Projekt - Wykrywacz ludzi palących

Martyna Kaczmarczyk

16 maj 2024

Spis treści

1 Wstęp

Poniższe opracowanie wyjaśnia działanie aplikacji rozróżniającej ludzi palących od niepalących oraz zawiera wyniki szkolenia modelu rozponawania na różnych etapach. Część backendowa kodu została napisana w pythonie, a część frontendowa - w react. Baza danych, na podstawie której był szkolony model zawiera 1221 obrazków i została ściągnięta ze strony keggle pod linkiem: $https://www.kaggle.com/datasets/sujaykapadnis/smoking$.

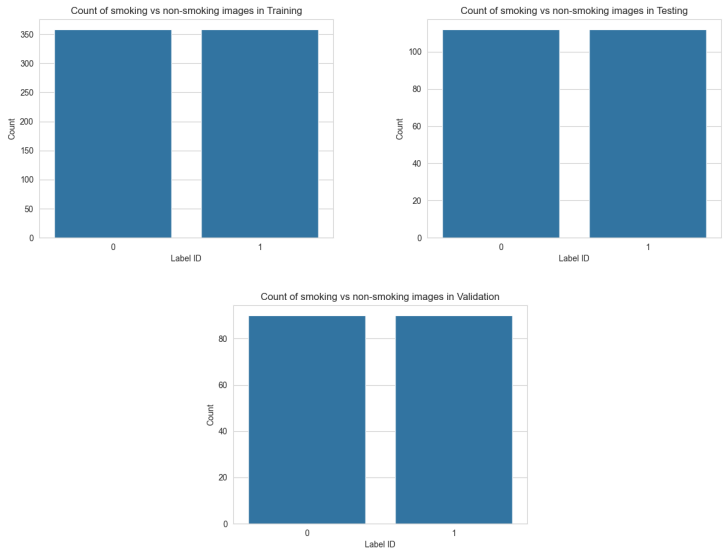
2 Działanie aplikacji

Aplikacja jest prosta w obsłudze. Aby ją uruchomić należy uruchomić serwer z poziomu termianala komendą: $nodeserver.js$, a następnie uruchomić frontendową część poleceniem: $npmstart$. Po uruchomieniu aplikacji należy wgrać jakieś zdjęcie, a następnie kliknąć przycisk "upload and predict".

Wtedy program wyśle zapytanie do servera i poprosi o predykcję obrazu: czy znajduje się na nim palacz, czy nie. Następnie serwer wyśle w odpowiedzi swoją predykcję, która zostanie pokazana poniżej zdjęcia.

3 Preprocessing

Podczas przygotowania danych najpierw zdjęcia są wypakowywane z archiwum *archive.zip*. Następnie zliczana jest ilość zdjęć w części trenigu (716) i testów (404). Następnie wykresy pokazują rozłożenie zdjęć: ile zawiera w sobie osób niepalących, a ile palących.



Następnie, program wykorzystuje bibliotekę TensorFlow do wczytania obrazu z pliku i dekodowania go w formacie JPEG z trzema kanałami (RGB). Po tym zmienia rozmiar obrazu do zdefiniowanej wartości IMAGE_SIZE, która wynosi w tym wypadku 224 oraz tasuje kolejność obrazów, by zapobiec uczenia się modelu w zachowanym porządku, i dzieli zbiór na mniejsze części (batche). Następnie dane są jeszcze raz obrabiane przez funkcje *augumentation* i *loadDatasetWithAugmentation*, gdzie na zdjęciach stosowane są różne operacje, takie jak losowe odwrócenie obrazu w poziomie, losowa zmiana jasności, kontrastu, nasycenia i odcienia.

4 Szkolenie modelu z funkcją aktywacyjną sigmoid

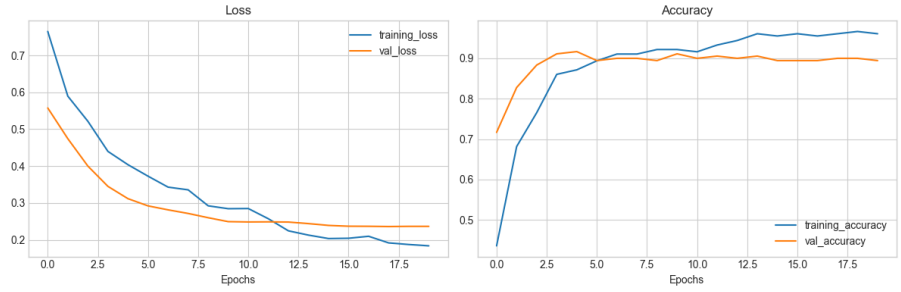
Fukcja sigmoid jest uznawana za najlepiej pasującą do szkolenia modelów o charakterze binarnym, dlatego uznałam, że od niej najlpeiej jest zacząć kształcić model..

4.1 Szkolenie pierwszego modelu

W tym etapie tworzony jest model z warstwą wejściową przyjmującą obrazy o wielkośći 224 i 3 kanałach (RGB). Do modelu dodane jest warstwa Dense z dwoma wyjściami, które są aktywowane funkcją "sigmoid". Ta warstwa jest odpowiedzialna za przewidywanie dwóch klas na podstawie cech wyekstrahowanych przez model.

Layer (type)	Output Shape	Param #
input layer (InputLayer)	(None, 224, 224, 3)	0
efficientnetv2-b0 (Functional)	(None, 7, 7, 1280)	5,919,312
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1280)	0
dense_2 (Dense)	(None, 2)	2,562

Na obrazku powyżej widnieje podsumowanie modelu sieci neuronowej użytego do treningu. Wykorzystuje architekturę EfficientNetV2B0 wyodrębniającą cechy z obrazków w połączeniu z warstwą GlobalAveragePooling2D, która uśrednia wszystkice kanały cech. Przekształca tensor(strukturę danych) o wymiarach (None, 7, 7, 1280) na tensor o wymiarach (None, 1280), czyli uśrednia po wysokościach i szerokościach każdej cechy, uzyskując w ten sposób jedną wartość dla każdego kanału cechy.
Po tym model jest kompilowany optymizatorem Adam z metryką "accuracy"i szkolony podczas n ilości epok, gdzie n wynosiło 20. Podczas trenowania epoch wykorzystywany jest callback ModelCheckpoint, który automatycznie zapisuje wagi modelu podczas treningu co epoch, jeżeli wyniki są lepsze od tych już zapisanych. Po ewaluacji tego modelu na danych testowych "accuracy"wynosi aż 89.99%, a starta danych - 28.04%.



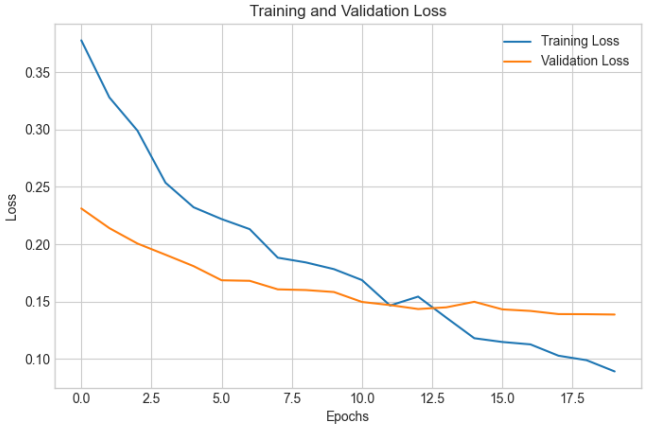
Na powyższym wykresie zestwione zostały malejące straty danych w zależności od epochu (których jest 20 w zamieszczonym przypadku) i rosące

accuracy - też zależne od epochów. Można zauważyć że krzywe uczenia danych treningowych i walidacyjnych się ze sobą w miarę przeplatają, jednak wspólnie rosną lub maleją w tym samym kierunku. To sugeruje, że model nie przejawia nadmiernego dopasowania (overfitting) ani niedopasowania (underfitting) do danych.

4.2 Szkolenie drugiego modelu

Podczas szkolenia pierwszego modelu do pliku zapisywane są wagi. Te oto wagi wczytywane są z wytrenowanego, pierwszego modelu do modelu drugiego. Dodatkowo, drugi model ma współczynnik uczenia ustawiony początkowo na 0.0001 oraz przeprowadzane jest trenowanie z różnymi współczynnikami uczenia za pomocą LearningRateScheduler.

Na wykresie pokazane jest jak zmienia się współczynnik uczenia. Jak widać, stopniowo maleje przy każdym epochu.



Tutaj z kolei widać jak jeszcze bardziej zmienisza się starta danych przy każdym z 20 epochów trenowania drugiego modelu.

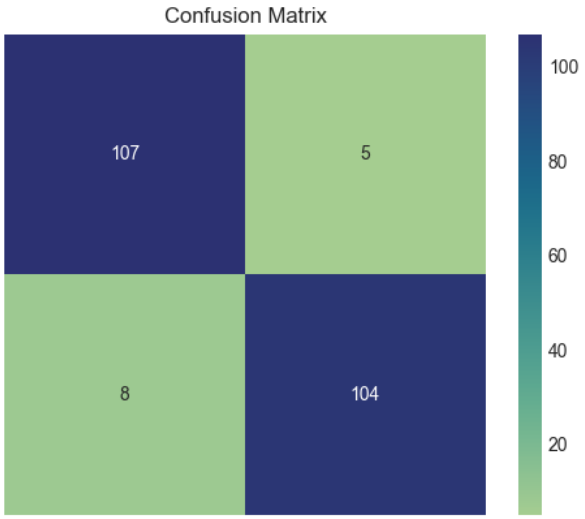
4.3 Testy działania

Do testowania losowych zdjęć użyte zostały obrazy z google. Pobierane są przez link url.

Smoking with probability 99.50520992279053
Non-Smoking with probability 0.6944173015654087

Smoking with probability 97.35099077224731
Non-Smoking with probability 3.0050326138734818

Smoking with probability 21.142558753490448
Non-Smoking with probability 80.61017990112305
Poniżej przedstawiona jest macierz błędów po przeprowadzeniu badań na danych testowych. Jak widać, większość zdjęć została prawidłowo sklasyfikowana. Jest tylko pięć osób niepalących, które przez program zostały uznane za osoby palące i 8 osób palacych uznanych za niepalące.



Accuracy dla drugiego modelu badane na danych testowych: 94.19642857142857%

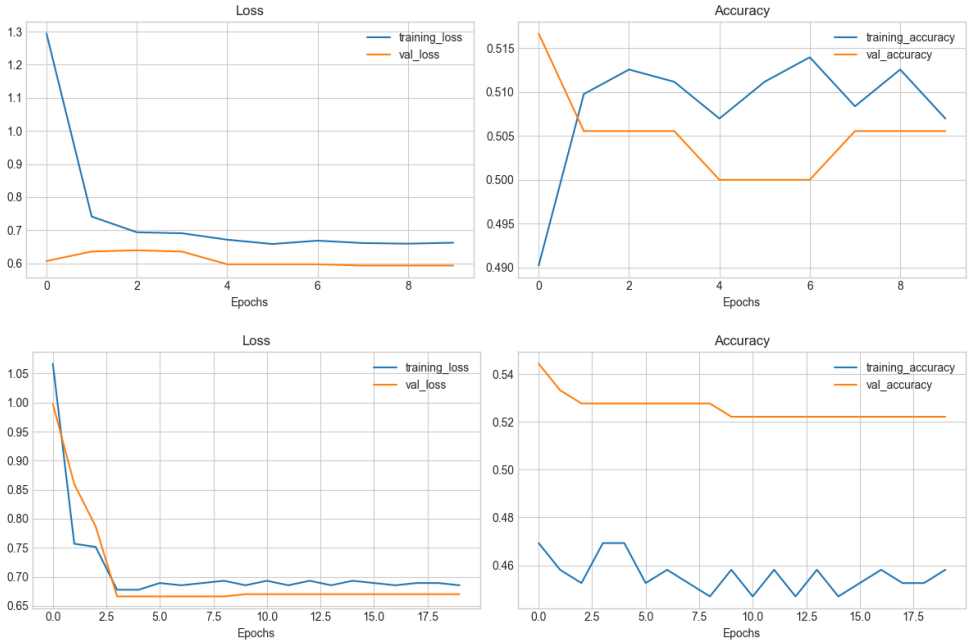
Incorrect classifications: 13

	precision	recall	f1-score	support
non-smoking	0.93	0.96	0.94	112
smoking	0.95	0.93	0.94	112
accuracy			0.94	224
macro avg	0.94	0.94	0.94	224
weighted avg	0.94	0.94	0.94	224

5 Szkolenie modelu z funkcją aktywacyjną relu

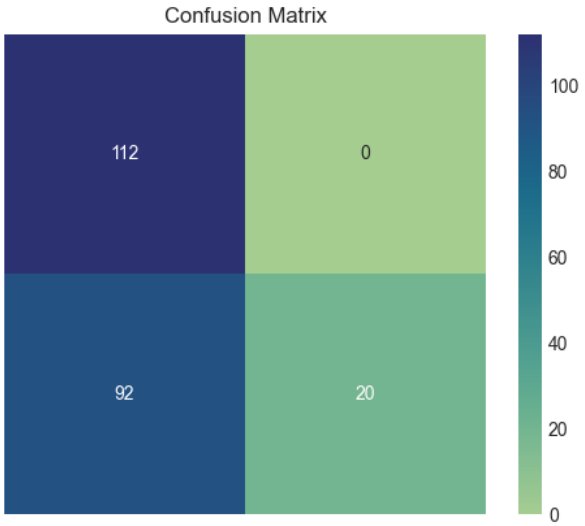
5.1 Szkolenie modelu

Ten model jest szkolony w ten sam sposób, co model opisany w sekcji powyżej. Jedyną różnicą jest zamienienie funkcji aktywacyjnej sigmoid z funkcją relu. Jak pokazują wyniki poniżej, accuracy bardzo drastycznie spadło.



W obu fazach szkolenia accuracy tego modelu graniczy przy 50% jak pokazują wykresy. Dodatkowo, krzywe uczenia się przelatają, jednak nu widać, by znacznie rosły bądź malały. Trzymają się jednej wartości.

5.2 Testy działania



Accuracy on the test dataset is 0.5892857142857143

Incorrect classifications 92

6 Podsumowanie

Jak widać, model działa całkiem dobrze. Zdarzają mu się pomyłki, jednak jego accuracy jest w miarę wysokie. Zakładam, że działałby znacznie lepiej, gdyby powiększyć treningową bazę danych o parę tysięcy zdjęć. Również można zauważyć, że dla klasyfkiacji binarnej lepiej działa dunkcja aktywacyjna sigmoid zamiast relu.