Sztuczna Inteligencja i Inżynieria Wiedzy

Opis projektu

„System do detekcji samochodów oraz rozpoznawania tablic rejestracyjnych”

Mateusz Pakuła
238336

1. Cel projektu

Celem projektu jest opracowanie systemu, w skład którego wchodzą funkcjonalności takie jak:
- detekcja samochodów i innych obiektów
- określanie stanów samochodu takich jak: odjazd i przyjazd z wyznaczonego miejsca parkingowego, ruch
- określanie stanu oświetlenia (włączone/wyłączone)
- wykrywanie tablicy rejestracyjnej, segmentacja symboli z tablicy rejestracyjnej i ich klasyfikacja

2. Opis działania systemu

System zaimplementowany został w języku Python z wykorzystaniem bibliotek OpenCV, PyTorch i NumPy. Działanie opiera się na: określeniu stanu oświetlenia, wykryciu obiektów, śledzeniu, klasyfikacji stanu samochodów, wykryciu tablicy rejestracyjnej, segmentacji i następnie klasyfikacji znaków z tablicy rejestracyjnej.

Detekcja

Do wykrywania obiektów na obrazie wykorzystano rozwiązanie o nazwie Single Shot MultiBox Detector (https://arxiv.org/abs/1512.02325) , bazujące na konwolucyjnej sieci neuronowej MobileNetV1.

Dane uczące

Dane uczące, na których został wytrenowany model składają się z danych utworzonych na bazie dostarczonych materiałów wideo obrobionych w narzędziu Scalabel (https://www.scalabel.ai/), oraz zbioru Open Images (https://storage.googleapis.com/openimages/web/index.html). Do pobierania danych ze zbioru Open Images przygotowany został skrypt **open_images_downloader.py**, które można wywołać poleceniem o składni:

python open_images_downloader.py --root ŚCIEŻKA **--class_names** ”nazwa1,nazwa2,...” **[--include_depiction] [--num_workers** L_WĄTKÓW**] [--retry** L_PRÓB**] [--filter_file** ”id1,id2,id3”**] [--remove_overlapped]**

gdzie:

--root – katalog główny, w którym przechowywany ma być zbiór danych
--class_names – nazwy klas, tj. samochód, które mają zostać pobrane
--include_depiction – mówi, że dane mają zawierać opis
--num_workers – liczba pracujących wątków
--retry – liczba prób podczas pobierania
--filter_file – identyfikatory plików, które mają zostać zignorowane
--remove_overlapped – mówi, że etykiety przesłonięte przez inne etykiety mają zostać usunięte

Uczenie modelu

Szczegóły dotyczące funkcji celu dostępne są w dokumentacji modelu Single Shot MultiBox Detector. Do optymalizacji funkcji celu wykorzystana została metoda stochastycznego spadku wzdłuż gradientu. Do trenowania modelu został przygotowany skrypt **train_ssd.py**, uruchamiany za pomocą polecenia o składni:

python train_ssd.py --train_datatype {bdd | open_images}
--validation_datatype {bdd | open_images} --datasets ”sciezka1 sciezka2 ...”
--validation_dataset SCIEŻKA **[--labels** ŚCIEŻKA**] [--freeze_base_net]**
[{--lr | --learning_rate} WARTOŚĆ**] [--momentum** WARTOŚĆ**] [--weight_decay** WARTOŚĆ**]**
[--gamma WARTOŚĆ**] [--base_net_lr** WARTOŚĆ**] [--extra_layers_lr** WARTOŚĆ**]**
[{--base_net ŚCIEŻKA **| --pretrained_net** ŚCIEŻKA **| --resume** ŚCIEŻKA**}]**
[--scheduler {multi-step [--milestones WARTOŚĆ**] | cosine [--t_max** WARTOŚĆ**]}]**
[--batch_size WARTOŚĆ**] [--num_epochs** WARTOŚĆ**] [--num_workers** WARTOŚĆ**]**
[--validation_epochs WARTOŚĆ**] [--debug_steps** WARTOŚĆ**] [--use_cuda {True | False}]**
[--checkpoint_folder ŚCIEŻKA**] [--balance_data]**

gdzie:

--train_datatype – typ zbioru danych uczących; musi być podany osobno dla każdej podanej ścieżki, np. mając 2 zbiory: jeden z narzędzia Scalabel, a drugi z Open Images należy napisać „--train_datatype bdd open_images”
--validation_datatype – typ zbioru walidacyjnego
--datasets – ścieżka lub ścieżki do zbiorów treningowych
--validation_dataset – ścieżka do zbioru walidacyjnego
--labels – ścieżka do pliku tekstowego z etykietami oddzielonymi przecinkiem, który mówi jaki numer powinien zostać przypisany danej etykiecie, np. mając etykiety „car”, „bike” i „person”, umieszczając w pliku „car,person,bike” etykieta „car” będzie miała nr 1, etykieta „person” nr 2, a etykieta „bike” nr 3
--freeze_baze_net – mówi o tym, że wagi bazowej sieci (MobileNetV1) nie powinny być zmieniane
--lr lub --learning_rate – początkowy współczynnik długości kolejnego kroku w gradiencie
--momentum – współczynnik pędu gradientu stochastycznego
--weight_decay – współczynnik regularyzacji L2 gradientu stochastycznego
--gamma – wspołczynnik gamma gradientu stochastycznego
--base_net_lr – współczynnik długości kroku dla sieci bazowej (MobileNetV1)
--extra_layers_lr – współczynnik długości kroku dla pozostałych warstw modelu
--base_net – ścieżka do sieci bazowej (sieć jest inizjalizowana przez metodę **init_from_base_net**)
--pretrained_net – ścieżka pretrenowanej sieci (inicjalizacja przez metodę **init_from_pretrained_ssd**)
--resume – ścieżka do trenowanej wcześniej sieci („zwykła” inicjalizacja przez metodę **load**)
--scheduler – wybór planisty do dynamicznego dostosowywania kroku w gradiencie
--milestones – parametr określający „kamienie milowe” w planiście MultiStepLR
--t_max – maksymalna liczba iteracji w planiście Cosine Annealing
--batch_size – rozmiar batcha
--num_epochs – liczba epok, które mają się wykonać

--num_workers – liczba wątków
--validation_epochs – określa liczbę epok, co które ma być dokonywana walidacja i zapis modelu
--debug_steps – liczba kroków, co które ma zostać wyświetlony log
--use_cuda – wartość logiczna wskazująca, czy używać GPU, czy nie
--checkpoint_folder – ścieżka do zapisu modelu
--balance data – użycie powoduje wyrównanie liczby etykiet każdej klasy

Śledzenie

Śledzenie samochodów jest procesem podzielonym na 3 etapy:

– pobranie wstępnego zbioru z detekcji obiektów

– utworzenie unikalnego identyfikatora dla każdego z samochodów

– monitorowanie każdego obiektu-samochodu podczas kolejnych klatek, zachowując przydzielone im unikalne identyfikatory

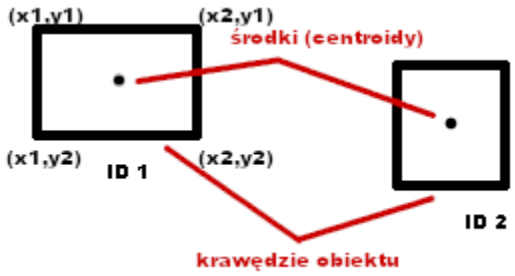
Pobranie wstępnego zbioru

Podczas pobrania początkowego zbioru wykrytych obiektów, obiekty oznaczone jako samochód są odseparowywane od pozostałych.

Utworzenie unikalnego ID

Każdy odseparowany samochód otrzymuje swój własny identyfikator.

Monitorowanie



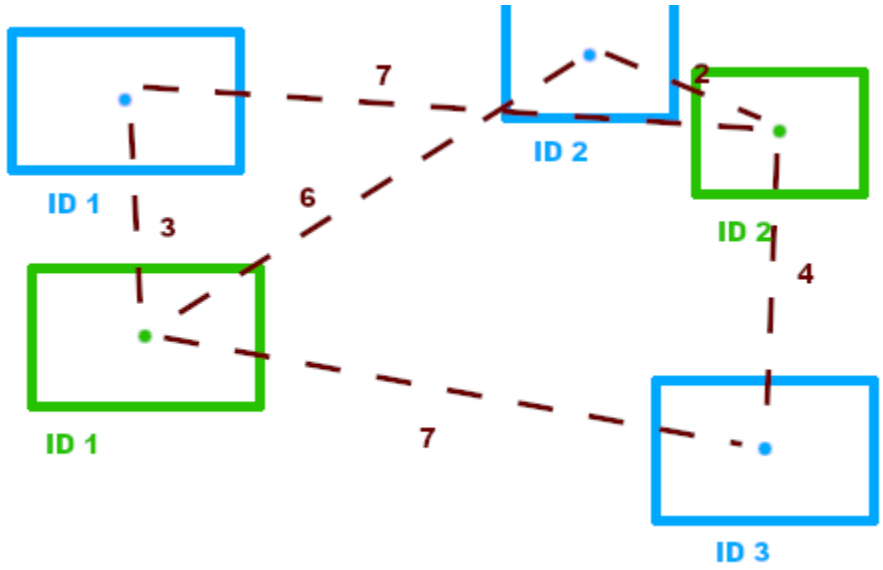
Rys. 1

Pierwszym krokiem jest obliczenie współrzędnych środka każdego z obiektów (samochodów) w układzie współrzędnych. Dla każdego prostokąta-ramki współrzedne wyliczane są ze wzorów:

$$x = \frac{x_1 + x_2}{2}$$

$$y = \frac{y_1 + y_2}{2}$$

Kolejnym krokiem jest porównanie środków obiektów z poprzedniej iteracji do obiektów z nowej iteracji.



Rys. 2 – Odległości pomiędzy środkami prostokątów

Dla każdego środka prostokąta z nowej iteracji (kolor niebieski) liczona jest odległość do każdego środka prostokąta z nowej iteracji (kolor zielony) zgodnie ze wzorem:

$l(a,b)=|b_x-a_x|+|b_y-a_y|$ (a, b – środki prostokątów, będące punktami w układzie współrzędnych)

Następnie każdemu obiektowi z nowej iteracji przyznawany jest identyfikator tego prostokąta, którego środek był najbliżej środka nowego prostokąta. W rysunku drugim – niebieski prostokąt w lewym górnym rogu jest najbliżej prostokąta o ID 1, dlatego też zostaje mu przyznany jego identyfikator. W przypadku większej liczby nowych obiektów-samochodów, niż w poprzedniej iteracji, przyznawany jest nowy identyfikator.

Określanie stanu samochodów

W systemie istnieją 4 stany opisujące samochód: domyślny – bez nazwy, w ruchu – MOVE, samochód zaparkował w pewnym miejscu parkingowym – ARRIVED, samochód wyjeżdża z ww. miejsca parkingowego – LEFT. Etykieta MOVE przyznawana jest samochodowi, którego środek przesunął się w jednej iteracji o 20 jednostek według wzoru na odległość między punktami stosowanego przy śledzeniu obiektów. Etykieta ARRIVED przyznawana jest wtedy, gdy prostokąt otaczający samochód i prostokąt, którym oznaczone jest miejsce parkingowe mają część wspólną. Etykieta LEFT przypisywana jest na określoną liczbę iteracji samochodowi, który poprzednio posiadał etykietę ARRIVED, a jego prostokąt-ramka i prostokąt oznaczający miejsce parkingowe nie nachodzą już na siebie.