

Koncepcje i zastosowania sztucznej inteligencji

Jakub Kasprowicz

Streszczenie referatu wygłoszonego 3 IV 2023

Spis treści:

Czym jest sztuczna inteligencja?

Dlaczego wybrano sieci neuronowe dla rozpoznawania obrazów w mojej pracy inżynierskiej?

Jak działa proces tworzenia i nauki sieci neuronowej

Metryki opisujące parametry modelu

Podsumowanie

1.Czym jest sztuczna inteligencja?

Najprościej rzecz ujmując, sztuczna inteligencja jest częścią informatyki, która bada i opisuje zagadnienia obejmujące rozwiązania problemów informatycznych, dla których klasyczne algorytmy są bardzo trudne do stworzenia. Jednym z takich problemów może być na przykład rozpoznawanie obrazów. W tym streszczeniu skrótowo omówię rozpoznawanie tablic rejestracyjnych z obrazów.

2. Dlaczego wybrano sieci neuronowe dla rozpoznawania obrazów w mojej pracy inżynierskiej?

W systemach obsługi parkingowej kluczowym elementem jest wpuszczanie oraz wypuszczanie pojazdów z obszaru centrum dystrybucyjnego. Czas obsługi na bramach wjazdowej i wyjazdowej ma kluczowy wpływ na czas dostawy towaru do wybranego punktu. Opóźnienia wjazdów i wyjazdów ciężarówek są bardzo problematyczne dla pracowników, a także klientów. W końcu klient czeka na dostarczenie towaru w terminie. Dla rozwiązania problemu obsługi bramowej zdecydowano się wykorzystać biblioteki języka programowania Python do obsługi sieci neuronowych oraz wizji komputerowej. Omawianym modelem jest model YOLOv5 stworzony przez UltraLytics.

3. Jak działa proces tworzenia i nauki sieci neuronowej

1. Zbieranie danych: Na początku procesu konieczne jest zebranie zbioru danych, który będzie reprezentatywny dla problemu, który sieć ma rozwiązać. Te dane mogą pochodzić z różnych źródeł, takich jak obrazy, tekst, dźwięk, dane sensorów itp., w zależności od rodzaju zadania.
2. Przetwarzanie danych: Zebrane dane mogą wymagać przetworzenia, normalizacji lub innych operacji, które pomogą w lepszym przystosowaniu ich do uczenia maszynowego. W tym przypadku skalowanie obrazów do określonego rozmiaru.
3. Podział na dane treningowe i testowe: Zbiór danych jest następnie podzielony na dwie główne części: dane treningowe i dane testowe. Dane treningowe są używane do nauczania modelu, podczas gdy dane testowe służą do oceny jego wydajności po zakończeniu procesu uczenia.



Rysunek 1 Obrazy wsadowe z tablicami rejestracyjnymi dla sieci neuronowej

Źródło: Opracowanie własne

Dla tablic rejestracyjnych – zbiór danych (dataset) znajdował się na stronie Kaggle.com. Był w formacie Pascal VOC, który nie jest kompatybilny z YOLOv5. Dlatego należało go przerobić na standard COCO zgodny z YOLOv5.

Pascal VOC	YOLO
<pre> <annotation> <folder>images</folder> <filename>Cars0.png</filename> <size> <width>500</width> <height>268</height> <depth>3</depth> </size> <segmented>0</segmented> <object> <name>licence</name> <pose>Unspecified</pose> <truncated>0</truncated> <occluded>0</occluded> <difficult>0</difficult> <bndbox> <xmin>226</xmin> <ymin>125</ymin> <xmax>419</xmax> <ymax>173</ymax> </bndbox> </object> </annotation> </pre>	<pre> 0 0.645 0.5559701492537313 0.386 0.1791044776119403 </pre>

Rysunek 2 Porównanie kodu zawierającego informacje o ramkach ograniczających dla Pascal VOC oraz YOLOv5

Źródło: Opracowanie własne

Posłużył do tego poniższy kod:

```
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

!mkdir "/kaggle/working/labels"
!cp -r "/kaggle/input/car-plate-detection/images"
"/kaggle/working/images"

imgs = os.listdir('/kaggle/working/images')
imgs_train, imgs_val = train_test_split(imgs, test_size=0.05)

df = pd.read_csv('../input/car-plate-get-annotation-info-from-xml/annotation.csv')

import yaml
cwd = '/kaggle/working/'
data = dict(
    train = cwd + 'train.txt',
    val = cwd + 'val.txt',
    nc = 1,
    names = ['licence'],
)

with open(cwd + 'bgr.yaml', 'w') as outfile:
    yaml.dump(data, outfile, default_flow_style=False)
with open(cwd + 'train.txt', 'w') as f:
    for path in imgs_train:
        f.write(cwd+'images/'+path+'\n')
with open(cwd + 'val.txt', 'w') as f:
    for path in imgs_val:
        f.write(cwd+'images/'+path+'\n')

for file in imgs:
    file = file.split('.')[0]
    bboxes = []
    for _, row in df[df['file'] == file].iterrows():
        bbox = [str(0), str(row['Xcent']), str(row['Ycent']),
str(row['boxW']), str(row['boxH'])]
        bbox = ' '.join(bbox)
        bboxes.append(bbox)
        with open(cwd+'labels/'+file+'.txt', 'w') as f:
            bboxes = '\n'.join(bboxes)
            f.write(bboxes)
```

Opis powyższego programu wygląda następująco:

Importowanie niezbędnych bibliotek:

-os: Biblioteka do operacji na systemie plików.

-numpy as np: Biblioteka do obliczeń numerycznych.

-pandas as pd: Biblioteka do manipulacji danymi w formie tabel.

-train_test_split z sklearn.model_selection: Funkcja do podziału danych na zbiór treningowy i walidacyjny.

-yaml: Biblioteka do obsługi plików YAML.

Tworzenie katalogów:

- Skrypt tworzy katalog o nazwie "labels" w bieżącym folderze pracy.
- Kopiuje zawartość katalogu `"/kaggle/input/car-plate-detection/images"` do nowo utworzonego katalogu "images" w bieżącym folderze pracy.

Podział zbioru danych na treningowy i walidacyjny:

- Lista plików obrazów znajdujących się w katalogu "images" jest podzielona na zbiór treningowy (imgs_train) i zbiór walidacyjny (imgs_val) w stosunku 95:5.

Wczytanie danych z pliku CSV:

- Skrypt wczytuje dane z pliku CSV o nazwie "annotation.csv" (zawierającego informacje o adnotacjach dla obrazów).

Tworzenie pliku YAML dla konfiguracji zbioru danych:

- Tworzony jest słownik data zawierający informacje o konfiguracji zbioru danych, takie jak ścieżki do plików treningowych i walidacyjnych, liczba klas (nc) oraz nazwy klas (names).
- Ten słownik jest zapisywany do pliku "bgr.yaml" w formacie YAML.

Tworzenie plików treningowych i walidacyjnych:

- Pętla tworzą pliki "train.txt" i "val.txt", zawierające ścieżki do obrazów treningowych i walidacyjnych, odpowiednio.

Tworzenie plików etykiet (ramek ograniczających):

- Pętla iteruje przez listę plików obrazów.
- Dla każdego pliku, odczytywane są odpowiednie adnotacje z ramki danych na temat ramek ograniczających zawierających tablice rejestracyjne.
- Tworzone są pliki tekstowe o nazwie odpowiadającej nazwie pliku obrazu w katalogu "labels". Każdy plik zawiera informacje o bounding boxach w formacie, który jest później używany w procesie uczenia modelu detekcji.

4. Inicjalizacja wag: Sieć neuronowa jest inicjalizowana losowymi wagami, co jest jej stanem początkowym. Te wagi są stopniowo dostosowywane podczas procesu uczenia w celu optymalizacji wydajności modelu.

5. Uczenie modelu: Model sieci neuronowej jest trenowany przy użyciu danych treningowych. Proces ten polega na dostosowywaniu wag sieci w taki sposób, aby minimalizować błąd między przewidywaniami modelu a rzeczywistymi danymi treningowymi.

4. Metryki opisujące parametry modelu

Następnym etapem jest weryfikacja metryk precyzji (ang. *Precision*) oraz czułości (ang. *Recall*). Są to podstawowe metryki związane z sieciami neuronowymi. Precision (precyzja) jest miarą proporcji poprawnie sklasyfikowanych instancji danej klasy (np. pozytywnej) wśród wszystkich instancji, które zostały sklasyfikowane jako ta klasa. Wyraża to, jak dobrze model identyfikuje prawdziwie pozytywne

przypadki wśród wszystkich przypadków, które zaklasyfikował jako pozytywne. Precyzja jest obliczana przy użyciu wzoru:

$$\frac{TP}{(TP + FP)}$$

Gdzie:

-TP (ang. *True Positive*) oznacza liczbę prawdziwie pozytywnych przypadków

-FP (ang. *False Positive*) oznacza liczbę fałszywie pozytywnych przypadków.

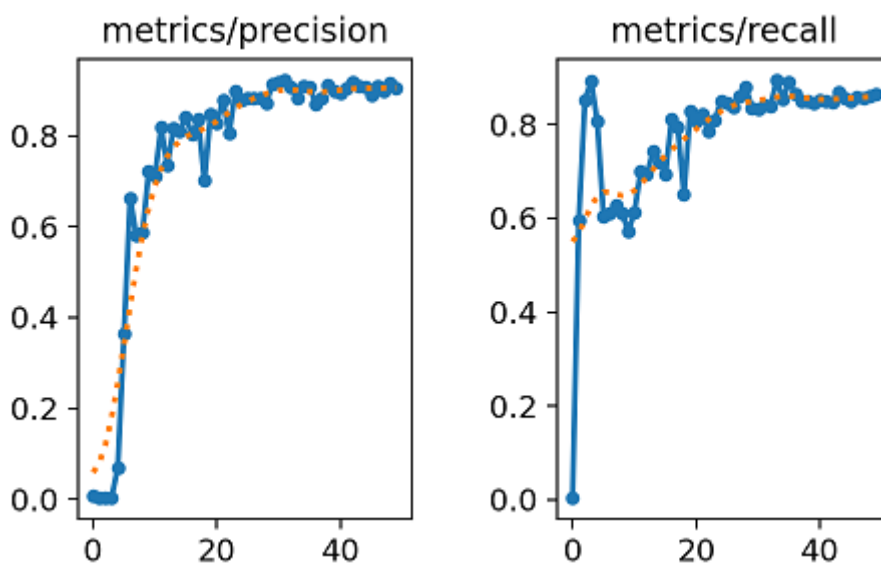
Recall (czułość) to miara proporcji poprawnie sklasyfikowanych instancji danej klasy (np. pozytywnej) wśród wszystkich instancji, które powinny być sklasyfikowane jako ta klasa. Wyraża to, jak dobrze model identyfikuje wszystkie prawdziwie pozytywne przypadki. Czuość jest obliczana przy użyciu wzoru:

$$\frac{TP}{(TP + FN)}$$

Gdzie:

-TP oznacza liczbę prawdziwie pozytywnych przypadków

-FN (ang. *False Negative*) oznacza liczbę fałszywie negatywnych przypadków.



Rysunek 3 Omawiane metryki dla 50 epok treningowych modelu

Źródło: Opracowanie własne

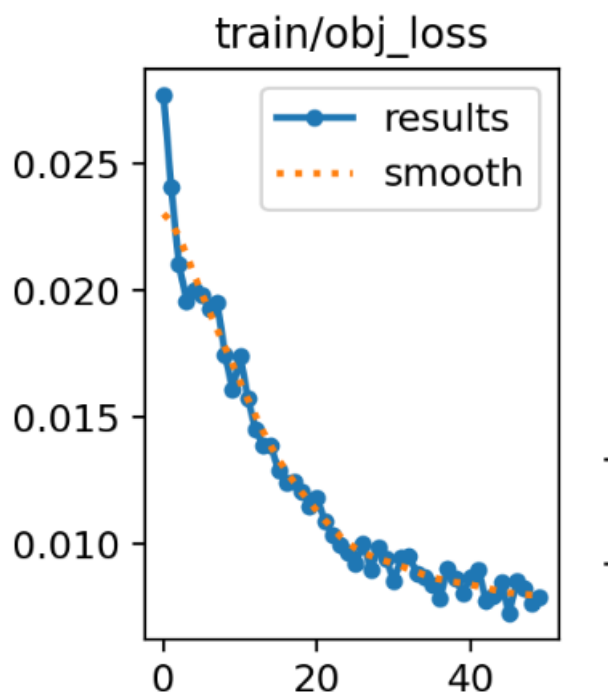
Detekcja obiektów koncentruje się także na wyodrębnianiu części obrazu związanych z konkretnymi obiektami w celu ich sklasyfikowania i w celu znalezienia ograniczającego wielokąta, który opakuje obiekt z największą dokładnością, zazwyczaj jest to kwadrat, dlatego nazywany jest ramką ograniczającą. Najpopularniejszą metryką stosowaną do określenia tych wartości jest część wspólna całości (ang. *Intersection Over Union*), definiowana jest jako:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

Gdzie:

- A ramka ograniczająca rzeczywista
- B ramka ograniczająca powstała w wyniku predykcji modelu

Współczynnik ten powinien maleć wraz z coraz większą liczbą treningów modelu



Rysunek 4 Wynik IoU dla stworzonej sieci neuronowej

Źródło: Opracowanie własne

Skrótowy opis w pseudokodzie realizacji wykrywania tablic rejestracyjnych na obrazie w pracy inżynierskiej.

1. Inicjalizacja modelu
2. Odczyt przesłanych plików
3. Konwersja pliku na obraz.
4. Detekcja obiektów na obrazie.
5. Identyfikowanie i filtrowanie wyników detekcji
6. Wycięcie tablic rejestracyjnych z obrazu
7. Rozpoznanie znaków: Wykorzystując bibliotekę Tesseract, która jest narzędziem do optycznego rozpoznawania znaków (OCR), tekst na tablicach rejestracyjnych jest rozpoznawany. Tesseract analizuje obraz tablicy i przekształca go na postać tekstową.
8. Filtrowanie rozpoznanych znaków niepoprawnie rozpoznane znaki i zachować jedynie te, które są poprawne.

9. Zwrot wyników w formacie JSON:

POST http://0.0.0.0:8000/detect Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
☒	file	polska2.png ×			
	Key	Value	Description		

Body Cookies Headers (4) Test Results Status: 200 OK Time: 405 ms Size: 150 B Save Response

Pretty Raw Preview Visualize JSON ≡

```

1  [
2    "plates": [
3      "ASB 8903R"
4    ]
5  ]

```

Rysunek 5 Żądanie zawierające obraz pojazdu

Źródło: Opracowanie własne



Rysunek 6 Obraz z pojazdem

Źródło: Opracowanie własne

Podsumowanie

Podsumowując, sieci neuronowe są przydatne w rozpoznawaniu obiektów na obrazach. Zgodnie z powyższymi przykładami, można je wykorzystać przy rozpoznawaniu tablic rejestracyjnych pojazdów. Umożliwia to wykorzystanie sieci neuronowych w systemach logistycznych, gdzie klasyczne podejście do rozwiązywania problemów detekcji tablic rejestracyjnych mogłoby być problematyczne.