

A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Praca dyplomowa

System wizyjny do detekcji ogórków gruntowych w oparciu o głęboką sieć YOLO

Vision system for cucumber recognition based on deep YOLO network

Autor:

Jan Tyc

Kierunek studiów:

Inżynieria Biomedyczna

Opiekun pracy:

dr hab. inż. Ziemowit Dworakowski

Kraków, 2024

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpośczechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

*Serdecznie dziękuję mojej rodzinie, dziewczynie
oraz promotorowi.*

Spis treści

1. Wprowadzenie	7
1.1. Jak robotyzacja zmieniła rolnictwo?	7
1.2. Cel i zakres pracy	9
1.3. Organizacja pracy	10
2. Przegląd literatury	11
2.1. Podstawy uczenia głębokiego i cyfrowego przetwarzania obrazów w kontekście interpretacji danych wizyjnych	11
2.1.1. SGD - sposób działania.....	11
2.1.2. “Label smoothing”	12
2.1.3. ResNet.....	12
2.1.4. Transfer learning	13
2.2. Detekcja obiektów - algorytm YOLO	13
2.2.1. Architektura “Głowy”	14
2.2.2. YOLOv7.....	16
2.3. Podobne rozwiązania	17
2.3.1. Detekcja obiektów - YOLO	17
2.3.2. Cyfrowe przetwarzanie obrazu	19
3. Metody i opis danych	21
3.1. Dane.....	21
3.2. Algorytm YOLO.....	21
3.2.1. Wstępne przetwarzanie danych algorytmu YOLOv7	23
3.2.2. Uczenie algorytmu YOLO v7	23
3.3. Sieć przedtreningowa	24
3.4. Self-paced learning.....	24
3.5. Wykrywanie koloru kwiatów.....	26
3.6. Metryki dokładności	27
4. Wyniki badań	29

4.1.	Sieć przedtreningowa	29
4.2.	Optymalizacja "label smoothing" w YOLO	31
4.3.	Self-paced learning	32
4.4.	Wykrywanie koloru kwiatów.....	32
4.5.	Omówienie wyników.....	32
5.	Podsumowanie.....	35

1. Wprowadzenie

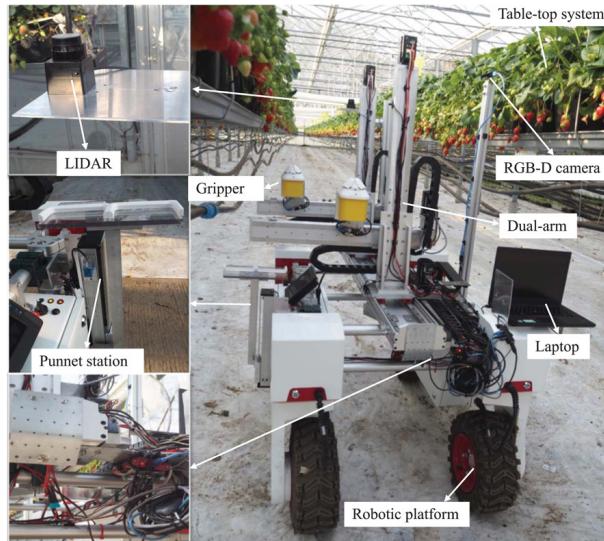
1.1. Jak robotyzacja zmieniła rolnictwo?

Rolnictwo tak samo jak pozostałe gałęzie gospodarki rozwija się gwałtownie pod wpływem nowych technologii. Dzięki osiągnięciom ostatnich lat zautomatyzowane zostały takie obszary jak nawożenie, nawadnianie, a także zbiory niektórych rodzajów owoców i warzyw takich jak kukurydza, pomidory czy różne rodzaje zbóż. Ponadto, w rolnictwie coraz częściej wprowadzane są rozwiązania, które wymagają zaawansowanych technologii oraz metod przetwarzania danych [1]. Ograniczenie elementu ludzkiego w tych procesach doprowadziło do znacznej redukcji kosztów produkcji, co w konsekwencji redukuje cenę rynkową plonów. Każda nowa technologia wprowadzana na rynek rolniczy jest dopasowywana do danego owocu lub warzywa, które tworzy plantację. Szczególnie widać to na przykładzie automatycznego zbioru plonów gdzie np. kombajn do zbioru pomidorów, wymaga specjalnego układu pola i nadaje się tylko do zbioru pomidorów. Zawsze podczas zbiorów, część plonów ulega zniszczeniu lub zostaje pominięta [2]. Wydajność zbioru, czyli jaką część plonów udało się zebrać z pola, jest jedną z kluczowych składowych, które wpływają na ilość produkowanej żywności. Automatyczny zbiór owoców i warzyw nie jest jednak obecnie możliwy w przypadku wszystkich rodzajów plonów [3]. Żeby dany rodzaj owocu lub warzywa został zebrane przez kombajn, lub inną ciężką maszynę rolniczą, musi on:

1. Nie ulegać zniszczeniu podczas wielokrotnego obijania się o siebie
2. Owocować jednokrotnie podczas sezonu
3. Być uprawiany na płaskim terenie

Gdyby chociaż jeden z tych warunków nie był spełniony to automatyzacja takich zbiorów nie byłaby możliwa, ponieważ:

- Zbiory uległyby zniszczeniu na skutek obijania się o siebie (maliny, truskawki)
- Wjazd maszyny na pole jest równoznaczny ze zniszczeniem uprawy, brakiem kolejnego owocowania (ogórki, maliny)
- Nie istnieje możliwość wjazdu ciężkiego sprzętu w dane miejsce (winogrona) [4]



Rys. 1.1. Maszyna do zbioru truskawek na plantacji [8]

Z wyżej wymienionych powodów nie istnieje możliwość automatycznego zbioru niektórych plonów np. malin, truskawek czy ogórków. Zbiór jest często jednym z największych kosztów produkcji.

Alternatywą pozostaje stworzenie maszyny, która w sposób precyzyjny, przy użyciu różnego rodzaju manipulatorów naśladujących ludzkie ręce, będzie w stanie dokonać zbioru, bez niszczenia plonu (na przykład poprzez zbyt mocne ściśnięcie). Dodatkowo maszyna do zbioru nie może niszczyć plantacji i musi charakteryzować się określona wydajnością, żeby miała szansę zostać wprowadzona do zastosowania przemysłowego.

Takie maszyny muszą być zdolne do pracowania w złożonym środowisku co sprawia, że ich dokładność, szybkość i stabilność pracy mogą być ograniczone w praktycznych zastosowaniach co może utrudniać faktyczny zbiór [5].

Głównym elementem maszyny do samodzielnego zbioru plonów jest algorytm, najczęściej oparty na metodach uczenia maszynowego, który musi w szybki i dokładny sposób rozpoznawać plon, a ciągły rozwój metod wizyjnych, algorytmów sztucznej inteligencji [6] oraz miniaturyzacji technologii [7] w ostatnich latach, doprowadził do przyśpieszenia prac nad tworzeniem takiej maszyny.

Poprawianie dokładności tych algorytmów, a także zapewnienie ich działania w zmiennych warunkach, jest krokiem w stronę stworzenia w pełni autonomicznej maszyny do zbioru wrażliwych i wielokrotnie owocujących owoców i warzyw. O ile istnieją już algorytmy, które są w stanie całkiem dokładnie lokalizować obiekty na podstawie koloru (truskawki) [8] (maszyna implementująca algorytm widoczna na rysunku 1.1), to detekcja plonów, których kolor jest podobny do ich naturalnego otoczenia jest zadaniem bardziej złożonym.

Szczególnym przykładem mogą być tutaj ogórki gruntowe, które z względu na swój kolor są wyjątkowo trudne do zbioru, ponieważ zlewają się z otoczeniem liści i łodyg w których otoczeniu rosną. Powstały próby stworzenia programów, które dokonywałyby segmentacji na podstawie koloru, jednak badania były przeprowadzane na konkretnej odmianie (jaśniejszej niż większość odmian) [9].

Ponadto z uwagi na wielkość gospodarstw rolniczych w europie, często nie są one w stanie nadążać za najnowszymi technologicznymi osiągnięciami co może skutkować pogłębieniem się monopolu wielkich firm w tym sektorze. Rozwiązania technologiczne w rolnictwie są często drogie i wymagają długofletnych inwestycji, na co małe firmy nie mogą sobie pozwolić, z uwagi na możliwe ryzyko. Cena i stopa zwrotu rozwiązania są zatem głównymi czynnikami, które mają wpływ na podejmowane inwestycje i modernizację małych i średnich gospodarstw rolniczych. Szczególnie ważne jest zatem wprowadzenie systemu, który oferowałby znaczącą stopę zwrotu, a przy tym nie charakteryzował się wysoką ceną. Alternatywnie system może być skalowalny, czyli np. koszt jednostkowej maszyny jest niewielki i może ona pracować niezależnie od innych, oraz nie wymaga dodatkowej infrastruktury.

Motywacją do podjęcia pracy jest stworzenie algorytmu, który będzie wykonywał zadanie detekcji ogórków w środowisku naturalnym lepiej niż inne algorytmu uczenia głębokiego, w oparciu o cechy szczegółowe ogórków oraz sposób uczenia algorytmu. Stworzenie takiego rozwiązania znacząco przybliżyłoby świat i rynek do stworzenia maszyny do automatycznego zbioru wszystkich owoców i warzyw, które pracowałyby bezstratnie i przyczyniłyby się do obniżenia cen żywności w Europie i w konsekwencji na całym świecie, a przy tym nie wzmaczałyby monopolizacji przemysłu żywieniowego i rolniczego.

1.2. Cel i zakres pracy

Celem niniejszej pracy jest stworzenie systemu wizyjnego składającego się z kamery i programu, który w oparciu o algorytm YOLO [10] dokonywałby detekcji ogórków gruntowych w ich naturalnym środowisku (tzn. na polu rolniczym). W oparciu o algorytmy uczenia głębokiego oraz klasyczne metody przetwarzania obrazów, stworzono program, który poprawia rezultaty działania algorytmu YOLO i sprawia, że detekcja ogórków jest dokładniejsza, czyli mniej ogórków jest pomijanych przez program oraz mniej obiektów jest błędnie klasyfikowanych jako ogórek. Ponadto jednym z głównych celów algorytmu jest poprawienie detekcji ogórków, które posiadają charakterystyczny kwiat żółtego koloru. Motywacją pracy, było zaprogramowanie mechanicznego manipulatora, który w oparciu o system wizyjny, stworzyłby system do automatycznego zbioru ogórków gruntowych, jednak nie jest to przedmiotem pracy, ale będzie kontynuowane w dalszych badaniach. Obecne rozwiązania są zbyt drogie do wprowadzenia na rynek rolniczy lub nie są konkurencyjne w porównaniu do pracy człowieka.

Głównymi założeniami przedstawionego algorytmu detekcji obiektów są:

1. System musi dokonywać detekcji w czasie niekrótszym niż człowiek (ok. 0.1 sekundy)
2. Dokładność detekcji powinna być wyższa niż bazowego algorytmu YOLO
3. Detekcja obiektów z żółtymi kwiatami zostanie poprawiona względem bazowego algorytmu YOLO

1.3. Organizacja pracy

Rozdział 1 ("Wprowadzenie") opisuje sytuację zbioru plonów na rynku rolniczym oraz przedstawia podstawy automatycznego zbioru owoców i warzyw ze szczególnym wyróżnieniem zbioru ogórków gruntowych. Ponadto rozdział przedstawia motywację, cel i zakres pracy autora.

Rozdział 2 ("Przegląd literatury") przedstawia obecne osiągnięcia algorytmów stosowanych do lokalizacji owoców i warzyw i omawia ich zastosowania w odniesieniu do ich użycia w maszynach rolniczych w środowisku naturalnym.

Rozdział 3 ("Opis metod i zbioru danych") zawiera opis akwizycji i cech zbioru danych użytego w tej pracy oraz zaproponowane przez autora rozwiązania i poszczególne techniki, które miały na celu poprawienie skuteczności działania algorytmu detekcji obiektów.

Rozdział 4 ("Omówienie wyników") przedstawia uzyskane rezultaty eksperymentów i wpływ poszczególnych metod na ostateczny wynik algorytmu. Omówione zostają też potencjalne poprawki, które mogłyby usprawnić działanie algorytmu.

Rozdział 5 ("Podsumowanie") prezentuje wszystkie wnioski wynikające z przeprowadzonych badań i obrazuje największe zagrożenia i szanse związane z przyszłością detekcji obiektów i automatycznego zbioru owoców i warzyw. Dodatkowo zostają przedstawione osiągnięcia pracy oraz co udało się osiągnąć.

2. Przegląd literatury

2.1. Podstawy uczenia głębokiego i cyfrowego przetwarzania obrazów w kontekście interpretacji danych wizyjnych

Konwolucja i sieci konwolucyjne (ang. Convolutional Neural Networks, CNN) stały się głównym elementem współczesnego przetwarzania i klasyfikacji obrazów. Od kiedy sieć AlexNet [11], która była pierwszą nowoczesną wielowarstwową siecią konwolucyjną, uzyskała najlepszy na świecie wynik podczas wyzwania klasyfikacji obrazów ze zbioru ImageNet [12], który jest jedną z największych baz obrazów do klasyfikacji na świecie (zawiera ponad 14 milionów obrazów z 1000 różnych klas), praktycznie wszystkie nowoczesne rozwiązania w dziedzinie przetwarzania obrazów implementują architekturę sieci CNN. Uczenie takiej sieci w przypadku klasyfikacji polega na podawaniu jej na wejściu obrazów i oczekiwaniu danego wyjścia.

Przykładowo, podajemy do sieci naprzemiennie zdjęcia kotów i psów. Kiedy podajemy zdjęcie kotów to oczekujemy, że program zwróci nam wartość 0, zaś kiedy podajemy mu zdjęcia psów to oczekujemy wartości 1. W ten sposób, poprzez ciągłe podawanie obrazów i korygowanie błędu klasyfikacji, algorytm uczy się, który obraz jest kotem i kiedy powinien zwrócić 0, oraz który obraz jest psem i kiedy powinien zwrócić 1. Algorytm uczy się tego poprzez aktualizację wag dzięki wstępnej propagacji błędu. Określa on o ile pomylił się podczas klasyfikacji, a potem propaguje błąd wsteczny, czyli sprawdza, jaki był udział w tym błędzie poszczególnych neuronów. W czasie propagacji poprzez warstwy, uaktualniane są wszystkie wagi w sieci. Algorytm propagacji wstecznej określa udział błędu poszczególnych neuronów, ale nie określa dokładnie jak i o ile mają zostać zmienione wagi danego neuronu, ponieważ za to odpowiada optymalizator. Algorytm spadku stochastycznego gradientu (ang. Stochastic Gradient Descent, SGD) [13] jest jednym z najczęściej stosowanych optymalizatorów.

2.1.1. SGD - sposób działania

Optymalizator SGD oblicza gradient funkcji kosztu (czyli pochodnej funkcji kosztu względem wag modelu). Ten gradient wskazuje, w którym kierunku i jak mocno należy dostosować wagi, aby zmimimalizować błąd predykcji. Wzór opisujący zmianę wag modelu:

$$w_{i+1} = w_i - \alpha * L_i \quad (2.1)$$

$$L_i = \beta * L_{i-1} + (1 - \beta) * G + \lambda * w_i \quad (2.2)$$

gdzie:

w_i - wartości wag w i-tej iteracji

G - gradient funkcji kosztu

α - krok uczenia (ang. learning rate)

β - momentum

λ - "weight decay"

Krok uczenia odpowiada udziałowi gradientu w zmianie wag i jest podstawowym hiperparametrem wpływającym na szybkość uczenia modelu. Parametr momentum odpowiada za wpływ wcześniejszej składowej gradientu w obliczaniu nowego. Pozawala on na bardziej stabilne uczenia, ponieważ lepiej uwzględnia globalny kierunek uczenia. Parametr "weight decay" odpowiada za zmniejszanie wartości wag, dzięki czemu kontroluje ich rozmiar i zapobiega przeuczeniu. Stosowanie "weight decay" pomaga w tworzeniu bardziej ogólnych modeli, które lepiej generalizują do nowych danych, a nie tylko dobrze dopasowują się do danych treningowych.

2.1.2. "Label smoothing"

"Label smoothing", czyli wygładzanie etykiet, jest narzędziem stosowanym w klasyfikacji obrazów, które polega na modyfikacji etykiet obrazów w celu poprawy ogólnej zdolności generalizacji modelu. W tradycyjnym podejściu w zadaniach klasyfikacji, funkcja kosztu często zachęca model do przydzielania 100% pewności jednej konkretnej klasy, nawet jeśli model nie jest całkowicie pewny. "Label smoothing" wprowadza pewien stopień niepewności w przypisanych etykietach, co pomaga w tworzeniu bardziej odpornych na przeuczenie modeli.

Formalnie, nowy wektor etykiet można zdefiniować jako:

$$y' = (1 - \epsilon) * y + \frac{\epsilon}{C} \quad (2.3)$$

gdzie:

C - ilość klas

ϵ - parametr "label smoothing"

Parametr "label smoothing" jest najczęściej wyznaczany eksperymentalnie w zależności od problemu, jednak najczęściej stosuje się wartości z przedziału od 0 do 0,2. Korzyści zastosowania "label smoothing" obejmują zwiększenie odporności modelu na szумy w danych treningowych, co może prowadzić do lepszego generalizowania do nowych, nieznanych danych. "Label smoothing" jest szczególnie przydatne w przypadku modeli, które mają tendencję do zbytniego dopasowywania do danych treningowych.

2.1.3. ResNet

Po sukcesie AlexNet zaczęto badać możliwości głębokich sieci konwolucyjnych. Jednak z czasem kiedy zaczęto tworzyć bardzo głębokie, składające się z dziesiątek warstw sieci, to okazało się, że ich

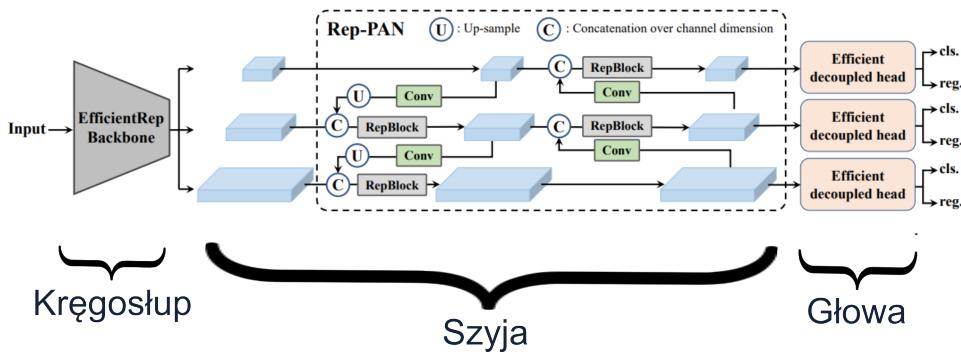
możliwości są ograniczone. W przypadku wykorzystania strategii uczących wykorzystujących metody gradientowe dla wielu warstw (np. sieci CNN) zwykle natrafiamy na problem zanikającego gradientu, ponieważ pochodne są z przedziału $[0, 1]$, więc wielokrotne przemnażanie prowadzi do bardzo małych liczb prowadząc do coraz mniejszych zmian wag w warstwach. Problem ten udało się rozwiązać poprzez zastosowanie bloków residualnych w sieci ResNet [14]. Znaczącą różnicą w odróżnieniu od innych podejść było zrezygnowanie wyłącznie z liniowości kolejnych warstw konwolucyjnych tzn. informacja zawarta w danej warstwie jest przekazywana tylko do następnej warstwy. Zamiast bezpośrednio przekazywać informacje przez warstwy, przekazywane są rezydua (ang. residuals). Warstwy residualne są zdefiniowane jako $F(x)+x$, gdzie $F(x)$ to transformacja warstwowa, a x to wejście do tej transformacji. Cała idea polega na przekazywaniu rezydu, co pozwala na zachowanie informacji z warstwy wejściowej, nawet jeśli warstwa dokonuje swojej własnej transformacji.

2.1.4. Transfer learning

Sieci konwolucyjne, które uzyskają najlepsze statystyki dokładności, często składają się z wielu warstw konwolucji i posiadają miliony trenowalnych parametrów. Żeby uniknąć przeuczenia sieci, tzn. idealnego dopasowania sieci do danych wejściowych, należy używać możliwie jak największej ilości danych, ponieważ tylko wtedy do sieci dostarczane jest dość przypadków i dochodzi do faktycznej generalizacji informacji o klasach. Istnieją jednak przypadki, kiedy ilość danych jest zbyt mała i sama w sobie nie wystarczy do wytrenowania sieci bez poświęcenia generalizacji. Wtedy możemy wykorzystać "transfer learning", czyli transfer wiedzy z obrębu innego, szerszego zagadnienia, do podobnego problemu. Polega to na uczeniu sieci na większym zbiorze w celu generalizacji wiedzy i nauki pierwszych warstw sieci poprawnej ekstrakcji cech, a następnie trenowanie na mniejszym, bardziej specyficznym zbiorze danych. Sprawia to, że nasz model posiada ogólną możliwość ekstrakcji cech, z uwagi na przetrenowanie na większym zbiorze i nauczył się rozpoznawać przydatne cechy, a przy tym potrafi rozwiązywać nasz problem, ponieważ został na nim powtórnie wytrenowany. Częstą praktyką jest zamrażanie pierwszych warstw sieci po uczeniu na większym zbiorze z uwagi na zachowanie cech, które są ekstrahowane w trakcie pierwszego treningu. W ten sposób modelując sposobem pierwszego i drugiego treningu, oraz ilością zamrożonych warstw, możemy dopasować sieć do naszego pierwotnego problemu.

2.2. Detekcja obiektów - algorytm YOLO

Algorytm "You Only Look Once"[15] (YOLO) jest obecnie jednym z najlepszych rozwiązań w detekcji obiektów. Charakteryzuje się on jednocześnie bardzo wysoką dokładnością detekcji, jak i bardzo dużą szybkością działania. [16] Algorytm osiąga najlepsze metryki dokładności na zbiorze MS COCO (ang. Microsoft Common Objects in Context) [17], który jest jednym z największych zbiorów obrazów do detekcji obiektów. Zbiór MS COCO posiada aż 80 klas różnych obiektów codziennego użytku (np.



Rys. 2.1. Przykładowa architektura algorytmu YOLOv6 z wyszczególnionymi "Kręgosłupem", "Szyją" i "Główą" [19]

osoba, auto, krzesło itp.) i jest międzynarodowym benchmarkiem detekcji obiektów w czasie rzeczywistym. Najnowsze odsłony YOLO są w stanie dokonywać detekcji obiektów na obrazie w tempie przekraczającym 100 FPS co sprawia, że algorytm może być wykorzystywany do przetwarzania danych w czasie rzeczywistym. [18]

Do tej pory powstało ponad 8 wersji tego algorytmu. Poszczególne wersje różnią się od siebie, jednak poszczególne segmenty pozostają wspólne dla wszystkich wersji tego algorytmu. W ogólnej architekturze algorytmu wyróżniamy 3 główne składowe. Są nimi "Kręgosłup"(ang. Backbone), który odpowiada za wstępna ekstrakcję cech, "Szyja"(ang. Neck), który dokonuje agregacji cech wyekstrahowanych przez "Kręgosłup" oraz "Główą", która jest najważniejszą częścią algorytmu, ponieważ na podstawie wejścia otrzymanego od wcześniejszych warstw dokonuje ostatecznej decyzji, gdzie znajdują się obiekty. Przykładowa architektura algorytmu z podziałem na wyszczególnione sekcje jest widoczna na rysunku 2.1.

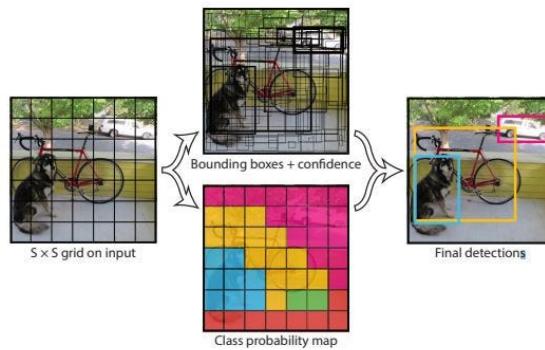
2.2.1. Architektura "Głów"

Sposób działania "Głów" można podzielić na 3 części:

1. Rysowanie siatki i obliczanie pewności
2. Rysowanie "bounding boxes"
3. "Non-maximum-suppression"

2.2.1.1. Rysowanie siatki i obliczanie pewności

Każdy obraz poddawany detekcji dzielony jest na siatkę o wymiarach NxN, gdzie wartość liczbową N różni się w zależności od wersji algorytmu. Następnie w każdej komórce ów siatki obliczana jest klasa, do której dana komórka należy, oraz pewność detekcji, czyli na ile algorytm jest pewien, że obiekt, faktycznie się w niej znajduje.



Rys. 2.2. Schemat działania głównego w oryginalnej wersji YOLO [15]

2.2.1.2. Rysowanie “bounding boxes”

Każda komórka siatki dokonuje predykcji określonej liczby "bounding boxes", czyli kandydatów na obiekt. Algorytm YOLO określa parametry każdego z kandydatów i przedstawia je w postaci pojedynczego wektora.

$$Y = [pc, bx, by, bh, bw, c1, c2], \text{ gdzie:}$$

pc - określa prawdopodobieństwo, że wewnątrz komórki znajduje się obiekt

bx, by - określa koordynaty środka prostokata, który obrysowuje potencjalny obiekt

bh, bw - określa wysokość i szerokość prostokąta

c1, ..., cn - odpowiada za określenie prawdopodobieństwa, że w prostokącie znajduje się obiekt danej klasy

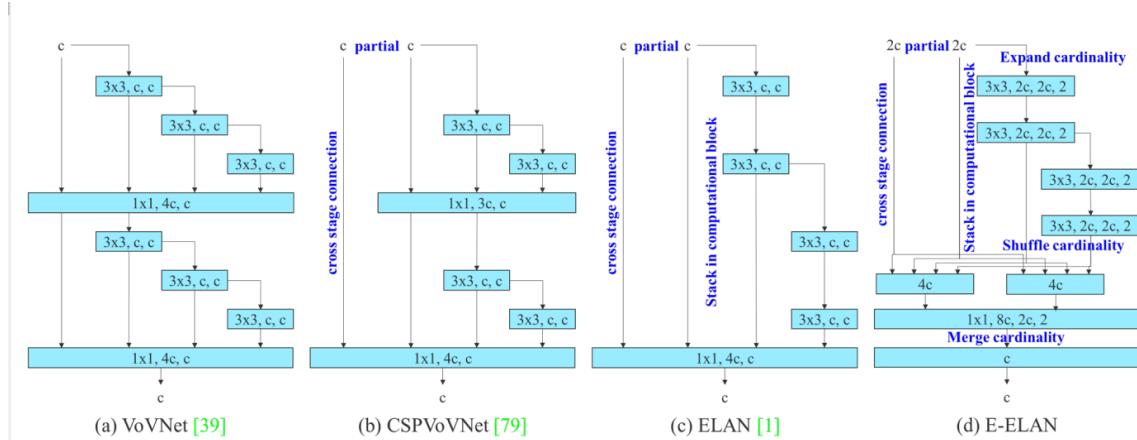
2.2.1.3. “Non-maximum-suppression” (NMS)

Każda komórka siatki określa swoje predykcje obiektu, ale nie wszystkie z nich są znaczące i dobrze obrazują faktyczne położenie obiektu. Zadaniem NMS jest wykluczenie predykcji o niskiej pewności detekcji obiektów oraz usunięcie duplikatów. W ten sposób wyjściem algorytmu zostają tylko "bounding boxes" o najwyższej wartości pewności detekcji, które są odpowiedzialne za różne obiekty.

Sposób działania NMS przedstawia się następująco:

1. Wszystkie predykcje obiektu sortowane są malejąco pod względem pewności detekcji, czyli na ile model jest pewny, że w danym "bounding box" znajduje się obiekt.
2. Następnie wybierany jest obiekt z najwyższym współczynnikiem pewności i staje się on punktem startowym.
3. Pomiędzy punktem startowym, a wszystkimi pozostałymi "bounding boxes" oblicza się IoU (ang. Intersection over Union), które określa się jako iloczyn pól obu prostokątów podzieloną przez ich sumę.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (2.4)$$



Rys. 2.3. Porównanie sieci ELAN [22] oraz E-ELAN [18]

4. Wszystkie "bounding boxes" o IoU większym niż określony próg, są usuwane ze zbioru. Próg najczęściej zostaje dostosowany eksperymentalnie lub w zależności od potrzeby działania algorytmu, a najczęściej przyjmowaną wartością jest 0,5.
5. Kroki 2 – 4 są powtarzane do momentu, aż nie jest możliwe usunięcie większej ilości "bounding boxes".

Zbiór uzyskanych w ten sposób "bounding boxes" jest ostatecznym zbiorem predykcji. Celem tego kroku jest uzyskanie "bounding boxes" opisujących różne obiekty, które posiadają wysoką pewność detekcji obiektu. Ogólny schemat działania głównego dla algorytmu YOLO [15] przedstawia rysunek 2.2.

2.2.2. YOLOv7

Sieć YOLOv7 [18] powstała w 2022 roku, była w czasie swojej publikacji jednym z najlepszych algorytmów do detekcji obiektów. Jej architektura opiera się na wcześniejszych sieciach YOLO [15, 20, 21], ale wprowadza udoskonalenia, które podnoszą skuteczność detekcji i zmniejszają jej czas działania. Główną zmianą jest użycie rozszerzonej wydajnej sieci agregacji warstw (ang. Extended efficient layer aggregation networks, E-ELAN), która pozwala na lepszą kontrolę ścieżki gradientu. Sieć oparta o ELAN [22] wykorzystuje rozszerzanie, mieszanie, a następnie łączenie poszczególnych bloków obliczeniowych. Porównanie sieci ELAN i E-ELAN znajduje się na rysunku 2.3. Rozszerzanie i tasowanie bloków ma na celu zwiększenie dywersyfikacji ekstrahowanych cech oraz zwiększenie się możliwości uczenia modelu.

Drugą największą zmianą jest sposób skalowania modelu. Twórcy YOLOv7 doszli do wniosku, że skalowanie głębokości i szerokości modelu nie jest rozłączne i wpływa na siebie nawzajem, dlatego powinno być rozpatrywane jednocześnie. W YOLOv7 skalowanie głębokości odbywa się tylko w blokach obliczeniowych, zaś reszta warstwy transmisyjnej jest odpowiednio skalowana w szerokości modelu. Sprawia to, że model lepiej dostosowuje swoje atrybuty. Algorytm implementuje również planowaną

ponownie sparametryzowaną konwolucję. Autorzy doszli do wniosku, że warstwa z połączeniami resztowymi lub konkatenacyjnymi nie powinna posiadać połączenia tożsamościowego. Dlatego bloki RepConv (konwolucja 3x3, konwolucja 1x1, połączenie tożsamościowe) zastąpiono blokami RepConvN, które nie zawierają przejścia tożsamościowego.

2.3. Podobne rozwiązania

Lokalizacja owoców i warzyw w środowisku naturalnym jest przedmiotem prac i badań od wielu lat [23]. Tworzenie lepszego algorytmu detekcji plonów jest ważne w kontekście stworzenia maszyny do samodzielnego zbioru, ale również do stworzenia maszyn, które będą pomagać szacować plony, określić sposób rozwinięcia danych owoców i warzyw, czy pomagać podejmować strategiczne decyzje na podstawie ilości i wyglądu plonu. [24] Poniżej przedstawię najczęstsze podejścia wykorzystywane w algorytmach do detekcji obiektów oparte o:

1. YOLO
2. Cyfrowe przetwarzanie obrazów

2.3.1. Detekcja obiektów - YOLO

Algorytm YOLO jest wszechstronnym rozwiązaniem w przypadku detekcji obiektów i bardzo dobrze sprawdza się w przypadku bardzo różnego rodzaju danych. Lokalizacja przechodniów [25], znaków drogowych [26] czy różnorakich obiektów medycznych [27] to tylko niektóre z przypadków jego użycia. Algorytm ten znalazł również swoje zastosowanie w przypadku detekcji owoców i warzyw, z uwagi na swoją dokładność i szybkość działania. W celu dopasowania działania algorytmu do konkretnego rozwiązania, czyli np. detekcji konkretnego rodzaju owocu lub warzywa, często wprowadza się zmiany do architektury algorytmu i do jego sposobu uczenia w celu zwiększenia dokładności detekcji.

W badaniach przeprowadzonych na Uniwersytecie Rolniczym Shanxi w Chinach [28] sprawdzono wpływ zmiany algorytmu bazowego YOLOv3 [20] na dokładność wykrywania pomidorów w szklarniach. Główną nowością wprowadzoną do architektury sieci jest zmiana niektórych ukrytych bloków residualnych na bloki gęste (eng. Dense) [29], które chociaż wymagają większej ilości obliczeń to lepiej radzą sobie z problemem "zanikającego gradientu" i są w stanie ekstrahować więcej przydatnych do detekcji cech. Dodatkowo w artykule sprawdzane są 3 różne wersje sieci, które różnią się funkcją aktywacji. Wyniki przedstawione w artykule pokazują, że umiejętna manipulacja architekturą YOLO może zwiększyć jego możliwości detekcji w konkretnych problemach dlatego warto dostosowywać ją indywidualnie do każdego zastosowania. Jednakże pomimo udoskonalenia wersji YOLOv3 to osiągała ona gorsze wyniki niż nowsza wersja YOLOv4 oraz potrzebowała więcej czasu na przetworzenie obrazu.

Badania przeprowadzone na Uniwersytecie w Dalian [30] były poświęcone modyfikacji architektury bazowego algorytmu sieci YOLOv4 [21] i jej wpływu na detekcję owoców wiśni w środowisku naturalnym. Głównym problemem detekcji wiśni jest ich duża liczba na jednym zdjęciu i częste nakładanie

Tabela 2.1. Porównanie metryk dokładności różnych wersji YOLO [32]

Algorytm	P(%)	R(%)	mAP@0.5(%)	mAP@0.5–0.95(%)
YOLOv7	92.9	88.3	94.7	83.1
YOLOv5	93.1	88	94	79

się na siebie owoców. Może to prowadzić do klasyfikacji klastra owoców jako jednego obiektu lub po minięcia częściowo zakrytych przypadków owoców w czasie detekcji. W celu uniknięcia tego problemu zdecydowano się na zmianę "ogona" algorytmu z cspdarknet53 [21], na DenseNet [29]. Wyniki przeprowadzonych badań pokazały, że lepiej radził on sobie z ekstrakcją wartościowych cech, co przełożyło się na wyższą dokładność detekcji. Kolejną nowelizacją zaproponowaną przez autorów artykułu była zmiana kształtu "bounding boxa" z prostokąta na koło. Owoce wiśni charakteryzują się swoim kolistym kształtem, a dokładniejsze dopasowanie "bounding box" do faktycznego obiektu sprawia, że algorytm jest sobie w stanie lepiej poradzić z jego lokalizacją. Ponadto w przypadku "bounding box" w kształcie koła, małe odchylenia środka detekcji od środka obiektu "ground truth" są obarczone mniejszym błędem.

Dokładność detekcji jest najważniejszą metryką, która świadczy o aplikowalności modelu, jednak w przypadku rozwiązań praktycznych i implementacji algorytmów w maszynach rolniczych, równie ważny jest jego czas działania oraz wielkość modelu. Algorytmy z rodziny YOLO, cechują się relatywnie szybką pracą i ograniczoną liczbą parametrów, dlatego często są wykorzystywane w maszynach do automatycznego zbioru owoców i warzyw. Dalsza miniaturyzacja modelu i przyśpieszenie jego działania jest jednak bardzo pożądane, ponieważ mogłoby ograniczyć koszty rozwiązania i zwiększyć jego wydajność. W swoim artykule [31] Dandan Wang i Dongjian He przedstawiają sposób ponad dziesięciokrotnego zmniejszenia wielkości modelu YOLOv5, którego zadaniem jest estymacja ilości owoców jabłoni przed przerzedzaniem w celu między innymi szacowania wielkości plonów. Poprzez wygaszanie wag, które nie przyczyniały się znacząco do dokładności detekcji, udało się znacząco zmniejszyć wielkość modelu i przyspieszyć czas jego działania, przy jednoczesnym zachowaniu dokładności pierwotnego modelu.

Uczenie głębokie może być wspomagane klasycznymi metodami przetwarzania obrazów. W artykule poświęconym detekcji owoców Camellia oleifera [32] przedstawione zostało porównanie działania algorytmu YOLOv7 oraz YOLOv5. Dodatkowo użyta została nowatorska metoda określania środka zlokalizowanego obiektu oparta w pełni na cyfrowym przetwarzaniu obrazów.

Porównanie algorytmów na podstawie metryk przedstawia tabela 2.1. Na podstawie porównania skuteczności można wywnioskować, że skuteczność detekcji jest bardzo podobna dla obu algorytmów z niewielką przewagą na rzecz wersji v7. Algorytm ten był w stanie wykrywać obiekty, które zostały pominięte przez wersję v5. Czas detekcji algorytmu v7 jest również krótszy co dodatkowo przemawia na jego korzyść.

W celu dokładnego określenia centroidu obiektu zostało zastosowanych kilka różnych metod. Na początku obiekt, który został zlokalizowany jest wyodrębniany z obrazu. Następnie wokół obrazu konstruowana jest czarna ramka o szerokości 10 pikseli. Potem w celu pozbycia się szumu z obrazu, przy jednoczesnym zachowaniu informacji o krawędziach stosowany jest filtr bilateralny. Następnie dokonywana jest konwersja do modelu kolorów HSV, z których parametr V jest poddawany wyrównaniu histogramu (ang. histogram equalization). Następnie obraz jest na powrót konwertowany do modelu kolorów RGB. W ten sposób dokonywane jest wstępne kontrastowanie obrazu. Następnie w celu segmentacji owocu stosowane jest progowanie w skali HSV. Następnie operacje morfologiczne otwarcia i zamknięcia są stosowane na obrazie binarnym w celu pozbycia się artefaktów. Następnie obraz zostaje poddany filtrowi medianowemu, a uzyskana w ten sposób binarna maska nakładana jest na obraz. Potem algorytm detekcji krawędzi Canny używany jest w celu znalezienia obrysu owocu. Na podstawie uzyskanej krawędzi szukany jest środek ciężkości obrazu i on uznawany zostaje za środek obiektu. Ten sposób pozwala na bardzo szybkie przetworzenie informacji z obrazu w celu zlokalizowania prawidłowego centra obrazu i może okazać się pomocny w dokładnej lokalizacji obiektów. Algorytm YOLO został poddany również próbom wykrywania różnego rodzaju owoców i warzyw. W pracy z Uniwersytetu w Guangxi [33] zaproponowane zostaje Lemon-YOLO, które bazując na YOLOv3 [20] dokonuje detekcji cytryn w środowisku naturalnym. Algorytm uzyskał średnią dokładność (eng. Average Precision - AP) na poziomie 96,28%, co było możliwe dzięki zamianie modułu Darknet-53, zaimplementowanego w bazowym YOLOv3 na SE_ResGNet34. Artykuł A. Koirala [34] prezentuje algorytm do detekcji owoców mango, który osiągnął metrykę AP na poziomie 98,3%. Badacze z Południowo-chińskiego uniwersytetu rolniczego w Guangzhou [35] zaproponowali algorytm oparty o YOLOv3 i YOLOv4 do detekcji bananów w złożonym środowisku naturalnym i uzyskali dokładność detekcji na poziomie 99.29%.

Wyniki uzyskane przez algorytmy oparte o YOLO na różnych rodzajach owoców i warzyw świadczą o wysokim potencjale takich rozwiązań w kontekście tworzenia maszyny do automatycznego zbioru plonów. Metryki uzyskane przez badaczy mogą być mocno związane z poziomem trudności danych uczących i złożonością środowiska naturalnego, poziomem oświetlenia, porą dnia akwizycji danych oraz innych czynników, które wpływająby na trudność treningu. Wynikać może z tego, że meryki uzyskane przez algorytmy na bazie YOLO mogą być bardzo mocno skorelowane z trudnością danych i obiektów.

2.3.2. Cyfrowe przetwarzanie obrazu

W związku z rozwojem metod uczenia głębokiego, coraz częściej są one używane klasyfikacji, detekcji obiektów i przetwarzaniu obrazów, ponieważ charakteryzują się one największymi dokładnościami, ale jednocześnie wymagają dużej mocy obliczeniowej, która mocno wydłuża czas działania programu. Metody oparte o klasyczne metody cyfrowego przetwarzania obrazów charakteryzują się bardzo dużą szybkością i mogą zastępować rozwiązań uczenia głębokiego w przypadkach, gdzie czas działania programu jest kluczowym elementem jego powodzenia. W artykule Roemi Fernandez [9] przedstawiono algorytm segmentacji ogórków gruntowych w ich naturalnym środowisku bez użycia sieci CNN ani

innych metod uczenia głębokiego, który osiąga metryki porównywalne z innymi metodami "state-of-the-art" w dziedzinie detekcji owoców i warzyw do zastosowania w maszynach do zbiorów. Używając jedynie zdjęcia w formacie kolorów RGB (Red Green Blue) algorytm osiąga metrykę F1 na poziomie 0.878 co jest wynikiem porównywalnym do innych algorytmów do detekcji owoców i warzyw na przykład mango, melonów czy słodkiej papryki. Praca zaproponowana przez Xiaojun Yu [36] prezentuje algorytm do detekcji jabłek oparty o wykrywanie koloru, detekcję krawędzi oraz analizę kształtu na podstawie których określa położenie owoców. Uzyskał on dokładność detekcji na poziomie 82.5%, czyli niższą niż większość rozwiązań "state-of-the-art", ale jest w stanie dokonać detekcji zdecydowanie szybciej niż algorytmy detekcji obiektów oparte o głębokie sieci konwolucyjne.

O ile algorytmy oparte wyłącznie o cyfrowe przetwarzanie obrazów mogą nie nadawać się do zastosowań technicznych z uwagi na zbyt małą dokładność detekcji, to biorąc pod uwagę ich krótki czas działania mogą świetnie uzupełniać algorytmy oparte o uczenie głębokie i poprawiać dokładność detekcji, czy położenie "bounding boxa".

3. Metody i opis danych

3.1. Dane

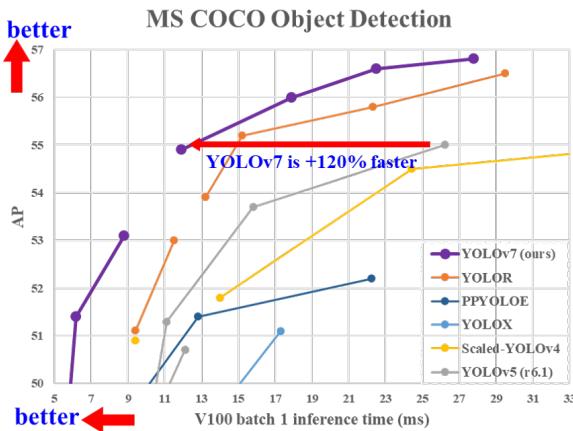
Zbiór danych, którego część jest widoczna na rysunku 3.1, tworzą 594 autorskie zdjęcia ogórków gruntowych w środowisku naturalnym. Średnio na każdym zdjęciu znajduje się około 5 ogórków. Zdjęcia zostały zrobione w Wielkopolsce w lipcu 2022 r. na dwóch różnych polach. Wszystkie zdjęcia zostały zrobione jednego dnia w dwóch porach, rano i popołudniu. W czasie robienia zdjęć rano pogoda była pochmurna i zdjęcia są ciemniejsze, a w czasie robienia zdjęć popołudniu świeciło pełne słońce, co sprawia, że niektóre obiekty są bardzo jasne, a inne zacienione. Zdjęcia były robione w odstępie ok. 3 metrów od siebie, czyli każdy na każdym zdjęciu znajduje się zbiór unikalnych ogórków, z których żaden nie występuje na innych zdjęciach. Do ich zrobienia użyto aparatu z telefonu Xiaomi Redmi Note 8 Pro, a rozdzielcość każdego zdjęcia to 4624x3472. Dane zostały poddane etykietowaniu tzn. na każdym zdjęciu zostały zaznaczone ogórki, które następnie posłużyły jako 'ground truth' w uczeniu algorytmu. Każde zdjęcie zostało sprawdzone dwukrotnie przez autora pracy. Dane zostały losowo podzielone na zbiory treningowy walidacyjny i testowy w proporcji 7:1:2. Komputer użyty do uczenia i przetwarzania algorytmu posiada procesor AMD Ryzen 7 4800HS, 16GB pamięci RAM oraz kartę graficzną NVIDIA GeForce RTX 2060 Max-Q. Algorytm został zaimplementowany w Pythonie 3.10.

3.2. Algorytm YOLO

Bazowym algorytmem detekcji obiektów zostało YOLOv7, z uwagi na kompromis pomiędzy czasem działania a dokładnością detekcji uzyskaną na zbiorze MS-COCO. Porównanie jego czasu działania i dokładności względem innych algorytmów jest widoczne na rysunku 3.2. Szczególnie istotnym faktem jest skrócenie czasu wykonywanej detekcji w stosunku do poprzednich wersji YOLO co sprawia, że algorytm jest lepiej dostosowany do detekcji obiektów w czasie rzeczywistym. Dodatkowo YOLO zapewnia preprocessing oraz zbalansowany sposób treningu zapewniający na generalizację danego problemu detekcji.



Rys. 3.1. Przykładowe dane



Rys. 3.2. Wydajność i dokładność YOLOv7 w porównaniu do innych algorytmów [18]

Tabela 3.1. Parametry dotyczące wstępnego przetwarzania danych

Nazwa parametru w YOLOv7	Wartość parametru
hsv _h	0,015
hsv _s	0,7
hsv _v	0,4
fliplr	0,5
mosaic	1,0

3.2.1. Wstępne przetwarzanie danych algorytmu YOLOv7

W celu zwiększenia generalizacji modelu, obrazy wejściowe zostają przetworzone przez algorytm "mosaic". Polega on na stworzeniu nowego obrazu z kilku prostokątnych części pierwotnych obrazów. Przetwarzanie wszystkich danych następuje dopiero po podziale na zbiory, w celu uniknięcia transferu informacji między zbiorami. Dodatkowe funkcje zaimplementowane w YOLOv7 obejmują losowy obrót w poziomie, skalowanie parametrów kolorów obrazu w skali HSV i skalowanie obrazu. Dokładne parametry użyte podczas przetwarzania zostały przedstawione w tabeli 3.1.

3.2.2. Uczenie algorytmu YOLO v7

Podczas uczenia algorytmu został wykorzystany optymalizator SGD z ustawionym krokiem uczenia na poziomie 0,001. Rozmiar parametru "momentum" optymalizatora został ustawiony na 0,937, "weight decay" na poziomie 0,0005, liczba klas algorytmu wynosi 1, a liczba epok treningowych wynosi 200. Liczba danych po których następuje uaktualnienie wag (ang. batch size) wynosi 4. Powyższe parametry zostały ustalone w sposób eksperymentalny i skutkują najlepszym dopasowaniem modelu do zadanej problemu detekcji. Optymalizacja parametru "label smoothing" jest opisana w sekcji z wynikami badań.

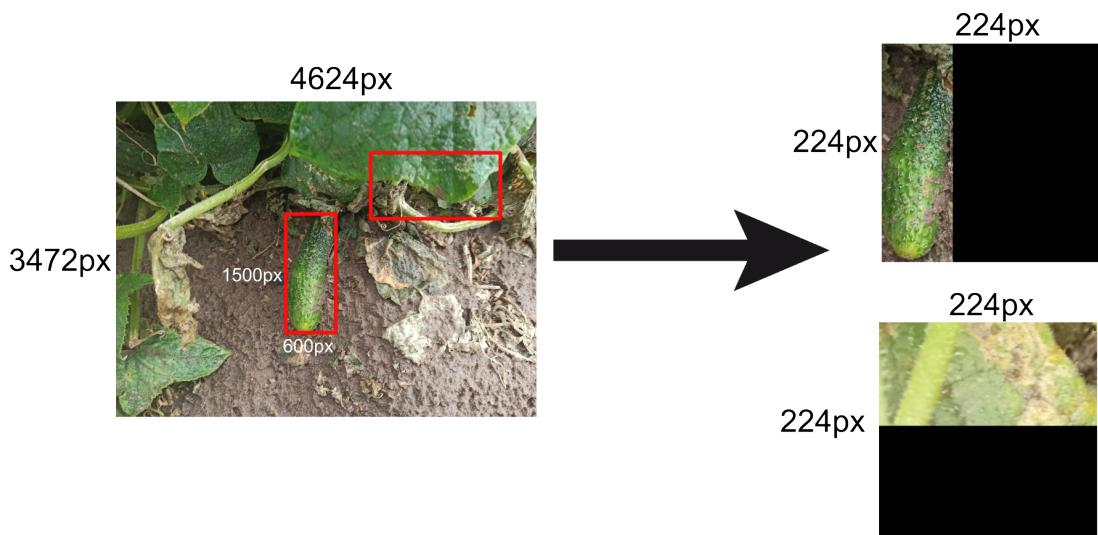
3.3. Sieć przedtreningowa

Zbiór danych został ręcznie etykietowany, a jego poprawność jest kluczową składową dobrej nauki algorytmu detekcji obiektów. Złe zaznaczenie obiektów w zbiorze uczącym, może przekładać się na nie-dokładne uczenie, a przypadki wykrycia obiektów byłyby karane przez algorytm przez co generalizacja modelu byłaby znacznie niższa, a metryki nie oddawałyby faktycznego stanu dokładności modelu. Standardowa procedura etykietowania zbioru danych do detekcji obiektów polega na sprawdzeniu każdego zdjęcia przez 3 lub więcej sędziów, którzy orzekają czy w danym miejscu znajduje się obiekt. Z uwagi na wyżej wymienione ograniczenia stworzony został algorytm, który w oparciu o posiadane już dane jest w stanie rozpoznać czy zdjęcie jest obiektem i na jego podstawie aktualnić zbiór danych. Procedura algorytmu wygląda następująco:

1. Część zdjęcia, na której zaznaczony był obiekt została z niego wycięta. Następnie przeskalowano dłuższą krawędź obrazu obiektu do wymiaru 224, zaś krótszą o taką samą wartość. Do obrazu dołączono czarny prostokąt o takich wymiarach, żeby jego złączenie z obrazem obiektem, tworzyło wymiary 224x224. Następnie z obrazu wybrano losowy obszar o wymiarach od 0,5 do 1,5 wartości obrazu obiektu, który nie pokrywał się z żadnym obiektem (IOU mniejsze niż 0,1) i powtórzono procedurę skalowania i dołączania czarnego prostokąta. Schemat tworzenia danych przedstawia rysunek 3.3. W ten sposób uzyskano 3400 próbek zbioru obrazów, w którym stosunek klasy pozytywnej (ogórki) do negatywnej (liście i tło) wynosił idealnie 1:1.
2. Głęboka sieć konwolucyjna została zainicjalizowanymi wagami pochodząymi z treningu na większym zbiorze danych (ImageNet). Hiperparametry algorytmów oraz ilość epok były osobno ustalone dla każdej testowanej sieci w celu zwiększenia dokładności klasyfikacji. Szczególnie ważne jest tutaj trenowanie z naciskiem na jak najwyższą precyzję klasyfikacji, ponieważ zmniejszy ona ryzyko przedostania się do zbioru przypadków prawdziwie fałszywych.
3. Na wcześniej wytrenowanym algorytmie YOLO dokonano detekcji obiektów przy niskim parametrze 'confidence' (algorytm wykrywa wszystkie obiekty, które mogą być ogórkami) i położenie obiektów zostały zapisane do pliku.
4. Wcześniej wytrenowana sieć neuronowa sprawdza każdy zlokalizowany przez YOLO obiekt i jeśli jest on ogórkiem to dodaje go do zbioru.
5. Procedurę przeprowadzono dla zbiorów treningowego, walidacyjnego i testowego.

3.4. Self-paced learning

Algorytmy detekcji obiektów mają szczególny problem z wykrywaniem małych obiektów i jest to obecnie jeden z głównych kierunków ich rozwoju i udoskonalania. Dzieje się tak dlatego, że małe obiekty



Rys. 3.3. Schemat tworzenia danych, do sieci przedtreniowej - z obrazu wycinany jest obiekt, oraz obszar, który nie nakłada się znacząco z innymi obiekttami. Następnie następuje skalowanie obu długości krawędzi w taki sposób, żeby dłuższa krawędź miała wartość 224 pikseli. Przykładowo: jeśli obiekt ma na pierwotnym zdjęciu wymiary 1500x600 to dłuższą krawędzią jest 1500. Następuje jej skalowanie do 224, czyli jej długość jest zmniejszana około 6,7 razy. Tyle samo razy zmniejszana jest druga krawędź, czyli jest skalowana do wartości ok. 90. Następnie dołączany jest czarny prostokąt (odpowiednio od dołu lub prawej strony w zależności od tego która krawędź jest dłuższa) w celu uzyskania obrazu o wymiarach 224x224.



Rys. 3.4. Po lewej zdjęcie pierwotne, po prawej zdjęcie z zasłoniętymi łatwymi obiek-tami

mają inną skalę i są inaczej reprezentowane przez piksele niż obiekty z tej samej klasy o większym rozmiarze. Ponadto problematyczne może być również odróżnienie niektórych struktur od tła w momencie kiedy wielkość obiektu nie jest wystarczająca. Obiekty małe można uznać więc za trudne dla algorytmu do klasyfikacji. Problem polega na tym, że nie tylko małe obiekty bywają pomijane, ale również te średnie i większe mogą być z jakiegoś powodu przez algorytm pomijane. "Self-Paced Learning"[37] polega na wyznaczeniu trudności próbek danych za pomocą algorytmu i sukcesywnym uczeniu algorytmu na coraz bardziej trudnych danych w celu optymalizacji uczenia. Algorytm rozpoznaje, które obiekty na danym zdjęciu są dla niego najtrudniejsze do detekcji i następnie może być uczony tylko na tych przypadkach, których nie był w stanie rozpoznać wcześniej. W momencie kiedy na jednym zdjęciu pojawia się kilka obiektów, jedne mogą okazywać się trudniejsze, a inne bardzo trywialne do rozpoznania, więc zaklasyfikowanie takiego zdjęcia jako "trudne" nie byłoby sprawiedliwe. Z uwagi na powyższy problem zdecydowano się na poniższą procedurę selekcji przypadków prostych i trudnych:

- Algorytm detekcji obiektów jest wytrenowany na pierwotnym zbiorze danych
- Na jego podstawie zastosowany jest podział przypadków na trudne i łatwe w zależności od pewności detekcji (parametr "confidence" na poziomie 0,1).
- Dla zbioru trudnego zakrycie obiektów łatwych przez ciemnozielony prostokąt w celu ich usunięcia ze zbioru (schemat na rysunku 3.4).
- Dla zbioru łatwego odpowiednio zakrycie przypadków trudnych.

Następnie sprawdzone zostały różne metody treningu algorytmu YOLOv7 tzn. kolejność podawania danych, długość jego treningu oraz wartości hiperparametrów.

3.5. Wykrywanie koloru kwiatów

Dekrekcja małych ogórków jest szczególnie ważna podczas trenowania algorytmu. Ogórki do pewnej wielkości nie są zbierane z pola z uwagi na swoją możliwość wzrostu oraz poziom dojrzałości. Mimo



Rys. 3.5. Po lewej obraz oryginalny, po prawej obraz po segmentacji koloru żółtego

to nie uwzględnianie ich w detekcji może zmniejszyć jej dokładność, ponieważ ich struktura niczym nie różni się od większych ogórków co może niepotrzebnie prowadzić do błędного treningu algorytmu.

W celu wsparcia detekcji małych ogórków na podstawie koloru ich kwiatów zaproponowano następujący algorytm:

1. Po detekcji obiektów dokonywana jest detekcja koloru żółtego ([0:30, 125:255, 125:255] w modelu HSV).
2. Usuwane są segmenty koloru żółtego poniżej wielkości 200 pikseli dla obrazu 640x640 (usuwanie szumu).
3. Obiekt o pewności detekcji niższej niż próg (ale wyższej niż ustalona wartość) w których niedalekiej odległości znajduje się wysegmentowany żółty są zaliczane jako poprawne.

Obraz po segmentacji koloru żółtego przedstawia rysunek 3.5.

3.6. Metryki dokładności

W celu określenia dokładności działania modelu zastosowano wskaźniki precyzji (P), recall (R) oraz średniej precyzji (ang. mean Average Precision, mAP), których wzory przedstawiono poniżej:

$$P = \frac{TP}{TP + FP} \quad (3.1)$$

$$R = \frac{TP}{TP + FN} \quad (3.2)$$

$$AP_i = \int_0^1 P(R) d(R) \quad (3.3)$$

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (3.4)$$

TP oznacza liczbę przypadków poprawnie zaklasyfikowanych jako pozytywne, FP oznacza liczbę przypadków, w których algorytm wykrył obiekt, który nie był zaznaczony w "ground truth", natomiast FN oznacza liczbę przypadków pominięcia przez algorytm obiektu, który został oznaczony jako "ground truth".

4. Wyniki badań

4.1. Sieć przedtrenigowa

W trakcie wyboru sieci do klasyfikacji ogórków zostało wziętych pod uwagę wielu kandydatów, miedzy innymi AlexNet [11], VGG16 [38], ResNet50 i Resnet101 [14] oraz EfficientNet [39]. Wszystkie sieci zostały zainicjalizowane wagami wytrenowanymi na zbiorze ImageNet. Z uwagi na wstępne wyniki uzyskane po wytrenowaniu, zdecydowano się na sieć ResNet50, która jako jedyna uzyskała dokładność ponad 80% na zbiorze walidacyjnym w pierwszych testach. Dalsze testy optymalizacyjne zostały wykonywane tylko na sieci ResNet50.

Trenowanie całej sieci doprowadzało często do przeuczenia, dlatego zdecydowano się na zamrożenie części wag sieci. Na początku zamrożono całą architekturę sieci, a następnie, w kolejnych testach zaczęto odmrażać kolejne górne warstwy sieci.

Wyniki uczenia na zbiorze walidacyjnym oraz rodzaj zamrożonych sieci przedstawia tabela 4.1.

Najlepsze wyniki uzyskała sieć z odmrożoną warstwą liniową oraz ostatnimi 9 warstwami konwolucyjnymi. Znaczy to, że odmrożenie większej ilości warstw skutkuje przeuczeniem odmrożonych warstw sieci i zmniejsza możliwości generalizacji modelu.

Najważniejszą metryką sieci przedtrenigowej jest precyza, ponieważ nie chcemy przypadków, w których obiekt, który nie jest ogórkiem znajdzie się w "ground truth". Takie przypadki mogłyby znacząco obniżyć zdolność generalizacji modelu i uczyć się złych wzorców i rozpoznawać nieadekwatne do

Tabela 4.1. Dokładność i precyza sieci ResNet50 w zależności od ilości i rodzaju warstw odmrożonych. FC odnosi się do ostatniej warstwy liniowej (ang. fully connected, FC), zaś conv5a oraz conv4a odnosi się do warstw konwolucyjnych przedstawionych na Rysunku 4.1.

Warstwy sieci ResNet z odmrożonymi wagami	Dokładność	Precyza
Wszystkie warstwy zamrożone	0,761	0,783
FC	0,801	0,792
FC, conv5a	0,863	0,889
FC, conv5a, conv4a	0,855	0,849
FC, conv5a, conv4a, conv3a	0,846	0,853

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Rys. 4.1. Opis architektur różnych wersji sieci ResNet [14]**Tabela 4.2.** Metryki finalnej sieci przedtreningowej

Dokładność	Precyzja	Recall
0,841	0,965	0,707

problemu cechy podczas treningu. Dlatego w celu zwiększenia precyzji zdecydowano się na klasyfikację obiektu jako ogórek jeśli:

- Wartość dla klasy 'ogórek' większa niż klasy 'brak ogórka'
- Wartość dla klasy 'ogórek' jest większa niż 0,8

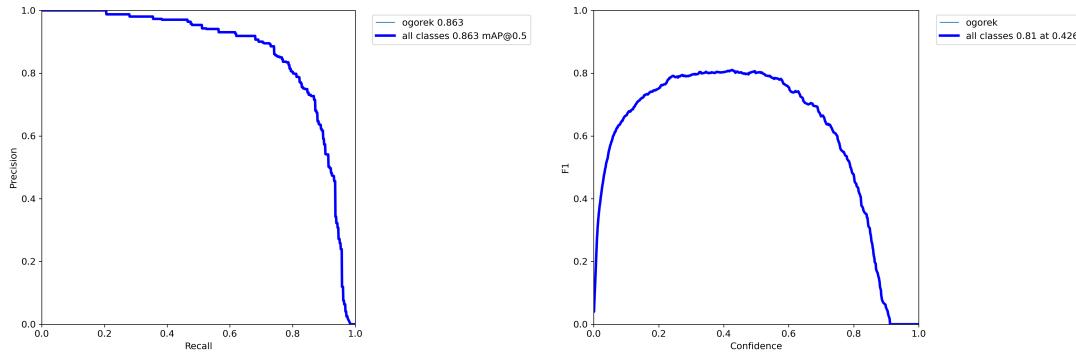
W ten sposób uzyskana sieć zmniejszy swoją dokładność, czyli większa część ogórków zostanie pominięta podczas dodawania do "ground truth", ale zwiększy się precyzja, czyli mniej przypadków fałszywie pozytywnych zostanie dodanych do zbioru. Metryki finalnej sieci, która posłużyła w dodaniu etykiet do zbioru, przedstawia tabela 4.2.

Sieć walidacyjna mimo stosunkowo niskiej dokładności (ok. 84,1% dla klasyfikacji binarnej) dodała 512 nowych etykiet do zbioru danych. W celu weryfikacji dokładności działania sieci, wszystkie nowe etykiety zostały ręcznie sprawdzone. Wyniki weryfikacji przedstawia tabela 4.3.

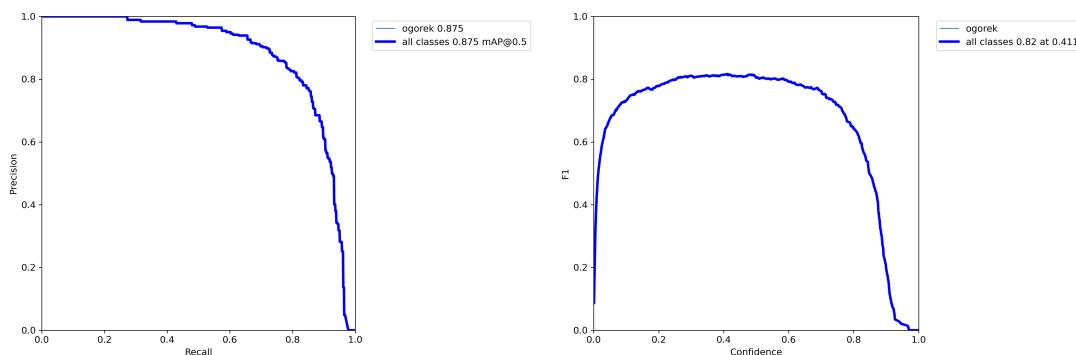
Następnie przeprowadzono uczenie sieci YOLOv7 na pierwotnym zbiorze oraz zbiorze z wszystkimi nowymi etykietami dodanymi przez sieć przedtreningową. Wykresy PR oraz F1-score dla danych przed i po dodaniu etykiet są zamieszczone odpowiednio na rysunkach 4.2 i 4.3.

Tabela 4.3. Metryki sieci przedtreningowej podczas uzupełniania etykiet

Ilość nowych etykiet	Ilość prawidłowo zaklasyfikowanych nowych etykiet	Precyzja
512	485	0,947



Rys. 4.2. Wykresy dla pierwotnego zbioru



Rys. 4.3. Wykresy dla zbioru z nowymi etykietami

4.2. Optymalizacja "label smoothing" w YOLO

Po wytrenowaniu algorytmu na pierwotnych danych i ustaleniu najlepszych wartości dla podstawowych hiperparamterów (krok uczenia, momentum, weight decay) optymalizacji został poddany parametr "label smoothing", który odpowiada za wygładzenie etykiet i poprawę generalizacji modelu. Przeprowadzono 4 eksperymenty w których wartość parametry mieściła się w przedziale od 0 do 0,1. Uzyskane wyniki zaprezentowane są w tabeli 4.4.

Wartość średniej precyzji dla jest największa dla wartości 0,05 co świadczy o najlepszym dopasowaniu do danego problemu detekcji.

Tabela 4.4. Wyniki eksperymentów optymalizacji "label smoothing"

Parametr "label smoothing"	mAP	F1-score
0	0,863	0,81
0,01	0,865	0,8
0,05	0,878	0,81
0,1	0,855	0,81

Tabela 4.5. Wyniki "self-paced learning"

Rodzaj danych użytych podczas uczenia	Doszkalane na całym zbiorze	mAP	F1-Score
Dane łatwe, średnie i trudne	Nie	0,842	0,83
Dane łatwe, średnie i trudne	Tak	0,880	0,82
Dane trudne	Nie	0,792	0,78
Dane trudne	Tak	0,876	0,82

Tabela 4.6. Wyniki kontrolne (bez użycia detekcji kolorów)

F1-score	Precyza	Recall
0,811	0,819	0,803

4.3. Self-paced learning

Sposób podawania danych może mieć kluczowe znaczenie dla optymalnego sposobu uczenia. Przetestowano dwa sposoby w tym:

- Uczenie od najłatwiejszych do najtrudniejszych danych
- Uczenie tylko na trudnych danych

Ponadto zbadano wpływ trenowania algorytmu na pełnym zbiorze przed ostateczną ewaluacją i uwzględniono jej wpływ. Uzyskane wyniki przedstawiono w tabeli 4.5.

4.4. Wykrywanie koloru kwiatów

Dwoma głównymi parametrami, które wymagają dopasowania w detekcji kwiatów ogórków są:

- Dolny próg pewności detekcji dla obiektów, które przejdą powtórna weryfikację
- Odległość od "bounding box" w otoczeniu której można znajdować się kwiat.

Wyniki kontrolne oraz te uzyskane podczas próby optymalizacji tych dwóch parametrów przedstawiają tabele odpowiednio 4.6 i 4.7.

4.5. Omówienie wyników

Wyniki uzyskane przez sieć przedtreningową świadczą o niedokładnym dopasowaniu danych treningowych do modelu. Świadczy to o trudności problemu klasyfikacji. Dane użyte w modelu są skalowane do jednego wymiaru co wpływa na ich ostrość oraz skalę. W celu uzyskania lepszych rezultatów sieci przedtreningowej należałoby użyć bardziej wyszukanych metod optymalizacji i generalizacji lub wy-trenować model przy użyciu większej ilości danych. Sama sieć przedtreningowa, mimo nie do końca

Tabela 4.7. Optymalizacja parametrów pewności detekcji i zasięgu otoczenia

Pewność detekcji	Otoczenie[%]	F1-score	Precyza	Recall
0,1	0	0,806	0,792	0,821
0,05	0	0,805	0,770	0,843
0,1	5	0,791	0,761	0,824
0,05	5	0,792	0,744	0,847

zadawalających metryk, spełniła swoje zadanie i uaktualniła zbiór danych o ponad 500 nowych etykiet z czego około 95% z nich zostało przyporządkowanych poprawnie. Nowy zbiór uzyskał lepsze rezultaty podczas uczenia modelu YOLOv7 i był w stanie rozpoznawać więcej obiektów poprawnie, mimo obecności źle zaklasyfikowanych obiektów w "ground truth". Potwierdza to, że jakość zbioru uczącego jest podstawą dobrego treningu podczas uczenia głębokich sieci neuronowych, a ponadto pokazuje, że brak obiektów w zbiorze danych, które powinny się tam znaleźć, może być bardziej szkodliwy dla powodzenia uczenia, niż obecność w nim kilku błędnie zaklasyfikowanych obiektów.

Optymalizacja "label smoothing" poprawiła jakość modelu z uwagi na marginalizację przypadków, które są łatwe do detekcji dla algorytmu i zwiększenie nacisku na obiekty, które nie chcą się poprawnie dopasować.

Sposób podawania danych również okazał się kluczowy dla metryk detekcji obiektów. Kluczem do zwiększonej detekcji okazało się wstępne kontrolowane podawanie danych ze względu na poziom trudności, a następnie doszkolenie modelu na całym dostępnym zbiorze. Model uczony wyłącznie na trudnych przykładach, lub przypadkach o progresywnie rosnącej trudności, tracił zdolność do detekcji niektórych prostych przypadków. Może to wynikać z tego, że przypadki zaklasyfikowane przez model jako trudne do detekcji mogą się znacznie różnić pod względem tekstury, koloru, oświetlenia czy rozmycia od przypadków łatwych. W ten sposób, w czasie treningu przypadków wyłącznie trudnych, część informacji może być tracona. Doskalanie na całym zbiorze danych może zmniejszać dokładność detekcji trudnych przypadków, ale zwiększa dokładność przypadków średnich i łatwych co w rozrachunku, zwiększa dokładność całego modelu.

Detekcja koloru żółtego okazała się nie polepszać metryk detekcji obiektów. Głównym problemem algorytmu jest fakt, że niektóre kwiaty wysegmentowane przez algorytm nie były złączone z ogórkami, ponieważ odpadły od nich. W tych przypadkach wszelaka detekcja kończyła się zaznaczeniem obiektów, które nie były ogórkami. Innym problemem było to, że nawet w momencie kiedy kwiat był częścią ogórków, to algorytm YOLO nie rozpoznawał go z dostatecznie dużą pewnością i wszelkie detekcji, również dołączły obiektu niebędące ogórkami. Jednym z kolejnych problemów jest detekcja kwiatów na podstawie koloru, który w zależności od oświetlenia może się różnić i przyjmować inne barwy. Sprawiło to, że nie wszystkie kwiaty zostały poprawnie wysegmentowane, a ponadto, niektóre elementy otoczenia

(łodygi, części liści) również były segmentowane. Rozwiązaniem tych problemów może być wprowadzenie do nauki YOLO 2 klas, ogórków i ogórków z żółtym kwiatem. W ten sposób algorytm pozbędzie się problemu nietrafnej segmentacji, segmentacji błędnych instancji oraz rozłączności kwiatów i ogórków.

5. Podsumowanie

W powyższej pracy udało się zaprezentować algorytm oparty na YOLOv7, który działa lepiej niż bazowy algorytm, co jest równoznaczne ze zrealizowaniem zamierzonego celu. Udało się usprawnić sposób uczenia sieci YOLO poprzez optymalizację hiperparametrów i kolejność podawania danych, oraz uaktualniono autorski zbiór danych o ponad 500 etykiet, które zostały pominięte podczas pierwotnego etykietowania.

Algorytm YOLOv7 jest świetnym narzędziem do detekcji obiektów i bardzo dobrze sprawdza się w przypadku detekcji owoców i warzyw, w tym ogórków. Przeprowadzone badania pokazały, że uzyskanie maksymalnej skuteczności detekcji wymaga dostosowania algorytmu i jego sposobu uczenia do konkretnego problemu. Zaproponowane przez autora rozwiązanie wykazuje lepszą średnią precyzję detekcji niż bazowy algorytm YOLO, a długość jego wykonywania nie przekracza 50 ms. W ten sposób udało się zrealizować dwa z trzech pierwotnych celów. Nie udało się poprawić detekcji ogórków z żółtym kwiatem z uwagi na algorytm, którego działanie nie zostało poprawnie zoptymalizowane. Przyczyny takiego stanu rzeczy przedstawione są we wnioskach w sekcji 4.5.

Niemniej jednak, zaproponowane rozwiązanie nadaje się do wstępnej implementacji i testów automatycznej maszyny do zbioru ogórków. Zbiór danych wymaga uaktualnienia o inne najpopularniejsze odmiany ogórków, żeby algorytm zwiększył swoje możliwości generalizacji i nie wymagał doszkalań podczas detekcji nowych odmian. Kolejnym krokiem mogłoby być uaktualnienie zbioru o zdjęcia zrobione w nocy przy sztucznym oświetleniu. W ten sposób zapewnialibyśmy możliwość jego użytku również w nocy, co mogłoby znacząco podnieść wydajność rozwiązania w ujęciu praktycznym. Wymagane będą również dalsze badania mające na celu poprawę detekcji małych ogórków oraz ogórków z kwiatem żółtym, ponieważ, mimo że nie powinny być one zbierane, to pomijanie ich podczas detekcji mogłoby znacząco wpływać na działanie algorytmu z uwagi na znaczne podobieństwo ich tekstury i koloru do większych ogórków.

Bibliografia

- [1] Nawab Khan i in. „Current Progress and Future Prospects of Agriculture Technology: Gateway to Sustainable Agriculture”. W: *Sustainability* 13.9 (2021). ISSN: 2071-1050. DOI: [10.3390/su13094883](https://doi.org/10.3390/su13094883).
- [2] Aysel Elik i in. „Strategies to reduce post-harvest losses for fruits and vegetables”. W: *Strategies* 5.3 (2019), s. 29–39.
- [3] Eduardo Navas i in. „Soft Grippers for Automatic Crop Harvesting: A Review”. W: *Sensors* 21.8 (2021). ISSN: 1424-8220. DOI: [10.3390/s21082689](https://doi.org/10.3390/s21082689).
- [4] *W jaki sposób zbiera się winogrona i dlaczego warto robić to ręcznie?* <https://www.viatempia.pl/w-jaki-sposob-zbiera-sie-winogrona-i-dlaczego-warto-robić-to-recznie/>. Accessed: 2024-01-04.
- [5] Guangyu Hou i in. „An Overview of the Application of Machine Vision in Recognition and Localization of Fruit and Vegetable Harvesting Robots”. W: *Agriculture* 13.9 (2023). ISSN: 2077-0472. DOI: [10.3390/agriculture13091814](https://doi.org/10.3390/agriculture13091814).
- [6] Stanford University. *SQ2. What are the most important advances in AI?* 2021. URL: <https://ai100.stanford.edu/gathering-strength-gathering-storms-one-hundred-year-study-artificial-intelligence-ai100-2021-1/sq2#:~:text=In%20the%20last%20five%20years, and%20integration%20of%20vision%20and>.
- [7] Robin Taylor, David Baron i Daniel Schmidt. *The world in 2025 - predictions for the next ten years.* 2015, s. 192–195. DOI: [10.1109/IMPACT.2015.7365193](https://doi.org/10.1109/IMPACT.2015.7365193).
- [8] Ya Xiong i in. „An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation”. W: *Journal of Field Robotics* 37.2 (2020), s. 202–224.
- [9] Roemi Fernández i in. „Automatic Detection of Field-Grown Cucumbers for Robotic Harvesting”. W: *IEEE Access* 6 (2018), s. 35512–35527. DOI: [10.1109/ACCESS.2018.2851376](https://doi.org/10.1109/ACCESS.2018.2851376).
- [10] Joseph Redmon i in. *You Only Look Once: Unified, Real-Time Object Detection.* 2016. arXiv: [1506.02640 \[cs.CV\]](https://arxiv.org/abs/1506.02640).
- [11] Alex Krizhevsky, Ilya Sutskever i Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks”. W: *25* (2012). Red. F. Pereira i in.

- [12] Jia Deng i in. „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).
- [13] Robert Mansel Gower i in. „SGD: General analysis and improved rates”. W: *International conference on machine learning*. PMLR. 2019, s. 5200–5209.
- [14] Kaiming He i in. *Deep Residual Learning for Image Recognition*. 2015. arXiv: [1512.03385 \[cs.CV\]](https://arxiv.org/abs/1512.03385).
- [15] Joseph Redmon i in. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: [1506.02640 \[cs.CV\]](https://arxiv.org/abs/1506.02640).
- [16] Tausif Diwan, G Anirudh i Jitendra V Tembhere. „Object detection using YOLO: Challenges, architectural successors, datasets and applications”. W: *multimedia Tools and Applications* 82.6 (2023), s. 9243–9275.
- [17] Tsung-Yi Lin i in. „Microsoft coco: Common objects in context”. W: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*. Springer. 2014, s. 740–755.
- [18] Chien-Yao Wang, Alexey Bochkovskiy i Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: [2207.02696 \[cs.CV\]](https://arxiv.org/abs/2207.02696).
- [19] Chuyi Li i in. *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. 2022. arXiv: [2209.02976 \[cs.CV\]](https://arxiv.org/abs/2209.02976).
- [20] Joseph Redmon i Ali Farhadi. „Yolov3: An incremental improvement”. W: *arXiv preprint arXiv:1804.02767* (2018).
- [21] Alexey Bochkovskiy, Chien-Yao Wang i Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: [2004.10934 \[cs.CV\]](https://arxiv.org/abs/2004.10934).
- [22] Chien-Yao Wang, Hong-Yuan Mark Liao i I-Hau Yeh. „Designing network design strategies through gradient path analysis”. W: *arXiv preprint arXiv:2211.04800* (2022).
- [23] Feng Xiao i in. „Object Detection and Recognition Techniques Based on Digital Image Processing and Traditional Machine Learning for Fruit and Vegetable Harvesting Robots: An Overview and Review”. W: *Agronomy* 13.3 (2023), s. 639.
- [24] Anand Koirala i in. „Deep learning–Method overview and review of use for fruit detection and yield estimation”. W: *Computers and electronics in agriculture* 162 (2019), s. 219–234.
- [25] Wenbo Lan i in. „Pedestrian detection based on YOLO network model”. W: *2018 IEEE international conference on mechatronics and automation (ICMA)*. IEEE. 2018, s. 1547–1551.
- [26] Yuchen Liu i in. „M-YOLO: Traffic sign detection algorithm applicable to complex scenarios”. W: *Symmetry* 14.5 (2022), s. 952.

- [27] Rizwan Qureshi i in. „A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)”. W: *Authorea Preprints* (2023).
- [28] Olarewaju Lawal. „Tomato detection based on modified YOLOv3 framework”. W: *Scientific Reports* 11 (sty. 2021). doi: [10.1038/s41598-021-81216-5](https://doi.org/10.1038/s41598-021-81216-5).
- [29] Forrest Iandola i in. „Densenet: Implementing efficient convnet descriptor pyramids”. W: *arXiv preprint arXiv:1404.1869* (2014).
- [30] Rongli Gai, Na Chen i Hai Yuan. „A detection algorithm for cherry fruits based on the improved YOLO-v4 model”. W: *Neural Computing and Applications* 35.19 (2023), s. 13895–13906.
- [31] Dandan Wang i Dongjian He. „Channel pruned YOLO V5s-based deep learning approach for rapid and accurate apple fruitlet detection before fruit thinning”. W: *Biosystems Engineering* 210 (2021), s. 271–281. ISSN: 1537-5110. doi: <https://doi.org/10.1016/j.biosystemseng.2021.08.015>.
- [32] Yunhe Zhou i in. „Adaptive Active Positioning of Camellia oleifera Fruit Picking Points: Classical Image Processing and YOLOv7 Fusion Algorithm”. W: *Applied Sciences* 12.24 (2022). ISSN: 2076-3417. doi: [10.3390/app122412959](https://doi.org/10.3390/app122412959).
- [33] Guojin Li i in. „Lemon-YOLO: An efficient object detection method for lemons in the natural environment”. W: *IET Image Processing* 15.9 (2021), s. 1998–2009.
- [34] Anand Koirala i in. „Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of ‘MangoYOLO’”. W: *Precision Agriculture* 20 (2019), s. 1107–1135.
- [35] Lanhui Fu i in. „Fast and accurate detection of banana fruits in complex background orchards”. W: *IEEE Access* 8 (2020), s. 196835–196846.
- [36] Xiaojun Yu i in. „A lab-customized autonomous humanoid apple harvesting robot”. W: *Computers Electrical Engineering* 96 (2021), s. 107459. ISSN: 0045-7906. doi: <https://doi.org/10.1016/j.compeleceng.2021.107459>.
- [37] Lu Jiang i in. „Self-paced curriculum learning”. W: *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 29. 1. 2015.
- [38] Karen Simonyan i Andrew Zisserman. „Very deep convolutional networks for large-scale image recognition”. W: *arXiv preprint arXiv:1409.1556* (2014).
- [39] Mingxing Tan i Quoc Le. „Efficientnet: Rethinking model scaling for convolutional neural networks”. W: *International conference on machine learning*. PMLR. 2019, s. 6105–6114.