

Model YoloV5

Adrian Łukasik 19388

VIII 2023

Spis treści

Wstęp.....	1
Cechy, zalety i wady tego modelu	1
Sieć neuronowa konwolucyjna (CNN)	2
Konwolucja	2
Łączenie	2
Normalizacja.....	2
Aktywacja	3
Przykładowe wykorzystanie modelu	3
Implementacja.....	4
Podsumowanie	5
Źródła:.....	6

Wstęp

YoloV5 jest zaawansowanym modelem do wykrywania obiektów, segmentacji instancji i klasyfikacji obrazów, opartym na bibliotece PyTorch. Jest to piąta wersja modelu "You Only Look Once", który zapewnia szybkie i dokładne wyniki w czasie rzeczywistym. YoloV5 buduje na sukcesach poprzednich wersji YOLO i wprowadza nowe funkcje i ulepszenia, aby zwiększyć wydajność i elastyczność. YoloV5 działa na zasadzie przetwarzania obrazu za pomocą sieci neuronowej konwolucyjnej, która dzieli obraz na siatkę komórek i przewiduje dla każdej komórki pewną liczbę prostokątów otaczających i prawdopodobieństwa klas obiektów. Następnie stosuje się algorytm Non-Maximum Suppression, aby wyeliminować nadmiarowe detekcje i zachować tylko te z najwyższym poziomem zaufania. YoloV5 ma kilka wariantów o różnej wielkości i dokładności, takich jak YoloV5s, YoloV5m, YoloV5l i YoloV5x. Można je łatwo trenować i wdrażać za pomocą narzędzi dostarczonych przez Ultralytics, twórców YoloV5. Niektóre z zastosowań YoloV5 to monitorowanie ruchu drogowego, rozpoznawanie twarzy, analiza sportowa, inspekcja przemysłowa i wiele innych.

Cechy, zalety i wady tego modelu

YOLO v5 to piąta wersja algorytmu wykrywania obiektów w czasie rzeczywistym, który jest szybki, dokładny i łatwy w użyciu. Główne cechy i zalety YOLO v5 to:

- Możliwość wykrywania obiektów o różnych rozmiarach i kształtach, dzięki zastosowaniu architektury FPN (*ang. Feature Pyramid Network*), która łączy cechy z różnych warstw sieci.

- Zastosowanie techniki MOSAIC, która polega na losowym łączeniu czterech obrazów treningowych w jeden, co zwiększa różnorodność danych i poprawia generalizację modelu.
- Użycie funkcji aktywacji SiLU (*ang. Sigmoid Linear Unit*), która jest szybsza i bardziej efektywna niż ReLU (*ang. Rectified Linear Unit*) i zapewnia lepszą jakość wykrywania.
- Zmniejszenie liczby parametrów modelu o ponad 80% w porównaniu do poprzedniej wersji YOLO v4, co zmniejsza wymagania sprzętowe i czas treningu.
- Możliwość dostosowania modelu do różnych zadań i domen, dzięki łatwej zmianie liczby klas, rozmiaru obrazu wejściowego i architektury sieci.

Z wad możemy wymienić:

- Brak oficjalnej publikacji naukowej,
- Kontrowersje wokół nazwy i zgodności z poprzednimi wersjami modelu „YOLO”,
- Także potencjalne problemy z licencjonowaniem i prawami autorskimi,
- Brak wsparcia dla innych zadań niż wykrywanie obiektów
- Brak wsparcia dla innych frameworków niż PyTorch

Sieć neuronowa konwolucyjna (CNN)

Sieć neuronowa konwolucyjna to rodzaj sieci neuronowej, która jest specjalnie zaprojektowana do przetwarzania danych o strukturze siatki, takich jak obrazy. Sieć ta składa się z kilku warstw, które wykonują różne operacje na danych wejściowych, takie jak: konwolucja, łączenie, normalizacja i aktywacja.

Konwolucja

Proces konwolucji polega na nakładaniu na obraz wejściowy filtrów (lub jąder), które mają narzucony rozmiar i głębokość. Filtry te poruszają się po obrazie z określonym krokiem i wykonują operację mnożenia macierzy z fragmentem obrazu. Wynikiem mnożenia macierzy jest mapa cech, która przechowuje informacje o występowaniu filtrów w różnych częściach obrazu. Jest to parametr uczony, który pozwala na wykrywanie krawędzi, kształtów, tekstur i innych cech obrazu.

Łączenie

Łączenie (*ang. Pooling*) – jest to operacja redukcji wymiarowości mapy cech (którą otrzymaliśmy po konwolucji), która polega na wybieraniu maksymalnej lub średniej wartości z okna o określonym rozmiarze i kroku. Łączenie pomaga zredukować liczbę parametrów sieci, żeby zwiększyć odporność na szумы i zmiany położenia obiektów oraz uchwycić informacje o kontekście przestrzennym.

Normalizacja

Normalizacja to operacja skalowania wartości mapy cech, aby oscylowały one w podobnym zakresie i wariancji. Za pomocą normalizacji jesteśmy w stanie poprawić stabilność i prędkość uczenia się sieci neuronowej oraz zapobiegać zjawisku przetrenowania (*ang. Overfeeding*).

Aktywacja

Aktywacja to operacja nieliniowa stosowana do każdego elementu mapy cech, która wprowadza zdolność sieci do modelowania złożonych zależności między danymi.

Sieci neuronowe są na tyle skomplikowanym tematem, że ciężko byłoby zawrzeć je w tym streszczeniu i chciałbym się skupić wyłącznie na samej metodzie „YOLO”. Natomiast poniżej załączam źródła, które są ciekawym uzupełnieniem.

[Training a Classifier — PyTorch Tutorials 2.0.1+cu117 documentation](#)

[Inteligencja obliczeniowa - Deep Learning \(Sieć konwolucyjna jako przykład modelu głębokiego\)](#)

Przykładowe wykorzystanie modelu

Model ten można wykorzystać na różny sposób w zależności od oczekiwań użytkownika. Jest to użyteczne w momentach kiedy potrzebujemy znaleźć interesujący użytkownika obiekt na pliku wideo, live czy zdjęciu. W tym celu został użyty domyślny zasób bazy danych modelu YOLOv5, który umożliwia detekcję prostych elementów. Jako element testowy zostało wykorzystane [zdjęcie](#) (Rys.1.1.) z witryny internetowej Akademii Nauk Stosowanych w Elblągu. Jak możemy zaobserwować na Rys.1.2. model YoloV5 domyślnie poradził sobie bardzo dobrze z detekcją ludzi oraz elementów, które znajdują się na zdjęciu.



Rys.1.1. – Zdjęcie testowe wykorzystane do detekcji elementów



Rys.1.2. – Zdjęcie otrzymane po użyciu metody „YoloV5” z parametrem –conf 0.25

Jak możemy zaobserwować model „YOLOv5” narysował na obrazie czerwone, niebieskie oraz zielone obwódki, które zidentyfikował jako elementy, które ma już zdefiniowane w swojej bazie danych. Za pomocą etykiet (*ang. Label*) podpisał zidentyfikowane obiekty oraz podał wartość od 0 do 100, która przedstawia procent, jak bardzo jest tożsamy to co widzi na zdjęciu, z tym co posiada w bazie danych. W następnym etapie została przedstawiona implementacja modelu i jego możliwości.

Implementacja

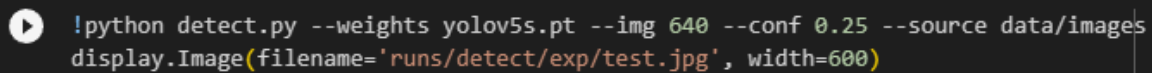
Efekt przedstawiony na Rys.1.2. został osiągnięty przy wykorzystaniu biblioteki PyTorch oraz modelu „YOLOv5” dostępnego na platformie GitHub. Kod przedstawiony poniżej przedstawia proces implementacji.

```
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml # install

import torch
import utils
display = utils.notebook_init() # checks
```

W pierwszym etapie kopiujemy repozytorium YoloV5 oraz instalujemy potrzebne komponenty tego modelu. Wszystkie potrzebne elementy są zawarte w notatniku „requirements.txt” znajdującego się w repozytorium.

Następnie importujemy bibliotekę PyTorch oraz utils i inicjalizujemy ją do zmiennej „display”.



```
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images  
display.Image(filename='runs/detect/exp/test.jpg', width=600)
```

Rys.1.3. – Inicjalizacja rozpoznawania elementów przykładowego obrazu

Na przedstawionym Rys.1.3. wykonujemy rozpoznawanie obrazu na przykładowym zdjęciu „test.jpg”, który znajduje się w ścieżce „runs/detect/exp/test.jpg” przy użyciu różnych parametrów.

„Python detect.py” - Odpowiada za uruchomienie programu nazwanego „detect” z wybranymi parametrami.

--weights yolov5s.pt – Ten parametr tłumaczy nam, że korzystamy z mniejszej wersji modelu „Yolov5” w wersji małej (*ang. Small*)

--img 640 – Parametr odpowiedzialny za rozmiar zdjęcia, które otrzymamy po przetworzeniu go przez ten model.

--conf 0.25 – Parametr ten to „pewność” (*ang. Confidence*) – Służy on do ustawiania progu ufności dla wykrywania obiektów. Określa, jak bardzo model musi być pewny, zanim wygeneruje ramkę otaczającą potencjalny obiekt. Oznacza to, że model wygeneruje tylko ramki, które mają co najmniej 25% ufności. Wyższa wartość conf spowoduje mniejszą liczbę obiektów, ale bardziej dokładnych ramek, podczas gdy niższa wartość spowoduje większą liczbę, ale mniej dokładnych ramek.

Podsumowanie

Biorąc pod uwagę wszystko co zostało przedstawione powyżej możemy zaobserwować, że model "You Only Look Once" w wersji 5, umożliwia użytkownikowi szybki i bezproblemowy dostęp do prostych metod detekcji na obrazie, wideo czy transmisji na żywo. Jego implementacja jest prosta, ponieważ potrzebuje tylko dwóch bibliotek oraz wymaganych elementów z dokumentu tekstowego „requirements” znajdującego się w repozytorium. Można ten model uruchomić w środowisku lokalnym oraz w chmurze, tak jak zostało to przedstawione w streszczeniu przy użyciu „Google Colab”. Prezentacja skupia się wyłącznie na przedstawianiu możliwości modelu „out of the box”.

Źródła:

- <https://github.com/ultralytics/yolov5>
 - https://pytorch.org/hub/ultralytics_yolov5/
 - <https://docs.ultralytics.com/yolov5/>
 - <https://roboflow.com/model/yolov5>
- https://colab.research.google.com/github/pytorch/pytorch.github.io/blob/master/assets/hub/ultralytics_yolov5.ipynb