Wrocław, 11 grudnia 2022

Maciej Franikowski
Indeks 249436

Sprawozdanie z projektu

1 Temat i cel projektu

Temat: „Aplikacja mobilna pozwalająca na rozpoznawanie owoców, z możliwością obracania zdjęć"

Cel projektu: Stworzenie systemu pozwalającego użytkownikowi na rozpoznanie owoców z 3 klas - bananów, jabłek i pomarańczy. Ważną cechą systemu jest możliwość obracania zdjęć.

Kod źródłowy projektu jest przechowywany w repozytorium na GitHubie[5].

2 Wykorzystywane technologie

Do realizacji części projektu związanej z uczeniem maszyn wykorzystany został język Python[2]. Głównym narzędziem była biblioteka Keras[3], która jest teraz częścią technologii TensorFlow. Aplikacja mobilna napisana została w języku Kotlin[1].

3 Opis działania i funkcje systemu

Projekt składa się z dwóch oddzielnych części. Pierwsza z nich odpowiada za wytrenowanie oraz utworzenie sieci konwolucyjnej. Wynikiem działania tej części jest model TensorFlowLite[4], który może zostać zaimportowany do drugiej części, czyli aplikacji mobilnej. Pierwsza aplikacja oferuje możliwości testowania oraz prezentowania wyników działania sieci. Aplikacja, oprócz samego wytrenowania modelu, oferuje następujące funkcjonalności:

- Wyświetlenie obrazów z podpisami - oryginalna klasa oraz klasa która została przydzielona obrazowi w wyniku klasyfikacji.
- Wydrukowanie w konsoli, dla grupy elementów, wyników działania sieci w postaci liczbowej - tablicy 3 elementowej z wynikiem liczbowym, gdzie najwyższy oznacza przynależność do odpowiadającej klasy,
- Przeprowadzenie testu mającego na celu wykrycie próbek na których model się myli, w celu wyciągnięcia wniosków,
- Przeprowadzenie ewaluacji modelu na podstawie danych testowych,

4 Dane wejściowe

Dane wejściowe podzielone zostały na 3 kategorie:

- Dane do trenowania modelu, 460 obrazów, 70% wszystkich danych
- Dane do walidacji, 66 obrazów, 10% wszystkich danych. Są używane podczas trenowania modelu, w celu douczenia go na przykładach, których nie zna.
- Dane do testowania, 130 obrazów, 20% wszystkich danych.
- Przeprowadzenie ewaluacji modelu na podstawie danych testowych,

Rysunek 1: Przykładowy obraz jabłka ze zbioru uczącego

Dane są importowane do systemu, gdzie przetrzymywane są w dataset'ach jako obrazy w rozmiarze 32x32 pikseli.

5 Sieć oraz tworzenie modelu

Sieć utworzona jest za pomocą kilku warstw, gdzie początkowo obraz jest przetwarzany. Każda wartość piksela była wcześniej dzielona przez 255, aby otrzymać obraz w skali szarości, w celu poprawy szybkości trenowania modelu. Aktualnie te piksele nie są już przeskalowywane, a sieć uwzględnia kolor. Następnie obraz poddawany jest operacji konwolucji, gdzie wykorzystany filtr jest rozmiaru 3x3 piksela. Wartości tego filtra dobierane są przez sieć automatycznie. Po konwolucji, wynikowy obraz poddawany jest maksymalnemu poolingowi, czyli wybierani są z grup 4 pikseli przedstawiciele o najwyższej wartości i taka wartość przypisywana jest jednemu pikselowi w nowej, mniejszej macierzy. Powyżej opisane przekształcenia, czyli konwolucja oraz pooling, przeprowadzane są trzykrotnie w celu ekstrakcji cech. Kolejnym etapem jest spłaszczenie obrazu do jednowymiarowej macierzy oraz podanie tej macierzy do typowej sieci neuronowej. Ta sieć ma 2 warstwy, pierwsza ma 128 neuronów,

natomiast druga, wyjściowa - 3 neurony. W ten sposób otrzymujemy wynik w postaci 3 liczb, co pozwala na przypisanie klasy podanemu na wejście obrazowi. Następnie model jest kompilowany oraz trenowany wykorzystując dane do trenowania oraz walidacji. Trenowanie trwa 10 epok. W wyniku wyżej opisanych operacji otrzymujemy model, któremu można podawać kolejne obrazy do klasyfikacji, lub wyeksportować w celu wykorzystania w aplikacji mobilnej.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 32)          9248

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 32)         0
 2D)

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 128)               16512

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 36,291
Trainable params: 36,291
Non-trainable params: 0
```

Rysunek 2: Architektura sieci

6 Skuteczność utworzonego modelu

Po wytrenowaniu modelu, w momencie osiągnięcia 10-tej epoki, model osiągnął 97% procent trafności predykcji.

```
Epoch 1/10
23/23 [==============================] - 2s 22ms/step - loss: 4.6004 - accuracy: 0.5217 - val_loss: 1.1339 - val_accuracy: 0.7727
Epoch 2/10
23/23 [==============================] - 0s 15ms/step - loss: 0.7762 - accuracy: 0.8043 - val_loss: 0.5040 - val_accuracy: 0.8788
Epoch 3/10
23/23 [==============================] - 0s 15ms/step - loss: 0.4494 - accuracy: 0.8652 - val_loss: 0.3617 - val_accuracy: 0.8939
Epoch 4/10
23/23 [==============================] - 0s 14ms/step - loss: 0.2611 - accuracy: 0.9283 - val_loss: 0.5471 - val_accuracy: 0.8333
Epoch 5/10
23/23 [==============================] - 0s 15ms/step - loss: 0.3136 - accuracy: 0.9109 - val_loss: 0.1939 - val_accuracy: 0.9394
Epoch 6/10
23/23 [==============================] - 0s 15ms/step - loss: 0.1949 - accuracy: 0.9391 - val_loss: 0.2563 - val_accuracy: 0.8939
Epoch 7/10
23/23 [==============================] - 0s 14ms/step - loss: 0.1398 - accuracy: 0.9500 - val_loss: 0.2574 - val_accuracy: 0.9394
Epoch 8/10
23/23 [==============================] - 0s 14ms/step - loss: 0.0765 - accuracy: 0.9717 - val_loss: 0.3254 - val_accuracy: 0.8939
Epoch 9/10
23/23 [==============================] - 0s 15ms/step - loss: 0.0644 - accuracy: 0.9739 - val_loss: 0.1797 - val_accuracy: 0.9242
Epoch 10/10
23/23 [==============================] - 0s 16ms/step - loss: 0.0745 - accuracy: 0.9761 - val_loss: 0.2050 - val_accuracy: 0.9242
```

Rysunek 3: Trafność predykcji oraz pozostałe parametry modelu w trakcie uczenia

Kolejnym sposobem działania modelu, było wykorzystanie funkcji *evaluate* na modelu, podając mu zupełnie obcy dla niego zbiór danych - czyli zbiór testowy. Wynikiem tej ewaluacji był sukces na poziomie 93%.

```
Evaluation on the test dataset:
7/7 [==============================] - 0s 2ms/step - loss: 0.3213 - accuracy: 0.9308
```

Rysunek 4: Wynik ewaluacji na zbiorze testowym.

6.1 Przypadki dla których sieć popełnia błąd

Sprawdzono również konkretne obrazy dla których model popełnia błędy, w celu identyfikacji słabych punktów. Uruchomiane były testy dla modelu, podając mu obrazy testowe z jednej z klas, gdzie dla każdego uruchomienia zapisywane były obrazy, które zostały błędnie sklasyfikowane. Wyniki tych testów przedstawione są poniżej.

6.1.1 Jabłka

Najczęściej jabłka były identyfikowane jako banany w sytuacjach gdy zdjęcie było zrobione z góry, jabłko miało spory liść, lub widoczny fragment innego jabłka. Można z tego wywnioskować, że jabłka o nietypowych kształtach kwalifikowane są jako banany, które mają najmniej regularny kształt z wszystkich owoców.
Najczęściej jabłka były identyfikowane jako banany w sytuacjach gdy jabłko miało spory ogonek lub ogonek w nietypowym miejscu. Można z tego wywnioskować, że jabłka z sporymi ogonkami były mylone przez sieć z pomarańczami, ponieważ mają one charakterystyczne zakończenie.

(a) (b) (c) (d)

Rysunek 5: Jabłka zidentyfikowane jako banany

(a) (b) (c)

Rysunek 6: Jabłka zidentyfikowane jako pomarańcze

6.1.2 Banany

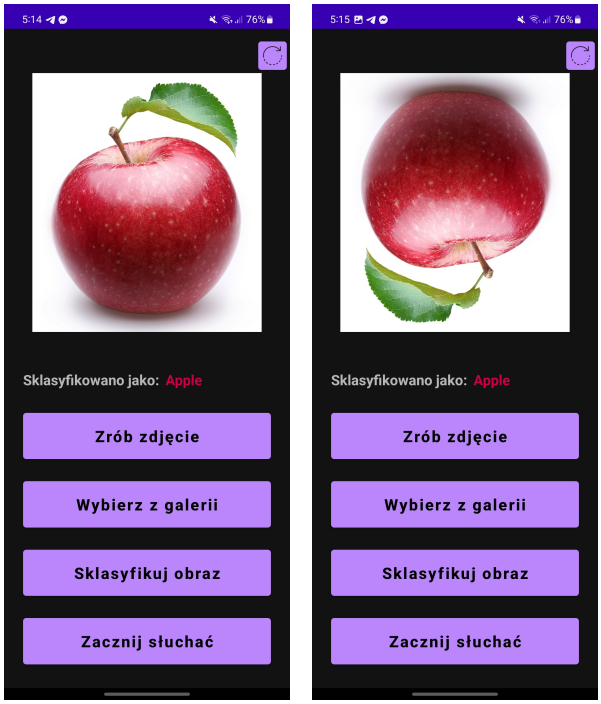
W przypadku bananów pomyłki były bardzo rzadkie, a obrazy na których mylił się model były nieregularne i nie powtarzały się.

6.1.3 Pomarańcze

Podobnie jak dla bananów, model nie miał problemu z danymi testowymi klasy pomarańczy. Dla 20 uruchomionych testów, zaobserwowano brak pomyłek.

7 Aplikacja mobilna

Stworzona aplikacja mobilna pozwala na wybranie zdjęcia z galerii, zrobienie go samodzielnie oraz przeprowadzenie klasyfikacji na załadowanym obrazie. Obecna jest również opcja obracania załadowanego obrazu, lecz po testach okazało się że ta funkcjonalność nie wpływa znacznie na wynik klasyfikacji. Aplikacja posiada również interfejs głosowy, któremu można wydawać komendy w celu uruchomienia wcześniej wspomnianych funkcjonalności.



(a) (b)

Rysunek 7: Wygląd aplikacji mobilnej

8 Podsumowanie

Realizacja projektu pozwoliła na praktyczne przypomnienie sobie informacji związanych z uczeniem maszynowym. Sporym plusem była możliwość wykorzystania biblioteki Keras, która była znacznie prostsza do wykorzystania niż rozwiązania z OpenCV. Działanie wytrenowanej sieci jest zadowalające i spełnia swoje zadanie. Zaskakujący był stały brak pomyłek modelu przy klasie pomarańczy. Dodatkowo proste okazało się utworzenie modelu, który można zaimportować do Android Studio i wykorzystanie tego modelu w aplikacji mobilnej.

Literatura

[1] Dokumentacja języka kotlin. https://kotlinlang.org/docs/home.html.

[2] Dokumentacja języka python. https://www.python.org/doc/.

[3] Dokumentacja narzędzia keras. https://keras.io/api/.

[4] Dokumentacja TensorFlow na temat modeli lite. https://www.tensorflow.org/lite/models/.

[5] Repozytorium projektu. https://github.com/MaciejFranikowski/Photo_vocalizer/.