Rozpoznawanie tablic rejestracyjnych z wykorzystaniem technik uczenia maszynowego

Jakub Dmitruk, Julita Kulesza

7 czerwca 2024

1 Wprowadzenie

Rozpoznawanie kształtów geometrycznych na obrazach jest kluczowym zagadnieniem w dziedzinie przetwarzania obrazu i widzenia komputerowego. Jednym z najczęściej analizowanych kształtów są prostokąty, które występują w wielu zastosowaniach, takich jak analiza obrazów satelitarnych, przetwarzanie dokumentów, systemy monitoringu czy robotyka. Automatyczne wykrywanie prostokątów może znacząco ułatwić analizę danych w formie obrazu, przyczyniając się do efektywniejszego i szybszego przetwarzania informacji.

Projekt, który właśnie ukończyliśmy, miał na celu stworzenie systemu zdolnego do rozpoznawania prostokątnych tablic na różnorodnych zdjęciach aut. Łączył on w sobie techniki przetwarzania obrazu z zaawansowanymi metodami uczenia maszynowego, aby zapewnić wysoką dokładność detekcji.

Warto podkreślić, że opracowany model wykrywania prostokątów może stanowić kluczowy element systemu automatycznego odczytywania tablic. W takim systemie model umożliwia precyzyjną lokalizację tablicy, co pozwala na dokładne odczytanie tekstu zawartego w wyznaczonym obszarze.

2 Projekt

2.1 Zakres prac

Zakres prac obejmował:

1. **Analizę dostępnych algorytmów uczenia maszynowego.** Istnieją klasyczne metody wykrywania kształtów, w tym różne detektory. Nasza uwaga skupiona była na nowocześniejszych metodach wykrywania kształtów, tj. sieciach neuronowych.

2. **Wybór odpowiednich technik przetwarzania obrazu.** W kontekście rozpoznawania prostokątów, kluczowym wyzwaniem było maksymalne wykorzystanie dostępnych danych treningowych. Zmuszała nas ona do świadomego zarządzania danymi i przemyślanej obróbki, aby osiągnąć jak najlepsze wyniki.

3. **Implementację wybranych metod w środowisku Python.** W tym procesie przydatne okazały się być biblioteki takie jak `Numpy`, `PIL`, `TensorFlow/Keras`.

4. **Testowanie algorytmu na zestawach danych obrazów zawierających prostokątne tablice.** W tym przypadku, ocena działania algorytmu była stosunkowo prosta i szybka, ponieważ wystarczyło zastosować go na kilku obrazach i wizualnie ocenić wyniki. Dzięki temu można łatwo było zidentyfikować ewentualne problemy i ewentualnie dokonać niezbędnych korekt (już nie tak łatwo).

2.2 Dane

Do budowy modelu wykorzystaliśmy dane typu *open-source*, które można znaleźć pod linkami www.kaggle.com/datasets/achrafkhazri/labeled-licence-plates-dataset oraz www.kaggle.com/datasets/andrewmvd/car-plate-detection.

2.3 Metodologia

Do realizacji projektu zastosowaliśmy następujące kroki:

Preprocessing obrazów: Każde zdjęcie zostało poddane przeskalowaniu do rozmiaru 224×224 przy użyciu filtru `Image.LANCZOS`, co daje wygładzenie obrazu. Ten krok był dość istotny, ponieważ większość modeli UM wymaga jednakowych rozmiarów wejściowych, a przeskalowanie do ustalonego rozmiaru pozwala uniknąć problemów z różnymi rozmiarami obrazów. Konieczne było też przekształcenie etykiet tak, aby były zgodne z nowym rozmiarem obrazu. Następnie, cały zbiór zdjęć został przekształcony do postaci `numpy.array`, a wartości pikseli zostały znormalizowane do zakresu od 0 do 1. Zdecydowaliśmy się też uzupełnić obrazy o szum gaussowski.

Wybór modelu (I): Używając podejścia **transfer learning**, zdecydowaliśmy się wykorzystać wcześniej wytrenowany model do zadań klasyfikacji obrazów. Miało to pozwolić na skrócenie czasu trenowania i poprawę wydajności.

Wybór modelu (II): Aby mieć punkt odniesienia, zdecydowaliśmy się też zbudować sieć neuronową od zera. Pozwoliło nam to również lepiej zrozumieć i kontrolować każdy aspekt architektury naszego modelu. Wszystko to mogło potencjalnie okazać się cenną lekcją przy dalszych modyfikacjach.

Architektura (I): Model który wykorzystaliśmy, `VGG16`, który jest gotowym modelem sieci neuronowej wytrenowanym na ogromnym zbiorze danych ImageNet. Wzbogaciliśmy gi o dwie dodatkowe warstwy, warstwę w pełni połączoną oraz warstwę wyjściową modelu, składającą się z czterech neuronów, co odpowiadało czterem rogom prostokąta.

Architektura (II): Nasza sieć neuronowa składała się z różnego typu warstw. Przykładowo, za pomocą warstwy *BatchNormalization* normalizowaliśmy wejścia do warstw, warstwa `MaxPooling2D` pozwoliła nam na redukcję rozmiaru danych w procesie, a warstwa `Dropout`, wyłączała losowo część neuronów, regularyzując model.

Trenowanie: Podczas trenowania modeli użyliśmy metryki IoU do oceny dopasowania przewidywanych obiektów do rzeczywistych na obrazie. W trenowaniu użyliśmy optymalizatora `Adam` ze stałą uczenia $0,0001$ i funkcji straty `MSE` celem zminimalizowania błędu między przewidywaniami sieci a rzeczywistymi etykietami. Ta kombinacja optymalizatora i funkcji straty jest powszechnie stosowana do treningu różnych typów sieci neuronowych. Na trenowanie finalnego modelu przeznaczyliśmy 50 epok, korzystając z danych walidacyjnych do oceny jakości modelu. Wybór 50 epok nie był przypadkowy, w naszym doświadczeniu wtedy model osiągnął punkt, w którym dalsze uczenie nie przynosiło już istotnych korzyści.

3 Wyniki

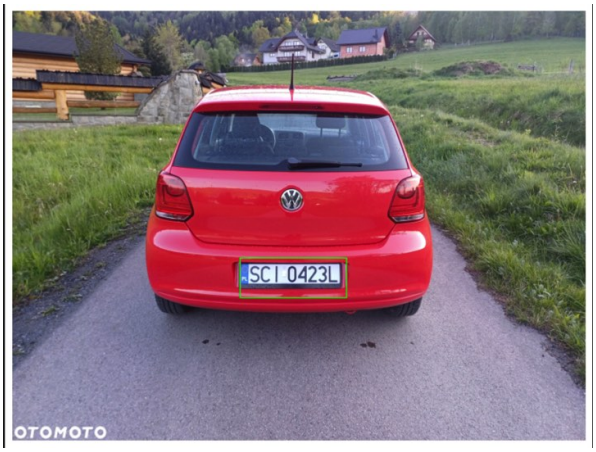
Średnie IoU na zbiorze walidacyjnym modelu I wyniosło 0.48, co jest dobrym wskaźnikiem skuteczności detekcji. Testy na danych testowych również potwierdziły, że nasz algorytm działa zadowalająco. Poniżej prezentujemy kilka przykładów (1,2,3), aby lepiej zilustrować działanie naszego modelu.

4 Wnioski

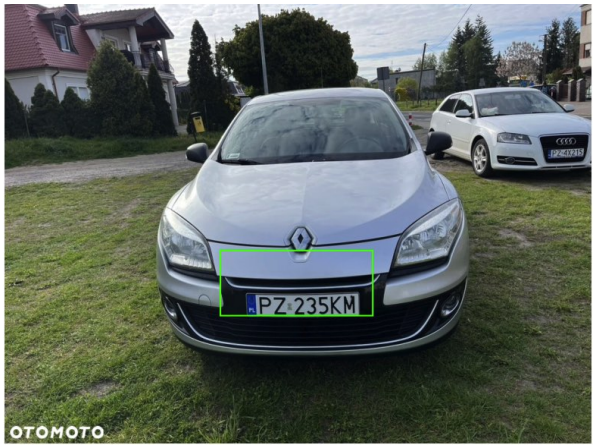
Analiza naszych wyników sugeruje, że zdecydowanie istnieje pole do poprawy naszego modelu; mimo że uzyskane wyniki na zbiorze testowym były przyzwoite, średnie IoU dla zbioru walidacyjego mogłoby wynosić więcej. Możliwe dalsze kierunki rozwoju obejmują zmianę architektury modelu, optymalizację hiperparametrów, czy zwiększenie rozmiaru zbioru treningowego.

Celowo podjęliśmy próbę budowy własnej sieci neuronowej od podstaw równolegle z wykorzystaniem modelu VGG16, jednak nasze doświadczenia sugerują, że konkurencja z gotowymi architekturami, takimi jak VGG16, jest trudna. Gotowe modele, jak VGG16, zapewniają solidne wyniki już na starcie i są trudne do zrównania pod względem skuteczności. W związku z tym, choć nasze wysiłki były cenne, nadal pozostaje nam wiele do nauki i doskonalenia w kontekście budowy efektywnych modeli wykrywania prostokątów.

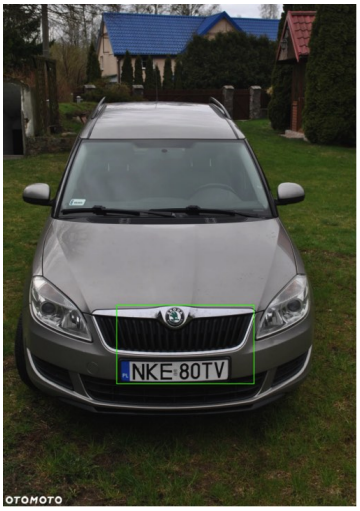
Poza samymi wynikami, należy podkreślić, że zadanie detekcji obiektów jest trudne i czasochłonne. Wymaga to precyzyjnego dostrojenia parametrów oraz znajomości różnych architektur. Rozbudowanie tego projektu o model OCR wymagałaby jeszcze większych zasobów, których, niestety, nie mieliśmy.



Rysunek 1: Prawie perfekcyjnie rozpoznanie granic tablicy rejestracyjnej auta z portalu OTOMOTO przez model I.



Rysunek 2: Dobre rozpoznanie granic tablicy rejestracyjnej auta z portalu OTOMOTO przez model I. Tablica w całości znajduje się w zaznaczonym obszarze.



Rysunek 3: Poprawnie rozpoznane dwie z granic tablicy rejestracynej auta z portalu OTOMOTO przez model I.