

Projekt - Wykrywacz ludzi palących

Martyna Kaczmarczyk

16 maj 2024

Spis treści

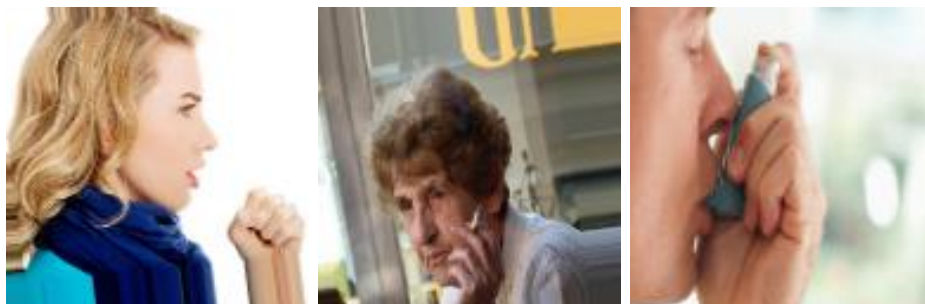
1	Wstęp	1
2	Działanie aplikacji	1
3	Szkolenie modelu	2
3.1	Preprocessing	2
3.2	Szkolenie pierwszego modelu	3
3.3	Szkolenie drugiego modelu	4
3.4	Testy działania	5
4	Podsumowanie	7

1 Wstęp

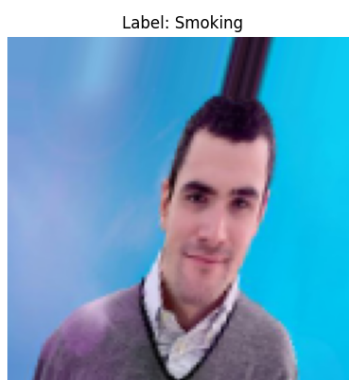
Poniższe opracowanie wyjaśnia działanie aplikacji rozróżniającej ludzi palących od niepalących oraz zawiera wyniki szkolenia modelu rozpoznań na różnych etapach. Część backendowa kodu została napisana w pythonie, a część frontendowa - w react. Baza danych, na podstawie której był szkolony model zawiera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

2 Działanie aplikacji

Aplikacja jest prosta w obsłudze. Najpierw 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 server wyśle w odpowiedzi swoją predykcję. Poniżej znajduje się druga część aplikacji, która umożliwia generowanie obrazków na podstawie bazy danych. Wystarczy kliknąć przycisk "get image", by strona servera wygenerowała jakiś obrazek i wysłała go na stronkę. Poniżej znajdują się przykładowe, wygenerowane obrazki:



Generator nie działa jednak idealnie. Zdarzają mu się przypadki, gdy myli badane elementu zdjęć, przez co generuje błędy takie jak te:

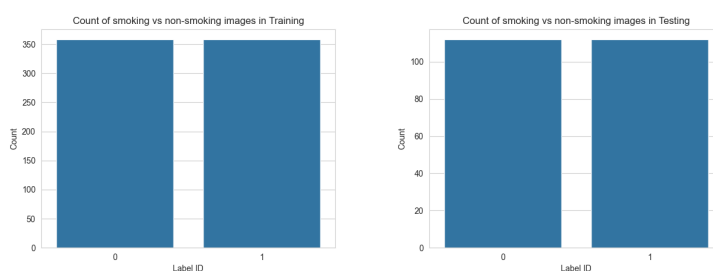


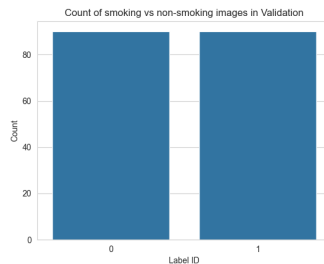
Na tym zdjęciu widzimy człowieka oraz używkę, która zdecydowanie znajduje się w nieodpowiednim miejscu na obrazie.

3 Szkolenie modelu

3.1 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 osoby niepalące, a ile palące.





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 uczeniu się modelu w zachowanym porządku, i dzieli zbiór na mniejsze części (batche). Następnie dane są jeszcze raz obrabiane przez funkcje *augmentation* i *loadDatasetWithAugmentation*, gdzie na zdjęciach stosowane są różne operacje, takie jak losowe odwrócenie obrazu w poziomie, losowa zmiana jasności, kontrastu, nasycenia i odcienia.

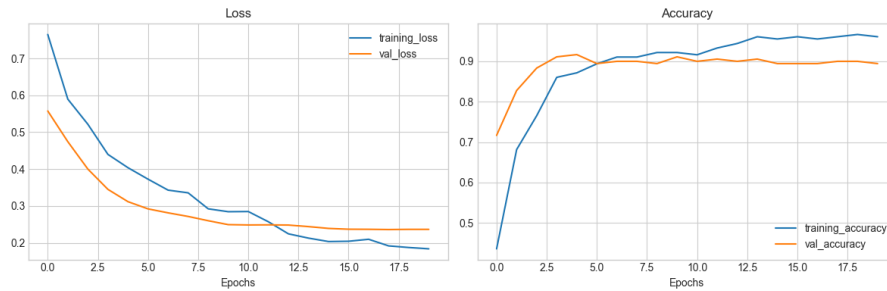
3.2 Szkolenie pierwszego modelu

W tym etapie tworzony jest model z warstwą wejściową przyjmującą obrazy o wielkości 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 wszystkie 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.

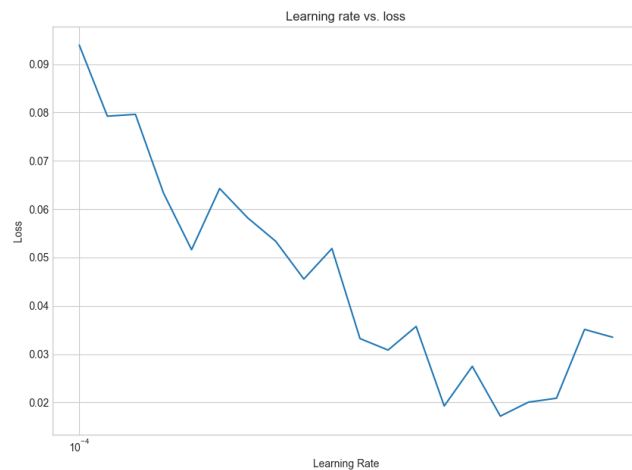
Po tym model jest kompilowany optymistą Adam z metryką "accuracy" i szkolony podczas n ilości epok, gdzie n wynosiło 10 i 20. Podczas trenowania epoch wykorzystywany jest callback ModelCheckpoint, który automatycznie zapisuje wagi modelu podczas treningu co epoch, jeżeli wyniki są lepsze od tych już zapisanych. Po ewaluacji tego modelu na danych testowych "accuracy" wynosi aż 89.99%, a starta danych - 28.04%.



Na powyższym wykresie zestawione zostały malejące straty danych w zależności od epoki (których jest 20 w zamieszczonym przypadku) i rosnące 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.

3.3 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 zmienia się starta danych przy każdym z 20 epochów trenowania drugiego modelu.

3.4 Testy działania

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



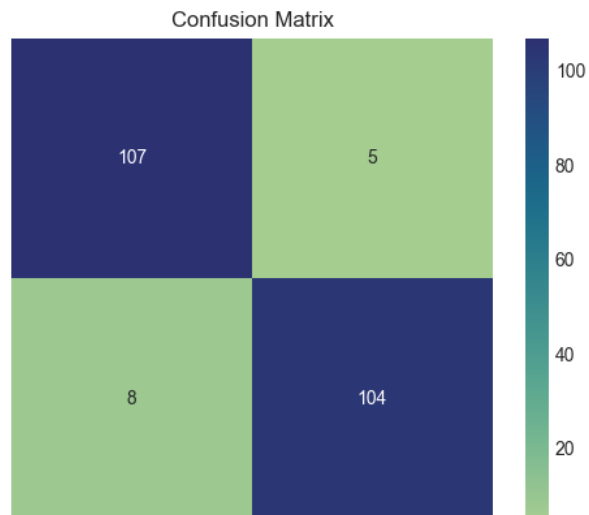
Smoking with probability 99.50520992279053
Non-Smoking with probability 0.6944173015654087



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 palących 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

4 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.