

Politechnika Warszawska

Warsztaty z Technik Uczenia Maszynowego

**APLIKACJA DO WYKRYWANIA
TWARZY I OKREŚLANIA EMOCJI
WYKRYTYCH OSÓB**

Wojciech Gajda 304494

Jakub Brzóskowski 313180

Aleksy Bałaziński 313173

Jan Cichomski 313201

Prowadzący: dr inż. Janusz Rafałko

Data oddania: **25 maja 2023**

Spis treści

1 Wstęp	3
1.1 Opis	3
1.2 Plan pracy	4
1.3 Podział zadań	4
2 Przebieg projektu	5
2.1 Wybór narzędzi	5
2.2 Wyodrębnianie twarzy ze zdjęć	6
2.2.1 Przegląd gotowych rozwiązań	6
2.2.2 Wybór typu sieci	6
2.2.3 Opis danych treningowych	7
2.2.4 Proces uczenia	7
2.2.5 Ocena efektów uczenia	8
2.3 Rozpoznanie emocji	9
2.3.1 Wybór typu sieci	9
2.3.2 Opis danych treningowych	11
2.3.3 Proces uczenia	11
2.3.4 Ocena efektów uczenia	14
2.4 Aplikacja użytkownika – GUI	15
2.5 Podsumowanie	19
3 Literatura	20

1 Wstęp

Rozpoznawanie twarzy w obrazach wykorzystywane jest obecnie w niezliczonych dziedzinach. Systemy monitoringu przetwarzają dziennie petabajty informacji, które bez odpowiedniego sklasyfikowania stanowią bezużytecznych zbiór nagrani. Aplikację do przechowywania zdjęć starają się porządkować przechowywany kontent, rozpoznając podobne twarze.

Jednocześnie samo wyodrębnienie twarzy nie niesie dużej wartości informacyjnej. Można, co prawda, zliczyć ilość osób objętych danym ujęciem, jednak bez dalszej analizy takie zdjęcia przedstawiają po prostu, ludzi. Więcej informacji uzyskujemy podając dopiero tak przygotowaną i wyodrębnioną twarz dalszej analizie. Obecnie dostępne narzędzia pozwalają oszacować wiele cech rozpoznanej osoby. Twarz skrywa информацию o wieku, płci, rasie, ale również bardziej ulotne cechy jak uśmiech, smutek, zmęczenie itp. W ramach tego projektu skupimy się właśnie tych ulotnych cechach które w ogólności można nazwać emocjami.

1.1 Opis

Celem projektu jest przygotowanie aplikacji do rozpoznawania emocji osób znajdujących się na zdjęciu. Działanie programu można podzielić na kilka etapów. Najpierw do aplikacji wczytywane jest dowolne zdjęcie. W kolejnym kroku na załadowanym zdjęciu wykrywane są ludzkie twarze. Wykrycie twarzy pozwala wizualizować efekt klasyfikacji przez dorysowanie kontrastujących ramek wokół twarzy. Jednocześnie każdą z twarzy można wyekstrahować ze zdjęcia i dodać do zbioru. Każdy z elementów tak utworzonego zbioru jest następnie poddany analizie i przypisywana jest mu jedna z przyjętych klas, reprezentujących emocję.

Powyższa aplikacja do działania wymaga jednak wytrenowania dwóch odmiennych modeli sieci: do binarnej klasyfikacji twarzy i klasyfikacji wyodrębnionych twarzy względem emocji. Proces treningu ww. modelu przedstawiony zostanie w odrębnych skryptach (notatnikach).

1.2 Plan pracy

1. Przegląd literatury.
2. Przygotowanie modułu do wyodrębniania twarzy przy pomocy pretrenowanego modelu sieci.
3. Obróbka danych uczących i wytrenowanie modelu do klasyfikacji emocji.
4. Przygotowanie modułu do rozpoznawania emocji.
5. Połączenie modułów w jednorodną aplikację z GUI.
6. Obróbka danych uczących i wytrenowanie własnego modelu do wykrywania twarzy.

1.3 Podział zadań

Praca nad projektem realizowane są w formie wewnętrznych spotkań. W naturalny sposób powoduje to, że postęp realizowany jest wspólnie, bez większego podziału ról. By łatwiej utrzymywać jakość projektu, do poszczególnych obszarów przydzielana została osoba odpowiedzialna. Podział ten prezentuje się następująco:

- Wojciech Gajda:
 - opracowanie architektury projektu i integracja modułów projektu
 - przygotowanie aplikacji użytkownika (GUI)
- Jan Cichomski:
 - przegląd literatury
 - dobór danych uczących
- Jakub Brzóskowski:
 - opis danych uczących do rozpoznawania twarzy
 - wytrenowanie modelu do klasyfikacji twarzy na zdjęciach
- Aleksy Bałaziński:
 - opis danych uczących do rozpoznawania emocji
 - wytrenowanie modelu do rozpoznawania emocji

2 Przebieg projektu

2.1 Wybór narzędzi

Jako główny język programowania wykorzystany zostanie **Python**. Język ten zdobył olbrzymią popularność w domenie Machine Learningu, Deep Learning i sztucznej inteligencji. Za popularnością Python'a przemawia jego prostota i niski próg wejścia. Dodatkowo ogromna społeczność zgromadzona wokół języka ułatwia poszukiwanie informacji i rad.

Do rozwiązania problemu ekstrakcji twarzy z zdjęcia oraz ogólnie pojętej obróbki wykorzystana zostanie biblioteka **OpenCV**. Biblioteka stanowi rozbudowany zbiór narzędzi do obróbki obrazu oraz zbiór podstawowych modeli sztucznych sieci neuronowych ukierunkowany na wykrywanie wzorców i wizję komputerową. Biblioteka zawiera model kaskadowej sieci z funkcjami Haara, która w ostatnich latach uznawana był jako główne narzędzie do klasyfikacji twarzy. W plikach źródłowych projektu znaleźć można pretrenowany model sieci, przeznaczony do odnajdywania twarzy w obrazach. Biblioteka ~~umożliwia również samodzielny trening takiego modelu~~. Niestety wsparcie dla tego rozwiązania zostało porzucone. Przyczyną tego jest przewaga, którą dają współczesne sieci DNN. Najnowsze artykuły wskazują, że rezultaty działania takich detektorów (np. z biblioteki TensorFlow lub YOLO) dają znacznie większą dokładność, przy zachowaniu zbliżonej prędkości obliczeń.

Klasyfikacja emocji na wydzielonych twarzach jest natomiast zadaniem dla którego nie istnieje powszechnie przyjęte rozwiązanie. W literaturze istnieje wiele podejść tego problemu jednak znacząca większość opiera swoje działanie na różnych odmianach konwolucyjnych sieci neuronowych. Większość z tych modeli dostępna jest w popularnej bibliotece **TensorFlow**. Ponadto biblioteka ta umożliwia łatwe wykorzystanie układów graficznych do trenowania modelu. W projekcie wykorzystany zostanie taki model, którego struktura da najlepsze rezultaty.

Oba modele wykorzystywane zostaną we wspólnej aplikacji z interfejsem użytkownika. Do przygotowania interfejsu wykorzystana zostanie biblioteka **Qt**.

Wszelkie źródła projektowe zawarte są w wspólnym repozytorium **GitHub**.

2.2 Wyodrębnianie twarzy ze zdjęć

2.2.1 Przegląd gotowych rozwiązań

Zadanie wyodrębnienia twarzy ze zdjęć stanowi typowe zadanie klasyfikacji. Pozwala to zatem na stosowanie popularnych uniwersalnych modeli m. in. dostępnych w bibliotece TensorFlow. Temat rozpoznawania twarzy zyskał jednak dużą popularność co spowodowało powstanie metod zoptymalizowanych do tego zadania.

Klasyczne podejście zakłada wykorzystanie sieci opartej o funkcji Haara oraz optymalizację działania przez ułożenie warstw w kaskadę. Takie rozwiązanie znaleźć można m.in. w bibliotece OpenCV. Rozwiązanie to charakteryzuje się przede wszystkim prostotą. Działanie poszczególnych wezłów można łatwo zwizualizować co ułatwia zrozumienie. Niestety metoda ta posiada znaczną wadę związaną ze zbiorem danych uczących. Do poprawnego działania wymaga ona, aby zbiór danych liczony był w milionach, a zbiór danych negatywnych (niezawierających twarzy) był ok 1000x większy. Samodzielne wytrenowanie tego modelu do klasyfikacji twarzy stanowi zatem spore wyzwanie. Na szczęście w ramach projektu OpenCV taki model został wytrenowany i może zostać wykorzystany do porównania.

Ze względu na rosnącą moc obliczeniową komputerów, problem dużej liczby warstw ukrytych powoli zanika i do klasyfikacji powszechnie korzysta się z DNN. Modele głębokich sieci neuronowych zaimplementowane zostały m. in. w bibliotekach TensorFlow i Torch. Warto jednak przyjrzeć się metodom zoptymalizowanym do klasyfikacji obiektu. Wśród nich dużą popularność zdobył model YOLO i jego kolejne wersje. Przewaga tego modelu polega na nieszablonowym podejściu do zadania klasyfikacji. Zamiast oddzielnego rozpatrywania każdego z podprostokątów, YOLO (ang. You look only once) przetwarza cały obraz "na raz", przypisując każdemu z pikseli przynależność do klasy. Następnie regiony w których dominująca liczba pikseli została zakwalifikowana do danej klasy zostają sklasyfikowane.

2.2.2 Wybór typu sieci

Obecnie za najbardziej aktualny uznaje się model YOLOv7. Zgodnie z artykułem opisującym ten model jest to najlepsza wersja YOLO ze względu na zarówno szybkość jak i dokładność. Głównymi ulepszeniami względem poprzednich iteracji jest E-ELAN, czyli projekt architektury opartej na blokach ELAN, które pozwalają na lepsze uczenie modelu. Używa się metod znanych ogólnie jako BoF - Bag of Freebies, np. reparametryzacji, która zwiększa czas potrzebny na trenowanie, ale ostatecznie poprawia rezultaty na przestrzeni kolejnych epok. YOLOv7 jest również przygotowane do wykrywania szkieletów wykrytych ludzi oraz zaznaczania konturów wykrytego modelu, a nie tylko opisującego prostokąta. W modelu podstawowym jest to sieć o 407 warstwach ukrytych, a pretrenowany model jest w stanie wykryć 80 różnych klas takich jak ludzie, rowery, laptopy, książki, owce, samochody itd.

2.2.3 Opis danych treningowych

Dane treningowe do wykrywania twarzy na zdjęciach zostaną użyte ze zbioru **WIDER FACE**. Jest to ogólnodostępny zbiór 32,203 zdjęć, na których znajduje się 393,703 twarzy. Są one wysoce zróżnicowane ze względu na rozmiar, a twarze znajdują się w różnych okolicznościach, pozycjach i rozmiarach. Cały dataset jest podzielony na dane treningowe, walidacyjne oraz testowe w proporcjach 40:10:50. Jednakże dane testowe są przeznaczone do sprawdzania funkcjonalności rozwiązań przez autorów zbioru przez co nie zostały udostępnione dane dotyczące informacji o pozycjach twarzy na tych zdjęciach. Zatem zbiór danych, które będą mogły zostać wykorzystane do testowania autorskiej sieci znacznie maleje.

W kontekście wykrywania twarzy pojawia się zagadnienie negatywnych próbek, czyli otoczenia twarzy, aby móc lepiej wyodrębnić pozycje twarzy. Oznacza to, że będziemy musieli przygotować z dostępnych zdjęć wiele próbek, które będą automatycznie wycinane z tego samego zbioru.



Rysunek 1: Przykład danych

2.2.4 Proces uczenia

Pierwszym krokiem było dopasowanie formatu danych dostępnych w zbiorze WIDER do tego wymaganego przez YOLO. Wynika to z faktu, że dane są opisane poprzez współrzędne lewego górnego rogu oraz prawego dolnego rogu obiektu widocznego na zdjęciu. Natomiast YOLO wymaga punktu środkowego oraz szerokości i wysokości, a to wszystko w znormalizowanych danych.

Następnie uczenie odbywa się specjalnie przygotowanym programem, któremu podajemy pretrenowany model do generalnych zadań. Jest to typowe uczenie maszynowe, które dostosowuje sieć do wykrywania jedynie ludzkich twarzy. Cały proces dostrajania modelu trwał około 48h.

2.2.5 Ocena efektów uczenia

Ostateczny model okazał się bardzo skuteczny. Rzadko kiedy nie jest w stanie wykryć twarzy, a podczas testowania nie daliśmy rady dostrzec false positive wykryć twarzy. Model dobrze generalizuje, nie ma problemu z wykrywaniem twarzy osób w różnym wieku, o różnej karnacji twarzy jak i owłosieniu na głowie.

2.3 Rozpoznanie emocji

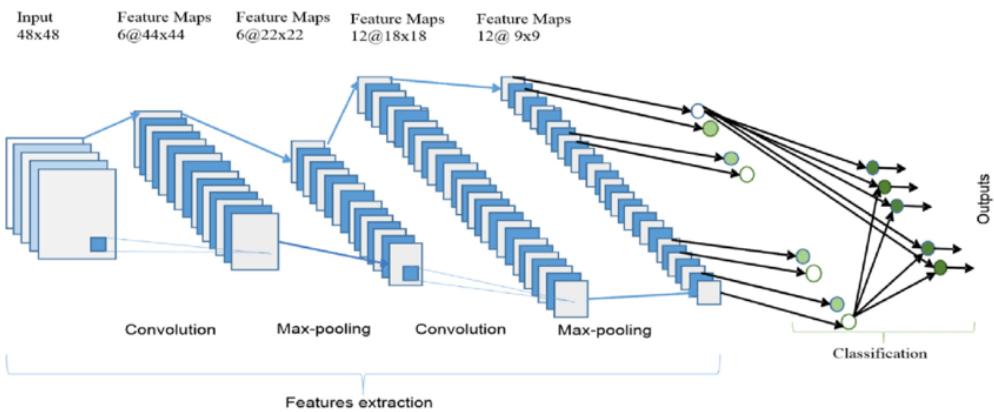
2.3.1 Wybór typu sieci

Badania nad wykorzystaniem uczenia głębokiego (*deep learning*) do zadania rozpoznawania emocji (*facial expression recognition, FER*) wykazały dużą skuteczność konwolucyjnych sieci neuronowych (*convolutional neural networks, CNN*) w tym zakresie. Okazuje się że sieci CNN dobrze radzą sobie ze różnymi ustawieniami twarzy na analizowanym zdjęciu oraz działają poprawnie dla różnych zmian skali. Z tego powodu, do projektu wykorzystaliśmy właśnie sieć CNN.

W przypadku zadania klasyfikacji, działanie sieci CNN można podzielić na dwa zasadnicze etapy:

1. uczenie się (ekstrakcja) cech,
2. klasyfikacja.

Schemat sieci CNN przedstawia rysunek 2. Najważniejsze operacje, które mają miejsce



Rysunek 2: Sieć konwolucyjna do klasyfikacji

w trakcie pierwszego etapu to wykonowywanie operacji *konwolucji*, wprowadzanie nielinowości dzięki zastosowaniu *funkcji aktywacji* oraz redukcja rozmiaru kolejnej warstwy poprzez zastosowanie *poolingu*. Omówimy teraz działanie sieci CNN nieco bardziej szczegółowo, uwzględniając informacje o funkcjonalnościach dostępnych w bibliotece **TensorFlow** (`import tensorflow as tf`).

Konwolucja (`tf.keras.layers.Conv2D`) jest podstawową operacją sieci konwolucyjnych, polegającą na przesuwaniu okna (filtra) po całym obrazie i obliczaniu iloczynu skalarnego między filtrem a fragmentem obrazu pokrytym przez okno. Koncepcyjnie, jeden filtr odpowiada pojedynczej czerwonej kropce, którą chcemy zidentyfikować w obrazie. Przykładem filtra może być

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

który wykrywa krawędzie (filtry powstają w trakcie uczenia modelu). Wynikiem operacji konwolucji (z dodanym *biasem*) jest wartość pojedynczego neuronu w kolejnej warstwie; warstwa uzyskana w ten sposób nazywana określana jest jako *feature map*, „mapa cech”. Przykładowo, jeśli rozważanym filtrem jest macierz wag $[w_{ij}]$ wymiaru 4×4 , to wartość neuronu (p, q) w warstwie zostanie obliczona jako

$$\sum_{i=1}^4 \sum_{j=1}^4 w_{ij} x_{i+p,j+q} + b,$$

gdzie x są wartościami neuronów warstwy poprzedniej (lub wartości pikseli, jeśli rozważamy pierwszą warstwę ukrytą), a b to *bias*.

Nieliniowość jest wprowadzana po każdym zastosowaniu konwolucji (`tf.keras.activations.*`). Najczęściej do tego celu wykorzystuje się operację ReLU (*rectified linear unit*), która przekształca wszystkie wartości ujemne na zero ($g(z) = \max(0, z)$).

Operacja *poolingu* (`tf.keras.layers.MaxPool2D`) jest stosowana do zmniejszania wymiarowości obrazów w trakcie przetwarzania. Polega na dzieleniu obrazu na mniejsze fragmenty (np. kwadraty 2×2) i zastępowaniu każdego z tych fragmentów pojedynczym pikselem, który jest reprezentatywny dla całego fragmentu. Dzięki temu operacja ta pozwala zmniejszyć liczbę parametrów sieci, a tym samym zwiększyć jej efektywność. Często stosowaną w praktyce odmianą *poolingu* jest *maxpooling*, która polega na wyborze elementu o maksymalnej wartości z danego fragmentu. Przykładowo dla fragmentów rozmiaru 2×2 operacja *maxpoolingu* przedstawia się następująco

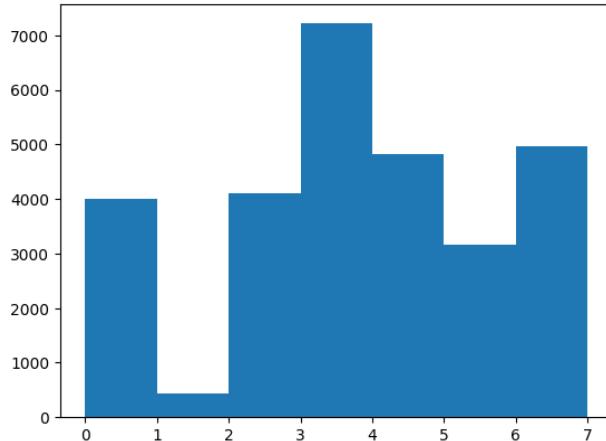
$$\left[\begin{array}{cc|cc} 1 & 1 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ \hline 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{array} \right] \xrightarrow{\text{maxpooling}} \left[\begin{array}{c|c} 6 & 8 \\ \hline 3 & 4 \end{array} \right]$$

W kolejnym etapie działania sieci, wyekstrahowane cechy podawane są na wejście sieci typu *fully connected*, której zadaniem jest określenie rozkładu prawdopodobieństwa przynależności obrazu do określonych klas. Mówiąc dokładniej, na wyjściu ostatniej warstwy konwolucyjnej znajdują się mapy cech, które są przetwarzane przez warstwy w pełni połączone (`tf.keras.layers.Dense`). W tym kroku, każda mapa cech jest „spłaszczana” (ang. *flattening*) w wektor liczb (`tf.keras.layers.Flatten`). Kolejnym krokiem jest przetworzenie tych wektorów w warstwach w pełni połączonych, które wykorzystują algorytm uczenia maszynowego do klasyfikacji obrazów (najczęściej wykorzystywaną funkcją aktywacji na tym etapie jest *softmax*). Na wyjściu tych warstw otrzymujemy wektor wyników, który jest interpretowany jako prawdopodobieństwo należenia obrazu do każdej z klas. Ostatecznie, klasa obrazu jest określana jako klasa z najwyższym prawdopodobieństwem.

Ważnym elementem projektowania sieci jest zadbanie o to, aby sieć się „nie przeuczyła”. Zjawisku, w którym działanie sieci jest zbyt dopasowane do zboru danych treningowych (*overfitting*) możemy przeciwdziałać stosując *regularyzację*. Jedną ze znanych technik regularyzacji jest *dropout* (`tf.keras.layers.Dropout`), który polega na losowym ustawnianiu wartości niektórych neuronów w warstwie na zero.

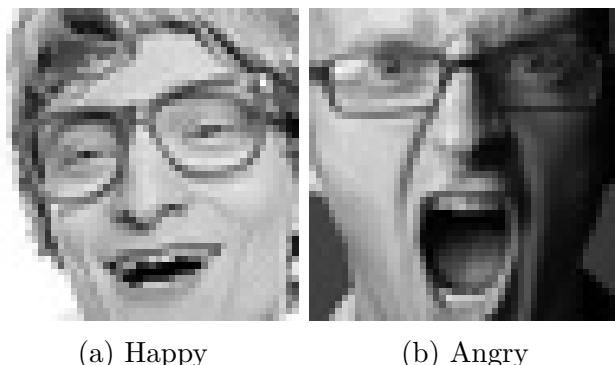
2.3.2 Opis danych treningowych

Dane treningowe pochodzą ze zbioru FER-2013, który był częścią zadania konkursowego zamieszczonego na platformie *kaggle* w 2013 roku. Zbiór treningowy składa się z 28709 zdjęć czarno-białych wymiaru 48×48 przedstawiających twarze ludzkie wyrażających jedną z sześciu emocji: 0=złość, 1=zniesmaczenie, 2=strach, 3=szczęście, 4=smutek, 5=zaskoczenie, 6=neutralność. Na rysunku 3 przedstawiono liczęność wymienionych grup. Dane są zawarte



Rysunek 3: Liczęność poszczególnych klas w zbiorze FER2013

w pliku `train.csv`. Pierwsza kolumna zawiera liczby z zakresu od 0 do 6 włącznie, które odpowiadają emocji przedstawianej na danym zdjęciu. Kolumna druga zawiera napis otoczony cudzysłowami dla każdego zdjęcia. Napis ten zawiera rozdzielone spacjami wartości pikseli zapisane „wiersz po wierszu”. Przykładowe zdjęcia ze zbioru danych przedstawia rysunek 4.



Rysunek 4: Przykłady danych FER2013

2.3.3 Proces uczenia

Zaimplementowane zostały dwa modele sieci CNN. Pierwszy z nich zawierał sześć warstw konwolucyjnych i dwie warstwy *fully-connected*. Jest on przedstawiony na listingu 1.

```

1 tf.keras.Sequential([
2     tf.keras.layers.Conv2D(64, (5,5), activation='relu', input_shape=(48, 48,
3         → 1), padding='same'),
4     tf.keras.layers.Conv2D(64, (5,5), activation='relu', padding='same'),
5     tf.keras.layers.BatchNormalization(),
6     tf.keras.layers.MaxPooling2D((2,2)),
7
8     tf.keras.layers.Conv2D(128, (5,5), activation='relu', padding='same'),
9     tf.keras.layers.Conv2D(128, (5,5), activation='relu', padding='same'),
10    tf.keras.layers.BatchNormalization(),
11    tf.keras.layers.MaxPooling2D((2,2)),
12
13    tf.keras.layers.Conv2D(256, (3,3), activation='relu', padding='same'),
14    tf.keras.layers.Conv2D(256, (3,3), activation='relu', padding='same'),
15    tf.keras.layers.BatchNormalization(),
16    tf.keras.layers.MaxPooling2D((2,2)),
17
18    tf.keras.layers.Flatten(),
19    tf.keras.layers.Dense(128, activation='relu'),
20    tf.keras.layers.Dropout(0.2),
21    tf.keras.layers.Dense(7, activation='softmax')
22 ])

```

Listing 1: Pierwszy model sieci konwolucyjnej do klasyfikacji emocji

Za pomocą tego modelu udało nam się osiągnąć maksymalną dokładność (*val. accuracy*) równą 0.56. Część procesu trenowania modelu jest przedstawiona na listingu 2.

```

Epoch 1/20
572/575 [=====>.] - ETA: 0s - loss: 1.7768 -
→ accuracy: 0.3033
Epoch 1: val_accuracy improved from -inf to 0.32368, saving model to
→ checkpoint/best_model.h5
575/575 [=====] - 51s 23ms/step - loss: 1.7763 -
→ accuracy: 0.3035 - val_loss: 1.7804 - val_accuracy: 0.3237
...
Epoch 19/20
573/575 [=====>.] - ETA: 0s - loss: 0.1870 -
→ accuracy: 0.9399
Epoch 19: val_accuracy did not improve from 0.55616
575/575 [=====] - 12s 22ms/step - loss: 0.1869 -
→ accuracy: 0.9399 - val_loss: 3.1051 - val_accuracy: 0.5438
Epoch 20/20

```

```

573/575 [=====>.] - ETA: 0s - loss: 0.1680 -
→ accuracy: 0.9446
Epoch 20: val_accuracy did not improve from 0.55616
575/575 [=====] - 12s 22ms/step - loss: 0.1680 -
→ accuracy: 0.9446 - val_loss: 3.0555 - val_accuracy: 0.5420

```

Listing 2: Trenowanie modelu 1.

Drugi model był o wiele bardziej złożony. Z uwagi na jego rozmiar nie został on umieszczony w tekście tej pracy; można go natomiast znaleźć pod adresem https://github.com/Xevi8X/WTUM_2023_gr5/blob/main/train_emotion/emotion_recognition.ipynb. Model został podzielony na 5 bloków, z czego każdy zawierał 4 warstwy konwolucyjne; model zawierał również 2 warstwy *fully-connected*. Za jego pomocą udało uzyskać się dokładność (*val. accuracy*) wynoszącą ok. 65%. Część procesu trenowania przedstawia listing 3.

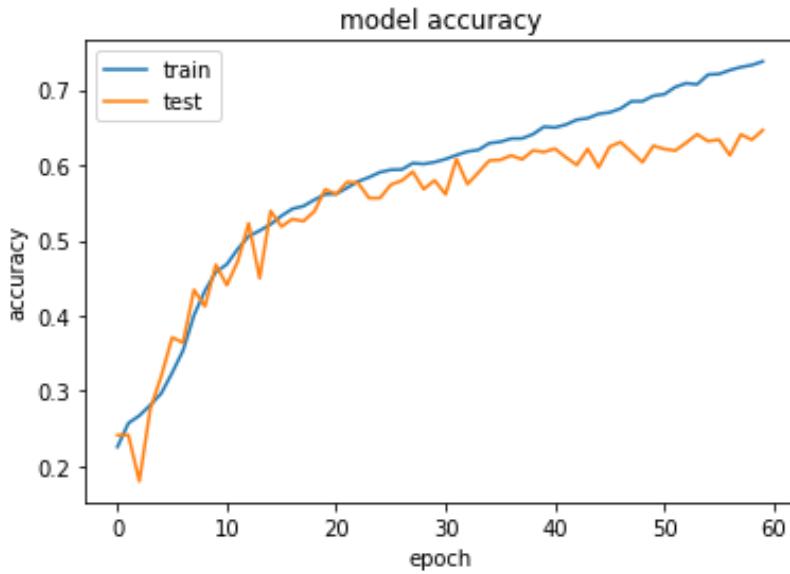
```

Epoch 1: val_accuracy improved from -inf to 0.24138, saving model to
→ /content/drive/MyDrive/datasets/checkpoint/best_model.h5
180/180 [=====] - 199s 929ms/step - loss: 2.0244 -
→ accuracy: 0.2258 - val_loss: 1.8347 - val_accuracy: 0.2414
...
Epoch 59/60
180/180 [=====] - ETA: 0s - loss: 0.7338 -
→ accuracy: 0.7329
Epoch 59: val_accuracy did not improve from 0.64107
180/180 [=====] - 147s 817ms/step - loss: 0.7338 -
→ accuracy: 0.7329 - val_loss: 1.1277 - val_accuracy: 0.6338
Epoch 60/60
180/180 [=====] - ETA: 0s - loss: 0.7097 -
→ accuracy: 0.7377
Epoch 60: val_accuracy improved from 0.64107 to 0.64646, saving model to
→ /content/drive/MyDrive/datasets/checkpoint/best_model.h5
180/180 [=====] - 150s 831ms/step - loss: 0.7097 -
→ accuracy: 0.7377 - val_loss: 1.0892 - val_accuracy: 0.6465

```

Listing 3: Trenowanie modelu 2.

W tym modelu zastosowano również dodatkową warstwę, która losowo obraca zdjęcia w czasie trenowania w zakresie kąta $[-2/5\pi, 2/5\pi]$ (`tf.keras.layers.RandomRotation(0.2)`). Stanowi ona przykład warstwy typu *preprocessing layer*, której zadaniem jest wprowadzenie większej różnorodności danych poprzez wprowadzanie losowych, ale jednocześnie realistycznych transformacji (w tym przypadku obrotów zdjęć). Technikę tą określa się czasem mianem *data augmentation*. Jej celem jest stworzenie różnorodności danych treningowych, aby modele uczenia maszynowego mogły lepiej uogólniać i radzić sobie z różnymi przypadkami. Wykres dokładności modelu w zależności od epoki przedstawia wykres 5.



Rysunek 5: Uczenie modelu 2. do wykrywania emocji

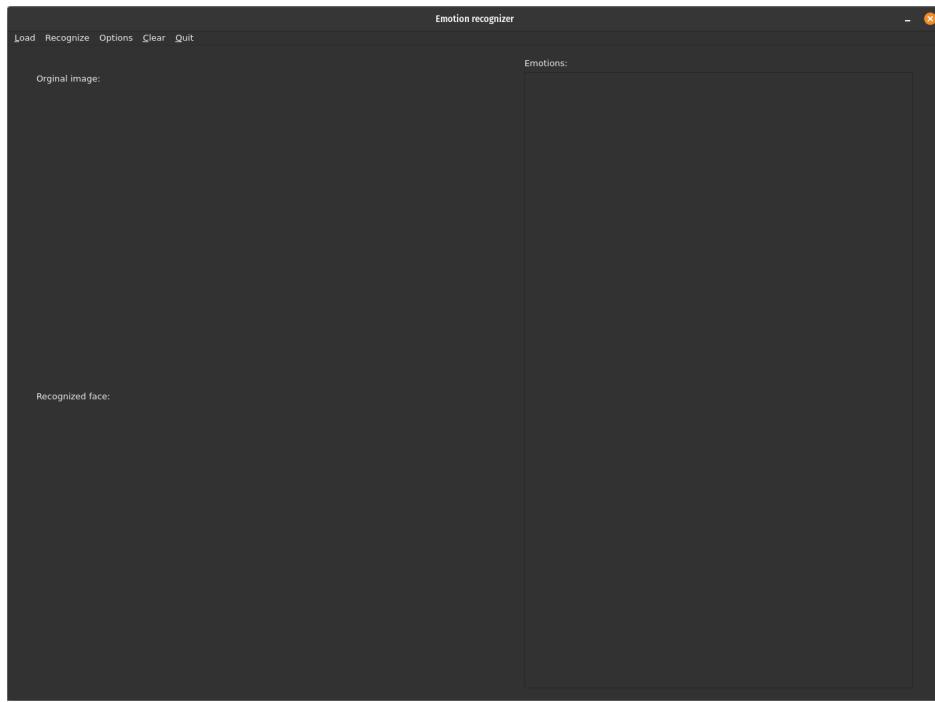
2.3.4 Ocena efektów uczenia

Dokładność, którą udało się osiągnąć korzystając ze zbioru danych FER2013 wyniosła 56% i 65% odpowiednio dla pierwszego i drugiego modelu. Celem określenia jak nasze modele wypadły na tle innych opisywanych w literaturze, posłużyono się zestawieniem zawartym w pracy *Deep Facial Expression Recognition* Li i Dengi. Wynika z niego, że największa dokładność jaką udało się osiągać z wykorzystaniem zbioru FER2013 wyniosła 75.2% (Christopher Pramerdorfer i in.). Pozostałe przedstawione wyniki zawierały się w przedziale od 67.21% do 75.10%; w każdym z tych rozwiązań zastosowana była sieć CNN.

Można na tej podstawie uznać dokładność opracowanych przez nas modeli (szczególnie modelu drugiego) za satysfakcyjną.

2.4 Aplikacja użytkownika – GUI

Efekty pracy opisane w poprzednich rozdziałach oraz wytrenowane modele zostały wykorzystane w finalnej aplikacji. Celem aplikacji jest załadowanie zdjęcia z pliku lub kamery, rozpoznanie twarzy oraz określenie ich emocji.

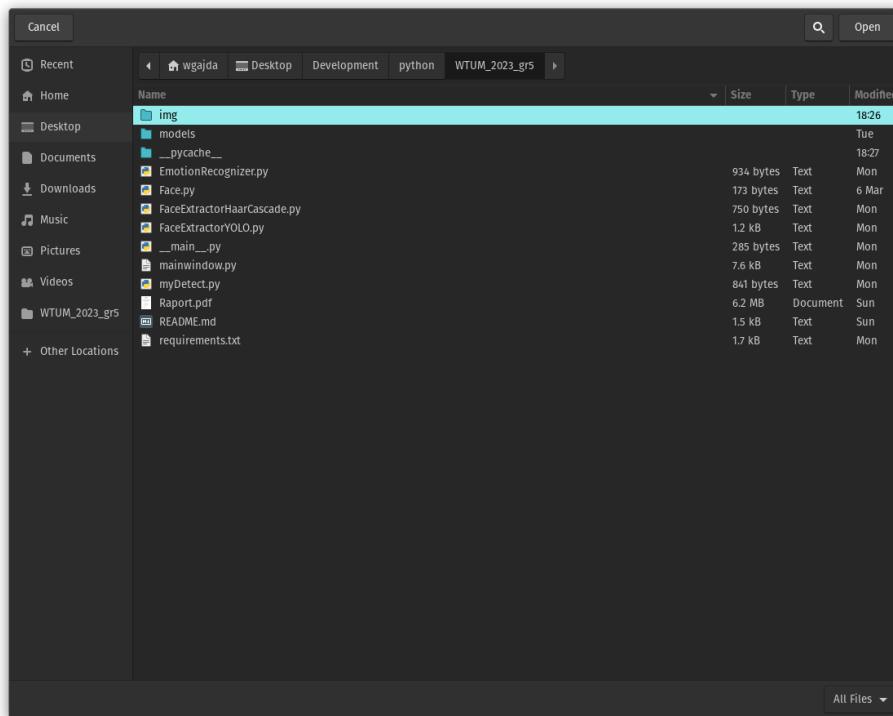


Rysunek 6: Aplikacja przed załadowaniem obrazu

Sterowania aplikacją dokonuje się z poziomu paska menu. W pasku dostępne są następujące akcje:

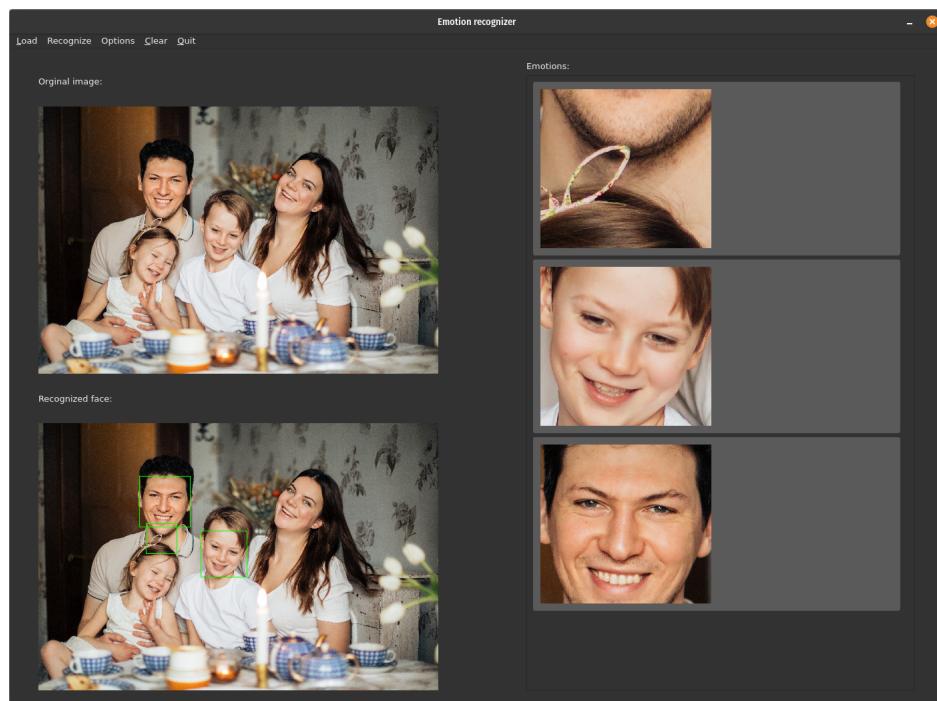
- Load
 - File - pozwala załadować plik z dysku
 - Camera
 - * Preview - powoduje rozpoczęcie przechwytywania obrazu z kamery
 - * Capture - pozwala zapisać aktualny obraz z kamery
- Recognize
 - Face - powoduje rozpoznanie twarzy wybranym Extractorem
 - Emotions - powoduje rozpoznanie emocji dla wyodrębnionych twarzy
- Option

- Haar - ustawia kaskade Haara jako Extractor
- YOLO - ustawia YOLO jako Extractor
- Clear - czyści obszary robocze
- Quit - zamknięcie aplikacji



Rysunek 7: Wybór pliku z obrazem

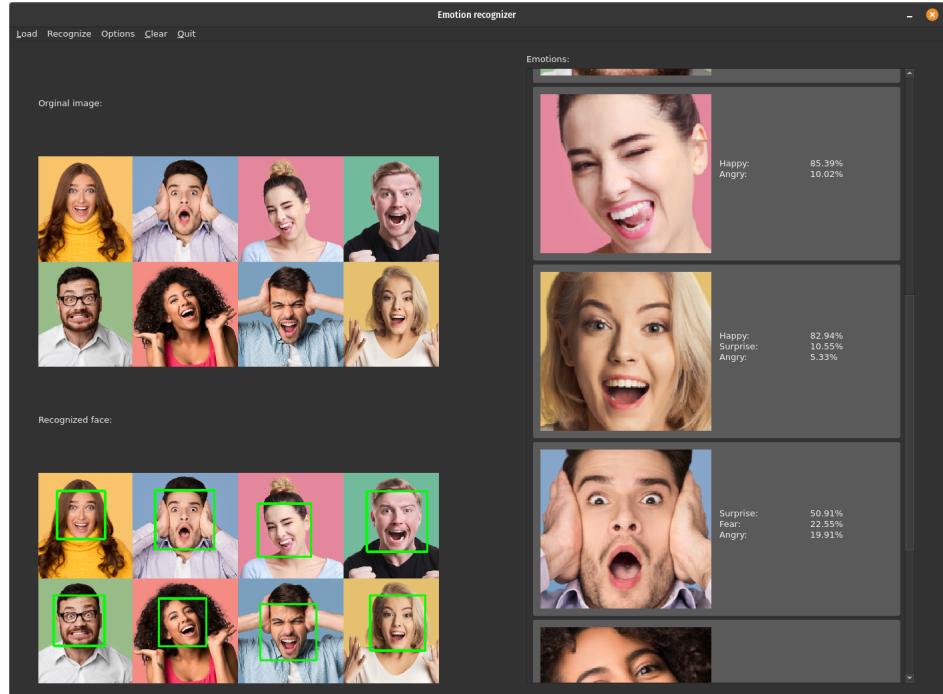
Do zaprezentowania działania aplikacji wykorzystanie obraz o rozmiarze 5617x3745 pikseli (stosunkowo duży obraz). Wszystkie obliczenia sieci realizowane są na CPU. Dla tego samego zdjęcia kaskada Haara nie wykryła poprawnie wszystkich twarzy, a ponadto jej czas obliczenia był niemal 20x dłuższy. Pokazuje to przewagę nowych metod klasyfikacji.



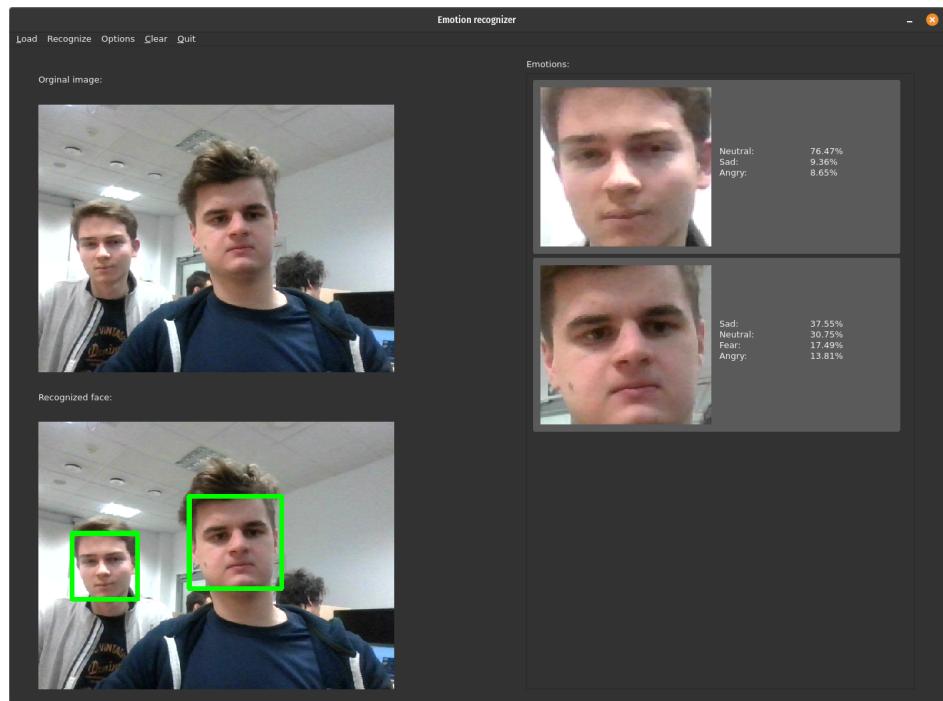
Rysunek 8: Wykrycie twarzy kaskadą Haara - 13.67s



Rysunek 9: Wykrycie twarzy DNN YOLOv7 - 0.498s



Rysunek 10: Przykład bardziej zróżnicowanego wykrycia emocji



Rysunek 11: Podgląd z kamery

2.5 Podsumowanie

W trakcie trwania projektu udało się stworzyć aplikacje GUI, która potrafi wyekstrahować twarze z zdjęcia, a następnie dla każdej z twarzy określić jakie emocje wyraża. Udało się uzyskać zadowalającą dokładność dla obu z powyższych zadań

Początkowo planowano wykorzystać algorytm kaskady Haara, jednak po dokładniejszej analizie i przeprowadzonych eksperymentach, uznano, że należy zmienić podejście do rozwiązania problemu ekstrakcji twarzy. Zdecydowano się na wykorzystanie głębokiej konwolucyjnej sieci neuronowej, a dokładniej sieci YOLOv7. Jednak jako, że część pracy zostało już wykonane, to postanowiono zostawić możliwość wyporu kaskady Haara, jako alternatywnej możliwości do ekstrakcji twarzy w aplikacji GUI. Do uczenia sieci wykorzystano publicznie dostępny zbiór danych **WIDER FACE**

Do rozpoznania emocji wyrażanej przez daną twarz wykorzystano głęboką konwolucyjną sieć neuronową, jednak tym razem stworzono autorską architekturę, gdyż okazało się, że daje ona w pełni satysfakcjonujące efekty. Do trenowania sieci wykorzystano zbiór danych z platformy Kaggle **FER-2013**

Podsumowując, wszystkie założone cele udało się zrealizować na zadowalającym poziomie dokładności, a nawet udało się zrobić więcej niż planowano, gdyż udostępniono użytkownikom dwa algorytmy wykrywania twarzy oraz dodano możliwość wykrywania twarzy i określania emocji robiąc zdjęcie kamerką internetową.

3 Literatura

- https://www.ripublication.com/ijaer18/ijaerv13n8_119.pdf
- Shan Li, Weihong Deng, *Deep Facial Expression Recognition: A Survey*. <https://arxiv.org/abs/1804.08348>.
- Alexander Amini, *Convolutional Neural Networks*, 6.S191, styczeń 2020, Massachusetts Institute of Technology https://www.youtube.com/watch?v=iaSUYvmCekI&t=83s&ab_channel=AlexanderAmini
- Alexander Amini, *Introduction to Deep Learning*, 6.S191, styczeń 2020, Massachusetts Institute of Technology https://www.youtube.com/watch?v=njKP3FqW3Sk&ab_channel=AlexanderAmini