większy podział, tym lepsza skuteczność dla mniejszych obiektów. Oczywiście dzieje się tak do pewnej wartości maksymalnej, powyżej której opisane wskaźniki zaczynają spadać. Mówiąc inaczej: rozmiaru wycinka nie można zmniejszać w nieskończoność; w pewnym momencie wartość metryki zacznie spadać. Dla każdego badanego wskaźnika udało się ten punkt przekroczyć, co pozwala szacować optymalny podział. Jak zostało wcześniej wspomniane, wartości podziału nie muszą być liczbami całkowitymi.

Oznacza to, że zmniejszanie rozmiaru analizowanego wycinka jest korzystne z punktu widzenia SOD. Zdaje się to być oczywiste, aczkolwiek według wiedzy autora nikt do tej pory nie wykazał tego formalnie ani dla wycinków o stałych wymiarach, ani dla wycinków o stałych proporcjach.

Porównując sieć nano i large, należy zwrócić uwagę na ciekawe zjawisko. Większa z nich osiąga wartości maksymalne dla mniejszego podziału. Można powiedzieć, że „sieć large preferuje większe wycinki”. Sieć YOLO11n osiąga maksymalną wartość głównego wskaźnika AP dla podziału 8, stąd kolejne eksperymenty były wykonywane dla tejże wartości.



Rysunek 23. Metryki uzyskane przez sieci YOLO11n i YOLO11l w funkcji podziału.

4.5 Ponowne zbadanie czułości modelu

Jak wcześniej wspomniano, wartość czułości została wybrana przez autora drogą eksperymentalną na małej części zbioru testowego. Aby zweryfikować tę wartość, wykonano cztery dodatkowe ewaluacje sieci nano, co razem daje pięć zbadanych wariantów. Porównywana była wartość AP. Otrzymane wyniki przedstawia Tabela 7. Wskazują one na słuszność pierwotnie wybranej czułości.

Tabela 7. Zestawienie wartości AP w zależności od ustawionej czułości.

Czułość	AP
0,05	18,07
0,10	17,46
0,15	16,93
0,20	16,31
0,25	15,70

4.6 Optymalizacja parametrów Filtrowania Krawędzi

Decyzja o odrzuceniu danego wycinka przez algorytm FK jest zależna od pięciu parametrów:

1. i 2. Threshold 1 i Threshold 2, czyli progi, według których funkcja Canny określa, czy w danym miejscu już jest krawędź, czy jeszcze nie. Te wartości modyfikowano w trakcie badań.
3. Rozmiar maski (filtru Sobela) w funkcji Canny – są 3 możliwe wartości, tj. 3, 5 i 7, ale dla 5 i 7 na obrazie pojawia się ogrom zbędnych malutkich krawędzi. Zostaje więc ona ustawiona na stałe na 3.
4. L2gradient, czyli parametru, który definiuje, według jakiego wzoru funkcja Canny liczy wynikowy gradient z gradientów pionowego i poziomego. Pozostawiono wartość domyślną.
5. Progu liczby białych pikseli (oznaczających krawędzie), powyżej którego algorytm uznaje wycinek za posiadający krawędzie. Zostaje na stałe wybrana wartość 100.

Zbadanie ostatniego z wymienionych parametrów również mogłoby być wartościowe. Niemniej ze względu na ograniczenia czasowe i czas trwania jednej ewaluacji, wynoszący wiele godzin, nie zostało to wykonane. Badanie różnych wartości oznaczałoby zwielokrotnienie liczby wszystkich prób. W subiektywnej ocenie autora wartość 100 pozwala pominąć pojedyncze „kropki” na wycinkach, jednocześnie nie wpływając na ostateczny wynik ewaluacji.

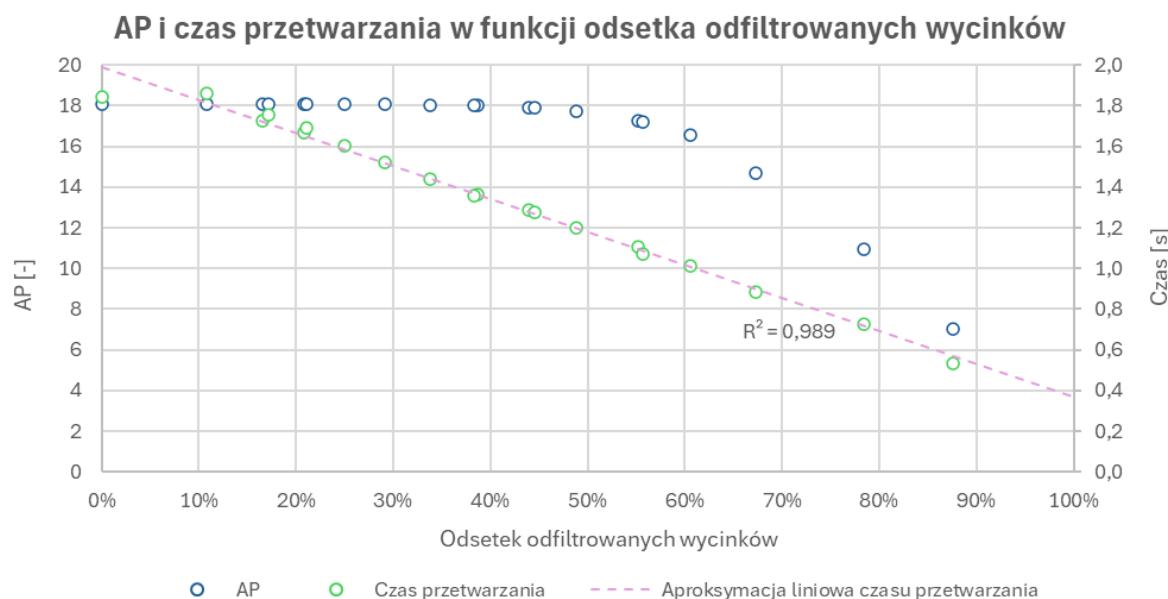
W ramach badania wykonano dziewiętnaście ewaluacji na zbiorze testowym przy podziale 8. Wielkościami modyfikowanymi były Threshold 1 i Threshold 2. Wartości AP, czasu przetwarzania, przyspieszenia i odsetka usuniętych wycinków zawarto w tab. 8. Próba bez wartości progów oznacza algorytm SAHI bez modyfikacji. Zgodnie z oczekiwaniami dla większych progów algorytm usuwa więcej wycinków i czas przetwarzania jest krótszy. Maksymalne uzyskane przyspieszenie to aż 3,47. Niestety wiąże się ono ze znacznym spadkiem AP do wartości 7,048. Co ciekawe, gdy wartości Threshold 1 i Threshold 2 są małe, AP nieznacznie rośnie. Najprawdopodobniej wynika to z odfiltrowania fałszywych pozytywów w pustych przestrzeniach.

Tabela 8. Parametry ewaluacji w zależności od wartości Threshold 1 i Threshold 2. Pierwszy wiersz odnosi się do SAHI bez Filtrowania Krawędzi. Wytluszczoną czcionką oznaczono wartości AP większe niż dla modelu bez FK.

Th. 1/Th. 2	AP	Usun. wyc.	Czas	Przysp.
-	18,067	-	1,84	1,00
50/50	18,069	11%	1,86	-
100/100	18,071	17%	1,73	1,07
150/150	18,076	21%	1,67	1,11
200/200	18,069	25%	1,60	1,15
300/300	18,047	34%	1,44	1,28
350/350	18,023	39%	1,37	1,35
400/400	17,916	44%	1,29	1,43
500/500	17,281	55%	1,11	1,66
600/600	14,667	67%	0,89	2,08
700/700	10,926	78%	0,72	2,55
800/800	7,048	88%	0,53	3,47
50/150	18,072	17%	1,76	1,05
100/200	18,071	21%	1,69	1,09
200/300	18,075	29%	1,52	1,21
300/400	18,031	38%	1,36	1,36
400/500	17,736	49%	1,20	1,54
500/600	16,558	61%	1,01	1,82
300/500	17,899	45%	1,27	1,45
400/600	17,180	56%	1,07	1,72

Kluczowa obserwacja została przedstawiona na rys. 24. Można pominąć znaczną liczbę wycinków i zachować wysoką skuteczność SAHI przy jednoczesnym skróceniu czasu przetwarzania zdjęcia. Wraz ze wzrostem odsetka usuniętych wycinków wartość AP pozostaje praktycznie stała aż do około 50%. Dla większych wartości zaczyna spadać. Z drugiej strony czas analizy maleje w przybliżeniu liniowo (współczynnik R^2 wynosi 0,989). Dowodzi to skuteczności algorytmu Filtrowania Krawędzi.

Dla Filtrowania 50/50 przyspieszenie było równe 0,99. Wynika to z tego, że przy niskich progach algorytm pomija niewiele wycinków – tutaj zaledwie 11%. Czas potrzebny na sprawdzenie, czy wycinki posiadają krawędzie, był dłuższy niż czas zaoszczędzony dzięki pominięciu wybranych fragmentów. Stąd łączny czas analizy się wydłużył, zamiast zmaleć, i uzyskane przyspieszenie jest mniejsze niż 1. Właśnie dlatego dla niewielkiego odsetka odfiltrowanych wycinków czas przetwarzania będzie nieznacznie odbiegał od zależności liniowej (stąd użyte wcześniej określenie „w przybliżeniu liniowo”). Jest to przyczyna stosunkowo dużej różnicy między czasem ewaluacji bez Filtrowania (odsetek usuniętych wycinków równy 0%) a linią trendu widoczną na rys. 24.



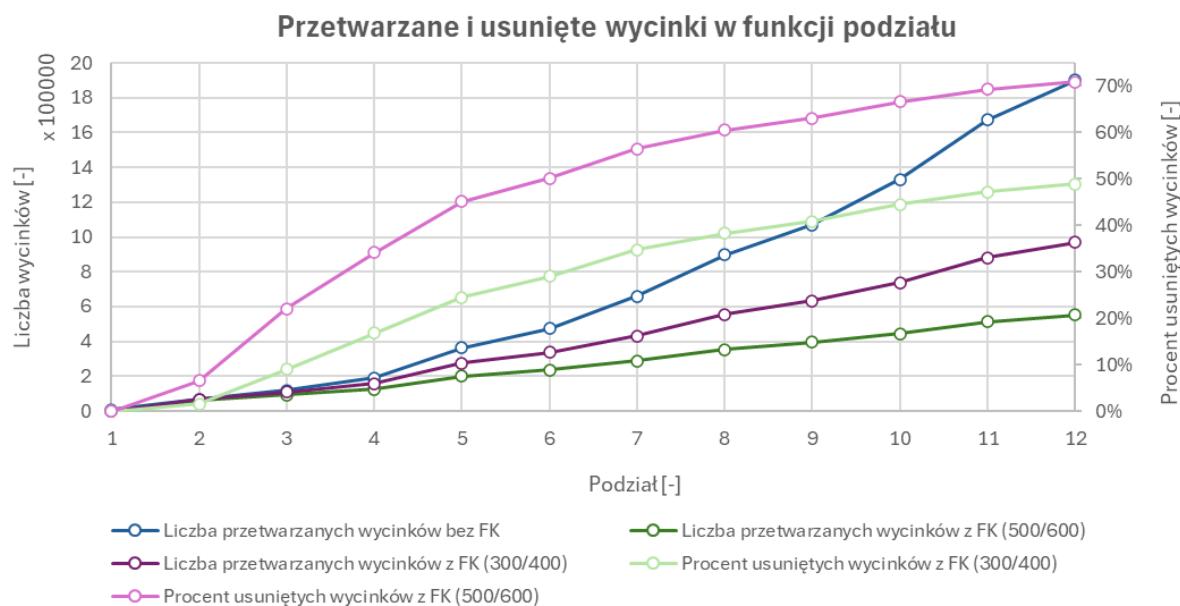
Rysunek 24. AP i czas przetwarzania w funkcji odsetka odfiltrowanych wycinków.

Na podstawie zebranych danych zostały wybrane dwie kombinacje wartości badanych progów do dalszych badań: 300 i 400 oraz 500 i 600 (zapisywane dalej w skrócie jako 300/400 i 500/600). Pierwsza z nich charakteryzuje się niewielkim spadkiem AP i zadowalającym przyspieszeniem równym 1,36. Druga powoduje jużauważalny spadek metryki, ale za to przyspieszenie działania algorytmu jest większe i wynosi 1,82 raza. Ma to na celu sprawdzenie FK przy parametrach nastawionych na jakość i na szybkość.

4.7 Serie ewaluacji z wykorzystaniem Filtrowania Krawędzi

Wykonano dwie kolejne serie ewaluacji przy progach funkcji Canny wynoszących 300/400 i 500/600. Ich wyniki zostały zestawione z serią odniesienia.

Większe wartości progów pozwalają usuwać więcej wycinków. Wraz ze wzrostem podziału rośnie względna liczba usuniętych wycinków, aż do wartości 48,9% dla progów 300/400 i 70,9% dla progów 500/600, co przedstawiono na rys. 25.

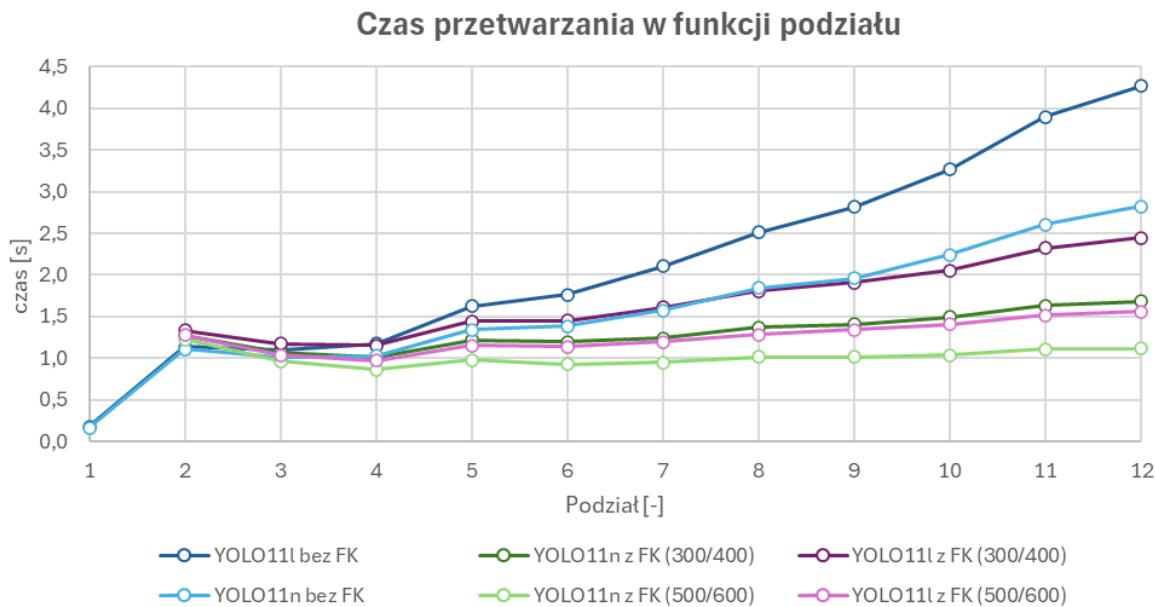


Rysunek 25. Liczba wycinków przetwarzanych i odsetek usuniętych w wyniku działania FK dla różnych przypadków sieci i progów FK.

Wynika to z faktu, że mniejsze wycinki mogą się dokładniej dopasować do kształtu pustych przestrzeni i lepiej je wypełnić. Jest to analogiczne do aproksymacji pola nieregularnej figury poprzez wypełnienie jej siatką kwadratów. Być może wycinki w kształcie kwadratów byłyby jeszcze lepsze. Należy jednak wziąć pod uwagę, że łączna liczba wycinków by wtedy wzrosła.

Tak wysokie wartości usuniętych wycinków są pozytywnie zaskakujące. Oznaczają one, że średnio aż połowa powierzchni całego obrazu może zostać pominięta w trakcie analizy bez utraty jakości. Oczywiście wartość ta jest charakterystyczna dla rodzaju analizowanych zdjęć. Niemniej uzyskane wyniki opisują praktyczne zastosowanie, jakim jest analiza obrazu z kamery samochodu.

Średni czas przetwarzania obrazu, przedstawiony na rys. 26, rośnie wraz z podziałem, co jest logiczne, ponieważ rośnie również liczba analizowanych wycinków. Dla małych wartości podziału (2 i 3) czas przetwarzania może wzrosnąć w wyniku zastosowania FK. Przyspieszenie jest wtedy mniejsze od 1. Dzieje się tak, ponieważ czas potrzebny na sprawdzenie, czy na wycinkach są krawędzie, jest dłuższy niż czas zaoszczędzony dzięki ich pominięciu. Dla większego podziału odsetek pominiętych wycinków rośnie i czas wnioskowania jest skrócony.



Rysunek 26. Czas przetwarzania zdjęcia i przyspieszenie w funkcji podziału dla badanych kombinacji sieci i progów Filtrowania Krawędzi.

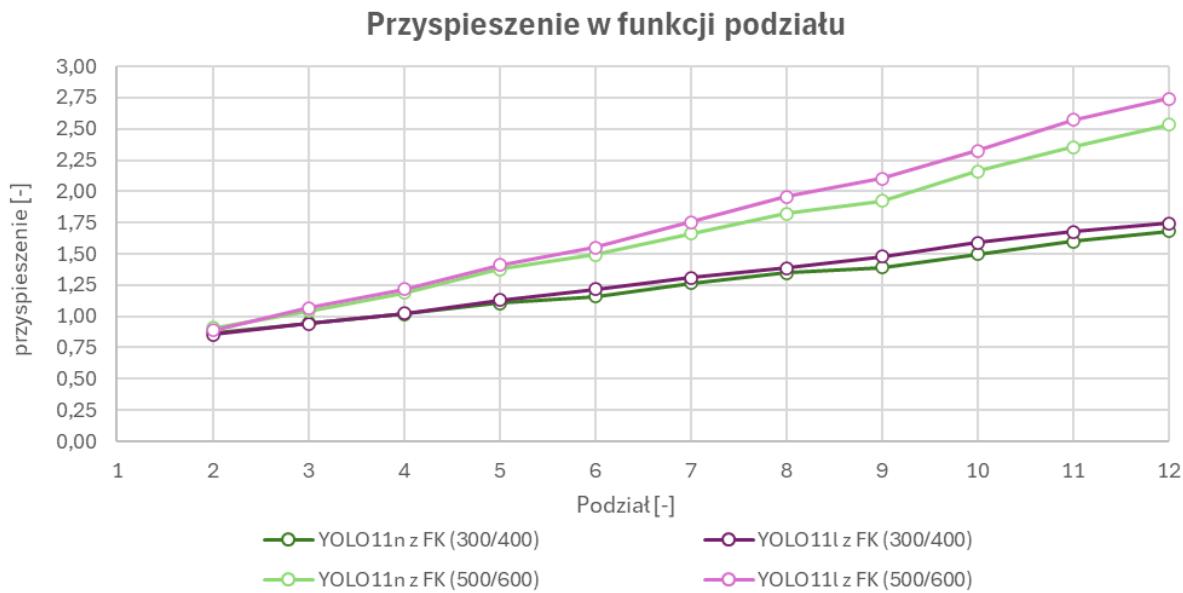
Czas przetwarzania przy podziale 2 był większy niż przy podziale 3. Wykonano dodatkową ewaluację sieci nano, w celu sprawdzenia, czy nie jest to efekt błędu lub innego nieprzewidzianego czynnika, ale uzyskany wynik był identyczny. Przyczyna tego stanu nie jest do końca jasna. Być może wynika to z rozmiarów wycinków.

Ciekawą obserwacją jest to, że czas dla sieci nano z FK 300/400 jest praktycznie równy czasowi dla sieci large z FK 500/600, podczas gdy większa sieć osiąga lepsze wartości AP. Analogicznie czas działania sieci YOLO11n bez żadnych modyfikacji jest zbliżony do czasu sieci YOLO11l z filtrowaniem 300/400, która znacznie ją przewyższa pod względem wspomnianej metryki. Oznacza to, że możliwe jest inne zastosowanie FK niż skracanie czasu detekcji, a mianowicie zwiększenie jakości przy zachowaniu stałego czasu.

Wartości przyspieszenia, widoczne na rys. 27, rosną liniowo wraz z podziałem. Sieć large osiąga nieco większe przyspieszenie przy takich samych parametrach filtrowania, co jest potwierdzeniem postawionej wcześniej hipotezy.

Wykresy AP, z rys. 28, dla SAHI bez modyfikacji niemalże nakładają się na te, gdzie zostało zastosowane FK 300/400. Okazuje się, że dla niskich wartości podziału również Filtrowanie 500/600 osiąga zbliżone wartości. Jest to szczególnie interesujące w połączeniu z faktem, iż sieć YOLO11l uzyskiwała lepsze wyniki właśnie dla mniejszego podziału. Gdy podział rośnie, różnica między wynikami dla różnych parametrów Filtrowania również rośnie.

Stosowanie większych wartości progów filtrowania zdaje się być zasadniejsze dla większych sieci, które nawet pomimo utraty jakości mogą dawać satysfakcyjujące rezultaty.



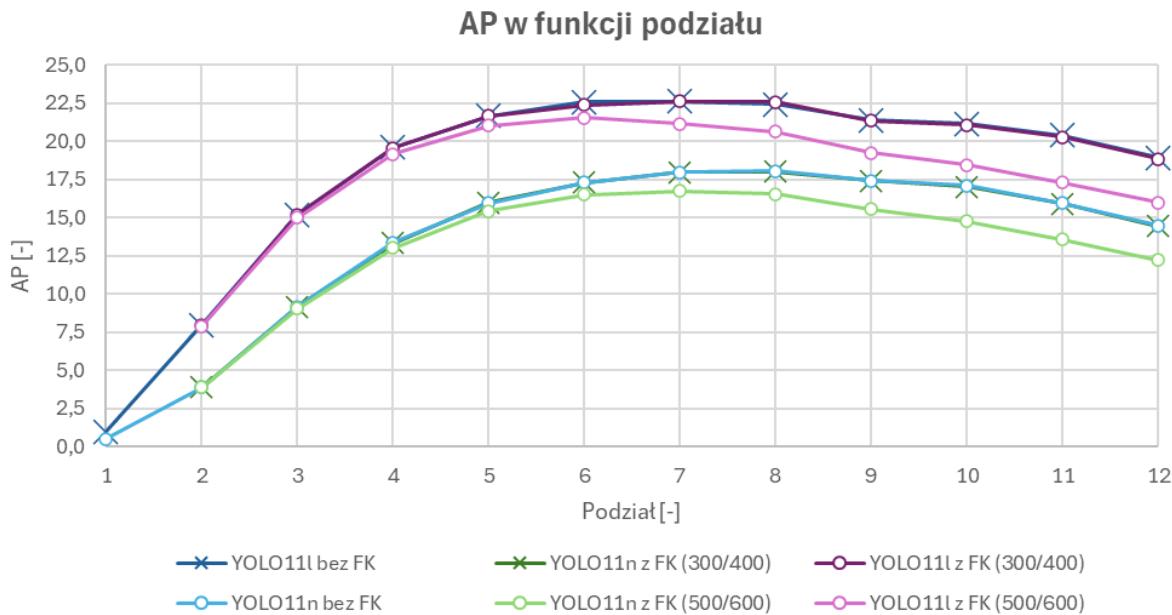
Rysunek 27. Czas przetwarzania zdjęcia i przyspieszenie w funkcji podziału dla badanych kombinacji sieci i progów Filtrowania Krawędzi.

4.8 Ewaluacje przy stałym rozmiarze wycinków

W ramach dalszych badań wykonano serię ewaluacji, w której wymiary wszystkich wycinków były stałe i miały kształt kwadratów (dokładnie: 640×640 , 480×480 i 320×320). Podobne rozwiązań wykorzystali oryginalnie twórcy SAHI [2]. Jest to alternatywne podejście do użytego przy wcześniejszych ewaluacjach, gdzie wycinki miały zmienne wymiary, obliczane przy użyciu podziału, charakterystyczne dla każdego zdjęcia. Przykładowo, założymy, że celem jest przeanalizowanie dwóch obrazów o wymiarach 800×400 i 2000×1000 . Jeżeli użyjemy podziału 4, wycinki analizowane na pierwszym zdjęciu będą miały wymiary 200×100 , a na drugim 500×250 . Jeżeli użyjemy wycinków o stałym rozmiarze 320×320 , to oba zdjęcia będzie analizowane przy użyciu wycinków 320×320 .

Dla każdego z rozmiarów 640×640 , 480×480 i 320×320 wykonano ewaluację sieci YOLO11n i YOLO11l z Filtrowaniem Krawędzi 300/400 i bez niego. Łącznie dało to dwanaście ewaluacji. Wszystkie wyniki znajdują się w tab. 16 w zał. 1. Poniżej, w tab. 9, zestawiono rezultaty sieci large z wynikami uzyskanymi wcześniej przy podziale 7. Uzyskane wskaźniki jakości ograniczono do AP. Podział 7 charakteryzowała się najwyższą wartością tej metryki ze wszystkich badanych podziałów. Porównanie to miało na celu sprawdzenie słuszności wcześniejszych założeń opisanych w pkt 4.1.1.

Okazuje się, że metryki otrzymane dzięki zastosowaniu stałych wymiarów wycinków przewyższają te uzyskane wcześniejszą metodą. Dla wymiarów 640×640 i 480×480 dzieje się to dodatkowo przy krótszym czasie przetwarzania. Uzyskane przyspieszenie jest podobne do wcześniejszych prób, które miały taki odsetek usuniętych wycinków.



Rysunek 28. Wartość AP w funkcji podziału dla różnych wariantów sieci i Filtrowania Krawędzi.

Mimo że wycinki 640×640 są równe wejściu sieci, model w tym przypadku uzyskał gorsze rezultaty niż dla wycinków 480×480 . Dzieje się tak najprawdopodobniej dlatego, że mniejszy wycinek jest w trakcie przetwarzania wstępного skalowany w górę, przez co późniejsza utrata informacji, spowodowana warstwami splotowymi z łączeniem, jest mniejsza, a obiekty mają większą skalę z punktu widzenia wejścia sieci.

Dodatkową zaletą może być fakt, że wycinki w ramach przetwarzania wstępnego nie zmieniają proporcji i są bardziej podobne do kwadratowych fragmentów, na których ćwiczono sieć.

Podsumowując, okazuje się, że wykorzystanie wycinków o stałych wymiarach jest lepszym podejściem niż opisane wcześniej zastosowanie podziału. Liczba sprawdzonych możliwych rozmiarów jest niestety mała. Do lepszego zbadania tej metody konieczne jest wykonanie większej liczby ewaluacji, podobnie jak zostało to wykonane dla różnych wartości podziału.

4.8. Ewaluacje przy stałym rozmiarze wycinków

Tabela 9. Porównanie wyników ewaluacji sieci YOLO11 przy stałych wymiarach wycinków lub podziale 7. „Liczba wyc.” „Pom.” i „Przysp.” oznaczają odpowiednio sumę wszystkich wycinków przeanalizowanych w ramach ewaluacji, odsetek pominiętych wycinków dzięki FK i przyspieszenie.

Rozmiar wycinków	FK	AP	Czas [s]	Liczba wyc.	Pom.	Przysp.
Stał - 640×640	-	23,51	1,43	279 627	-	-
Stał - 640×640	300/400	23,49	1,32	221 652	21%	1,08
Stał - 480×480	-	24,30	1,85	497676	-	-
Stał - 480×480	300/400	24,27	1,53	349 379	30%	1,21
Stał - 320×320	-	23,72	2,50	869 126	-	-
Stał - 320×320	300/400	23,63	1,79	538 600	38%	1,40
Zmienny - podział 7	-	22,64	2,10	660 694	-	-
Zmienny - podział 7	300/400	22,62	1,61	431 280	35%	1,31

Rozdział 5

Podsumowanie i wnioski

W niniejszej pracy inżynierskiej przeanalizowano zagadnienie rozpoznawania małych i licznych obiektów na zdjęciach przez głębokie sieci neuronowe. Opisano jego zastosowania, przyczyny towarzyszących mu problemów oraz techniki stosowane, aby te problemy zniwelować. W szczególności przeanalizowano algorytm SAHI. Jego autorska modyfikacja, nazwana Filtrowaniem Krawędzi, została zaimplementowana w praktyce. W celu jej zbadania, dwie sieci neuronowe CNN, YOLO11n oraz YOLO11l, zostały wytrenowane na zbiorze zdjęć sytuacji drogowych SODA-D, skierowanym na rozpoznawanie małych obiektów.

Przeprowadzone badania wykazały, że Filtrowanie Krawędzi jest w stanie skutecznie przyspieszyć działanie algorytmu SAHI przy niewielkiej utracie jakości. Alternatywnie, FK może zostać użyte, aby poprawić parametry detekcji przy zachowaniu stałego czasu przetwarzania. Działo się tak, gdy odfiltrowanie wycinków umożliwiało użycie większego modelu lub mniejszego rozmiaru wycinków. Jednakże to pierwsze z opisanych zastosowań zdaje się być cenniejsze.

Niestety nawet pomimo skrócenia czasu przetwarzania, analiza zdjęcia trwała średnio blisko sekundę na komputerze o ogromnej sprawności obliczeniowej. Wyklucza to zastosowanie SAHI z Filtrowaniem Krawędzi w systemie prowadzenia autonomicznego samochodu oraz innych zadaniach, gdzie wymagana jest wysoka liczba klatek na sekundę. Niemniej pozostaje ono nadal użyteczne w innych aplikacjach wykorzystujących algorytm SAHI, gdzie może skrócić czasochłonne analizy wielu zdjęć lub obrazów wysokiej rozdzielczości. Finalnie, autor postrzega takie właśnie zastosowanie jako najodpowiedniejsze dla Filtrowania Krawędzi. Należy jednak pamiętać o tym, że przetwarzane zdjęcia powinny zawierać puste przestrzenie. W przeciwnym wypadku czas analizy zostanie wydłużony.

Dla skutecznego działania FK istotne jest dopasowanie jego parametrów. Są one charakterystyczne dla badanego zagadnienia i pozwalają modyfikować to, jak duże będzie finalne przyspieszenie. Jednakże zbyt duże przyspieszenie wiąże się nieuchronnie z pewną utratą jakości w wyniku usunięcia wycinków zawierających obiekty. Dla wartości progów funkcji Canny 300/400 model uzyskał na wybranym zbiorze średnie przyspieszenie równe maksymalnie 1,74 i spadek AP równy jedynie 0,0007. Było to dla wartości podziału 12. Analogicznie, przy ustawieniu progów na 500/600, największe przyspieszenie wyniosło 2,74, a AP zmniejszyło się o 0,0293.

Zaobserwowano, iż w zależności od rozmiaru badanych wycinków zmienia się optymalny podział. Jest on mniejszy dla większego z badanych modeli sieci YOLO11, która poza tym uzyskuje nieznacznie większe przyspieszenie niż jej mniejszy wariant. Ponadto przyspieszenie, spadek AP i odsetek pominiętych wycinków rosną wraz z podziałem. Wykazano również, że pomimo usunięcia znaczej części analizowanych fragmentów obrazu (do ok. 50%) uzyskiwane wartości metryki AP pozostają w przybliżeniu stałe, podczas gdy czas przetwarzania maleje w przybliżeniu liniowo. Wartość, przy której dalsze zwiększenie przyspieszenia powoduje utratę jakości, jest charakterystyczna dla rodzaju analizowanych zdjęć. Dla niewielkiego odsetka pominiętych wycinków metryki nieznacznie wzrosły, najprawdopodobniej dzięki usunięciu wyników fałszywie pozytywnych.

Wykonano dodatkową serię badań, w której rozmiar wycinków pozostawał stały, a nie był skalowany względem wymiarów obrazu. Uzyskane wyniki sugerują, że jest to lepsze rozwiązanie niż zaproponowany w pracy podział. Dla całościowego porównania konieczne są dalsze badania. Niemniej zdaje się to być najbardziej obiecujące podejście.

Kod źródłowy projektu został publicznie udostępniony na platformie GitHub w postaci rozgałęzienia oryginalnego repozytorium SAHI uzupełnionego o kod autora. Jest on dostępny pod adresem https://github.com/Michal-Wojcieszek/SAHI_FK/tree/main.

5.1 Dalsze możliwości rozwoju

Autor widzi liczne perspektywy kontynuacji badań przedstawionych w pracy.

Jako że wycinki analizowane przez sieć w ramach eksperymentu miały zmienne wymiary, a stałe proporcje, wartościowe może się okazać przygotowanie zbioru treningowego w taki właśnie sposób. Różnica skali obiektów widzianych przez sieć będzie wtedy mniejsza. Należałyby wtedy określić z góry przewidywany rozmiar wycinków. Jednakże, zdaniem autora, cenniejsze będzie skupienie się na metodzie wycinków o stałych rozmiarach z powodu wyższości jej rezultatów. Przede wszystkim należy wykonać dla niej ewaluacje z różnymi rozmiarami wycinków i różnymi progami funkcji Canny.

Algorytm SAHI, poza analizą wycinków, wykonuje również wnioskowanie na pełnym obrazie. Możliwe jest wykorzystanie dwóch różnych sieci neuronowych, każdej skierowanej na obiekty w innej skali. Sieć do analizy pełnego obrazu mogłaby być ćwiczona na całych zdjęciach z oznaczonymi większymi obiektami. Sieć do analizy wycinków byłaby wytrenowana na małych fragmentach zdjęć ze zbioru treningowego z oznaczonymi jedynie małymi obiektami. Takie hybrydowe połączenie dwóch sieci w jednym układzie mogłoby umożliwić osiągnięcie lepszej ogólnej jakości.

Ciekawe rezultaty może również przynieść zbadanie wpływu pozostałych zmiennych parametrów Filtrowania Krawędzi, czyli L2gradient i progu białych pikseli. Z prostych analiz autora wynika, że wpływ pierwszego z tych parametrów jest złożony, a uzyskane dzięki niemu krawędzie są „bardziej gładkie”, ale jednocześnie jest ich mniej. Trudno przewidzieć finalny wpływ tej zmiany na działanie FK. Zwiększenie wartości progu spowodowałoby odrzucanie większej liczby wycinków. Być może dobrym rozwiązaniem byłoby ustalenie niskich wartości Threshold 1 i Threshold 2 wraz z wysokim progiem, aby algorytm wyłapywał więcej krawędzi.

Inną zmienną, która nie była modyfikowana w trakcie badań było nakładanie się wycinków. Wybrana wartość 0,25, mimo że typowa, może być nad wyraz duża w kontekście tak małych obiektów jak te w badanym zbiorze. Aby to zweryfikować, konieczne byłoby wykonanie dodatkowej serii ewaluacji. Zmniejszenie nakładania zmniejsza całkowitą liczbę wycinków. Z tego powodu czas wykonywania algorytmu powinien się skrócić. Ponadto, przynajmniej w teorii, również odsetek usuniętych wycinków i przyspieszenie powinny mieć mniejsze wartości.

Jak wspomniano w 2.3.6 do określenia prawdopodobieństwa znajdowania się obiektów w danym miejscu można użyć dodatkowej sieci neuronowej. Taki model mógłby potencjalnie zastąpić funkcję Canny w zadaniu klasyfikacji wycinków SAHI. Mógłby być on wytrenowany na wycinkach zbioru treningowego do określania obecności instancji szukanych klas. Może to rozwiązać problem analizowania wycinków, które mają krawędzie, ale nie posiadają obiektów. Są one przyczyną wykonywania redundantnych inferencji i mogą prowadzić do detekcji fałszywych pozytywów. W tym przypadku, w trakcie przygotowania zbioru treningowego, z oczywistych przyczyn, nie można by usunąć zdjęć niezawierających obiektów.

Potencjalny rozwój byłby również możliwy przez zastosowanie Filtrowania Krawędzi do przyspieszenia algorytmu ASAHI, który już wykazuje się oszczędnością obliczeniową względem klasycznego SAHI. Wyniki uzyskane w niniejszej pracy każdą podejrzewać, że konieczne byłoby zwiększenie liczby wycinków względem tej proponowanej przez autorów, aby FK nie zwiększyło łącznego czasu analizy.

Bibliografia

- [1] Akyon, F. C., Cengiz, C., Altinuc, S. O., Cavusoglu, D., Sahin, K. i Eryuksel, O., *SAHI: A light-weight vision library for performing large scale object detection and instance segmentation*, list. 2021. DOI: [10.5281/zenodo.5718950](https://doi.org/10.5281/zenodo.5718950).
- [2] Akyon, F. C., Onur Altinuc, S. i Temizel, A., „Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection”, w *2022 IEEE International Conference on Image Processing (ICIP)*, dostęp uzyskano: 4.01.2025, IEEE, paź. 2022, s. 966–970. DOI: [10.1109/icip46576.2022.9897990](https://doi.org/10.1109/icip46576.2022.9897990). adr.: <http://dx.doi.org/10.1109/ICIP46576.2022.9897990>.
- [3] Arumugham, K., „A Novel Computer Vision Based Deep Learning Model for Plant Disease and Pest Detection”, grud. 2023, dostęp uzyskano: 28.12.2024.
- [4] Bashir, S. M. A. i Wang, Y., „Small Object Detection in Remote Sensing Images with Residual Feature Aggregation-Based Super-Resolution and Object Detector Network”, *Remote Sensing*, t. 13, nr. 9, 2021, dostęp uzyskano: 28.12.2024, ISSN: 2072-4292. DOI: [10.3390/rs13091854](https://doi.org/10.3390/rs13091854). adr.: <https://www.mdpi.com/2072-4292/13/9/1854>.
- [5] Cheng, G. i in., „Towards Large-Scale Small Object Detection: Survey and Benchmarks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 45, nr. 11, s. 13 467–13 488, 2023.
- [6] Dai, X. i in., *Dynamic Head: Unifying Object Detection Heads with Attentions*, dostęp uzyskano: 28.01.2025, 2021. arXiv: 2106.08322 [cs.CV]. adr.: <https://arxiv.org/abs/2106.08322>.
- [7] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. i Fei-Fei, L., „ImageNet: A large-scale hierarchical image database”, w *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, s. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- [8] Ding, J. i in., „Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, s. 1–1, 2021. DOI: [10.1109/TPAMI.2021.3117983](https://doi.org/10.1109/TPAMI.2021.3117983).
- [9] Dollar, P. i Lin, T.-Y., *pycocotools*, dostęp uzyskano: 2.01.2025, 2014. adr.: <https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools>.
- [10] Duan, C., Wei, Z., Zhang, C., Qu, S. i Wang, H., „Coarse-grained Density Map Guided Object Detection in Aerial Images”, w *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, s. 2789–2798. DOI: [10.1109/ICCVW54120.2021.00313](https://doi.org/10.1109/ICCVW54120.2021.00313).

Bibliografia

- [11] Ferrari, A., Lombardi, S. i Signoroni, A., „Bacterial colony counting by Convolutional Neural Networks”, w *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, s. 7458–7461. DOI: [10.1109/EMBC.2015.7320116](https://doi.org/10.1109/EMBC.2015.7320116).
- [12] Fujii, S., Akita, K. i Ukita, N., „Distant Bird Detection for Safe Drone Flight and Its Dataset”, w *International Conference on Machine Vision Applications (MVA)*, 2021.
- [13] Fukushima, K., „Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements”, *IEEE Transactions on Systems Science and Cybernetics*, t. 5, nr. 4, s. 322–333, 1969. DOI: [10.1109/TSSC.1969.300225](https://doi.org/10.1109/TSSC.1969.300225).
- [14] Gia, B. T. i in., „Enhancing Road Object Detection in Fisheye Cameras: An Effective Framework Integrating SAHI and Hybrid Inference”, w *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, czer. 2024, s. 7227–7235.
- [15] Giannaros, A. i in., „Autonomous Vehicles: Sophisticated Attacks, Safety Issues, Challenges, Open Topics, Blockchain, and Future Directions”, *Journal of Cybersecurity and Privacy*, t. 3, nr. 3, s. 493–543, 2023, dostęp uzyskano: 25.12.2024, ISSN: 2624-800X. DOI: [10.3390/jcp3030025](https://doi.org/10.3390/jcp3030025). adr.: <https://www.mdpi.com/2624-800X/3/3/25>.
- [16] Gupta, A., Anpalagan, A., Guan, L. i Khwaja, A. S., „Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues”, *Array*, t. 10, s. 100057, 2021, dostęp uzyskano: 6.01.2025, ISSN: 2590-0056. DOI: <https://doi.org/10.1016/j.array.2021.100057>. adr.: <https://www.sciencedirect.com/science/article/pii/S2590005621000059>.
- [17] *Hawk vs. Drone! (Hawk Attacks Quadcopter)*, dostęp uzyskano: 4.01.2025. adr.: YouTube : https://www.youtube.com/watch?v=AhDG_WBIQgc.
- [18] Hou, H.-Y. i in., „Ensemble Fusion for Small Object Detection”, w *2023 18th International Conference on Machine Vision and Applications (MVA)*, 2023, s. 1–6. DOI: [10.23919/MVA57639.2023.10215748](https://doi.org/10.23919/MVA57639.2023.10215748).
- [19] Huang, H., Wang, B., Xiao, J. i Zhu, T., „Improved small-object detection using YOLOv8: A comparative study”, *Applied and Computational Engineering*, t. 41, s. 80–88, lut. 2024. DOI: [10.54254/2755-2721/41/20230714](https://doi.org/10.54254/2755-2721/41/20230714).
- [20] International, S., *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, SAE Standard J3016_202104, dostęp uzyskano: 30.12.2024, 2021. adr.: https://www.sae.org/standards/content/j3016_202104/.
- [21] Itseez, *Open Source Computer Vision Library*, <https://github.com/itseez/opencv>, dostęp uzyskano: 2.01.2025, 2015.
- [22] Jegham, N., Koh, C. Y., Abdelatti, M. i Hendawi, A., *Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLOv1 and Its Predecessors*, dostęp uzyskano: 29.12.2024, 2024. arXiv: [2411.00201 \[cs.CV\]](https://arxiv.org/abs/2411.00201). adr.: <https://arxiv.org/abs/2411.00201>.

- [23] Keles, M. C., Salmanoglu, B., Guzel, M. S., Gursoy, B. i Bostanci, G. E., „Evaluation of YOLO models with sliced inference for small object detection”, *arXiv preprint arXiv:2203.04799*, 2022, dostęp uzyskano: 26.12.2024.
- [24] Khan, S. A., Lee, H. J. i Lim, H., „Enhancing Object Detection in Self-Driving Cars Using a Hybrid Approach”, *Electronics*, t. 12, nr. 13, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12132768.
- [25] Khanam, R. i Hussain, M., *YOLOv11: An Overview of the Key Architectural Enhancements*, dostęp uzyskano: 27.12.2024, 2024. arXiv: 2410.17725 [cs.CV]. adr.: <https://arxiv.org/abs/2410.17725>.
- [26] Kim, E. i in., „SHOMY: Detection of Small Hazardous Objects using the You Only Look Once Algorithm.”, *KSII Transactions on Internet & Information Systems*, t. 16, nr. 8, 2022.
- [27] Krizhevsky, A., Sutskever, I. i Hinton, G. E., „ImageNet Classification with Deep Convolutional Neural Networks”, w *Advances in Neural Information Processing Systems*, Pereira, F., Burges, C., Bottou, L. i Weinberger, K., red., dostęp uzyskano: 29.12.2024, t. 25, Curran Associates, Inc., 2012. adr.: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
- [28] Lac, L., *Globox — Object Detection Toolbox*, <https://github.com/laclouis5/globox>, dostęp uzyskano: 23.12.2024, 2022.
- [29] Lam, D. i in., *xView: Objects in Context in Overhead Imagery*, 2018. arXiv: 1802.07856 [cs.CV].
- [30] Lin, J., Lin, H. i Wang, F., „STPM-SAHI: A Small Target Forest Fire Detection Model Based on Swin Transformer and Slicing Aided Hyper Inference”, *Forests*, t. 13, nr. 10, 2022, dostęp uzyskano: 30.12.2024, ISSN: 1999-4907. DOI: 10.3390/f13101603. adr.: <https://www.mdpi.com/1999-4907/13/10/1603>.
- [31] Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B. i Belongie, S. J., „Feature Pyramid Networks for Object Detection”, *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, s. 936–944, 2016, dostęp uzyskano: 4.01.2025. adr.: <https://api.semanticscholar.org/CorpusID:10716717>.
- [32] Lin, T.-Y. i in., *Microsoft COCO: Common Objects in Context*, dostęp uzyskano: 28.12.2024, 2015. arXiv: 1405.0312 [cs.CV]. adr.: <https://arxiv.org/abs/1405.0312>.
- [33] Liu, T., Kawanishi, Y., Komamizu, T. i Ide, I., „Tracking Small Birds by Detection Candidate Region Filtering and Detection History-aware Association”, *arXiv preprint arXiv:2405.17323*, 2024.
- [34] Mahaur, B., Mishra, K. i Kumar, A., „An improved lightweight small object detection framework applied to real-time autonomous driving”, *Expert Systems with Applications*, t. 234, s. 121036, 2023, dostęp uzyskano: 2.01.2025, ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2023.121036>. adr.: <https://www.sciencedirect.com/science/article/pii/S0957417423015385>.

Bibliografia

- [35] Mahaur, B. i Mishra, K., „Small-object detection based on YOLOv5 in autonomous driving systems”, *Pattern Recognition Letters*, t. 168, s. 115–122, 2023.
- [36] Makrai, L. i in., „Annotated dataset for deep-learning-based bacterial colony detection”, *Scientific Data*, t. 10, nr. 497, lip. 2023. DOI: [10.1038/s41597-023-02404-8](https://doi.org/10.1038/s41597-023-02404-8).
- [37] *Mavic 2-DJI*, dostęp uzyskano: 26.12.2024, data dostępu: 6.01.2025.
- [38] Muzammul, M., Algarni, A., Ghadi, Y. Y. i Assam, M., „Enhancing UAV Aerial Image Analysis: Integrating Advanced SAHI Techniques With Real-Time Detection Models on the VisDrone Dataset”, *IEEE Access*, t. 12, s. 21 621–21 633, 2024. DOI: [10.1109/ACCESS.2024.3363413](https://doi.org/10.1109/ACCESS.2024.3363413).
- [39] Nam, G. P., Choi, H., Cho, J. i Kim, I.-J., „PSI-CNN: A Pyramid-Based Scale-Invariant CNN Architecture for Face Recognition Robust to Various Image Resolutions”, *Applied Sciences*, t. 8, nr. 9, 2018, dostęp uzyskano: 2.01.2025, ISSN: 2076-3417. DOI: [10.3390/app8091561](https://doi.org/10.3390/app8091561). adr.: <https://www.mdpi.com/2076-3417/8/9/1561>.
- [40] *Phantom 3 get kidnapped by two eagles*, dostęp uzyskano: 4.01.2025. adr.: YouTube <https://www.youtube.com/watch?v=FX3u0QiZs0A>.
- [41] Rabbi, J., Ray, N., Schubert, M., Chowdhury, S. i Chao, D., *Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network*, dostęp uzyskano: 2.01.2025, 2020. arXiv: 2003.09085 [cs.CV]. adr.: <https://arxiv.org/abs/2003.09085>.
- [42] Rzeczpospolita Polska, *Ustawa z dnia 20 czerwca 1997 r. - Prawo o ruchu drogowym. z późn. zm.* Dz.U. 2024 poz. 1251, dostęp uzyskano: 26.12.2025, 2024. adr.: <https://isap.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU19970980602>.
- [43] Sethu Selvi, S., Pavithraa, S., Dharini, R. i Chaitra, E., „A Deep Learning Approach to Classify Drones and Birds”, w *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*, 2022, s. 1–5. DOI: [10.1109/MysuruCon55714.2022.9972589](https://doi.org/10.1109/MysuruCon55714.2022.9972589).
- [44] Sever, T. i Contissa, G., „Automated driving regulations – where are we now?”, *Transportation Research Interdisciplinary Perspectives*, t. 24, s. 101 033, 2024, dostęp uzyskano: 28.12.2024, ISSN: 2590-1982. DOI: <https://doi.org/10.1016/j.trip.2024.101033>. adr.: <https://www.sciencedirect.com/science/article/pii/S2590198224000198>.
- [45] Seymour, P., „The Summer Vision Project”, lip. 1966.
- [46] Shandilya, S. K., Srivastav, A., Yemets, K., Datta, A. i Nagar, A. K., „YOLO-based segmented dataset for drone vs. bird detection for deep and machine learning algorithms”, *Data in Brief*, t. 50, s. 109 355, 2023, dostęp uzyskano: 4.01.2025, ISSN: 2352-3409. DOI: <https://doi.org/10.1016/j.dib.2023.109355>. adr.: <https://www.sciencedirect.com/science/article/pii/S2352340923004742>.
- [47] Sun, Y., Song, J., Li, Y., Li, Y., Li, S. i Duan, Z., „IVP-YOLOv5: an intelligent vehicle-pedestrian detection method based on YOLOv5s”, *Connection Science*, t. 35, nr. 2168254, 2023. DOI: [10.1080/09540091.2023.2168254](https://doi.org/10.1080/09540091.2023.2168254).

- [48] *Territorial Bird Attacks Flying Drone*, dostęp uzyskano: 4.01.2025. adr.: YouTube: %20https://www.youtube.com/watch?v=1NY3Df_Wuc4.
- [49] Threewitt, C., „Vehicles That Are Almost Self-Driving”, *U.S. News & World Report*, grud. 2024, dostęp uzyskano: 3.01.2025. adr.: https://cars.usnews.com/cars-trucks/advice/cars-that-are-almost-self-driving.
- [50] Wang, H., Liu, C., Cai, Y., Chen, L. i Li, Y., „YOLOv8-QSD: An improved small object detection algorithm for autonomous vehicles based on YOLOv8”, *IEEE Transactions on Instrumentation and Measurement*, 2024.
- [51] Wang, J., Wang, C., Ding, W. i Li, C., „YOLOv5s-ACE: forest fire object detection algorithm based on improved YOLOv5s”, *Fire Technology*, s. 1–21, 2024.
- [52] Wang, Q.-J. i in., „Pest24: A large-scale very small object data set of agricultural pests for multi-target detection”, *Computers and Electronics in Agriculture*, t. 175, s. 105585, 2020, dostęp uzyskano: 26.12.2024, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105585. adr.: https://www.sciencedirect.com/science/article/pii/S0168169919324123.
- [53] Wang, J., Zhang, L., Huang, Y., Zhao, J. i Bella, F., „Safety of Autonomous Vehicles”, *Journal of Advanced Transportation*, t. 2020, s. 1–13, paź. 2020. DOI: 10.1155/2020/8867757.
- [54] Watson, A., *Deep Learning Techniques for Super-Resolution in Video Games*, dostęp uzyskano: 27.12.2024, 2020. arXiv: 2012.09810 [cs.NE]. adr.: https://arxiv.org/abs/2012.09810.
- [55] Yasin, M., *A Simple Trick To Increase YOLOv8's Accuracy On Small Objects With No Overhead*, dostęp uzyskano: 29.12.2024, 2024. adr.: https://y-t-g.github.io/tutorials/yolov8-increase-accuracy/, %20dost%C4%99p:%204.01.2025.
- [56] *YOLO11*, https://docs.ultralytics.com/models/yolo11, Ultralytics, dostęp uzyskano: 6.01.2025.
- [57] Yu, W., Sun, X., Yang, K., Rui, Y. i Yao, H., „Hierarchical semantic image matching using CNN feature pyramid”, *Computer Vision and Image Understanding*, t. 169, s. 40–51, 2018, dostęp uzyskano: 5.01.2025, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2018.01.001. adr.: https://www.sciencedirect.com/science/article/pii/S1077314218300018.
- [58] Yu, X., Gong, Y., Jiang, N., Ye, Q. i Han, Z., *Scale Match for Tiny Person Detection*, dostęp uzyskano: 29.12.2024, 2019. arXiv: 1912.10664 [cs.CV]. adr.: https://arxiv.org/abs/1912.10664.
- [59] Zhang, B., Zhou, Z., Cao, W., Qi, X., Xu, C. i Wen, W., „A New Few-Shot Learning Method of Bacterial Colony Counting Based on the Edge Computing Device”, *Biology*, t. 11, nr. 2, 2022, dostęp uzyskano: 3.01.2025, ISSN: 2079-7737. DOI: 10.3390/biology11020156. adr.: https://www.mdpi.com/2079-7737/11/2/156.

Bibliografia

- [60] Zhang, H., Hao, C., Song, W., Jiang, B. i Li, B., „Adaptive Slicing-Aided Hyper Inference for Small Object Detection in High-Resolution Remote Sensing Images”, *Remote Sensing*, t. 15, nr. 5, 2023, ISSN: 2072-4292. DOI: [10.3390/rs15051249](https://doi.org/10.3390/rs15051249).
- [61] Zhao, W. i in., „A High-Performance Accelerator for Super-Resolution Processing on Embedded GPU”, w *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, dostęp uzyskano: 30.12.2024, IEEE, list. 2021, s. 1–9. DOI: [10.1109/iccad51958.2021.9643472](https://doi.org/10.1109/iccad51958.2021.9643472). adr.: <http://dx.doi.org/10.1109/ICCAD51958.2021.9643472>.
- [62] Zhu, P. i in., „Detection and tracking meet drones challenge”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, t. 44, nr. 11, s. 7380–7399, 2021.

Wykaz skrótów i symboli

CBB	liczenie kolonii bakterii (ang. <i>Bacterial Colony Counting</i>)
CNN	splotowa sieć neuronowa (ang. <i>Convolutional Neural Network</i>)
CPU	procesor (ang. <i>Central Processing Unit</i>)
FI	wnioskowanie na pełnym obrazie (ang. <i>Full Inference</i>)
FK	Filtrowanie Krawędzi
FLOPS	liczba operacji zmiennoprzecinkowych na sekundę
GPU	procesor graficzny (ang. <i>Graphical Processing Unit</i>)
IDE	zintegrowane środowisko programistyczne (ang. <i>Integrated Development Environment</i>)
mAP	średnia precyzja sieci (ang. <i>mean Average Precision</i>)
RAM	pamięć o swobodnym dostępie (ang. Random-Access Memory)
SAHI	Slicing Aided Hyper Inference
SOD	rozpoznawanie małych obiektów (ang. <i>Small Object Detection</i>)
SSH	protokołów komunikacyjny (ang. <i>Secure Shell</i>)
VPN	wirtualna sieć prywatna (ang. <i>Virtual Private Network</i>)

Spis rysunków

1	Zdjęcie zawierające małe i liczne obiekty. Źródło: własne.	2
2	Zdjęcie sytuacji drogowej wraz ze zbliżeniem na oddalone znaki. Opracowano na podstawie: [5].	4
3	Zdjęcie zawierające drapieżnego ptaka na słupie energetycznym. Opracowano na podstawie: [12].	5
4	Zdjęcie kolonii bakterii <i>E. coli</i> na szalce Petriego, które rozwinęły się w 48 godzin. Źródło: [59].	6
5	Zdjęcie satelitarne zawierające wiele małych, licznych i różnie zorientowanych łodzi. Źródło: [8].	7
6	Zdjęcie przedstawiające scenę typową dla miejskiego monitoringu. Źródło: [62].	8
7	Przykładowe szkodliwe efekty działania chorób roślin. Źródło: [3].	9
8	Lewa strona: oryginalne zdjęcie, prawy górny róg i prawy dolny róg: efekty detekcji sieci YOLOv8 przy rozdzielczości wejściowej równej odpowiednio 640×640 i 2560×2560 . Źródło: własne.	10
9	Zdjęcie, na którym ze względu na niską rozdzielczość, obiekty są trudne do klasyfikacji. Opracowano na podstawie: [58].	11
10	Zdjęcie zawierające pojazdy w różnych skalach. Samochód ciężarowy po lewej jest znacznie większy niż położony dalej biały van, który z kolei jest większy niż inne znajdujące się w oddali samochody. Źródło: [5].	12
11	Zdjęcie wykonane z drona w niekorzystnych warunkach oraz zbliżenie na jego fragment zawierający pieszego o wymiarach 10×20 pikseli, który prawdopodobnie prowadzi rower. Zauważalny jest znaczny szum. Opracowano na podstawie: [62].	13
12	Zdjęcie oraz zbliżenie na region o znacznym zagęszczaniu małych obiektów, które nakładają się na siebie. Opracowano na podstawie: [5].	13
13	Zmiana wartości IoU w wyniku przesunięcia ramki dla obiektów różnych rozmiarów. Źródło: [5].	14
14	Schemat działania algorytmu SAHI. Obiekty rozpoznane w analizie pełnego obrazu są łączone z tymi rozpoznanymi na wycinkach, aby uzyskać efekt końcowy. Wycinki $P_1^I \dots P_n^I$ częściowo się na siebie nakładają. Opracowano na podstawie: [2]	16

15	Schemat architektury sieci YOLO11. Widoczne jest, jak część śródkowa (<i>neck</i>) pobiera informacje z różnych warstw ekstraktora cech (<i>backbone</i>), które po przetworzeniu są interpretowane przez trzy głowice detekcyjne w części <i>head</i> . Źródło: [22].	18
16	Klasy zbioru SODA-D. Źródło: [5]	26
17	Przykładowe zdjęcia ze zbioru SODA-D, obrazujące jego różnorodność. Opracowano na podstawie: [5].	26
18	Przykładowe zdjęcie oraz powstały z niego wycinek, zawierający znak, światła drogowe i osobę. Opracowano na podstawie: [5].	28
19	Wartości błędów na zbiorze treningowym i walidacyjnym oraz metryki mAP50 i mAP50-95 w trakcie trwania treningu sieci large. Box_loss, cls_loss oraz dfl_loss to odpowiednio funkcje celu odpowiadające za poprawność lokalizacji, klasy oraz rozkładu położenia ramek.	30
20	Wartości funkcji celu dla zbioru treningowego i walidacyjnego oraz metryki obliczone na zbiorze walidacyjnym w trakcie uczenia sieci YOLO11n.	30
21	Przykładowe wyniki detekcji na dwóch zdjęciach ze zbioru SODA-D przy użyciu SAHI i wytrenowanej sieci YOLO11n. Małe obiekty są skutecznie rozpoznawane z dużą dokładnością.	31
22	Przykład podziału równego 3 przy zerowym nakładaniu. Każdy wycinek ma szerokość i wysokość równą odpowiednio $\frac{1}{3}$ szerokości i $\frac{1}{3}$ wysokości pełnego obrazu. Źródło: własne.	34
23	Metryki uzyskane przez sieci YOLO11n i YOLO11l w funkcji podziału.	37
24	AP i czas przetwarzania w funkcji odsetka odfiltrowanych wycinków.	40
25	Liczba wycinków przetwarzanych i odsetek usuniętych w wyniku działania FK dla różnych przypadków sieci i progów FK.	41
26	Czas przetwarzania zdjęcia i przyspieszenie w funkcji podziału dla badanych kombinacji sieci i progów Filtrowania Krawędzi.	42
27	Czas przetwarzania zdjęcia i przyspieszenie w funkcji podziału dla badanych kombinacji sieci i progów Filtrowania Krawędzi.	43
28	Wartość AP w funkcji podziału dla różnych wariantów sieci i Filtrowania Krawędzi.	44

Spis tabel

1	Specyfikacja sprzętowa jednostki obliczeniowej.	22
2	Specyfikacja programowa	23
3	Specyfikacja bibliotek	23
4	Parametry treningu sieci neuronowych.	29
5	Metryki uzyskane przez sieci YOLO11n i YOLO11l. Dopisek „bez mozaiki” oznacza, że w treningu nie wykorzystano augmentacji danych poprzez mozaikę.	29
6	Przedstawienie rozmiarów, według których jest oceniana skuteczność sieci. Powierzchnia odnosi się do liczby pikseli.	35
7	Zestawienie wartości AP w zależności od ustawionej czułości.	38
8	Parametry ewaluacji w zależności od wartości Threshold 1 i Threshold 2. Pierwszy wiersz odnosi się do SAHI bez Filtrowania Krawędzi. Wytluszczoną czcionką oznaczono wartości AP większe niż dla modelu bez FK.	39
9	Porównanie wyników ewaluacji sieci YOLO11l przy stałych wymiarach wycinków lub podziale 7. „Liczba wyc.” „Pom.” i „Przyp.” oznaczają odpowiednio sumę wszystkich wycinków przeanalizowanych w ramach ewaluacji, odsetek pominiętych wycinków dzięki FK i przyspieszenie.	45
10	Wyniki ewaluacji sieci YOLO11n bez Filtrowania Krawędzi.	65
11	Wyniki ewaluacji sieci YOLO11l bez Filtrowania Krawędzi.	65
12	Wyniki ewaluacji sieci YOLO11n z Filtrowaniem Krawędzi 300/400.	66
13	Wyniki ewaluacji sieci YOLO11l z Filtrowaniem Krawędzi 300/400.	66
14	Wyniki ewaluacji sieci YOLO11n z Filtrowaniem Krawędzi 500/600.	67
15	Wyniki ewaluacji sieci YOLO11l z Filtrowaniem Krawędzi 500/600.	67
16	Wyniki ewaluacji przy stałym rozmiarze wycinka.	68

Spis załączników

1 Wyniki ewaluacji..... 65

Załącznik 1

Wyniki ewaluacji

„Liczba wyc”, „Pom.”, „Przypsp.” i „Wym. wyc.” oznaczają odpowiednio sumę wszystkich wycinków przeanalizowanych w ramach ewaluacji, odsetek pominiętych wycinków względem algorytmu bez FK, przyspieszenie i wymiary wycinka.

Tabela 10. Wyniki ewaluacji sieci YOLO11n bez Filtrowania Krawędzi.

Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Liczba wyc.
1	0,49	1,23	0,05	0,03	0,19	0,64	2,01	0,17	7 428
2	3,89	10,37	1,08	0,53	1,50	6,35	12,94	1,11	66 852
3	9,15	21,73	3,94	1,31	5,33	13,72	19,99	1,01	118 848
4	13,36	30,02	7,52	2,92	9,58	18,41	22,79	1,03	191 232
5	15,97	34,85	10,29	4,84	12,99	20,84	23,67	1,34	363 972
6	17,33	37,37	11,42	6,06	14,71	21,93	23,50	1,39	475 392
7	18,00	38,54	11,60	6,88	16,04	22,05	21,96	1,57	660 694
8	18,07	38,27	11,10	7,39	16,24	21,88	20,03	1,84	898 788
9	17,44	36,88	9,10	7,53	16,28	20,70	15,78	1,96	1 069 632
10	17,10	36,47	8,31	7,77	16,57	19,79	13,75	2,24	1 332 350
11	15,96	34,51	6,82	7,78	16,13	17,98	10,32	2,61	1 671 300
12	14,50	31,83	5,43	7,24	15,46	15,99	7,40	2,83	1 901 568

Tabela 11. Wyniki ewaluacji sieci YOLO11l bez Filtrowania Krawędzi.

Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Liczba wyc.
1	0,97	2,54	0,09	0,15	0,40	1,50	5,08	0,18	7 428
2	7,96	19,94	2,59	1,01	4,14	12,49	20,41	1,14	66 852
3	15,20	34,42	6,78	3,00	10,90	21,29	26,71	1,11	118 848
4	19,61	42,28	10,89	6,19	16,28	25,28	29,48	1,18	191 232
5	21,69	45,60	13,36	8,26	19,02	26,82	29,43	1,63	363 972
6	22,60	46,97	14,27	10,14	20,26	27,27	29,11	1,77	475 392
7	22,64	46,82	14,17	10,28	20,68	26,99	27,85	2,10	660 694
8	22,42	46,26	13,58	10,61	20,48	26,59	26,18	2,51	898 788
9	21,38	44,52	11,93	10,41	19,66	25,32	23,01	2,82	1 069 632
10	21,15	43,89	10,48	10,48	20,12	24,51	20,54	3,27	1 332 350
11	20,39	42,48	8,95	10,26	19,70	23,50	16,74	3,90	1 671 300
12	18,93	40,25	7,41	9,74	18,95	21,48	13,05	4,27	1 901 568

Tabela 12. Wyniki ewaluacji sieci YOLO11n z Filtrowaniem Krawędzi 300/400.

Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Pom.	Przysp.
2	3,89	10,37	1,08	0,53	1,50	6,35	12,94	1,28	2%	0,87
3	9,16	21,74	3,95	1,31	5,33	13,72	19,96	1,07	9%	0,94
4	13,33	30,00	7,53	2,92	9,59	18,35	22,71	1,01	17%	1,02
5	15,98	34,86	10,27	4,85	13,02	20,82	23,53	1,22	24%	1,11
6	17,31	37,34	11,36	6,07	14,74	21,87	23,36	1,19	29%	1,16
7	17,97	38,51	11,55	6,91	16,10	21,96	21,81	1,24	35%	1,27
8	18,03	38,23	11,00	7,42	16,32	21,74	19,85	1,37	38%	1,35
9	17,43	36,79	9,05	7,62	16,43	20,55	15,57	1,40	41%	1,39
10	17,04	36,34	8,23	7,83	16,60	19,65	13,47	1,50	45%	1,50
11	15,90	34,42	6,77	7,86	16,16	17,81	10,16	1,63	47%	1,60
12	14,47	31,81	5,40	7,30	15,49	15,83	7,30	1,68	49%	1,68

Tabela 13. Wyniki ewaluacji sieci YOLO11l z Filtrowaniem Krawędzi 300/400.

Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Pom.	Przysp.
2	7,95	19,89	2,59	1,01	4,14	12,48	20,42	1,34	2%	0,85
3	15,19	34,43	6,79	3,00	10,90	21,28	26,71	1,17	9%	0,94
4	19,57	42,22	10,86	6,19	16,30	25,20	29,45	1,15	17%	1,02
5	21,69	45,61	13,33	8,26	19,08	26,79	29,29	1,44	24%	1,13
6	22,37	46,24	13,47	10,62	20,56	26,44	25,93	1,45	29%	1,22
7	22,62	46,83	14,10	10,38	20,77	26,87	27,69	1,61	35%	1,31
8	22,59	46,93	14,24	10,17	20,30	27,20	28,97	1,81	38%	1,39
9	21,36	44,44	11,85	10,48	19,90	25,08	22,78	1,91	41%	1,48
10	21,08	43,77	10,38	10,55	20,15	24,31	20,34	2,06	45%	1,59
11	20,28	42,20	8,85	10,32	19,76	23,25	16,60	2,32	47%	1,68
12	18,86	40,07	7,37	9,87	18,99	21,27	12,94	2,45	49%	1,74

Tabela 14. Wyniki ewaluacji sieci YOLO11n z Filtrowaniem Krawędzi 500/600.

Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Pom.	Przysp.
2	3,87	20,35	1,05	0,53	1,50	6,31	12,71	1,23	7%	0,91
3	9,04	29,68	3,88	1,32	5,29	13,51	19,43	0,97	22%	1,05
4	13,03	33,49	7,30	2,90	9,43	17,90	21,68	0,87	34%	1,19
5	15,42	34,67	9,76	4,78	12,65	19,94	22,20	0,98	45%	1,38
6	16,53	33,51	10,60	5,97	14,35	20,61	21,43	0,93	50%	1,49
7	16,76	31,15	10,46	6,57	15,35	20,18	19,51	0,95	57%	1,66
8	16,56	35,05	9,65	7,11	15,39	19,62	17,31	1,01	61%	1,82
9	15,54	20,78	7,76	7,09	15,11	17,92	13,40	1,02	63%	1,93
10	14,78	17,44	6,85	7,28	14,68	16,76	11,45	1,04	67%	2,16
11	13,59	12,44	5,46	6,99	14,15	14,92	8,68	1,11	69%	2,35
12	12,24	8,71	4,38	6,51	13,40	13,05	6,17	1,11	71%	2,53

Tabela 15. Wyniki ewaluacji sieci YOLO11l z Filtrowaniem Krawędzi 500/600.

Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Pom.	Przysp.
2	7,91	30,37	2,58	1,01	4,12	12,41	20,18	1,28	7%	0,89
3	14,99	38,04	6,69	3,01	10,81	20,96	26,04	1,03	22%	1,07
4	19,17	40,98	10,56	6,14	16,08	24,60	28,29	0,97	34%	1,22
5	21,04	41,43	12,81	8,21	18,72	25,77	27,91	1,15	45%	1,41
6	21,56	40,14	13,27	9,99	19,82	25,58	26,86	1,14	50%	1,55
7	21,16	38,06	12,71	9,84	19,79	24,78	25,13	1,20	57%	1,75
8	20,63	34,87	11,92	10,22	19,40	23,92	23,03	1,28	61%	1,96
9	19,25	30,61	10,21	9,75	18,52	22,09	19,99	1,34	63%	2,10
10	18,46	26,03	8,66	9,67	18,05	20,93	17,61	1,41	67%	2,32
11	17,30	20,84	7,18	9,21	17,32	19,39	14,33	1,52	69%	2,57
12	16,00	16,06	5,93	8,93	16,45	17,63	11,15	1,56	71%	2,74

Tabela 16. Wyniki ewaluacji przy stałym rozmiarze wycinka.

Wym. wyc.	Sieć	Podział	AP	AP50	AP75*	APeS	APrS	APgS	APN	Czas [s]	Liczba wyc.	Pom.	Przysp.
640x640	nano bez FK	8	17,42	37,65	11,64	5,16	14,49	22,42	25,13	1,19	279627	-	-
	nano z FK	8	17,40	37,63	11,58	5,15	14,50	22,36	24,98	1,12	279 627	21%	1,05
	large bez FK	7	23,51	48,49	15,04	9,79	21,38	28,48	30,88	1,43	279 627	-	-
	large z FK	7	23,49	48,41	14,97	9,79	21,37	28,41	30,71	1,32	279 627	21%	1,08
480x480	nano bez FK	8	19,22	40,61	13,29	7,52	16,85	23,67	24,20	1,45	497676	-	-
	nano z FK	8	19,19	40,55	13,23	7,53	16,88	23,59	24,04	1,22	497 676	30%	1,20
	large bez FK	7	24,30	49,54	16,00	11,53	22,72	28,67	29,59	1,85	497 676	-	-
	large z FK	7	24,27	49,50	15,93	11,53	22,71	28,57	29,45	1,53	497 676	30%	1,21
320x320	nano bez FK	8	19,09	39,82	10,13	8,59	17,66	22,63	17,58	1,82	869126	-	-
	nano z FK	8	19,04	39,66	10,03	8,62	17,66	22,50	17,37	1,35	869 126	38%	1,34
	large bez FK	7	23,72	47,92	12,49	12,36	22,25	27,56	22,98	2,50	869 126	-	-
	large z FK	7	23,63	47,72	12,43	12,38	22,22	27,41	22,81	1,79	869 126	38%	1,40