18.05.2020

Podprojekt. Automatyczny kelner. Raport nr.4

Adam Wojdyła

1. Opis podprojektu:

Głównym celem realizowanego projektu jest implementacja funkcjonalności rozpoznawania obrazów przez kelnera tak aby posiłki przygotowane na kuchni zostały dostarczone do danych stolików na podstawie wyniku algorytmu

2. Metody uczenia:

a. Sieci neuronowe

3. Wykorzystane technologie

a. Tensorflow
b. Keras

4. Etapy projektu

Przygotowany projekt składał się z kilku etapów

a. Pobranie zbioru zdjęć pogrupowanych w klasy dań
b. Pogrupowanie zdjęć na dane do trenowania I do testowania
c. Implementacja sieci neuronowej
d. Trening sieci neuronowej zadanymi zdjęciami
e. Implementacja kodu do projektu, obliczanie predykcji za pomocą wytrenowanego modelu
f. Wykorzystanie modelu do rozpoznania dań przygotowanych przygotowanych na kuchni I wskazania stolika, który złożył takie zamówienie

5. Opis projektu.

Moim celem było osiągnięcie jak najlepszej predykcji, aby program był funkcjonalny dlatego zdecydowałem się wykorzystać konwolucyjne sieci neuronowe. Biblioteka Keras udostępnia częściowo przetrenowane modele na bazie wielu zdjęć, na które dokładamy kilka warstw, aby dostroić model do naszych danych co znacząco skraca proces uczenia.
W CNN wykorzystywany jest filtr (convolutional window), który aplikuje się na obraz I przemieszcza po całym obrazie. Każdy filtr zwraca mapę w jaki sposób filtr zgada się z danym obrazem (macierz z wartościami 0-1). Stosuje się wiele filtrów, które złożone ze sobą na stos tworzą warstwę konwolucyjną. W kolejnych krokach na wynikowych macierzach zastosowany jest tzw. Pooling, który wybiera maksymalne wartości z każdego okienka w macierzy. Następnie normalizujemy macierz funkcją Relu, która zamienia negatywne wartości na 0. Cały proces trzech kroków jest powtarzany, a wyniki z jednej warstwy stają się wejściem do kolejnej warstwy. Cała poprawność algorytmu opiera się na propagacji wstecznej, czyli aktualizowaniu wartości wag na podstawie funkcji obliczającej błąd między predykcją algorytmu a poprawną predykcją – badanie poprawności wyników. Na samym końcu następuje flattening, funkcja mapuje macierz na pojedynczą kolumnę, którą łączy z w pełni połączoną warstwą. Na podstawie wyliczonych wag obliczane wartość, która wskazuje na ile dane wejście przypomina oczekiwany wynik

6. Struktura projektu

Kelner/src/algorithms/CNN - Główny folder zawierający pliki potrzebne do wykonania algorytmu

PrepreData.py – plik odpowiedzialny za załadownie modelu poczas startu I przygotowanie danych do trenowania sieci neuronowej. Część kodu jest wykomentowana, ponieważ przygotowanie danych zostało wykonane wcześniej.

TrainModel.py – plik, w którym zaimplementowane jest uczenie sieci neuronowej z wykorzystaniem Tensorflow I Keras. Również niewykorzystywane, ponieważ uczenie sieci zajęło mi na GPU NvidiaMX150 około 7h.

PredictClass.py – Tu znajduje się klasa Predictor. Zawiera metodę predict_class, która przyjmuje ścieżkę zdjęcia I zwraca rozpoznaną klasę (Wyświetla tylko na terminalu) .

trainedModels/big_model_trained_3class.hdf5 – Wytrenowany model konwolucyjnej sieci neuronowej

CNNFood.ipynb – Notatnik z jupyter notebook, który został wykorzystany do trenowania sieci jak I przygotowania danych

Kelner/images/testDishes – folder z testowymi zdjeciami do rozpoznania przez kelnera.

!! Nie umieściłem w repozytorium zdjęć do uczenia sieci z powodu dużej wagi !!

7. Udział podprojektu w projekcie

Dzięki algorytmowi do rozpoznawania obrazów, kelner stał się prawdziwie inteligentny I potrafi rozpoznać zamówienie przygotowane na kuchni, sprawdzić predykcje na podstawie dostępnych zamówień, dopasować je do zamówień oczekujących stolików a następnie zanieść zamówienie do danego stolika.

Większość operacji odbywa się w klasie **WaiterManager**. Kelner sprawdza czy ilość jego zaakceptowanych zamówień przekroczyła daną liczbę. Jeśli tak, to zmienia się „target" kelnera I zanosi je do kuchni. Jeśli na kuchni jest zamówienie, to jest ono wyświetlane a następnie rozpoznane **self.predict_order.** Rezta funkcji zostanie omówiona podczas prezentacji.

Uwaga: Model musi zostać załadowany na początku rozruchu programu tak później działał płynnie, więc start programu wydłużył się do około 30 sekund!