Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Rozpoznawanie znaków drogowych

Warsztaty z Technik Uczenia Maszynowego

Mateusz Chmurzyński
Mateusz Chudek
Marcin Skrzypczak
Jakub Stachyra
Jakub Winiarski

Wersja 1.0

7 czerwca 2024

Spis treści

1 Wstęp

Celem projektu było stworzenie aplikacji umożliwiającej rozpoznawanie znaków drogowych na zdjęciach, nagraniach wideo oraz w czasie rzeczywistym. Aplikacja została opracowana przy użyciu zaawansowanego algorytmu YOLOv8, który dzięki swojej szybkości i precyzji jest idealny do zastosowań wymagających natychmiastowej analizy obrazów. Projekt obejmował wykorzystanie odpowiednich zbiorów danych, trenowanie modeli sieci neuronowych oraz stworzenie interfejsu graficznego umożliwiającego użytkownikom łatwą i intuicyjną obsługę aplikacji.

2 Wykorzystany algorytm

W projekcie wykorzystano algorytm YOLOv8 (You Only Look Once) do detekcji obiektów. YOLO jest zaawansowanym modelem sieci neuronowych, który analizuje obraz w czasie rzeczywistym i identyfikuje różne obiekty w jednym przejściu. Jak sama nazwa wskazuje cała praca zamyka się jednym procesie, jednym przebiegu algorytmu. Dzięki swojej szybkości i precyzji, YOLO jest idealnym wyborem do zastosowań wymagających natychmiastowej analizy wideo lub obrazu.

Algorytm YOLO (You Only Look Once) działa poprzez jednoczesne przetwarzanie całego obrazu. Najpierw dzieli obraz na siatkę, a każda komórka siatki przewiduje, czy znajduje się w niej obiekt. Algorytm generuje ramki ograniczające dla potencjalnych obiektów, przypisując im etykiety klas i wartości zaufania. Następnie technika non-max suppression (NMS) eliminuje nadmiarowe ramki, pozostawiając tylko najbardziej wiarygodne. Dzięki temu YOLO jest szybki i precyzyjny, idealny do zastosowań w czasie rzeczywistym.

3 Zbiory danych

W projekcie wykorzystaliśmy dwa pozyskane zbiory danych [1][2]. Dane przygotowane do zadania detekcji zawierają większe zdjęcia z dołączonym plikiem tekstowym opisującym pozycję występującego na obrazie znaku. Zdjęcia, które posłużyły do trenowania modelu klasyfikującego zawierają zdjęcia, które przedstawiają jedynie tarczę znaku, posortowana w folderach opisanych kodem znaku. Przy użyciu napisane skryptu w Python podzieliliśmy zebrane dane na zbiory do uczenia, testowania oraz walidacji modeli. Dane zostały podzielone w proporcji odpowiednio 7:2:1.

4 Modele

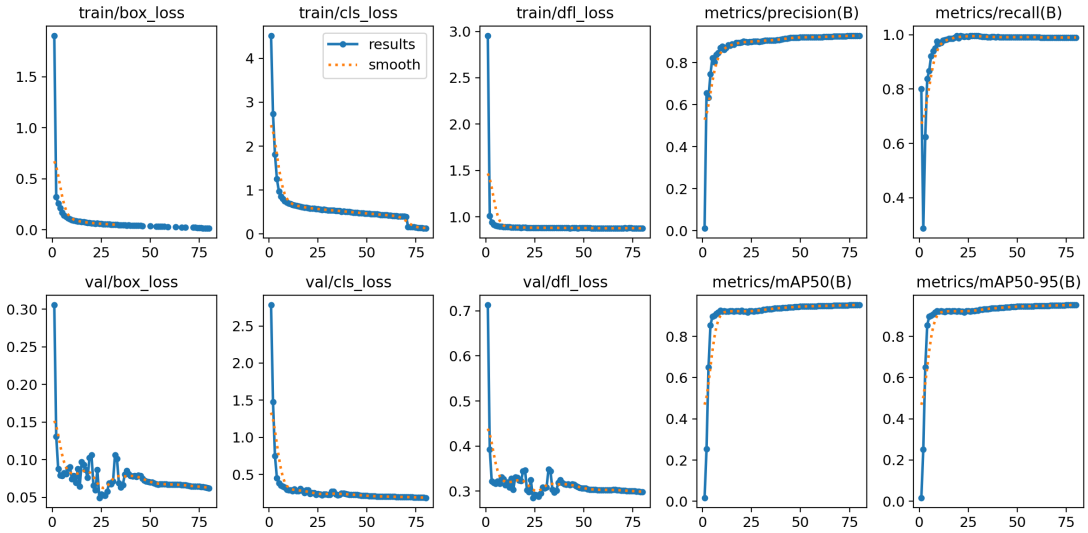
W celu poprawnego rozpoznawania znaków wykorzystaliśmy dwie sieci. Pierwsza z nich ma za zadanie wyekstrahować znak z obrazu, natomiast druga służy do klasyfikacji. Dzięki temu poprawność klasyfikacji znaków jest znacznie zwiększona, gdyż klasyfikacja przebiega tylko na fragmencie obrazu, w którym został wykryty i wyekstrahowany znak. Sieć do ekstrakcji znaków została wytrenowana przy użyciu 320 epok, natomiast sieć do klasyfikacji została wytrenowana w 80 epok. Takie wartości dały satysfakcjonujące efekty.

5 Ocena jakości modeli

Algorytm YOLO dostarcza informacje statystyczne na temat przeprowadzonego procesu trenowania. Do dyspozycji mamy wykresy:

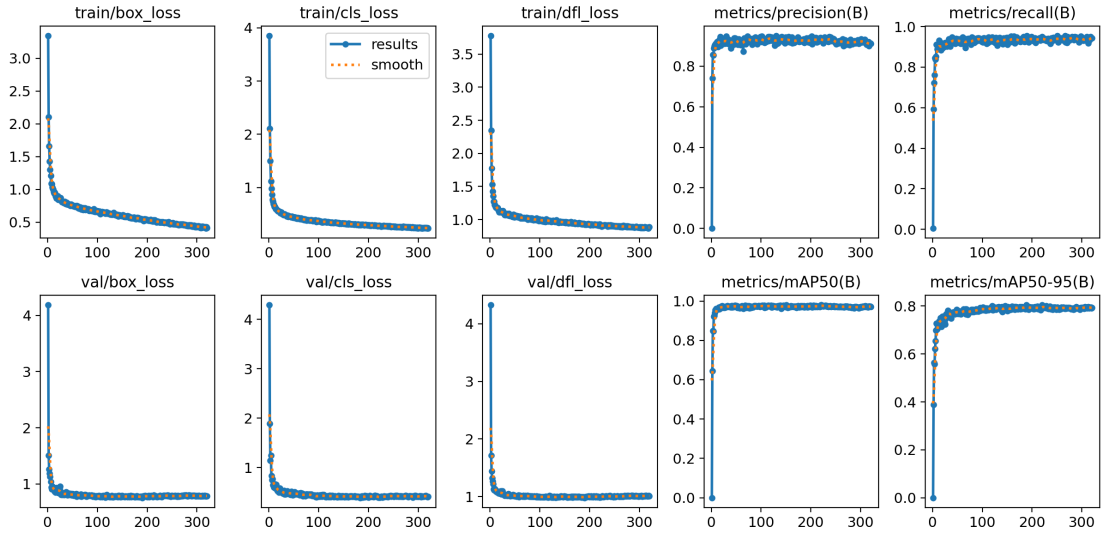
- box_loss - ten wykres pokazuje informacje związane z błędami w przewidywaniu położenia obiektów. Tworzony jest na podstawie różnicy między przewidywanymi a rzeczywistymi współrzędnymi ramki otaczającej każdy obiekt. Niższe wartości wskazują na to, że model dokładniej lokalizuje obiekty. YOLO zwraca wykres dla zbioru treningowego i walidacyjnego.
- cls_loss - wykres ten dotyczy klasyfikacji i pokazuje, jak dobrze model radzi sobie z przewidywaniem prawidłowej klasy dla wykrytych obiektów. Tworzony na podstawie różnica między przewidywanymi a rzeczywistymi klasami obiektów. Niższa wartość oznacza, że model lepiej klasyfikuje obiekty. YOLO zwraca wykres dla zbioru treningowego i walidacyjnego.
- dfl_loss - jest to zaawansowany wykres, który koncentruje się na lepszym modelowaniu rozkładów prawdopodobieństwa dla lokalizacji obiektów, co pomaga w precyzyjniejszym wyznaczaniu ramki. YOLO zwraca wykres dla zbioru treningowego i walidacyjnego.
- metrics/precision(B) - metryka, stosunek poprawnie wykrytych obiektów (prawdziwie pozytywne wyniki) do wszystkich obiektów, które model uznał za daną klasę (suma prawdziwie pozytywnych i fałszywie pozytywnych wyników). Wyższe wartości na wykresie są pożądane.
- metrics/recall(B) - metryka, czułość (recall) mierzy stosunek poprawnie wykrytych obiektów do wszystkich obiektów, które rzeczywiście należą do danej klasy (suma prawdziwie pozytywnych i fałszywie negatywnych wyników). Wysoka wartość czułości oznacza, że model efektywnie wykrywa obiekty danej klasy. Wykres czułości wskazuje, jak dobrze model wykrywa rzeczywiste obiekty w danych.
- metrics/mAP50(B) - metryka mAP (ang. mean Average Precision) przy IoU (ang. Intersection over Union) o progu 0.5. mAP50 oblicza średnią precyzję dla różnych poziomów progów detekcji dla IoU równego 0.5. IoU to miara nakładania się przewidywanej ramki na prawdziwą ramkę, gdzie 0.5 oznacza, że przewidywana ramka musi pokrywać przynajmniej 50% prawdziwej ramki.
- metrics/mAP50-95(B) - Ta metryka jest rozszerzeniem mAP50, gdzie mAP jest obliczane dla różnych progów IoU od 0.5 do 0.95 (zwykle z krokiem co 0.05). Jest to bardziej rygorystyczna i kompleksowa ocena modelu, ponieważ uwzględnia wydajność modelu przy różnych wymaganiach dotyczących precyzji lokalizacji obiektów. Wyższy wynik w tej metryce wskazuje na to, że model jest bardzo skuteczny w precyzyjnym lokalizowaniu obiektów przy różnych poziomach tolerancji błędów.

Prezentujemy wykresy dla modelu wykorzystanego do klasyfikacji



Rysunek 1: Wykresy strat i metryki dla modelu wykorzystanego do klasyfikacji

Wykresy strat (loss) wraz z kolejnymi epokami dążą do mniejszych wartości, co zasugerowało nam, że wytrenowany model będzie użyteczny. Również metryki dążące do większych wartości wskazują na dobrą precyzję modelu.



Rysunek 2: Wykresy strat i metryki dla modelu wykorzystanego do detekcji

Na podstawie tych wykresów można stwierdzić, że trenowanie modelu do detekcji znaków przebiegło wzorowo. Algorytm YOLO wyłączył trenowanie na 320 epoce, aby nie doszło do przetrenowania modelu.

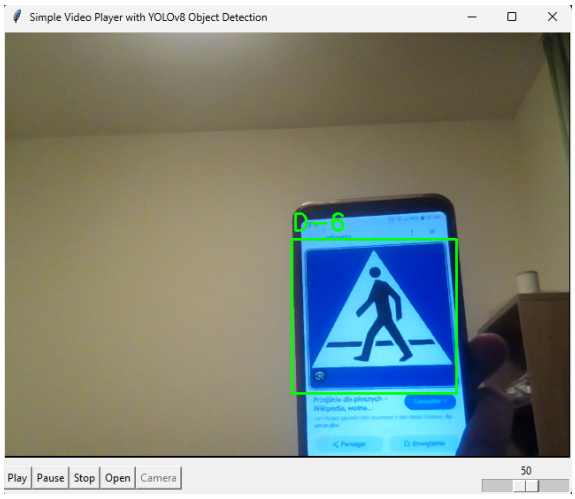
6 Aplikacja

W celu graficznej graficznej prezentacji uzyskanych wyników powstała aplikacja umożliwiająca rozpoznawanie znaków drogowych w kilku trybach:

- rozpoznawanie znaków w filmach,
- rozpoznawanie znaków na zdjęciach,
- rozpoznawanie znaków w czasie rzeczywistym (kamerka w laptopie).

Na rysunkach 3 i 4 przedstawiono zrzuty ekranu z działającej aplikacji

Rysunek 3: Przykład aplikacji rozpoznającej znaki na filmie pobranym z internetu



Rysunek 4: Przykład aplikacji rozpoznającej znaki w czasie rzeczywistym

GUI zostało stworzone w Pythonie za pomocą frameworku *tkinter*. Umożliwia on dodanie wszystkich potrzebnych funkcjonalności w prosty sposób, bez konieczności wystawiania osobnego API w celu przetwarzania obrazów przez sieć neuronową.

Obraz jest obrabiany przy pomocy frameworku *opencv*, umożliwiającego rysowanie prostokątów oraz oznaczeń klasyfikujących znaki. Dzięki temu użytkownik może w przejrzysty sposób zobaczyć na żywo efekty działania algorytmu YOLO.

7 Podsumowanie

Projekt dotyczący detekcji znaków drogowych z wykorzystaniem algorytmu YOLOv8 wykazał skuteczność tego zaawansowanego modelu w analizie obrazów w czasie rzeczywistym. YOLO, dzięki jednoprzebiegowemu przetwarzaniu obrazu, zapewnia szybkie i precyzyjne wykrywanie obiektów, co czyni go idealnym do zastosowań wymagających natychmiastowej analizy wideo lub obrazu.

W ramach projektu użyto odpowiednio przygotowanych zbiorów danych, które podzielono na zestawy treningowe, testowe i walidacyjne. Dwie sieci neuronowe zastosowane do ekstrakcji i klasyfikacji znaków drogowych przyczyniły się do zwiększenia precyzji rozpoznawania. Ocena jakości modeli oparta na metrykach takich jak precyzja, czułość i średnia precyzja (mAP) potwierdziła ich efektywność.

Stworzona aplikacja graficzna umożliwia rozpoznawanie znaków drogowych zarówno na zdjęciach, filmach jak i w czasie rzeczywistym za pomocą kamery. Dzięki zastosowaniu frameworku tkinter i opencv, aplikacja jest łatwa w obsłudze i pozwala na przejrzystą wizualizację wyników działania algorytmu YOLO. Projekt stanowi solidną podstawę do dalszych badań i rozwoju systemów wspomagania kierowców oraz autonomicznych pojazdów.

Podczas projektu nauczyliśmy się wielu nowych rzeczy. Zgłębiliśmy zasadę działania algorytmu YOLO oraz podstawy sieci neuronowych. Dla wszystkich było to również pierwsze zetknięcie z procesem trenowania sieci neuronowych. Zobaczyliśmy w praktyce jak długi jest to proces (oczywiście w zależności od sprzętu) oraz ile epok powinno trwać trenowanie. Całość udało się spiąć w ładny interfejs graficzny.

8 Bibliografia

[1] Kaggle. Dataset 1. https://www.kaggle.com/datasets/kasia12345/polish-traffic-signs-dataset. Dostęp: 2024-03-08.

[2] Kaggle. Dataset 2. https://www.kaggle.com/datasets/chriskjm/polish-traffic-signs-dataset. Dostęp: 2024-03-08.

[3] Ultralytics Inc. Ultralytics YOLOv8 docs. https://docs.ultralytics.com/. Dostęp: 2024-03-08.