POLITECHNIKA GDAŃSKA

PROJEKT – DETEKCJA TABLIC REJESTRACYJNYCH

Przedmiot: Sztuczna Inteligencja

Kierunek: Informatyka

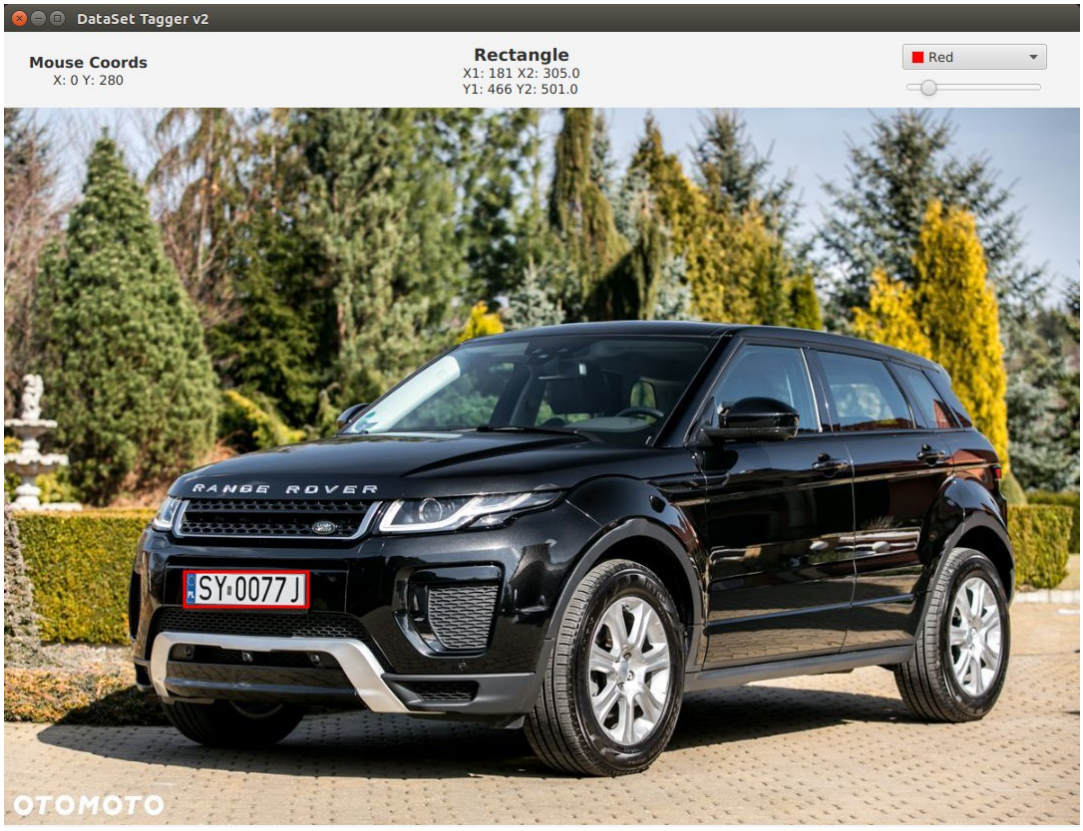
Paweł Kołakowski 165452
Tomasz Kurek 165572

1. Wprowadzenie

Celem naszego projektu była detekcja tablic rejestracyjnych na zdjęciach. Do tego zadania użyliśmy konwulcyjnych sieci neuronowych, zaimplementowach w algorytmie YOLOv3.

2. Dataset

Pierwszym problemem na który się natknęliśmy było zdobycie zbioru uczącego/testującego. Poszukiwanie gotowych zbiorów w internecie zakończyło się niepowodzeniem, więc postanowiliśmy stworzyć własny. Do tego celu stworzyliśmy aplikację w języku Java. Program pobiera ścieżkę do folderu, w którym znajdują się gotowe zdjęcia do obróbki w formacie .jpg, pobrane z portali aukcyjnych. Zaznaczamy myszką tablicę rejestracyjną. Spacją przechodzimy do nastęgo obrazka. Program zapisuje kopię obrazu z zaznaczoną przez nas tablicą rejestracyjną, oraz współrzędne zaznaczenia do pliku o nazwie nazwa_obrazu.txt. Z założeń wykorzystywanego przez nas algorytmu rozmiar obrazu oraz zaznaczonej tablicy rejestracyjnej nie jest istotny.

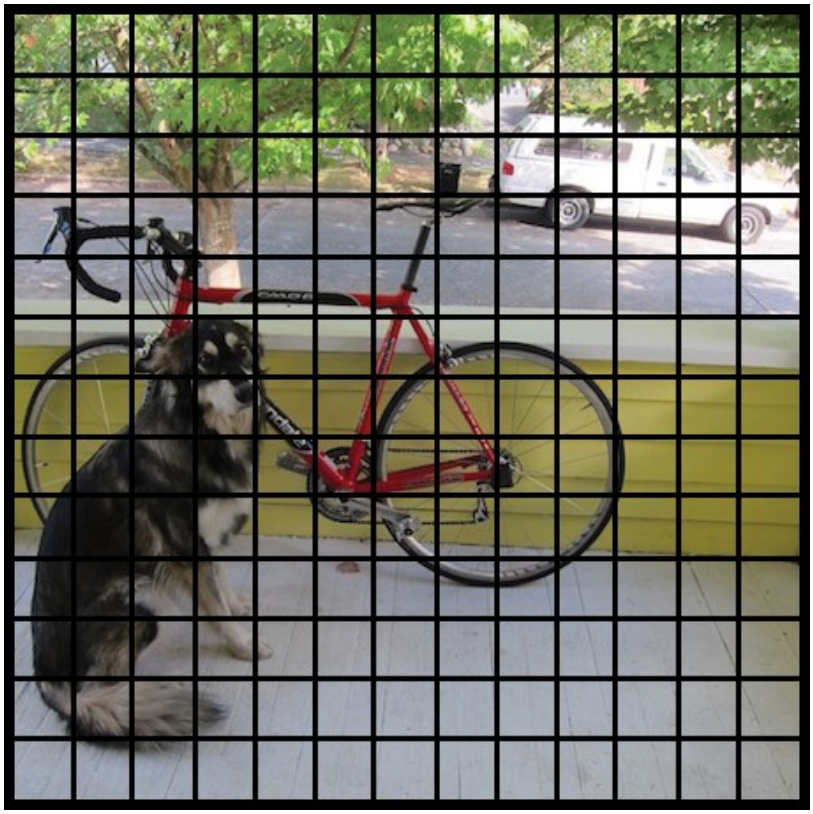


3. YOLOv3

Czym jest YOLO?

Yolo (You Only Look Once) jest algorytmem, który służy do detekcji obiektu na obrazie. Jego twórcami są Joseph Redmon oraz Ali Farhadi (University of Washington) https://pjreddie.com/darknet/yolo/. Yolo jest jest zupełnie innym podejściem, niż w poprzednich algorytmach np. R-CNN.

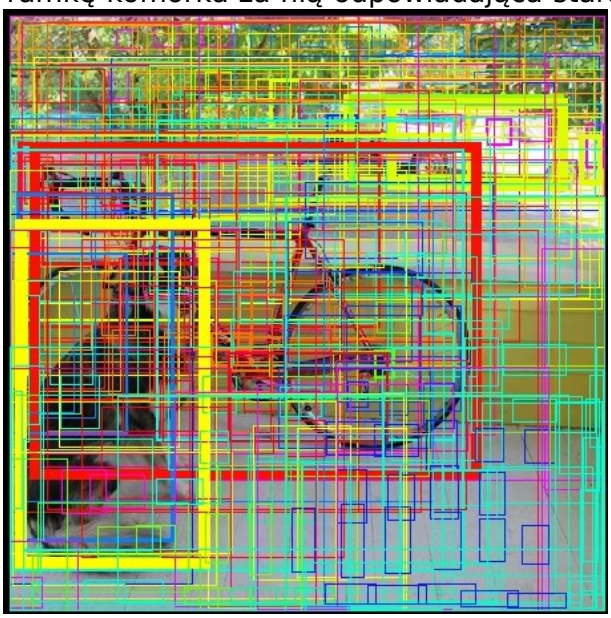
Tak jak nazwa wskazuje, YOLO nie wymaga wielokrotnego ładowania obrazu, wystarczy tylko raz. Na początku dzieli obraz na siatkę o wymiarach 13 x 13. Każda komórka jest odpowiedzialna za predykcję 5 przesuniętych względem komórki ramek brzegowych (ang. Bounding Box), w których mogą być prostokątem opisującym szukany obiekt.



Rys. 1

Jako dane wyjściowe YOLO daje nam prawdopodobieństwo z jakim z jest pewien, że w danej ramce istnieje dany obiekt. Wynik nie mówi nam jeszcze który obiekt może być tam zawarty, tylko czy jakikolwiek może się tam znajdować (Rys. 2). Za każą ramkę komórka za nią odpowiadająca stara się

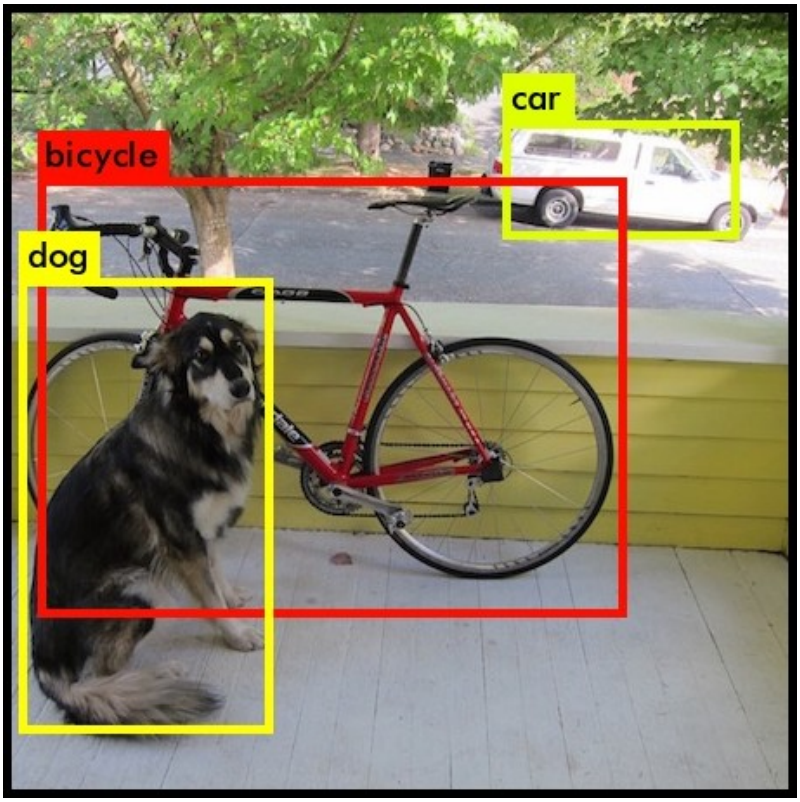
Rys. 2



Rys. 3

przewidzieć obiekt w nim zawarty, korzystając ze znanych już algorytmów klasyfikacji (Rys. 3).

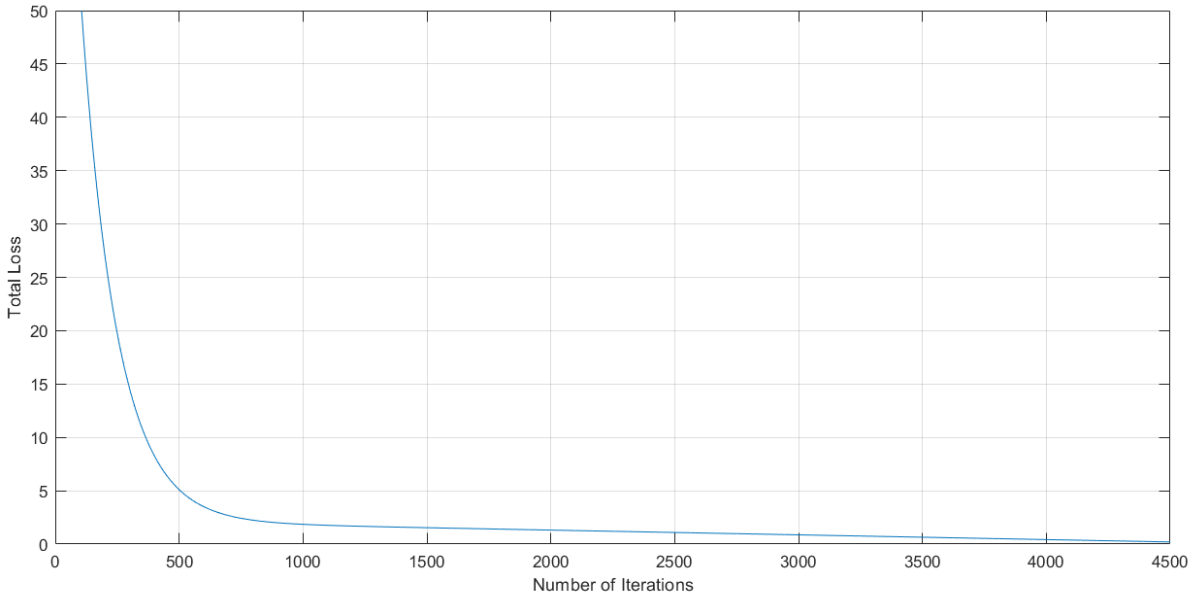
Jak widzimy na powyższych obrazkach, im grubszy prostokąt, tym większe prawdopodobieństwo, że obiekt się w nim znajduje. Likwidując wszystkie prostokąty z mniejszym prawdopodobieństwem niż 30% otrzymujemy ostateczny wynik (Rys. 4).



Rys. 4

4. Uczenie

Do uczenia własnej sieci neuronowej użyliśmy 1500 obrazów, z czego 1200 należało do zbioru uczącego, a 300 do zbioru testującego. Proces uczenia trwał ok 6h (4500 iteracji algorytmu uczącego), po którym uzyskaliśmy wytrenowany model. Okazało się jednak, że uzyskany model jest przetrenowany i nie jest w stanie wykryć tablic rejestracyjnych z obrazów innych niż ze zbioru uczącego/testującego. Dzięki kopiom zapasowym zapisywanym co 100 iteracji byliśmy wstanie cofnąć proces uczenia do iteracji 2700, w której uzyskaliśmy najlepsze parametry detekcji.



Powyżej jest przedstawiony wykres uzyskany podczas uczenia. Przedstawia on zależność błędu detekcji od liczby iteracji wyznaczonego następującym wzorem.

$$\mathcal{L} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_{ij} - \hat{C}_{ij})^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_{ij} - \hat{C}_{ij})^2 + \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

where:

Symbol	Explanation
1_i^{obj}	whether the cell i contains an object.
1_{ij}^{obj}	j-th bounding box predictor of the cell i is "responsible" for that prediction (See Fig. 9).
C_{ij}	confidence score of the j-th box in cell i, probability(containing an object) * IoU(pred, truth).
$\hat{C}_{ij}$	predicted confidence score.
$p_i(c)$	conditional probability of whether cell i contains an object of class c.
$\hat{p}_i(c)$	predicted conditional probability of whether cell i contains an object of class c.

5. Wyniki

Do sprawdzenia poprawności działania naszego detektora wykorzystaliśmy nowy zbiór 130 obrazów. Wyniki są następujące:

- Bezbłędne zaznaczenie: 88%
- Złe lub brak zaznaczenia: 3%
- Podwójne zaznaczenie tej samej tablicy: 3%
- Niedokładne zaznaczenie 6%

Przykłady

A. Brak odczytu

Możliwe przyczyny:

- Zła kolorystyka zdjęcia
- Zły stosunek rozmiarów tablicy do wielkości zdjęcia
- Orientacja obrazu
- Kolor tablicy
- Nieodpowiedni kąt wykonania zdjęcia

Przykład:

B. Błędny Odczyt

Możliwe przyczyny:

- Tablice zagraniczne z nietypowymi symbolami np. Niemcy, Szwajcaria, Chorwacja
- Nietypowa ilość znaków w tablicy
- Niejednorodne oświetlenie tablicy oraz refleksja światła od lakieru samochodu
- Odczytanie innego prostokąta z tekstem o podobnych proporcjach co tablica rejestracyjna, np. znak z nazwą ulicy

Przykłady:

C.Poprawny Odczyt

Przykłady:

6. Rozpoznawanie napisu (OCR)

Po detekcji tablicy rejestracyjnej wycinamy zaznaczony prostokąt z oryginalnego zdjecia. Kolejnym ważnym krokiem do poprawnego odczytania numeru rejestracyjnego jest progowanie i wyprostowanie obrazka, proces ten powinien zapewnić większą skuteczność algorytmu ocr. Do tego celu posłużyliśmy się różnymi algorytmami np Horizon Detection Algorithm. Dla naszych danych algorytmy wykazały się skuteczność mniejszą niż 5%, może to być spowodowane niekorzystnym kątem wykonania zdjęcia albo niską rozdzielczością wyciętych tablic rejestracyjnych. Kolejnym krokiem jest odczytanie numeru rejestracyjnego, do tego celu wykorzystaliśmy algorytm Google Tesseract. Do sprawdzenia poprawności działania algorytmu wykorzystaliśmy zbiór 130 obrazów. Algorytm tylko 5 razy zwrócił poprawną wartość. Uzyskana skuteczność spowodowana jest niską skutecznością algorytmów prostujących obraz oraz szumami występującymi na zdjęciach. Kolejnym czynnikiem obniżającym poprawny odczyt numeru rejestracyjnego są dodatkowe napisy występujące na tablicach rejestracyjnych. Poniżej przedstawiony został proces odczytywnaia numeru tablicy rejestracyjnej.

7. Podsumowanie

Zaimplementowanie detektora tablic rejestracyjnych używając algorytmu YOLOv3 przyniosło zadawalające efekty: 88% prawidłowych zaznaczeń i 9% niedokładmych zaznaczeń. Nasz program poprawnie zaznacza tablice o różnych rozmiarach oraz o różnych proporcjach, wyszukuje wszystkie rejestracje na zdjęciu. Natomiast próby zaimplementowania odczytywania numeru rejstracyjnego zakończyły się niepowodzeniem. Możliwym powodem tego jest niejednorodne oświetlenie obrazków oraz niewłasciwy kąt zrobienia zdjęcia.

Kod aplikacji (Java) do tworzenia zbioru uczącego można znaleźć na stronie:
https://github.com/Kreans/Picture_Tagger

Zmodyfikowany kod algorytmu yolo v3 (zaimplementowany w języku C) oraz napisane przez nas skypty(python) detekcji oraz rozpoznawania numeru rejestracyjengo można znaleźć na stronie:
https://github.com/Kreans/PlateDetection

Źródła:

https://pjreddie.com/media/files/papers/YOLOv3.pdf

https://pjreddie.com/darknet/yolo/

https://github.com/pjreddie/darknet

https://github.com/unsky/yolo-for-windows-v2

https://github.com/tesseract-ocr/

https://www.youtube.com/watch?v=4eIBisqx9_g

https://www.otomoto.pl

https://gaborvecsei.wordpress.com/2016/08/29/straighten-image-with-opencv/

https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/

https://blog.algorithmia.com/how-to-rotate-images-in-python-using-a-horizon-detection-algorithm/

https://github.com/tesseract-ocr/tesseract