Głębokie sieci neuronowe

Sterowowanie prezentacją slajdów w programie Google Slides za pomocą gestów dłoni

Konrad Sochacki 227958

Wstęp

Celem projektu było wykonanie programu, zdolnego przetwarzać obraz z kamerki tak, aby za pomocą gestów dłoni można było sterować prezentacją w programie Google Slides.

Wykorzystane technologie i bilbioteki:

- DroidCam – podłączenie Androida jako kamerki do komputera
- Python, Anaconda, cv2, numpy, tensorflow, pynput – detekcja, klasyfikacja, sterowanie
- Google Slides – internetowa aplikacja do prezentacji slajdów

Zadanie dzieli się na dwie zasadnicze części:

- detekcja i wyodrębnienie obszaru dłoni na obrazie
- klasyfikacja gestu

Obliczenia do powyższych zadań realizowano na karcie graficznej RTX 2060.

Detekcja dłoni

Do detekcji dłoni został wykorzystany gotowy model z internetowego źródła Real-time Hand-Detection using Neural Networks (SSD) on Tensorflow. W celu jego sprawnego wytrenowania zastosowana została metoda „transfer learning”. Nie był on trenowany od zera, ale wzięto inny model, dobrze wstępnie przetrenowany do danej dziedziny zadań - „ssd_mobilenet_v1_coco”. Następnie wytrenowano tylko ostatnie warstwy, do uściślenia celu - detekcji dłoni. Jako bazę danych do tego ostatniego kroku wykorzystany został „The Egohands Dataset”.

Uzyskany model został zapisany jako plik .pb, „inference graph”. Po jego pobraniu i wczytaniu możemy go użyć do detekcji dłoni. Jako wejście podajemy obraz. Jako wyjście otrzymujemy „bouding boxes” - pozycje obwodów obszarów zawierających dłonie, korespondujące z nimi wskaźniki pewności detekcji, klasy oraz ilość wykrytych obiektów.

Przed przekazaniem obrazu do klasyfikatora wykonywane są jeszcze 2 czynności. Za pomocą klasy „MOG2 Background Subtractor” biblioteki cv2 wycinane jest tło obrazu. Następnie „bouding boxes” wycinane są na takim obrazie pozbawionym tła. W ten sposób niskim kosztem znacząco ułatwione zostało zadanie klasyfikacji.

Aby uniknąć opóźnień do detekcji przekazywana jest co dziesiąta klatka.

Klasyfikacja gestu

Zadaniem klasyfikatora było rozróżnianie 7 gestów dłoni. Na wejściu otrzymywał on obraz w wymiarach 200x200. Eksperymentowano z różnymi modelami sieci, najlepiej sprawdził się poniżej przedstawiony, z trzema warstwami konwolucyjnymi:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 200, 200, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 100, 100, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 100, 100, 64)      18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 50, 50, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 50, 50, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 80000)             0
_________________________________________________________________
dense (Dense)                (None, 128)               10240128
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 903
=================================================================
Total params: 10,334,279
Trainable params: 10,334,279
Non-trainable params: 0
_________________________________________________________________
```

Prostsze modele uczyły się wolniej i osiągały gorsze końcowe wyniki. Przetestowano również bardziej złożony model z 4 warstwami konwolucyjnymi oraz normalizacją między ukrytymi warstwami. Nie udało się na nim osiągnąć żadnych sensownych wyników, zatem zaprzestano eksperymentów z bardziej złożonymi modelami.

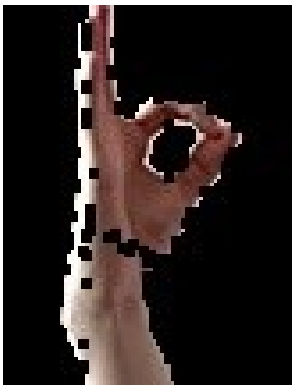
Danymi do uczenia był zestaw własnoręcznie zrobionych zdjęć gestów z wyciętym tłem. Każda klasa miała 500 próbek, w sumie 3500 zdjęć dla wszystkich klas. Detektor dłoni zwracał nierówne rozmiarowo obrazy, dlatego każdy z nich konwertowany był do zadanych wymiarów. Do sieci dostarczane były za pomocą generatora. Nie zastosowano augmentacji obrazu, wydawała się ona nadmiarowa.

Poniżej przykładowe zdjęcia:

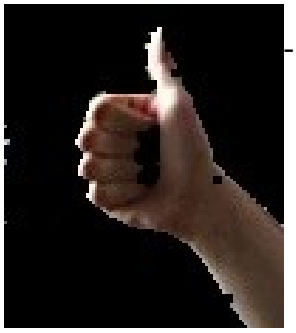
- „right” - slajd do przodu

- „left” - slajd do tyłu

- „close” - pozycja neutralna



- „play” - start / stop media



- „volume up” - podgłośnienie dźwięku

- „volume down” - ściszenie dźwięku

- „open” - zapasowy gest

- „pointer” - wskaźnik

Ostatecznie wybrany model osiągał bardzo zadowalające wyniki już po 3 epokach, trenowany był w 10. Nauka trwała ok. 1.5 minuty. „Value accuracy” było bliskie lub równe 100%. Niestety „na żywo” klasyfikacja sprawdzała się już nieco gorzej. Gest „play” wymagał chwili przećwiczenia przez osobę sterującą programem. Znak „volume down” okazał się trudny do rozróżnienia od znaku „pointer”. Dlatego został zastąpiony przez łatwo rozróżnialny „open”. Ponadto „pointer” - wskaźnik, okazał się niepraktyczny w użyciu ze względu na zbyt wolne przetwarzanie obrazu przez detektor dłoni.

Wyniki skuteczności całości programu w praktyce widoczne są w załączonym nagraniu.

Źródła

How to Build a Real-time Hand-Detector using Neural Networks (SSD) on Tensorflow

Using Deep Learning and CNNs to make a Hand Gesture recognition model

Using Deep Learning and CNNs to make a Hand Gesture recognition model

How to Classify Photos of Dogs and Cats (with 97% accuracy)