UNIWERSYTET WARMIŃSKO-MAZURSKI
WYDZIAŁ MATEMATYKI I INFORMATYKI

DOKUMENTACJA
«System rozpoznawania znaków drogowych»

Studenci grupy ISI:
Maciej Kała – lider zespołu
Alina Upyrowa
Ruslan Zhukotynskyi
Mateusz Kruszewski
Eryk Żabiński

Prowadzący: dr inż. Krzysztof Ropiak
Firma partnerska: Visimind

Olsztyn 2025

Spis treści

1 OPIS PROJEKTU

Celem systemu jest stworzenie nowoczesnej, intuicyjnej aplikacji mobilnej, umożliwiającej automatyczne wykrywanie i klasyfikację znaków drogowych przy wykorzystaniu zaawansowanych technik sztucznej inteligencji. Aplikacja obsługuje zdjęcia wykonywane aparatem oraz obrazy wybierane z galerii, zapewniając szybki i czytelny rezultat analizy.

Sercem aplikacji są specjalnie wytrenowane modele YOLOv8s Sign Detection, pierwszy trzymany na serwerze, pozwalając na lepszą detekcję dla wgranych zdjęć, drugi zoptymalizowany do działania w środowisku mobilnym dzięki konwersji do formatu TensorFlow Lite. Podejście to pozwala osiągnąć wysoką wydajność detekcji przy zachowaniu umiarkowanych wymagań sprzętowych, co czyni aplikację dostępną również dla urządzeń średniej klasy.

Interfejs został zaprojektowany z myślą o prostocie i przejrzystości – użytkownik po wybraniu zdjęcia otrzymuje wynik detekcji w czasie poniżej jednej sekundy. Rozpoznane znaki prezentowane są w formie ramek obiektowych z opisem klasy, co pozwala na szybkie zidentyfikowanie oznaczeń drogowych i lepsze ich zrozumienie.

1.1 Główne funkcje aplikacji

Wykrywanie znaków na zdjęciach – użytkownik może wybrać obraz z galerii, a aplikacja w ciągu poniżej 1 sekundy prezentuje wynik z ramkami oraz opisem wykrytych znaków drogowych.

Podstawowa obsługa klas znaków – aplikacja rozpoznaje najczęstsze znaki ostrzegawcze, zakazu i informacyjne zgodne z konwencją wiedeńską – np. STOP, przejście dla pieszych, ograniczenie prędkości.

Model wykrywania osadzony w aplikacji – aplikacja korzysta z wbudowanej wersji modelu YOLOv8s, TensorFlow Lite do wykrywania znaków w trybie wykrywania na żywo.

Model wykrywania przez serwer – aplikacja korzysta z serwerowej wersji modelu YOLOv8s, która wymaga połączenia z nim do działania, wgrywa na nie zdjęcie, odbierając informacje i bounding boxy nakładane przy detekcji na obraz.

1.2 Technologie

Aplikacja została zbudowana w oparciu o Ultralytics YOLOv8 (PyTorch 2.8), wykorzystując transfer learning na skonsolidowanym zbiorze GTSRB + BTSD + TT100K i 14 tys. własnych kadr. Model do wykrywania na żywo eksportowany jest do ONNX Runtime, a następnie konwertowany na TensorFlow Lite z delegatem NNAPI, co umożliwia uruchomienie na Androidzie z przyspieszeniem sprzętowym. Logikę i interfejs napisano w Kotlinie przy użyciu Jetpack Compose, Material 3, natomiast eksperymenty ML śledzone są w MLFlow. Pipeline danych obrazu korzysta z Albumentations do zaawansowanej augmentacji oraz z NumPy/OpenCV do operacji pomocniczych.

1.3 Bezpieczeństwo i zgodność

Aplikacja respektuje politykę Google Play dotyczącą minimalnych uprawnień – wymaga jedynie dostępu do kamery i pamięci – a funkcja wykrywania znaków drogowych działa na podstawie zdjęć wybranych z galerii urządzenia.

2 SŁOWNIK

YOLOv8 – (You Only Look Once, wersja 8) zaawansowana architektura sieci neuronowych do detekcji obiektów w czasie rzeczywistym. W projekcie użyto wersji YOLOv8s (small), zoptymalizowanej pod kątem urządzeń mobilnych.

TensorFlow Lite (TFLite) – lekkie środowisko inferencyjne dla modeli ML działające na urządzeniach mobilnych, pozwalające uruchamiać modele z minimalnym zużyciem zasobów.

ONNX (Open Neural Network Exchange) – otwarty format wymiany modeli uczenia maszynowego między różnymi frameworkami (np. PyTorch, TensorFlow).

Bounding box (ramka obiektowa) – prostokątna ramka wyznaczająca pozycję wykrytego obiektu na obrazie (w tym przypadku znaku drogowego).

mAP@50 / mAP@50:95 – średnia precyzja (mean Average Precision) mierząca dokładność detekcji. mAP@50 oznacza tolerancję 50% IoU (Intersection over Union), natomiast mAP@50:95 uwzględnia wiele progów IoU i jest bardziej rygorystyczna.

MLFlow – platforma do śledzenia eksperymentów ML, pozwalająca monitorować parametry treningowe, metryki oraz wyniki modeli.

Augmentacja danych – technika sztucznego zwiększania liczby danych treningowych poprzez modyfikacje obrazów (np. obrót, zmiana jasności, skalowanie).

NNAPI – Android Neural Networks API – interfejs przyspieszający działanie modeli ML na urządzeniach mobilnych z systemem Android.

Jetpack Compose – nowoczesny framework UI dla Androida, służący do deklaratywnego tworzenia interfejsów użytkownika w języku Kotlin.

Transfer learning – technika trenowania modelu, która wykorzystuje wcześniej nauczone cechy z innego zadania, przyspieszając i poprawiając skuteczność treningu nowego modelu.

Dark mode / Light mode – tryby interfejsu graficznego aplikacji: ciemny (oszczędzający wzrok i energię) oraz jasny (klasyczny, lepszy w świetle dziennym).

Albumentations – biblioteka służąca do zaawansowanej augmentacji obrazów i bounding boxów, zwiększająca różnorodność danych treningowych.

Real-time detection – wykrywanie obiektów na strumieniu obrazu z kamery w czasie rzeczywistym.

3 WYMAGANIA PROJEKTOWE

3.1 Wymagania funkcjonalne

Jako użytkownik, chcę aby aplikacja pozwalała na wykrywanie znaków w czasie rzeczywistym, rysując bounding boxy na kamerze i wyświetlając opis, abym mógł rozpoznawać znaki na żywo.

Jako użytkownik, chcę aby aplikacja pozwalała na wybranie zdjęcia znaku drogowego z galerii i natychmiastowe zobaczenie jego opisu oraz piktogramu, aby szybko upewnić się co do jego znaczenia.

Jako użytkownik, chcę móc przełączać tryb wyświetlania aplikacji między trybem dziennym i nocnym, aby zapewnić sobie lepszą widoczność i komfort korzystania w różnych warunkach oświetleniowych.

Jako użytkownik, chcę aby aplikacja posiadała przyjemny i nieskomplikowany interfejs z animacjami, żeby korzystanie z niej było intuicyjne i proste.

3.2 Wymagania niefunkcjonalne

Jako użytkownik aplikacji, chcę, aby system był kompatybilny z urządzeniami z Androidem 10 lub nowszym i posiadającymi co najmniej 4 GB RAM, abym mógł używać aplikacji na większości nowoczesnych smartfonów.

Jako kierowca, chcę, aby aplikacja utrzymywała płynność co najmniej 25 FPS w rozdzielczości 1080p i wykonywała detekcję jednej klatki w czasie nie dłuższym niż 80 ms, aby obraz były wyświetlane bez zauważalnych opóźnień.

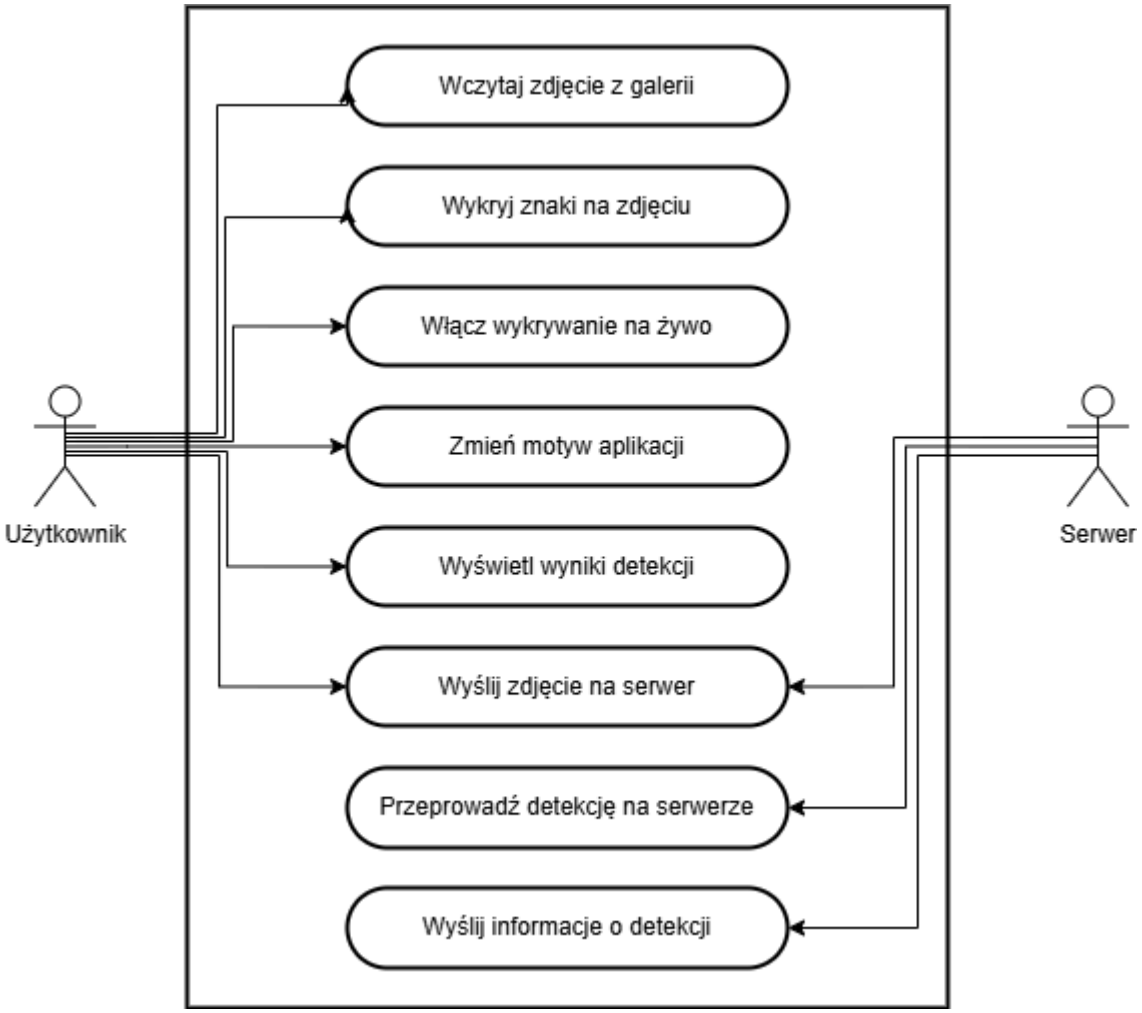
Jako właściciel produktu, chcę, aby model osiągał dokładność co najmniej mAP@50 = 0,95 oraz mAP@50:95 $\geq$ 0,75 na zbiorze testowym, aby ograniczyć liczbę fałszywych pozytywów i negatywów w pracy systemu.

Jako użytkownik, chcę, aby interfejs aplikacji był czytelny i prosty, a także umożliwiał komfortowe korzystanie z funkcji nawet przy ograniczonej widoczności – np. poprzez tryb nocny lub ciemny motyw.

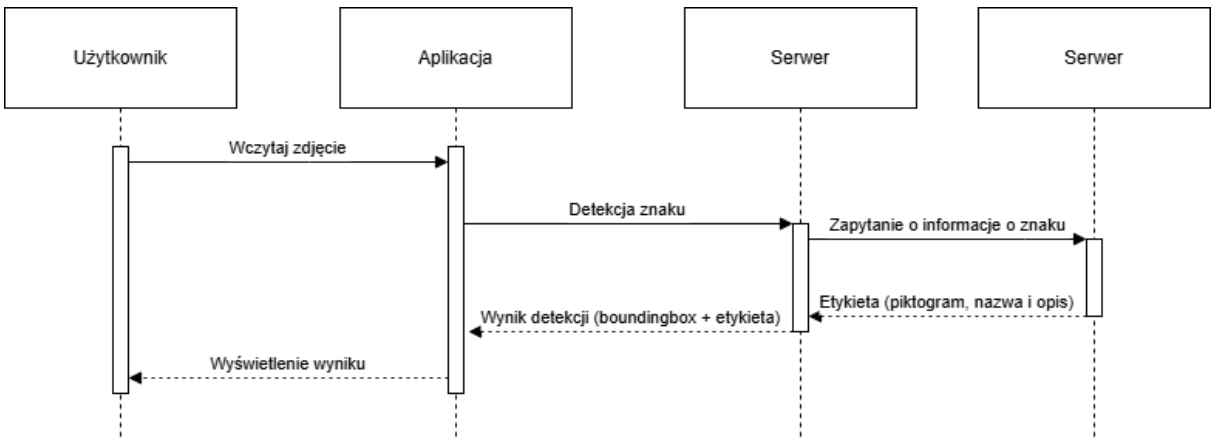
Jako tester, chcę, aby aplikacja działała stabilnie przez długi czas podczas przetwarzania zdjęć z galerii i analizy obrazu, bez nieoczekiwanych błędów lub zamknięcia, aby zapewnić użytkownikom niezawodność.

4 DIAGRAMY UML

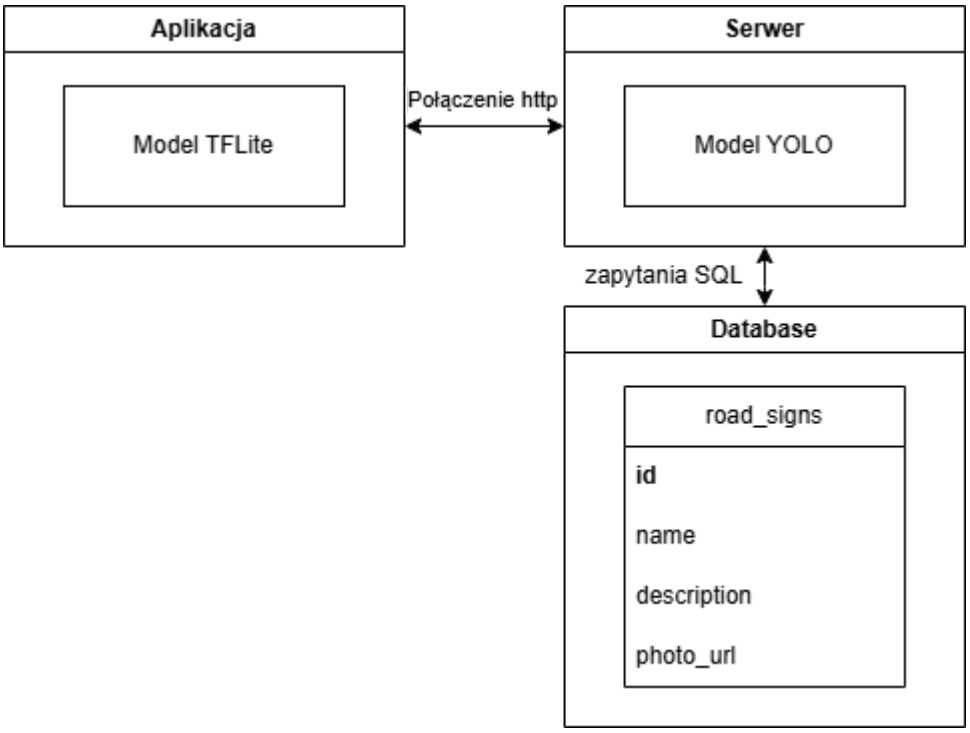
4.1 Diagram Użycia



4.2 Diagram Sekwencji dla obu sposobów detekcji

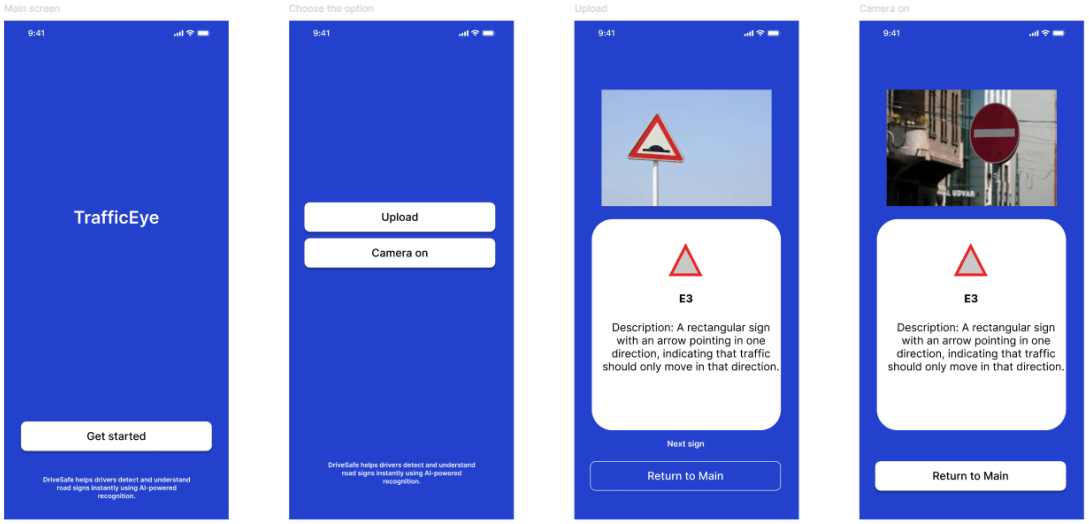


4.3 Struktura Projektu



5 IMPLEMENTACJA INTERFEJSU UŻYTKOWNIKA

Interfejs użytkownika aplikacji został zaprojektowany zgodnie z założeniami wymagań funkcjonalnych. Głównym celem było stworzenie środowiska intuicyjnego w obsłudze, estetycznego oraz wspierającego płynne i naturalne przechodzenie pomiędzy funkcjami aplikacji. Poniżej znajduje się zdjęcie mockupu UI.



Implementacja interfejsu została oparta na wytycznych Material Design 3 z wykorzystaniem Android Studio i biblioteki Jetpack. Zastosowano układy Constraint Layout w celu zapewnienia responsywności i poprawnego skalowania elementów na różnych urządzeniach. Zastąpiono komponenty zamiennikami Material 3, dając większą możliwość dostosowywania ich do potrzeb projektu. Główne widoki aplikacji obejmują ekran powitalny, panel detekcji znaków oraz sekcję ustawień. Użytkownik ma możliwość wyboru obrazu z galerii, uruchomienia detekcji na żywo oraz zmiany motywu aplikacji. Wszystkie przyciski zostały rozmieszczone logicznie, a ich etykiety są jednoznaczne. Wprowadzono również subtelne animacje tła, animacje przejścia między ekranami oraz efekt dotykowego feedbacku przy interakcji z elementami UI. Poniżej zdjęcie finalnej wersji UI ze wszystkimi ekranami.

6 SCENARIUSZE TESTÓW I REZULTATY

6.1 Test jednostkowy wykrycia pojedynczego znaku w Upload.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie „Get Started” 3. Kliknięcie „Upload” 4. Wybór pliku JPG z pojedynczym znakiem.	Prawidłowe dane: Na stronie Uploadu zostaje wyświetlony bounding-box wokół znaku. Na liście wyświetlony zostaje opis znaku wraz z jego piktogramem. Program rozpoznaje tylko jeden znak. Nieprawidłowe dane: Brak znaku lub niepoprawny format obrazu → brak wyświetlanego bounding-box’u na znaku, pusta lista wykrytych znaków.

6.2 Test jednostkowy wykrycia wielu różnych znaków w Upload.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie „Get Started” 3. Kliknięcie „Upload” 4. Wybór pliku JPG z wieloma różnymi znakami.	Prawidłowe dane: Na stronie Uploadu zostają wyświetlone bounding-box'y wokół znaków. Na liście wyświetlone zostają opisy znaków wraz z ich piktogramami. Program rozpoznaje każdy znak poprawnie. Nieprawidłowe dane: Brak znaku lub niepoprawny format obrazu → brak wyświetlanych bounding box'ów na znakach, pusta lista wykrytych znaków.

6.3 Test jednostkowy wykrycia wielu takich samych znaków w Upload.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie „Get Started” 3. Kliknięcie „Upload” 4. Wybór pliku JPG z takimi samymi znakami.	Prawidłowe dane: Na stronie Uploadu zostają wyświetlone bounding-box'y wokół znaków. Program rozpoznaje wiele znaków, ale na liście wyświetlony zostaje pojedynczy opis znaku wraz z jego piktogramem. Nieprawidłowe dane: Brak znaku lub niepoprawny format obrazu → brak wyświetlanych bounding boxów na znakach, pusta lista wykrytych znaków.

6.4 Test jednostkowy czasu wykrycia wielu znaków w Upload.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie „Get Started” 3. Kliknięcie „Upload” 4. Wybór pliku JPG z dużą ilością znaków.	Prawidłowe dane: Na stronie Uploadu zostają wyświetlone bounding-box'y wokół znaków, na liście wyświetlone zostają opisy znaków wraz z ich piktogramami, dobrze je rozpoznając. Czas oczekiwania na wyświetlenie się informacji wynosi mniej niż 1 sekundę.

6.5 Test wykrycia przez kamerę znaku drogowego.

Wykonywane kroki	Oczekiwany rezultat
1.Uruchomienie aplikacji i przyznanie uprawnień do kamery 2. Kliknięcie „Get Started” 3. Kliknięcie „Camera on” 3. Jazda testowa z widocznym znakiem drogowym w kadrze	Prawidłowe dane: W podglądzie kamera utrzymuje ~30 FPS, a każdy znak w kadrze jest otoczony ramką i etykietą w ≤ 80 ms od chwili pojawienia się w kadrze. Nieprawidłowe dane: Brak uprawnień do kamery → komunikat „Brak uprawnień do kamery”; lub bardzo słaba widoczność → brak wykryć i komunikat „Nie wykryto znaków.”

6.6 Test jednostkowy motywu Light.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie ikonki koła zębatego (opcje). 3. Wybranie z listy ustawień motywu Light. 4. Wyjście z aplikacji 5. Włączenie systemowego trybu nocnego. 6. Wejście do aplikacji w celu obserwacji zmian. 7. Wyjście z aplikacji 8. Wyłączenie systemowego trybu nocnego. 9. Wejście do aplikacji w celu obserwacji zmian.	Prawidłowe dane: Zmiana motywu aplikacji na dzienny, przeniesienie do ekranu początkowego. Pomimo obu zmian systemowego trybu nocnego, motyw zostaje taki sam.

6.7 Test jednostkowy motywu Dark.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie ikonki koła zębatego (opcje). 3. Wybranie z listy ustawień motywu Dark. 4. Wyjście z aplikacji 5. Włączenie systemowego trybu nocnego. 6. Wejście do aplikacji w celu obserwacji zmian. 7. Wyjście z aplikacji 8. Wyłączenie systemowego trybu nocnego. 9. Wejście do aplikacji w celu obserwacji zmian.	Prawidłowe dane: Zmiana motywu aplikacji na nocny, przeniesienie do ekranu początkowego. Pomimo obu zmian systemowego trybu nocnego, motyw zostaje taki sam.

6.8 Test jednostkowy motywu System default.

Wykonywane kroki	Oczekiwany rezultat
1. Uruchomienie aplikacji 2. Kliknięcie ikonki koła zębatego (opcje). 3. Wybranie z listy ustawień motywu System default. 4. Wyjście z aplikacji 5. Włączenie systemowego trybu nocnego. 6. Wejście do aplikacji w celu obserwacji zmian. 7. Wyjście z aplikacji 8. Wyłączenie systemowego trybu nocnego. 9. Wejście do aplikacji w celu obserwacji zmian.	Prawidłowe dane: Zmiana motywu aplikacji na systemowy, przeniesienie do ekranu początkowego. Przy zmianach systemowego trybu następują odpowiadające mu zmiany motywu aplikacji.

6.9 Wyniki testów

1. **Test jednostkowy wykrycia pojedynczego znaku w Upload.**
   Otrzymano oczekiwane rezultaty.

2. **Test jednostkowy wykrycia wielu różnych znaków w Upload.**
   Otrzymano oczekiwane rezultaty.

3. **Test jednostkowy wykrycia wielu takich samych znaków w Upload.**
   Otrzymano oczekiwane rezultaty.

4. **Test jednostkowy czasu wykrycia wielu znaków w Upload.**
   Otrzymano oczekiwane rezultaty.

5. **Test wykrycia przez kamerę znaku drogowego.**
   Wykrywane znaki zajmują więcej czasu niż oczekiwano do zostania wykrytym i poprawnie oznaczonym. Bounding boxy nakładane są czasem niepoprawnie, pojawiają się False Positives.

6. **Test jednostkowy motywu Light.**
   Otrzymano oczekiwane rezultaty.

7. **Test jednostkowy motywu Dark.**
   Otrzymano oczekiwane rezultaty.

8. **Test jednostkowy motywu System default.**
   Otrzymano oczekiwane rezultaty.

7 PODSUMOWANIE

Projekt „System rozpoznawania znaków drogowych”, oparty na modelu YOLOv8s, zakończył się niecałkowitym powodzeniem, osiągając zakładaną dokładność detekcji, jak i wymaganą wydajność obliczeniową w trybie Upload.Aplikacja mobilna została zintegrowana z serwerem backendowym, który umożliwia analizę obrazów przesyłanych przez użytkownika.