Wrocław, 11 listopada 2024

Przemysł 4.0

Laboratorium 5

System detekcji gestów cz. 1

prowadzący: *Dr inż. Radosław Idzikowski*

1 Wprowadzenie

Celem laboratorium jest budowa systemu do detekcji gestów. Zadanie będzie podzielone na dwa spotkania: (1) budowa zbioru danych oraz trening modelu i (2) uruchomienie modelu na `Raspberry Pi`. Praca będzie oceniana na bieżąco w trakcie zajęć wraz z postępem wykonywania zadań. Po ukończeniu każdego zadania należy zawołać prowadzącego w celu zaakceptowania etapu. Jednak ostatecznie całość zostanie ocenione na koniec drugiego spotkania.

Wyzwaniem treningu sieci neuronowych jest otrzymanie modelu z pomocą którego możliwe będzie uzyskanie wystarczająco dobrych wyników na wcześniej nie wykorzystanych danych – taką korzystną cechę modelu w literaturze [1] określa się jako zdolność do generalizacji (*generalization*). Jest ona bardzo ważna ze względu na praktyczne ograniczenia uczenia maszynowego: dane wykorzystane w ramach treningu przeważnie są jedynie ograniczoną, małą próbką w porównaniu do ogromnej populacji z której pochodzą. Co gorsze próbka taka może być jeszcze obarczona błędami lub naturalnie występującym szumem związanym z procesem pomiarowym. Świetny wynik wyłącznie na zbiorze treningowym nie jest celem uczenia maszynowego – szczególnie w sytuacji, gdzie znajomość poprawnych rozwiązania dla próbki jest konieczna do samego rozpoczęcia procedury treningu w ramach uczenia nadzorowanego.

2 Zadania

W ramach zajęć należy w zespołach wykonać następujące zadania:

1. Zebranie zdjęć do treningu.
2. Oznaczenie zdjęć.
3. Trening modelu.
4. Wdrożenie systemu do detekcji gestów na `Raspberry Pi`.

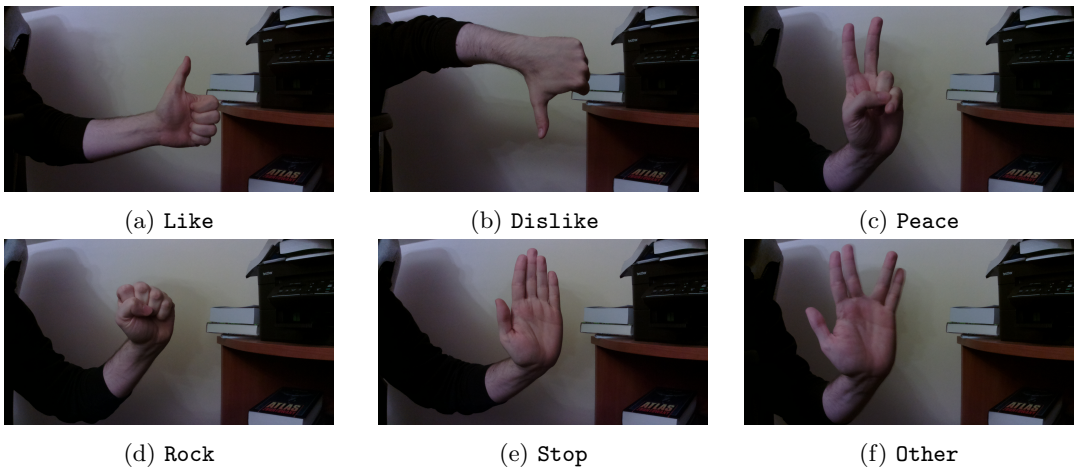
Zadania od 1 do 2 należy wykonać podczas pierwszego spotkania. Zadania 3 i 4 jest przewidziane na drugim oraz trzecim terminie. Na ocenę 3.0 należy zrobić poprawnie działający model na komputerze (zadania 1-3). Za uruchomienie wytrenowanego modelu na `Raspberry Pi` jest przewidziana ocena z zakresu od 4.0 do 5.0 w zależności od skuteczności modelu i liczby rozpoznawanych gestów.

3 Opis zadań

3.1 Zadanie 1

W ramach zadania będzie trzeba zebrać zdjęcia niezbędne zbudowania zbioru danych (`dataset`), który zostanie później wykorzystany podczas trenowania sieci neuronowej. Podczas zajęć należy zebrać następujące gesty: (a) `like`, (b) `dislike`, (c) `peace`, (d) `rock`, (e) `stop` oraz (f) `other`

(klasa zbiorcza, każdy inny gest wykonany dłonią traktujemy jako przynależny do tej kategorii, ponadto należy się ograniczyć do gestów niewulgarnych). Zdjęcia przykładowych poprawnych gestów dla każdej klasy przedstawiono na Rys. 1.



(a) `Like` (b) `Dislike` (c) `Peace`

(d) `Rock` (e) `Stop` (f) `Other`

Rysunek 1: Gesty do zebrania w ramach laboratorium

Zdjęcia należy zbierać według ścisłych wytycznych, ponieważ nietrzymanie się ich może powodować trudności w kolejnych krokach zadania. Wytyczne brzmią następująco:

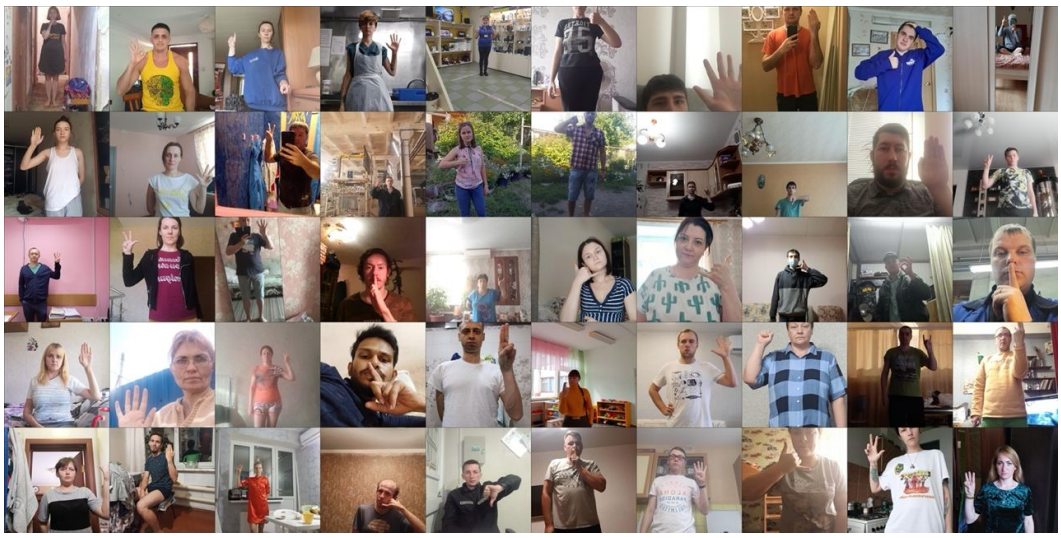
- zdjęcia należy zebrać z użyciem własnego telefonu komórkowego, ale z ograniczoną rozdzielczością (w pełni wystarczy 1920×1080);
- zdjęcia powinny być jak najbardziej zróżnicowane (oczywiście przy naturalnym uwzględnieniu ograniczeń wynikających z warunków) – chodzi między innymi o zróżnicowanie: odległość dłoni od kamery, jej pozycji od centrum obrazu, kąta pod jakim gest został uchwycony, widoczną stronę dłoni czy też tło zdjęcia tak jak pokazano na Rys. 2);
- dla zachowania porządku zdjęcia muszą być nazywane według następującej konwencji:

  `{day}_RP{pi}_{name}_{id}_{category}.jpg`

  gdzie `day` – dzień zajęć, `pi` – numer urządzenia z którego grupa korzysta, `name` – unikalna nazwa osoby wykonującej gest, `id` – numer zdjęcia, `category` – wykonany gest, przykładowo: `wt_RP7_kasztan_002_like.jpg`;
- w celu zwiększenia różnorodności, na tym etapie można się wymieniać zdjęciami między grupami, ale proszę o rozwagę, ponieważ w kolejnych korkach każda grupa będzie musiała samodzielnie wykonać ich oznaczenie (będzie to miało wpływ na działanie modelu, co w konsekwencji będzie się przekładać na ocenę końcową);
- w celu zachowania prywatności, warto **nie zbierać** zdjęć na których widoczna jest twarz (a w zależności od preferencji również sylwetka) – narzędzia, które będą wykorzystane sprawią, że zdjęcia staną się publiczne. Jak powszechnie wiadomo, internecie nic nie ginie;
- *dla chętnych* na zebranych zdjęciach można zastosować augmentację – proces sztucznego rozszerzania zbioru treningowego poprzez przekształcenie oryginalnych danych.

Jednym z nowych przykładów poprawnego wykonania zbioru danych jest publiczny zbiór danych (HaGRID)[2] do rozpoznawania gestów z badań naukowych – ponad 500 tysięcy przykładów dla 18 gestów. Przykładowe zdjęcie próbek w ramach zbioru pokazano na Rys. 3.

Rysunek 2: Próba zróżnicowania zdjęć w ramach pojedynczego gestu `like` – próba zmiany pozycji dłoni, jej odległości od kamery, jej kąta czy też tła



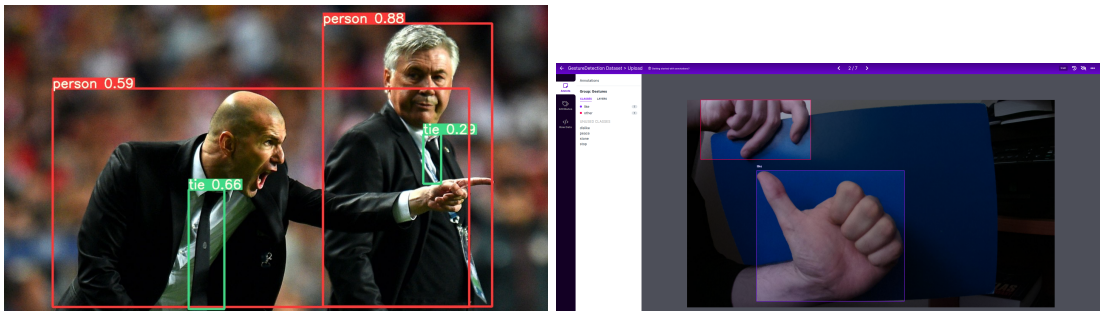
Rysunek 3: Przykłady zróżnicowania zdjęć ze zbioru *HaGRID* do rozpoznawania gestów

3.2 Zadanie 2

Celem kolejnego zadania jest oznaczenie zebranych zdjęć. **Poprawne** oznaczanie zdjęć jest niezbędne, aby możliwe było wytrenowanie sieci neuronowej do zadania detekcji obiektów (w naszym wypadku: detekcji gestów). W tym celu potrzebne są dane wejściowe (zdjęcia, które zostały zebrane w kroku poprzednim) oraz oczekiwane wyjście jakie sieć neuronowa ma zwrócić. Należy więc kolejno do **KAŻDEGO** zdjęcia dodać informację o oczekiwanym wyjściu w postaci `bounding boxes`, czyli **minimalnego prostokąta w którym zawiera się obiekt** jak na Rys. 4a oraz 4b.

Do oznaczania należy skorzystać z narzędzia `roboflow annotate`. W tym celu konieczne jest założenie darmowego konta, ale można się zalogować z użyciem konta `gmail` (preferowane konto studenckie). Po zalogowaniu zostaniemy poproszeni o utworzenie obszaru roboczego (*workspace*, dla ułatwienia dostępu można zrobić publiczny obszar roboczy), do którego można zaprosić innych użytkowników (w tym **obowiązkowo prowadzącego**). Następnie należy utworzyć nowy projekt, w kolejnym korku wybieramy detekcję obiektów (*Object Detecion*), proszę nie zmieniać tupu licencji. Nowe zdjęcia do projektu wrzucamy w karcie `Upload`. Zdjęcia może dodać poprzez: (1) upuszczenie ich , (2) zaznaczenie na dysku (`Select Files`) lub (3) wskazanie całego folderu (`Select Folder`). Po wybraniu zdjęć w celu ich przesłania należy kliknąć `Save and Continue`. Kolejne zdjęcia możemy przesłać w dowolnym momencie przechodząc zawsze do karty `Upload`.

Przesłane zdjęcia nigdy nie są oznaczone. Każde zdjęcia należy oznaczyć osobno w karcie `Annotate`. Na grupie zdjęć nieopisanych możemy wykonać operację `Assign Images`, co uruchomi



(a) Wynik nauczonego modelu YOLOv5 (b) Oznaczanie narzędziem `roboflow`

Rysunek 4: `Bounding boxes` w praktyce

edytor. Między zdjęciami możemy się przełączać na pasku u góry. Każde zdjęcie należy oznaczyć bardzo dokładnie przy użyciu narzędzia `Bounding Box Tools` (skrót klawiszowy `B`). Pojedynczy `Bounding Box` powinien **być minimalnym prostokątem zawierającym dany gest** (jak na Rys. 4b). Po zaznaczeniu `Bounding Box` **koniecznie musimy przypisać mu odpowiednią klasę!!!** (`like`, `dislikem peace`, `rock`, `stop`, `other`). Po oznaczeniu wszystkich nieopisanych zdjęć możemy się cofnąć strzałką w lewym górnym rogu do widoku projektu. Następnie należy dodać oznaczone zdjęcia do zbioru danych (opcja `Add X images to Dataset`). W przypadku dodawania większej liczby zdjęć możemy wybrać, aby system za nas przydzielił zdjęcia do odpowiednich zbiorów (zalecana opcja) lub ręcznie wybrać, do którego zbioru mają trafić oznaczone zdjęcia, mowa oczywiście o zbiorach: treningowym, walidacyjnym oraz testowym.

Po oznaczeniu wystarczającej liczby zdjęć możemy przejść do wygenerowania gotowego zbioru danych. Później będzie jeszcze możliwość tworzenia kolejnych wersji naszego zbioru. Przechodzimy do zakładki `Generate`. Mamy możliwość wykonania pewnego preprocessingu na naszym zbiorze, np: zmiana rozdzielczości (warto zmniejszyć w celu przyśpieszenia procesu trenowania, ponadto proszę zwrócić uwagę na różne ustawiania – rozciągnięcie, skalowanie, przycinanie itp.) czy przekonwertowanie na odcienie szarości wszystkich zdjęć itp. Następnie można użyć augmentacji (których dostarcza narzędzie `roboflow`, w pewnym stopniu ograniczone w darmowej wersji) w celu zwiększenia różnorodności zbioru, poprzez rotacje czy rozjaśnienie zdjęć.

Portal `roboflow` teoretycznie udostępnia możliwość trenowania modelu, jednak w wersji darmowej jesteśmy ograniczeni do trzech prób, więc w trakcie zajęć zostanie użyta inna metoda. Do kolejnego etapie utworzony zbiór danych należy wyeksportować (`Export Dataset`). Używany przez nas z kategorii `TXT` format to `YOLO v5 PyTorch` jako archiwum `zip` jeśli chcemy pobrać zbiór bezpośrednio na dysk lub `show download code` możemy pobrać później z poziomu języka `PYTHON` i pakietu `robofow` (wygodne w pracy z `Google Colab`).

3.3 Zadanie 3

W ramach zadania należy wytrenować model `Yolo5` z wykorzystaniem narzędzia `Google Colab` na utworzonym wcześniej zbiorze danych. W pierwszej kolejności należy utworzyć kopię projektu Google Colab i upewnić się, że mamy włączone korzystanie z `GPU`. Proszę przejść do `Ustawienia notatnika` w zakładce `Edytuj`, w polu `Akcelerator sprzętowy` powinien być wybrana opcja `T4 GPU`. Należy postępować zgodnie z kororkami wewnątrz `notebook`. Pierwsza komórka dotyczy przygotowania środowiska, a kolejna pobrania zbioru danych. W ostatnim akapicie poprzedniego rozdziału opisano sposób pobrania zbioru danych. Jednak należy się upewnić, że kod pobiera dane w prawidłowym formacie. Ostatnia linia powinna wyglądać następująco:

```
dataset = version.download("yolov5")
```

W przeciwnym wypadku, kolejne komórki nie zadziałają. Po wybraniu wersji `yolov5` domyślne może proponować wersje `obb` (*oriented bounding boxes*), która różni się formatem danych.

Pierwszym krokiem przed przystąpieniem do treningu jest wybór odpowiedniego modelu `YOLOv5` (`Nano`, `Small`, `Medium`, `Large`, `XLarge`). W naszym przypadku najlepszym rozwiązaniem

będzie wersja `Small`. Przy wywołaniu metody do trenowania trzeba określić następujące parametry: (1) `img` – rozmiar obrazu, domyślna wartość 640, przy bardziej złożonych obrazach warto trenować w wyższych rozdzielczościach, (2) `batch` rozmiar partii danych w epoce, (3) `epochs` – liczba epok, (4) `data` lokalizacja zbioru danych, w pliku `.yaml` są podane względne ścieżki do zbiorów, (5) `weights` – wagi początkowe, warto użyć wstępnie wytrenowanych niż całkowicie losowych. Poniżej przedstawiono przykładowy kod uruchomienia treningu:

```
!python train.py
                --img 640
                --batch 16
                --epochs 10
                --data GestureDetection-3/data.yaml
                --weights yolov5s.pt
```

Każde uruchomienie skryptu `train.py` powoduje utworzenie nowego eksperymentu w folderze `run/train/expX`, gdzie `X` to numer kolejnego uruchomienia. We wspomnianym folderze mamy dostęp do wyników (w formie wykresów) oraz wyuczonego modelu (folder `weights`). W ramach zadania warto przeanalizować przebieg różnych krzywych.

Istnieje możliwość wykonania ręcznie detekcji z wykorzystaniem skryptu `detect.py`. W tym celu musimy wczytać ścieżkę do wag zwróconych przez wyuczony model, rozdzielczość, poziom pewności, plik lub folder ze zdjęciami.

```
!python detect.py
                    --weights runs/train/exp5/weights/best.pt
                    --img 640
                    --conf 0.4
                    --source plik.jpg
```

Analogicznie jak do skryptu trenującego, w tym wypadku każde uruchomienie również powoduje utworzenie nowego eksperymentu w folderze `run/detect/expX`. W tym wypadku możemy zobaczyć wykryte obiekty na każdym zdjęciu z osobna wraz z określonym poziomem pewności.

3.4 Zadanie 4

Celem zadania będzie wdrożenie systemu na `Raspberry` Pi, a w kolejnym kroku uruchomienie wyuczonego modelu detekcji obiektów w czasie rzeczywistym. Zadanie zostanie rozpisane w ramach osobnej instrukcji na kolejnych zajęciach.

Literatura

[1] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[2] A. Kapitanov, A. Makhlyarchuk, and K. Kvanchiani. Hagrid-hand gesture recognition image dataset. arXiv preprint arXiv:2206.08219, 2022.

Zadanie opracowanie wspólnie z *dr inż. Teodorem Niżyńskim.*