AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA Infrormatyki

PROJEKT INŻYNIERSKI

Rozpoznawanie ekspresji twarzy w oparciu o sieci neuronowe konwolucyjne

Dokumentacja użytkownika

Autorzy: Jacek Obłaza, Dmytro Petruk
Kierunek studiów: Informatyka
Opiekun pracy: Dr hab. Bogdan Kwolek

Kraków, 2016

1. Wymagania biblioteki

Do poprawnego działania nasza biblioteka potrzebuje interpretera Pythona w wersji co najmniej 2.7. Oprócz tego potrzebne są takie biblioteki jak OpenCV2, Theano, Lasagne oraz wszystkie, których one same wymagają.

2. Korzystanie z biblioteki

2.1. Klasyfikowanie obrazu

Aby przeprowadzić klasyfikację ekspresji twarzy na danym obrazie należy najpierw przygotować dane wejściowe do sieci. W tym celu należy użyć funkcji *load_img(img_path).* Wejściem tej funkcji jest ścieżka do naszego obrazu. Następnie należy przygotować sieć za pomocą funkcji *build_cnn(model_path).* Jako argument należy podać ścieżkę do pliku z modelem wcześniej wytrenowanej sieci. Mając gotowe te dwa elementy można już rozpocząć rozpoznawanie ekspresji twarzy przy pomocy funkcji *evaluate(network, faces_matrix).* Wyjściem jest macierz o wymiarach Nx7, gdzie N to ilość wykrytych twarzy na obrazie, a 7 to nasycenie każdą z siedmiu rozpoznawalnych ekspresji w kolejności: złość, pogarda, zniesmaczenie, strach, radość, smutek, zaskoczenie. Jeżeli jedna z wartości dla danego obrazu jest zdecydowanie większa od innych, to możemy przyjąć odpowiadającą jej ekspresję jako rozpoznaną. Można też przyjąć jakiś stały poziom (sugerowane jest 0.8), powyżej którego musi być wartość w macierzy, abyśmy uznali ekspresję za rozpoznaną.

2.1.1. Przykład zastosowania

Takie zdjęcie zostało dostarczone bibliotece, do rozpoznania ekspresji twarzy:

Wynikiem były takie wartości przy każdej z emocji:

anger = 0.07%
contempt = 0.00%
disgust = 0.01%
fear = 0.00%
happy = 99.89%
sadness = 0.02%
surprise = 0.00%

Jak widać rozpoznaną emocją jest radość, co pokrywa się z naszymi przypuszczeniami.

Kod do przykładu:

```
faces = load_img("zdjecie.jpg")          # tworzenie tablicy twarzy z podanego zdjęcia
network = build_cnn('model.npz')         # budowanie sieci na podstawie dostarczonego modelu

tab = evaluate(network, faces)           # otrzymywanie predykcji sieci

print ("                                 #wypisanie odpowiednio sformatowanych danych na
        anger = {:.2f}%\n                # standardowe wyjście
        contempt = {:.2f}%\n
        disgust = {:.2f}%\n
        fear = {:.2f}%\n
        happy = {:.2f}%\n
        sadness = {:.2f}%\n
        surprise = {:.2f}%\n".format(tab[0][0]*100, tab[0][1]*100, tab[0][2]*100, tab[0][3]*100,
                tab[0][4]*100, tab[0][5]*100, tab[0][6]*100))
```

2.2. Uczenie sieci

Uczenie powinno się odbywać kiedy dla ponad połowy twarzy ekspresja rzeczywista ma wartość nasycenia podaną przez sieć mniejszą niż 60%. Czyli jeżeli dla zdjęć 20 smutnych twarzy 11 z nich będzie miało w szóstej komórce odpowiadającej jej tabeli wynikowej, otrzymanej z naszej sieci, wartość mniejszą niż 0.6.

Aby uczyć sieć na własnych przykładach należy przygotować wcześniej odpowiednią strukturę katalogów. W katalogu głównym utworzyć 2 katalogi. Jeden na zdjęcia, a drugi na podpisy z emocjami na odpowiednich zdjęciach. Każde zdjęcie, bądź też grupa zdjęć reprezentujących jedną ekspresję powinna znajdować się w jednym katalogu umieszczonym w katalogu na zdjęcia. W drugim katalogu na emocje powinien znaleźć się katalog o takiej samej nazwie jak ten z odpowiednim zdjęciem/zdjęciami, a w nim plik tekstowy z podaną wartością ekspresji, którą reprezentują zdjęcia.

Zalecane jest aby do zbioru danych dołączyć zapewniony przez nas zbiór, aby sieć była dalej przystosowana do różnych twarzy, nie tylko dostarczonych przez użytkownika. Proces staje się wtedy bardzo czasochłonny, ale zapewnia uniwersalność sieci. Jeżeli zależy nam na czasie nie należy uczyć sieci dłużej niż 3 epoki z wymienionych wcześniej względów.

Przykładowa struktura:

home

| -Data
| -Images
| - 1
photo1.jpg
photo2.jpg
| - 2
photo1.jpg
.
.
.

| - Labels
| - 1
label.txt
| - 2
label.txt
.
.
.

Mając taką strukturę katalogów należy wywołać funkcję *train_net(datadir, imagedir, labeldir, network, epochs = 1000, save_each = False)*, gdzie:
- datadir jest ścieżką bezwzględną naszego folderu z danymi (w przypadku przykładu "\\home\\Data\\"),
- imagedir jest ścieżką z obrazami względem datadir (w przypadku przykładu "Images\\"),
- labeldir ścieżką z podpisami względem datadir(dla przykładu "Labels\\"). Pliki tekstowe z opisami obrazów muszą zawierać tylko i wyłącznie liczbę odpowiadającą reprezentowanej ekspresji:
1 - złość, 2 - pogarda, 3 - zniesmaczenie, 4 - strach, 5 - radość, 6 - smutek, 7 - zaskoczenie.
- network - załadowana wcześniej sieć,
- epochs - ile epok chcemy aby trwało uczenie (domyślnie 1000),
- save_each - flaga mówiąca o tym, czy chcemy zapisywać model co każdą epokę (uwaga: może to zająć dużo miejsca na dysku, gdyż każdy model to około 30MB).

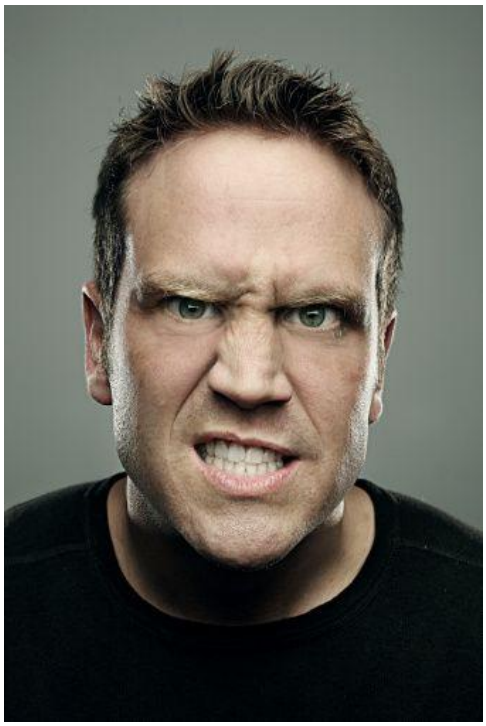
Sieć zostanie zapisana do pliku custom_model.npz (lub custom_model<nr epoki>.npz jeśli wybraliśmy opcję zapisywania co epokę). Można go wczytać tak samo jak dostarczony przez nas model sieci.

Trening sieci może zakończyć się automatycznie wcześniej niż zadeklarowana liczba epok, jeżeli różnica między wartością funkcji błędu dwóch kolejnych epok będzie mniejsza niż 0.0001.

3. Przykłady działania dla różnych obrazów

anger = 0.49%
contempt = 0.02%
disgust = 97.62%
fear = 1.47%
happy = 0.00%
sadness = 0.40%
surprise = 0.00%

anger = 0.00%
contempt = 0.00%
disgust = 0.00%
fear = 0.00%
happy = 0.00%
sadness = 0.08%
surprise = 99.92%



anger = 100.00%
contempt = 0.00%
disgust = 0.00%
fear = 0.00%
happy = 0.00%
sadness = 0.00%
surprise = 0.00%

anger = 0.00%
contempt = 0.00%
disgust = 14.98%
fear = 0.00%
happy = 0.00%
sadness = 85.02%
surprise = 0.00%