Inteligentne Systemy Autonomiczne

Instrukcja nauki modeli do śledzenie obiektów złożonych (*klasy nie występujące w gotowych modelach*)

Niniejszy opis dotyczy wykonania zadania z wykorzystaniem sieci głębokiej, opartej o architekturę **SSD MobileNet**, z wykorzystaniem frameworka PyTorch. Możliwe jest uruchomienie innych modeli i skorzystanie z innych narzędzi jak np. Tensorflow (szczegóły do ustalenia z prowadzącymi zajęcia, wymaga to indywidualnego doinstalowania bibliotek).

Kroki wykonania zadania:

1. Pobranie kodu z repozytorium, zbudowanie i instalacja pakietów. Instalacja biblioteki PyTorch
2. Wybór obiektu oraz zebranie danych treningowych i testowych
3. Wybranie i wytrenowanie modelu sieci z wykorzystaniem PyTorch na pojeździe
4. Konwersja sieci do formatu ONNX
5. Uruchomienie detekcji obiektów z wykorzystaniem modelu sieci i konwersja automatyczna do formatu TensorRT. Połączenie modułu wykrywającego obiekt z modułem sterowania pojazdem, opracowanym na potrzeby zadania 1

Szczegółowy opis wykonania zadania:

Ad. 1 (**Opis w celach informacyjnych – wszystkie biblioteki są już zainstalowane)**

a. Pobrać kod z repozytorium https://github.com/dusty-nv/jetson-inference do katalogu /home/jetson/Code
    a. `git clone --recursive https://github.com/dusty-nv/jetson-inference`
b. Zbudować kod w repozytorium zgodnie z instrukcją: https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md
    a. Wszystkie potrzebne pakiety są zainstalowane na autkach, nie należy wykonywać polecenia *sudo apt-get install git python3-numpy* (instalować pakietów git i python3-numpy)
    b. W trakcie budowania pojawi się okno pobrania modeli (Model Downloader), z punktu widzenia wykonania zadania żaden z nich nie jest potrzebny. W celu przetestowania inferencji można jednak pobrać modele z rodziny MobileNet SSD
    c. W trakcie budowania kolejnym krokiem będzie instalacja biblioteki PyTorch. Krok ten należy wykonać tylko dla Python 3.6
c. Poprawną instalację można przetestować przez wykonanie polecenia detectnet (https://github.com/dusty-nv/jetson-inference/blob/master/docs/detectnet-camera-2.md) – wymaga pobrania modeli w kroku budowania

Ad 2.

W celu wykonania zadania należy wybrać obiekt dla którego model śledzący zostanie nauczony. Może być to dowolny obiekt, posiadający nieregularne kształty i kolory. Ze względu na wygodę zalecane jest wybranie obiektu o rozmiarach nie przekraczających 20-30 cm w żadnym wymiarze, ale możliwe jest nauczenie rozpoznawania dowolnych obiektów.

Obiekty podstawowe (np. człowiek, monitor itp. są rozpoznawane przez wiele modeli wbudowanych, nauczenie modeli rozpoznawania nowych obiektów spowoduje „zapomnienie" tamtych klas).

Zebranie danych może odbywać się z wykorzystaniem dwóch metod: bezpośrednio na urządzeniu lub poprzez nagranie filmów i analizę zewnętrznymi narzędziami

a. Zebranie na urządzeniu. W tym celu należy postępować zgodnie z instrukcją https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md . Narzędzie *camera-capture* pozwala na bezpośrednie budowanie datasetu poprzez analizę obrazu z kamery. W domyślnej konfiguracji będzie dawało obraz odwrócony o 180 stopni, wykonanie polecenia *camera-capture --help* wskaże odpowiednie opcje naprawy.
    a. Przed przystąpieniem do analizy należy utworzyć plik ***labels.txt***, w którym należy wpisać od jednej do dziesięciu nazw rozpoznawanych obiektów (klas), oddzielonych znakiem nowej linii. Nazwy nie powinny zawierać spacji i znaków specjalnych. **Po ostatniej nazwie NIE NALEŻY umieszczać znaku nowej linii.**
    b. Przed kamerą należy umieścić wybrany obiekt, następnie po wybraniu opcji *Dataset Type – Detection* i ustawieniu folderu dla datasetu możliwe jest oznaczanie obiektów z wykorzystanie *Bounding Boxa.*
    c. Należy zawsze utworzyć zestaw testowy i treningowy. Liczba zdjęć w zestawie treningowym **dla każdego rozpoznawanego obiektu** to co najmniej 50, w zestawie testowym – co najmniej 20. Im większa liczba w zbiorze treningowym tym lepiej będzie działał model
    d. Obiekt należy umieszczać pod różnymi kątami, w różnej odległości od kamery oraz na różnorodnym tle. Im więcej obrazów i im większa różnorodność danych tym wyższa dokładność wykrywania.
b. Zebranie danych lokalnie i analiza zewnętrzna. Na udostępnionym repozytorium umieszczony został kod w pythonie umożliwiający pobranie obrazu z kamery w formie plików video. Pliki te można następnie podzielić samodzielnie, lub z wykorzystaniem innych narzędzi na zdjęcia i wykorzystać dowolny program do tworzenia oznaczeń (VOTT, labelImg). Oznaczenia i struktura katalogów są zgodne z formatem Pascal VOC. W przypadku wyboru tej opcji można skontaktować się z prowadzącym w celu uzyskania szczegółowych instrukcji.

Ad 3.

Trening modelu należy przeprowadzić z wykorzystaniem polecenia train_ssd.py (https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md#training-your-model). Domyślnie sieć tworzy model w oparciu o architekturę Mobilenet V1 SSD. Możliwe jest utworzenie modeli w oparciu o architektury MobileNetV2 SSD-Lite i VGG SSD. Wybór architektury modelu możliwy jest przez zmianę parametru *--net*. W celu wytrenowania modeli należy pobrać najpierw punkty startowe do folderu:
jetson-inference/python/training/detection/ssd/models

a. *wget https://storage.googleapis.com/models-hao/mobilenet-v1-ssd-mp-0_675.pth*
b. *wget https://storage.googleapis.com/models-hao/mb2-ssd-lite-mp-0_686.pth*
c. *wget https://storage.googleapis.com/models-hao/vgg16-ssd-mp-0_7726.pth*

Skrypt uruchomiony w domyślny sposób będzie poszukiwał punktów początkowych w katalogu *models. Opis parametrów:*

- *--model-dir* – ścieżka do katalogu, w którym zostanie zapisany nauczony model
- *--data – ścieżka do folderu głównego z danymi wejściowymi w formacie Pascal VOC*
- *--batch-size – ile obrazów jednocześnie używać do treningu (im więcej tym szybszy trening), domyślnie 4*

*Przykładowe polecenie (**NIE PRZEPISYWAĆ BEZ ZASTANOWIENIA**):*

python3 train_ssd.py --dataset-type=voc --data=data/custom --model-dir=models/custom_SSD_v2_lite --batch-size=3 --pretrained-ssd=models/mb2-ssd-lite-mp-0_686.pth --net mb2-ssd-lite

W przypadku uruchomienia środowiska graficznego może zabraknąć pamięci na urządzeniu. Aby zwolnić pamięć można ograniczyć *batch-size* do 2 lub 1, ewentualnie zamknąć powłokę graficzną poleceniem *sudo init 3* (przywrócenie *sudo init 5*) i wykonać uczenie zdalnie przez SSH.

Model należy uczyć przez co najmniej 20-30 epok. Im więcej zdjęć obiektów i w zależności od zakładanej dokładności wykonania parametr ten można zwiększyć do 50-70.

Ad. 4

Po wytrenowaniu modelu należy przekonwertować go do formatu ONNX (https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect-detection.md#training-your-model).

Przykładowe polecenie *(**NIE PRZEPISYWAĆ BEZ ZASTANOWIENIA**)*:

python3 onnx_export.py --labels=models/custom_SSD_v2_lite/labels.txt --model-dir=models/custom_SSD_v2_lite/ --net=mb2-ssd-lite

Wczytany model można przetestować globalnym poleceniem *detectnet.* Pierwsze uruchomienie modelu będzie trwało ok 1-2 minut, w tym czasie następuje jego konwersja do bardziej optymalnego formatu (obok pliku .onnx powstanie nowy o tej samej nazwie, ale innym rozszerzeniu).

WAŻNE: od teraz do wykonania modelu należy używać pliku labels.txt znajdującego się w katalogu z utworzonym modelem. Plik ten zawiera dodatkową klasę BACKGROUND i nie należy jej usuwać.

Ad. 5

Model przekonwertowany do formatu ONNX (rozszerzenie .onnx) można wykorzystać bezpośrednio w kodzie znajdującym się na repozytorium (katalog AI). Podany tam został przykład jego uruchomienia i interpretacji wyników. Wczytanie pliku labels.txt i interpretacja wykrytej klasy jest zadaniem do wykonania samodzielnego. Kod można wykorzystać bezpośrednio w opracowanych na potrzeby zadania 1 rozwiązaniach, został napisany tak, aby korzystać z JetsonVideoStream (tak, jak kody przykładowe).