JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 34 (2025) 8-13
Received: 9 August 2024
Accepted: 11 November 2024

Analysis of object recognition systems using augmented reality glasses

Analiza systemów rozpoznawania obiektów przy wykorzystaniu okularów do rozszerzonej rzeczywistości

Jan Figura*, Rafał Kuźmiczuk, Marcin Badurowicz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
Object recognition systems and augmented reality devices aim to bridge the gap between the virtual and real worlds. It is natural, then, to combine these two technologies to create devices that can assist us in various aspects of life. This paper compares four object detection models: Faster R-CNN ResNet-101 v1, YOLO v8s, SSD MobileNet v2, and EfficientDet Lite 2 in the context of their use with augmented reality glasses. Using a simulated test environment, we examined resource consumption, energy efficiency, precision, and speed of the models during real-time operation. The results confirm that models using single-stage architecture are more suitable for real-time operation directly on the device. Among the single-stage models, YOLO v8s proved to be the most efficient.

Keywords: object detection; smart glasses; augmented reality; neural networks

Streszczenie
Systemy rozpoznawania obiektów i urządzenia rozszerzonej rzeczywistości łączą świat wirtualny z rzeczywistym. Naturalnym krokiem wydaje się więc połączenie obu tych technologii by tworzyć urządzenia wspierające różne aspekty życia. Niniejsza praca porównuje cztery modele wykrywania obiektów: Faster R-CNN ResNet-101 v1, YOLO v8s, SSD MobileNet v2 i EfficientDet Lite 2 z perspektywy zastosowania w okularach rozszerzonej rzeczywistości. Wykorzystując symulowane środowisko testowe, badane są zużycie zasobów, energooszczędność, precyzja i szybkość modeli podczas działania w czasie rzeczywistym. Uzyskane wyniki badań potwierdziły, że modele wykorzystujące architekturę jednoetapową są lepiej przystosowane do działania w czasie rzeczywistym, a najwydajniejszym okazał się YOLO v8s.

Słowa kluczowe: wykrywanie obiektów; inteligentne okulary; rzeczywistość rozszerzona; sieci neuronowe

*Corresponding author
Email address: jan.figura@pollub.edu.pl (J. Figura)

Published under Creative Common License (CC BY 4.0 Int.)

1. Wstęp

Wizja komputerowa jest dziedziną, która zaczęła się rozwijać już w latach sześćdziesiątych dwudziestego wieku, jednak dopiero w ostatniej dekadzie doświadczyła rewolucji za sprawą wykorzystania sieci neuronowych. W tym czasie powstało wiele różnych architektur i podejść do trenowania modeli zorientowanych na wykrywanie obiektów. Zadanie systemów do wykrywania obiektów opartych na konwolucyjnych sieciach neuronowych można podzielić na dwie części: odnajdywanie pozycji obiektu i jego kategoryzacja.

Wśród detektorów opartych na sieciach neuronowych najpopularniejszy jest podział na jednostopniowe i dwustopniowe modele wykrywania obiektów. To dwa powszechne podejścia stosowane w wizji komputerowej do identyfikacji i lokalizacji obiektów na obrazie [1-3].

Detektory jednoetapowe mają na celu bezpośrednie przewidywanie ramek ograniczających i etykiet klas dla obiektów w jednym przejściu przez sieć neuronową. Przykłady detektorów jednoetapowych obejmują modele takie jak YOLO (You Only Look Once) i SSD (Single Shot Multibox Detector). Modele te zazwyczaj składają się z pojedynczej sieci neuronowej, która jednocześnie przewiduje klasy obiektów i współrzędne obwiedni dla wszystkich obiektów na obrazie.

Zalety:

- Prostota: Detektory jednoetapowe są prostsze i szybsze, ponieważ wykonują przewidywania w jednym kroku,
- Operacja w czasie rzeczywistym: Ze względu na swoją wydajność, detektory jednoetapowe są często wykorzystywane do zastosowań w czasie rzeczywistym.

Wady:

- Niższa dokładność: Detektory jednoetapowe mogą mieć trudności z dokładną lokalizacją małych obiektów i mogą mieć niższą precyzję w porównaniu do detektorów dwustopniowych,
- Brak kontekstu: Przewidywania są dokonywane bez uwzględnienia kontekstu lub relacji między różnymi regionami obrazu.

Detektory dwuetapowe dzielą proces wykrywania obiektów na dwa etapy: propozycję regionu i klasyfikację. Pierwszy etap proponuje zestaw regionów, które mogą zawierać obiekty, a drugi etap klasyfikuje te regiony i udoskonala przewidywania obszarów. Przykłady dwuetapowych detektorów obejmują modele takie jak R-CNN (regionalna konwolucyjna sieć neuronowa), Fast R-CNN i Faster R-CNN. Modele te obejmują sieć propozycji regionów w pierwszym etapie, aby zasugerować regiony

kandydujące, które są następnie przetwarzane w celu klasyfikacji i udoskonalenia obszarów w drugim etapie.

Zalety:

- Wyższa dokładność: Dwuetapowe detektory często osiągają wyższą dokładność, zwłaszcza w scenariuszach z małymi obiektami lub złożonymi scenami,
- Lepsza lokalizacja: Dwuetapowy proces pozwala na lepsze zlokalizowanie przewidywań obwiedni.

Wady:

- Wolniejsze wnioskowanie: Modele dwustopniowe mogą być bardziej wymagające obliczeniowo i wolniejsze w porównaniu do modeli jednoetapowych,
- Złożoność: Architektura wieloetapowa może być bardziej złożona i trudniejsza do wytrenowania.

Okulary rozszerzonej rzeczywistości mogą wykorzystywać model do rozpoznawania obrazów, który znajduje się na serwerze lub bezpośrednio na urządzeniu mobilnym podłączonym do okularów. Na potrzeby niniejszych badań został wybrany wariant z instalacją bezpośrednio na urządzeniach podłączonych do okularów. Do najważniejszych zalet wykorzystania urządzeń mobilnych jako platformy zaliczają się:

- Krótki czas reakcji: Jeśli istnieje potrzeba szybkiej reakcji na obiekty w czasie rzeczywistym, instalacja modelu na urządzeniu mobilnym może być korzystniejsza, aby zminimalizować opóźnienia związane z przesyłaniem danych do serwera,
- Brak zależności od zasobów sieciowych: W przypadku, gdy nie ma połączenia internetowego lub jest ono zbyt słabe, urządzenia z modelami zainstalowanymi lokalnie mogą nadal wykonywać swoje zadania. Jest to szczególnie istotne w zastosowaniach, które wymagają dużej niezawodności,
- Bezpieczeństwo danych: W przypadku, gdy analiza obrazów obejmuje dane poufne, warto rozważyć, gdzie będą one przetwarzane. Dla urządzeń rozszerzonej rzeczywistości jest to bardzo istotne, ponieważ są one wykorzystywane przez użytkownika w wielu miejscach i mogą przesyłać do sieci obrazy o poufnych treściach, takie jak hasła, PIN-y, dane osobowe i inne wrażliwe dane.

Okulary rozszerzonej rzeczywistości zaliczają się do urządzeń z kategorii technologii mobilnej. Z tego powodu jednym z kryteriów wyboru było zoptymalizowanie do pracy na tychże urządzeniach. Dlatego do testów wybrano 3 modele, które są dostosowane do działania na urządzeniach mobilnych [4-5].

2. Cel, zakres pracy i hipotezy badawcze

Celem badań jest porównanie modeli do wykrywania obiektów w kontekście ich wykorzystania z okularami rozszerzonej rzeczywistości. Do przeprowadzenia badania wybrano 4 modele wykorzystujące najnowocześniejsze architektury do wykrywania obiektów:

1. Faster R-CNN ResNet-101 v1 [wersja 640x640],
2. YOLO v8 [wersja s],
3. SSD MobileNet v2 [wersja FPNLite 320x320],
4. EfficientDet Lite [wersja 2].

Architektura Faster R-CNN jest dwuetapowa. Oznacza to rozdzielenie sieci na dwa etapy: propozycję regionów i klasyfikację, w odróżnieniu od pozostałych modeli, które wykonują oba te zadania w ramach jednej sieci neuronowej. Badania biorą pod uwagę skuteczność, zużycie zasobów oraz szybkość działania wyżej przedstawionych modeli podczas pracy w symulowanym środowisku wykorzystującym okulary do rozszerzonej rzeczywistości.

Zakres pracy obejmuje:

- Przegląd literatury i zagadnień związanych z tematem pracy,
- Przygotowanie oprogramowania i środowiska badawczego pozwalającego na mierzenie metryk wydajności i skuteczności modeli,
- Przeprowadzenie badań i analiza wyników,
- Sformułowanie wniosków.

W ramach pracy sformułowano trzy hipotezy badawcze:
H1: *Modele wykorzystujące jednoetapową architekturę są szybsze w wykrywaniu obiektów.*
H2: *Modele wykorzystujące jednoetapową architekturę zużywają mniej zasobów.*
H3: *Modele wykorzystujące architekturę dwuetapową są dokładniejsze.*

3. Przegląd literatury

W ostatnich latach, wraz z rosnącymi możliwościami systemów mobilnych, pojawia się coraz więcej badań [6-9] dotyczących implementacji modeli na przenośnych systemach w celu stworzenia użytecznego środowiska rozszerzonej rzeczywistości (AR). Dynamiczny postęp w tej dziedzinie otwiera nowe perspektywy dla praktycznych zastosowań, obejmujących nie tylko sferę rozrywki, ale także przemysł, usługi oraz do użytku prywatnego. W tym rozdziale przyjrzeliśmy się pracom, które podejmują tematykę związaną z uruchomieniem modeli opartych na sieciach neuronowych na urządzeniach mobilnych.

W 2022 roku opublikowano artykuł [10], w którym przeprowadzono kompleksową analizę artykułów dotyczących detekcji obiektów z wykorzystaniem rozszerzonej rzeczywistości. Praca zawiera przegląd aktualnych technologii i urządzeń stosowanych w AR oraz często używanych algorytmów detekcji obiektów na przestrzeni ostatniej dekady. Podkreślane są różnice między głębokim uczeniem a klasycznymi klasyfikatorami statystycznymi, ukazując liczne zalety stosowania głębokiego uczenia. Badania pokazują wzrost liczby publikacji w tej dziedzinie. Autorzy podkreślają konieczność uwzględnienia wielu czynników, takich jak typ algorytmu, rozmiar modelu, warunki sieciowe i moc obliczeniowa urządzeń AR, przy implementacji obliczeń po stronie serwera lub lokalnie na urządzeniach.

Artykuł [11] z konferencji IEEE International Conference on Edge Computing and Communications przedstawia rozwiązanie umożliwiające uruchomienie modelu YOLOv8 w czasie rzeczywistym na urządzeniu Microsoft HoloLens. Model YOLOv8, w 5 różnych wersjach przygotowanych przez autorów, został przetestowany pod kątem możliwości działania w czasie rzeczywistym, będąc uruchomionym bezpośrednio na urządzeniu

noszonym przez użytkownika, bez potrzeby korzystania z zewnętrznego serwera w chmurze. Najwydajniejsza wersja modelu, osiągająca około 11 klatek na sekundę, to YOLOv8n. Autorzy pracy zauważają jednak, że w niektórych sytuacjach model YOLOv8s, który oferuje większą dokładność w wykrywaniu obiektów, może mieć wystarczającą wydajność dla komfortowej pracy. We wnioskach podkreślono również znaczenie zmniejszenia rozdzielczości obrazu wejściowego w celu przyspieszenia pracy modeli, a także ich kwantyzacji.

Kolejną pracą podejmującą temat uruchamiania modeli na okularach rozszerzonej rzeczywistości jest artykuł [12], w którym autorzy zaprojektowali własny prototyp okularów rozszerzonej rzeczywistości oraz zmodyfikowali architekturę YOLO, aby uzyskać wydłużony czas pracy na urządzeniach mobilnych, przy jednoczesnym zachowaniu zadowalającej wydajności. Autorzy pracy osiągnęli 18 klatek na sekundę przy poborze mocy na poziomie 62.9mW, co zapewnia około 9 godzin pracy na baterii zastosowanej w prototypie okularów. Zwracają uwagę, że mimo wysokiej wydajności modeli, ich dokładność jest niższa niż w przypadku bazowych architektur, na których są one oparte.

Badania [13] przeprowadzone w celu porównania dokładności i prędkości wykrywania architektur YOLO, SSD i Faster R-CNN pokazują, że YOLO jest modelem, który oferuje największą wydajność wykrywania dla dużych zbiorów danych, takich jak COCO, użytego w badaniu. Model SSD wypada pośrodku stawki, jeśli chodzi o dokładność wykrywania. Autorzy zwracają uwagę na problemy wydajnościowe tego modelu, związane z wykrywaniem obiektów o małych rozmiarach, spowodowane używaniem zbyt dużej rozdzielczości w warstwie odpowiedzialnej za ten aspekt. Najgorzej w porównaniu wydajnościowym wypadł model oparty na architekturze Faster R-CNN, choć miał on największą dokładność wśród testowanych modeli.

Autorzy kolejnej pracy [14] przygotowali aplikację na urządzenia mobilne z systemem iOS, wspierającą użytkownika w procesie składania mebli. Zadaniem było wyświetlanie modelu 3D aktualnie składanego mebla oraz identyfikowanie jego elementów za pomocą modelu YOLO. Badania skupiają się na doświadczeniu użytkowników w pracy z aplikacją. Ankiety przeprowadzone na badanych pokazują, że wykorzystanie modelu AI do wykrywania składanych elementów pomaga w pracy. Jednakże używanie tego rozwiązania na smartfonie odbiera niezbędną swobodę ruchu. Autorzy wskazują na możliwe usprawnienia aplikacji, takie jak przeniesienie jej na urządzenia noszone, poprawa wykrywania mniejszych przedmiotów oraz rozwiązanie problemu zacieniania podczas wizualizacji planów na ekranie. Ponadto praca wskazuje, że zadowalająca dla użytkowników płynność aplikacji osiągana jest przy około 20 klatkach na sekundę.

4. Metodyka badawcza

Aby porównać modele w kontekście ich wykorzystania na okularach rozszerzonej rzeczywistości, zaplanowano eksperyment polegający na umieszczeniu kamery internetowej przed ekranem telewizora. Na telewizorze były wyświetlane obrazy przedstawiające obiekty, które należą do klas zbioru danych COCO [15], na którym były trenowane wszystkie testowane modele. Obraz z kamery był wejściem dla każdego z modeli. Podczas wykrywania zostały zebrane informacje o szybkości wykrycia, dokładności oraz wykorzystaniu zasobów danej maszyny.

4.1. Przygotowanie oprogramowania do testowania modeli

Na potrzeby badań uproszczono dane wyjściowe każdego z modeli. Zastosowano algorytm znajdujący obiekt, który znajduję się najbliżej środka ekranu, a jednocześnie wynik klasyfikacji jest nie mniejszy niż 90% wyniku dla obiektu z największą pewnością. Program do testowania modeli został napisany w języku Python. Pozwala na testowanie wybranego modelu w 3 trybach:

1. Wydajność – Podczas wykrywania zbierane są informacje o zużyciu zasobów systemowych oraz szybkości wykrywania,
2. Dokładność – Podczas wykrywania zbierane są informacje o wykrytej klasie w centrum obrazu i pewności, z jaką klasa została wykryta,
3. Na żywo – Na wyświetlaczu okularów jest wyświetlana nazwa wykrytej klasy przed polem widzenia użytkownika.

4.2. Przygotowanie oprogramowania do zbierania informacji o zużyciu zasobów i dokładności modelu

W celu przeprowadzenia analizy porównawczej modeli określono metryki, które były mierzone w czasie wykonywania testów. W tabeli poniżej przedstawiono metryki zbierane podczas testów wydajności w celu określenia zużycia zasobów wykorzystywanych przez komputer podczas wykrywania obiektów.

Tabela 1: Metryki zbierane podczas testu wydajności

Metryka	Opis
Czas wykrycia (s)	Czas potrzebny na wykrycie obiektu
Użycie procesora (%)	Procentowe wykorzystanie procesora przez program
Użycie pamięci (MB)	Pamięć RAM zaalokowana przez program
Pobór mocy procesora (W)	Moc pobrana przez procesor podczas działania programu
Temperatura procesora (°C)	Temperatura procesora podczas działania programu

Podczas badań dokładności zbierano metryki opisujące zdolność modeli do skutecznego wykrywania obiektów z testowego zbioru danych.

Tabela 2: Metryki zbierane podczas testu dokładności

Metryka	Opis
FPS	Frames Per Second – częstotliwość odświeżania informacji o wykryciu
Czas wykrycia (s)	Czas potrzebny na wykrycie obiektu
Dokładność (%)	Procent wykrytych obiektów na zdjęciach, które zostały poprawnie zidentyfikowane.
Pewność	Pewność wykrytej klasy w skali od 0 do 1
Poprawność	Wartość prawda/fałsz wskazująca, czy poprawnie wykryto klasę obiektu znajdującego się na zdjęciu

5. Przeprowadzenie badań

Do przeprowadzenia badań wykorzystano dwa komputery stacjonarne oraz dwa laptopy. Laptopy posiadały mniejszą ilość pamięci RAM i procesory starszych generacji Intela niż nowsze procesory AMD zainstalowane w komputerach stacjonarnych. Dodatkowo testy zostały przeprowadzone na każdej maszynie dwukrotnie z wykorzystaniem systemu operacyjnego Windows oraz Linux.

Tabela 3: Specyfikacje urządzeń testowych

Nazwa urządzania	L1	L2	PC1	PC2
Systemy operacyjne	Linux, Windows			
Model procesora	Intel Core I3-4200H	Intel Core I5-6300U	AMD Ryzen 5 5600	AMD Ryzen 5 2600
Ilość rdzeni / wątków	2 / 4	2 / 4	6 / 12	6 / 12
Taktowanie	2,8 GHz	2,4 GHz	3,5 GHz	3,4 GHz
Pamięć RAM	8GB	8GB	16 GB	32 GB

6. Wyniki badań

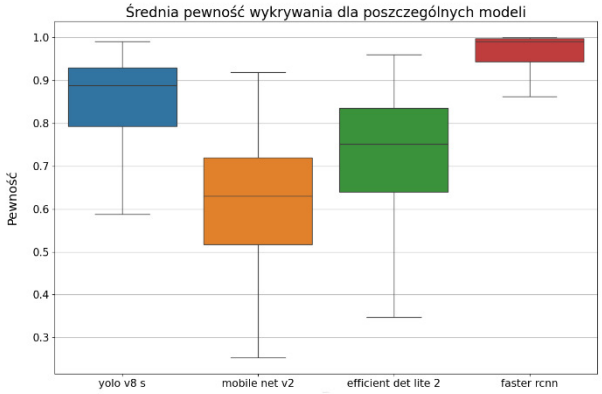
Podczas badań kamera ustawiona była w kierunku telewizora podłączonego do urządzenia, na którym wykonywany był test modeli. Dla każdego urządzenia wykonano testy wydajności i dokładności modeli. Przygotowano zbiór 240 zdjęć, po trzy zdjęcia dla każdej klasy ze zbioru danych COCO.

6.1. Wyniki badań dokładności

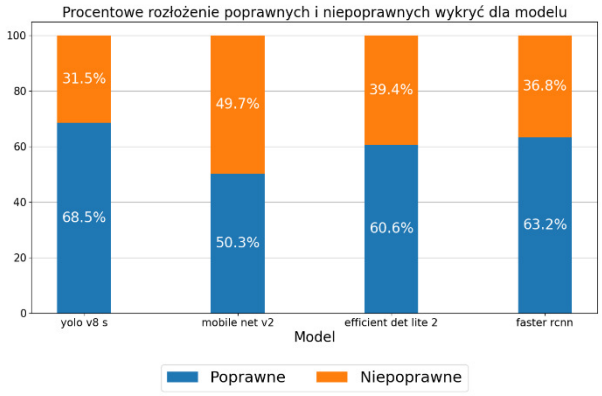
Po przeprowadzeniu badań dokładności modeli wyniki zostały uśrednione, aby uniezależnić je od konfiguracji maszyny i systemu, na którym działał program testowy. Tabela poniżej przedstawia wyniki dokładności wykrywania dla poszczególnych modeli.

Tabela 4: Podsumowanie wyników badań dokładności

Model	Yolo v8s	Faster R-CNN	SSD Mobile-Net v2	Efficient Det lite 2
Średni czas wykrycia (ms)	266 ms	2090 ms	170 ms	234 ms
Odchylenie standardowe czasu wykrycia (ms)	0,098 ms	0,865 ms	0,070 ms	0,080 ms
Średnia dokładność (%)	68,5%	63,2%	50,3%	60,0%
Średnia pewność	0,83	0,94	0,61	0,72
Odchylenie standardowe pewności	0,16	0,11	0,14	0,15



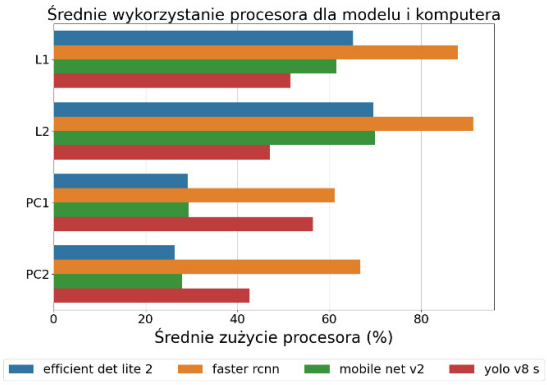
Rysunek 1: Wykres pudełkowy przedstawiający pewność wykrycia dla poszczególnych modeli.



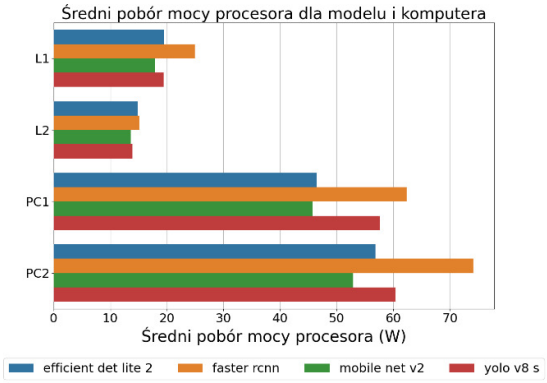
Rysunek 2: Wykres przedstawiający procentowe rozłożenie popranie i niepoprawnie sklasyfikowanych obiektów przez modele.

6.2. Wyniki badań wydajnościowych

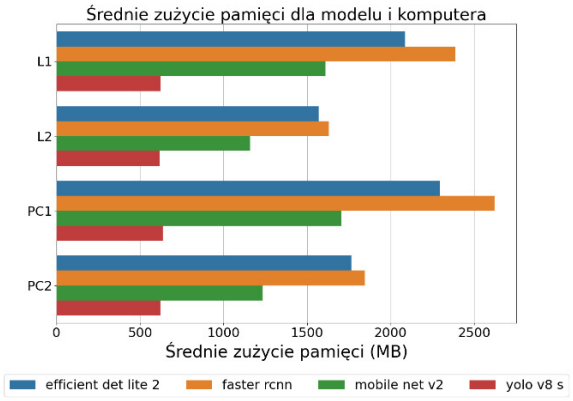
Celem badań wydajnościowych jest porównanie działania modeli na różnych maszynach. Ze względu na różnice w specyfikacji wyniki zostały przedstawione osobno dla każdej z testowanych maszyn.



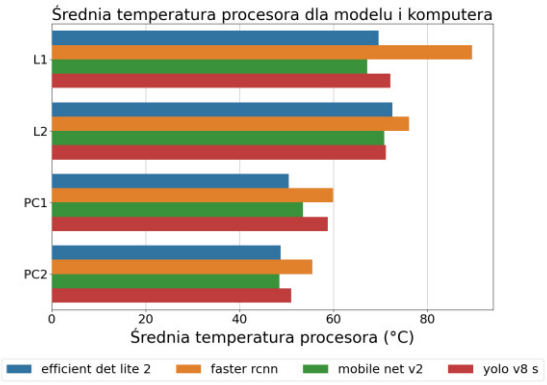
Rysunek 3: Wykres przedstawiający średnie zużycie procesora w procentach dla testowanych modeli.



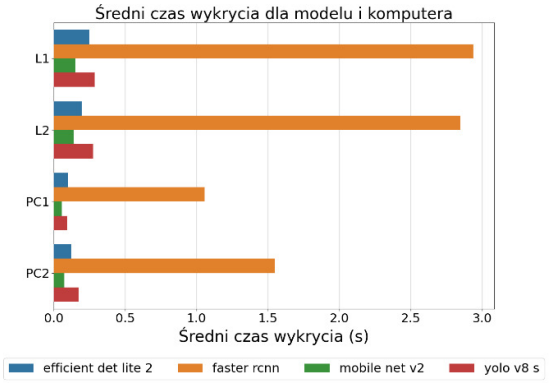
Rysunek 4: Wykres przedstawiający średni pobór mocy procesora w watach dla testowanych modeli.



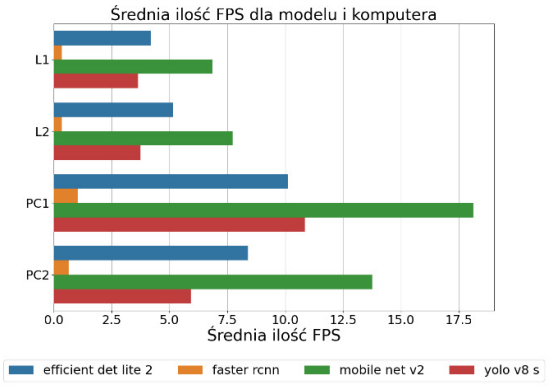
Rysunek 5: Wykres przedstawiający średnie zużycie pamięci w Megabajtach dla testowanych modeli.



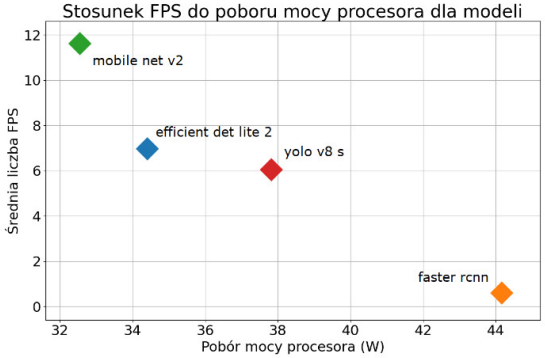
Rysunek 6: Wykres przedstawiający średnią temperaturę procesora w stopniach Celsjusza dla testowanych modeli.



Rysunek 7: Wykres przedstawiający średni czas wykrycia obiektu w sekundach dla testowanych modeli.



Rysunek 8: Wykres przedstawiający średnią ilość klatek na sekundę dla testowanych modeli.



Rysunek 9: Wykres przedstawiający średnią ilość klatek na sekundę w stosunku do poboru mocy procesora dla testowanych modeli.

7. Wnioski

Wyniki badań dokładności i wydajności pokazują znaczące różnice w zużyciu zasobów, szybkości i precyzji wykrywania obiektów przez testowane modele.

Wyniki z testów wydajności jasno potwierdziły słuszność hipotezy H1, pokazując, że modele oparte na detekcji jednoetapowej przewyższają szybkością modele oparte na architekturze dwuetapowej. Modele te miały szybsze czasy wykrywania obiektów, co zwiększało średnią liczbę klatek na sekundę. Przy wynikach w granicach 12 FPS ma to duże znaczenie dla wygody korzystania z interfejsu wykorzystującego detekcję obiektów. Jeśli chodzi o szybkość działania, najlepszym wyborem okazały się EfficientDet Lite 2 oraz SSD MobileNet v2.

Wykresy zużycia zasobów systemowych pokazują, że we wszystkich przypadkach najwięcej zasobów wykorzystywanych jest przez model Faster R-CNN w architekturze dwuetapowej, co potwierdza założenie hipotezy H2. Jeśli chodzi o wykorzystanie zasobów procesora, wszystkie modele jednoetapowe osiągają podobne wyniki z pojedynczymi wyjątkami. Natomiast w zakresie wykorzystania pamięci, model YOLO v8s zużywa jej średnio 2 razy mniej niż pozostałe modele, co pokazuje, że jest on dobrym wyborem, jeśli zależy nam na niskim zużyciu tego zasobu.

Największą dokładność wykrywania uzyskał model z rodziny YOLO, jednak największą pewność w wykrywaniu klas pokazał model z rodziny dwuetapowych detektorów Faster R-CNN i był w tej mierze przeważający nad pozostałymi modelami. Niestety, nie pozwala to przyjąć hipotezy H3, więc musi ona zostać odrzucona. Jeśli ważna jest dokładność wykrywanych obiektów, dobrymi wyborami będą więc YOLO lub Faster R-CNN.

Ostatecznie najlepszym wyborem w kontekście pracy na okularach rozszerzonej rzeczywistości są detektory w architekturze jednoetapowej. W szczególności model YOLO v8s jawi się jako dobry wybór, pokazując wysoką dokładność w detekcji obiektów, będąc jednocześnie oszczędnym w zużyciu zasobów, co ma znaczenie w przypadku pracy na baterii w terenie. Zachowuje on też odpowiednią prędkość wykrywania, dzięki czemu urządzenie może pracować w czasie rzeczywistym.

Literatura

[1] D. Garcia, J. Carias, T. Adão, R. Jesus, A. Cunha, L.G. Magalhães, Ten Years of Active Learning Techniques and Object Detection: A Systematic Review, Applied Sciences 13(19) (2023) 10667-10696, https://doi.org/10.3390/app31910667.

[2] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, J. García-Gutiérrez, On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data, Remote Sensing 13(1) (2021) 89-112, https://doi.org/10.3390/rs13010089.

[3] J. Brownlee, A Gentle Introduction to Object Recognition With Deep Learning, https://machinelearningmastery.com/object-recognition-with-deep-learning/, [02.10.2024].

[4] R. Gandhi, R-CNN, Fast R-CNN, Faster R-CNN, YOLO Object Detection Algorithms, Towards Data Science, https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e, [02.10.2024].

[5] How single-shot detector (SSD) works? ArcGIS API for Python, https://developers.arcgis.com/python/guide/how-ssd-works/, [02.10.2024].

[6] N. Zendehdel, H. Chen, M. Leu, Real-time tool detection in smart manufacturing using You-Only-Look-Once (YOLO)v5, Manufacturing Letters 35 Supplement (2023) 1052-1059, https://doi.org/10.1016/j.mfglet.2023.08.062.

[7] J.C. Arbeláez, On object recognition for industrial augmented reality, PhD dissertation, Politecnico di Milano, Milan, 2018.

[8] J. Estrada, P. Sidike, Y. Xiaoli, N. Quamar, Deep-Learning-Incorporated Augmented Reality Application for Engineering Lab Training, Applied Sciences 12(10) (2022) 5159-5178, https://doi.org/10.3390/app12105159.

[9] M. Mukhiddinov, C. Jinsoo, Smart Glass System Using Deep Learning for the Blind and Visually Impaired, Electronics 10(22) (2021) 2756-2786, https://doi.org/10.3390/electronics10222756.

[10] Y. Ghasemi, H. Jeong, S.H. Choi, K. Park, J.Y. Lee, Deep learning-based object detection in augmented reality: A systematic review, Computers in Industry 139 (2022) 103661-103676, https://doi.org/10.1016/j.compind.2022.103661.

[11] M. Łysakowski, K. Żywanowski, A. Banaszczyk, M.R. Nowicki, P. Skrzypczyński, S.K. Tadeja, Real-time onboard object detection for augmented reality: Enhancing head-mounted display with yolov8, In 2023 IEEE International Conference on Edge Computing & Communications (EDGE), 364-371, https://doi.org/10.1109/EDGE60047.2023.00059.

[12] J. Moosmann, P. Bonazzi, Y. Li, S. Bian, P. Mayer, L. Benini, M. Magno. Ultra-efficient on-device object detection on ai-integrated smart glasses with tinyissimoyolo, arXiv digital preprint 2311.01057v2 (2023), https://arxiv.org/abs/2311.01057v2.

[13] S. Srivastava, A.V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, V. Pattabiraman, Comparative analysis of deep learning image detection algorithms, Journal of Big Data 8(1) (2021) 66-93, https://doi.org/10.1186/s40537-021-00434-w.

[14] J. Atlas, J. Svensson, Object Recognition in Augmented Reality, Master thesis, Lund University, Lund, 2018.

[15] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. Zitnick, Microsoft COCO: Common Objects in Context, In Computer Vision - ECCV (8693) (2014) 740-755, https://doi.org/10.1007/978-3-319-10602-1_48.