UNIWERSYTET PEDAGOGICZNY
im. Komisji Edukacji Narodowej w Krakowie

INSTYTUT INFORMATYKI

Kierunek: INFORMATYKA

specjalność: -

Adrian Bury

Nr albumu: 149618

Zastosowanie głębokiej sieci neuronowej o architekturze FaceNet do rozpoznawania twarzy

Praca magisterska
napisana pod kierunkiem
dr hab. inż. Tomasza Hachaja

Kraków 2021

Spis treści

Streszczenie

W rozdziale pierwszym zostały poruszone zagadnienia związane z rozpoznawaniem twarzy oraz został postawiony cel napisania aplikacji, za pomocą której użytkownik będzie mógł uzyskać informację na temat osoby przedstawionej na wysłanym zdjęciu pod warunkiem, że do aplikacji zostały wcześniej dostarczone zdjęcia reprezentujące daną osobę. W rozdziale drugim została omówiona koncepcja działania aplikacji, a w rozdziale trzecim zostały wybrane i opisane technologie, które zostaną użyte do jej napisania. Rozdział czwarty szczegółowo opisuje proces implementacji poszczególnych elementów aplikacji oraz omawia sposób znalezienia progu, za pomocą której zostanie uznane, że dana osoba jest znana/nieznana aplikacji. Ostatni rozdział przedstawia sposób testowania oraz zawiera analizę otrzymanych wyników.

Abstract

In chapter one is discussed issues related to face recognition and is set the goal of writing an application to help find information who is pictured on the sent photo if the person has been previously uploaded to the serwer. The second chapter discusses the concept of the system operation and in the third chapter is chosen and described technologies that will be used to meet the expectations. The fourth chapter describes in detail the process of implementing individual system components and discusses the methodology of finding the threshold, by means of which the system recognizes that a given person is known. The last chapter shows tests of implemented system and analysis the results.

Rozdział 1

Cel i zakres pracy

Rozpoznawanie twarzy to zagadnienie z dziedziny klasyfikacji, które polega na rozpoznawaniu i identyfikacji tożsamości osoby na podstawie określonych wzorców [19]. Innymi słowy, rozpoznawanie twarzy to wykrycie obszaru na fotografii, w którym znajduje się twarz oraz weryfikacja tożsamości.

Problem identyfikacji tożsamości jest ogólnie znany jako problem klasyfikacji, czyli określenie etykiety dostarczonego zdjęcia twarzy na podstawie aktualnego zbioru zdjęć. Problem może być postrzegany jako zapytanie systemu "kim jest osoba przedstawiona na zdjęciu?". Przykładem procesu, gdzie jest stosowana identyfikacja, to proces wyszukiwana tożsamości. Po dostarczeniu pliku system przeszukuje bazę dostępnych zdjęć w celu znalezienia pasującego zdjęcia i jeżeli zdjęcie zostanie dopasowane, to system zwróci informację o tożsamości.

Weryfikacja to proces sprawdzający prawdziwość, przydatność lub prawidłowość czegoś [20]. Oznacza to, że weryfikacja na podstawie zdjęcia twarzy będzie oznaczać proces polegający na sprawdzeniu, czy deklarowana tożsamość zgadza się z przypisanym do deklaracji zdjęciem twarzy i jeżeli tak, to akceptowanie, jeżeli nie to odrzucanie oświadczenia. Zapytanie, jakie zadaje się wtedy systemowi to "czy podane oświadczenie zgadza się ze zdjęciem zapisanej w systemie osoby?". Przykładem weryfikacji tożsamości za pomocą twarzy jest odblokowywanie smartfonów z wykorzystaniem kamery. Aplikacja w smartfonie sprawdza wtedy, czy zdjęcie osoby z kamery zgadza się ze zdjęciem zapisanym w pamięci. Jeżeli zdjęcia, zapisane w pamięci oraz dostarczone z kamery, zgadzają się, to telefon zostanie odblokowany.

System, który implementuje weryfikację lub identyfikację na podstawie twarzy, może znaleźć zastosowanie do wykonywania automatycznego oznaczania osób na zdjęciach [6], przeszukiwania stron internetowych w celu znalezienie zdjęcia przedstawiającego daną osobę [14] czy zautomatyzowanego grupowania zdjęć na podstawie twarzy na nich się znajdujących [8].

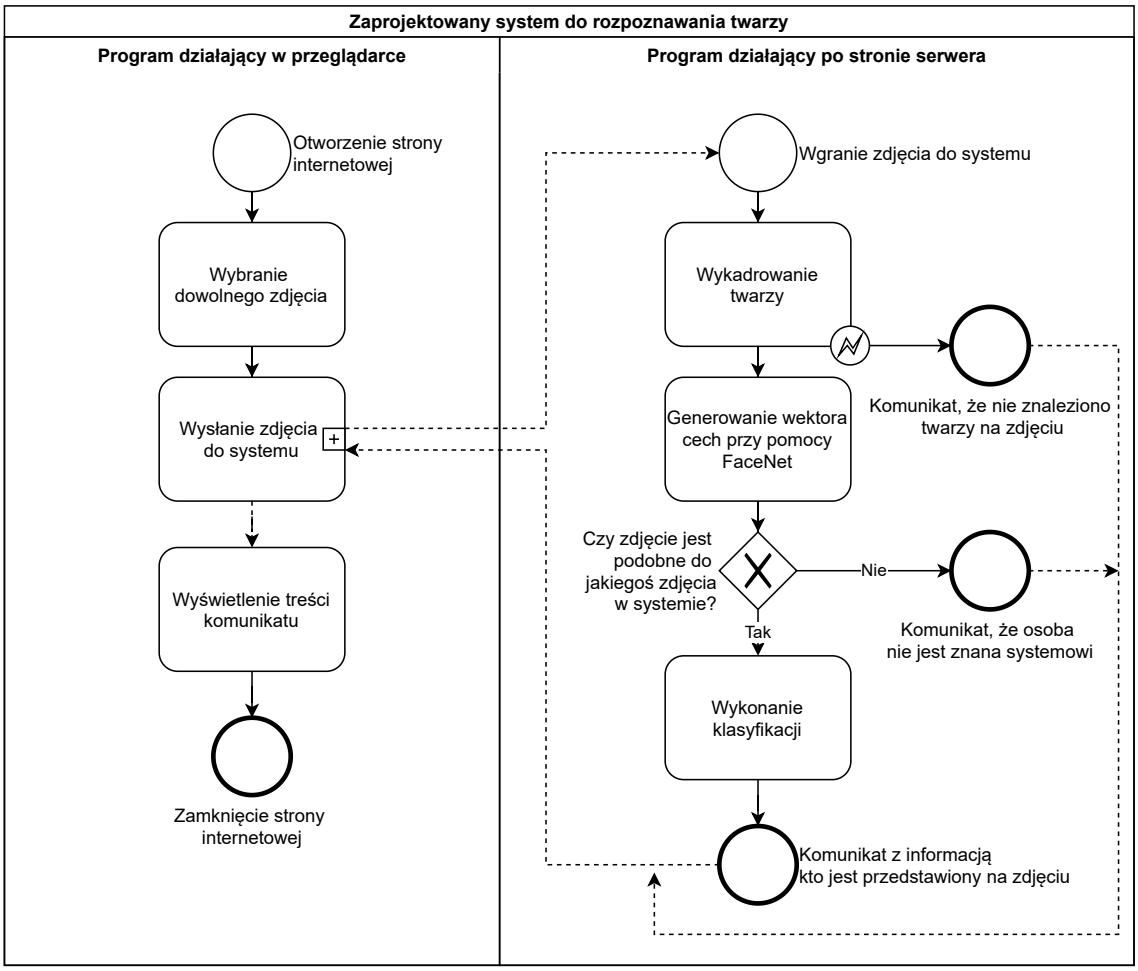
Celem pracy dyplomowej jest utworzenie systemu, którego zadaniem będzie rozpoznanie osoby przedstawionej na zdjęciu. Zakres prac obejmuje implementację strony internetowej, za pomocą której użytkownik będzie miał możliwość przesłania wybranego przez siebie zdjęcia, oraz systemu, który zwróci informację na temat osoby widniejącej na owym zdjęciu (przesyłanym za pomocą strony internetowej), pod warunkiem, że ta osoba zostanie znaleziona w bazie zdjęć. W systemie zostanie użyta głęboka sieć neuronowa o architekturze FaceNet do badania podobieństwa pomiędzy poszczególnymi twarzami. Aby spełnić wszystkie oczekiwania, które zostały postawione systemowi, należy wykonać następujące kroki:

1. sformułować koncepcję działania systemu,
2. wybrać odpowiednie technologie oraz narzędzia,
3. zaimplementować poszczególne elementy systemu,
4. przetestować system.

Rozdział 2

Koncepcja działania systemu

System ma udostępniać użytkownikowi interfejs graficzny, za pomocą którego będzie miał możliwość wysłania dowolnego zdjęcia. W tym celu zostanie przygotowana strona internetowa, która udostępni formularz z możliwością wyboru pliku graficznego. Po wybraniu zdjęcia z dysku użytkownika plik ten zostanie wysłany do systemu. System będzie miał za zadanie wykryć twarz znajdującą się na zdjęciu i tak wykadrować, aby znajdowała się na nim tylko twarz. Jeżeli na zdjęciu nie zostanie znaleziona twarz, system zwróci odpowiedni komunikat o błędzie. Kolejnym krokiem będzie sprawdzenie, czy osoba przesłana na zdjęciu znajduje się w bazie. Po uzyskaniu pozytywnego wyniku nastąpi klasyfikacja zdjęcia za pomocą klasyfikatora i zwrócenie odpowiedniej informacji użytkownikowi korzystającego z systemu. W przeciwnym wypadku zostanie zwrócony komunikat, że dana osoba nie istnieje w bazie zdjęć. Schemat blokowy opisanych procesów został przedstawiony na rysunku 2.1.



Rysunek 2.1: Schemat blokowy zaprojektowanego systemu
Źródło własne

Rozdział 3

Wybór technologii oraz narzędzi

3.1 Wykrywanie twarzy na zdjęciu

Wykrywanie twarzy to pierwszy krok w zaprojektowanym systemie. Z powodu tego, że jest on umieszczony na samym początku, proces ten musi być maksymalnie skuteczny. Jeżeli na zdjęciu nie zostanie wykryta twarz, to cały proces zakończy się z negatywnym rezultatem. **Twarzy nie będzie można zidentyfikować, jeżeli nie zostanie ona znaleziona.** Niestety twarz ludzka jest obiektem dynamicznym i ma duży stopień zmienności w wyglądzie, co sprawia, że wykrywanie twarzy jest trudnym problemem w widzeniu komputerowym [10].

W artykule z 2016 roku zatytułowanym "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks" [21] została zaproponowana wielozadaniowa kaskadowa konwolucyjna sieć neuronowa (ang. Multi-Task Cascaded Convolutional Neural Networks - MTCNN) do wykrywania twarzy. Sieć ta zyskała duża popularność, ponieważ osiągnęła wówczas najlepszy wyniki w wykrywaniu twarzy na zdjęciach dla wybranych zbiorów danych. Kolejną zaletą zaproponowanej sieci neuronowej jest to, że jest w stanie rozpoznać punkty orientacyjne twarzy, takie jak oczy, usta i nos.

W internecie znajduje się spora liczba implementacji MTCNN [7]. Spośród dostępnych została wybrana sieć napisana przez *Iván de Paz Centeno* i udostępniona na portalu GitHub[1)] (`https://github.com/ipazc/mtcnn`) na zasadach licencji MIT[2)] [11]. Główną zaletą wybranej implementacji jest łatwość użycia. Sieć

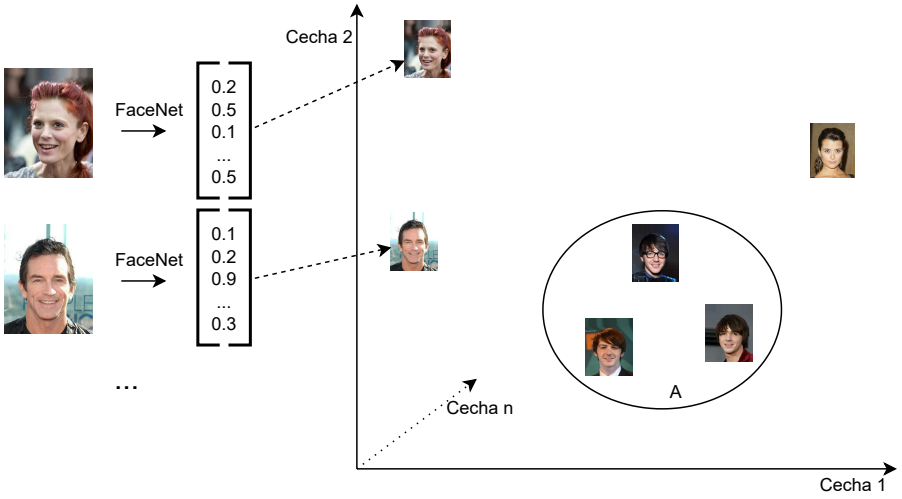
[1)] GitHub - dostawcą platformy internetowej do tworzenia oprogramowania i kontroli wersji za pomocą narzędzia Git udostępniania pod adresem `https://github.com`.

[2)] Licencja MIT daje użytkownikom nieograniczone prawo do używania, kopiowania, modyfikowania i rozpowszechniania (w tym sprzedaży) oryginalnego lub zmodyfikowanego programu. Jedynym wymaganiem jest, by we wszystkich wersjach zachowano warunki licencyjne i informacje o autorze.

została napisana w języku programowania python w bibliotece TensorFlow i całość udostępniona jako pakiet z możliwością instalacji przez PIP[3)].

3.2 FaceNet

FaceNet to architektura oparta na głębokiej sieci neuronowej, która bezpośrednio uczy się mapowania z obrazów twarzy do n-wymiarowych wektorów, gdzie poszczególne wartości wektora charakteryzują daną twarz. Wymiarowość wektora jest zależna od konkretnej implementacji. W przestrzeni składającej się z wygenerowanych wektorów odległości pomiędzy wektorami, mierzone za pomocą metryki euklidesowej, bezpośrednio odpowiadają mierze podobieństwa twarzy. Oznacza to, że zadania polegające na identyfikacji, weryfikacji czy grupowaniu sprowadzają się do pomiaru odległości pomiędzy poszczególnymi wektorami cech [16]. Wizualizacja procesu mapowania zdjęcia do przestrzeni euklidesowej została przedstawiona na rysunku 3.1.



Rysunek 3.1: Wizualizacja przestrzeni składającej się z wygenerowanych wektorów. Odległości mierzone metryką euklidesową pomiędzy wektorami reprezentującymi te same osoby są mniejsze względem wektorów reprezentujących inne osoby. Literą "A" zostały oznaczone zdjęcia przedstawiające tę samą osobę.
Źródło własne

[3)] PIP - narzędzie do instalowania pakietów python

3.2.1 Zasada działania

Po dostarczeniu danego zdjęcia do sieci następuje wieloetapowa filtracja przy użyciu filtrów splotowych, aby wydobyć z obrazu na wejściu wyróżniające się cechy (ang. convolution layer) oraz wykorzystywane są operatory łączenia w celu zredukowania wymiaru danych (ang. pooling layer). Operacje te są pewnego rodzaju kompresją, której zadaniem jest zmniejszenie ilości informacji na temat danej rzeczy (w tym wypadku zdjęcia twarzy) bez utraty kluczowych informacji. Jako wynik działania sieci otrzymywany jest wektor, który przechowuje skompresowane informacje na temat dostarczonego zdjęcia. Informacje jaką są w nim przechowywane są trudne, o ile w ogóle możliwe do zinterpretowania.

3.2.2 Trenowanie sieci

W celu wytrenowania sieci, aby generowany wektor dokładnie odwzorowywał dostarczone zdjęcie, wykorzystywana jest funkcja strat *triplet loss*. Jej zadaniem jest minimalizowanie odległości pomiędzy prawdziwymi wyrażeniami oraz maksymalizowanie dla fałszywych [3]. Innymi słowy, celem jest, aby odległości między wektorami cech tych samych osób były jak najmniejsze, a odległości pomiędzy wektorami różnych osób — maksymalne. Wzór matematyczny opisujący funkcję *triplet loss* został przedstawiony równaniem 3.1 [16], natomiast zasada działania w sposób graficzny przedstawiona na rysunku 3.2.

$$L = \sum_{i}^{N} [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha]_+$$

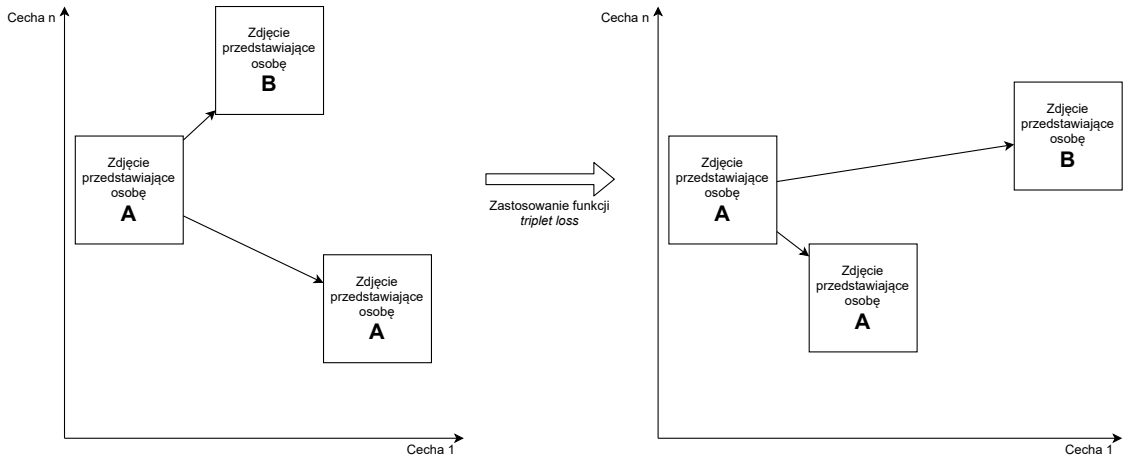
gdzie

x_i^a - wektor zdjęcia wzorcowego

x_i^p - wektor zdjęcia “prawdziwego” względem wzorca

x_i^n - wektor zdjęcia “fałszywego” względem wzorca

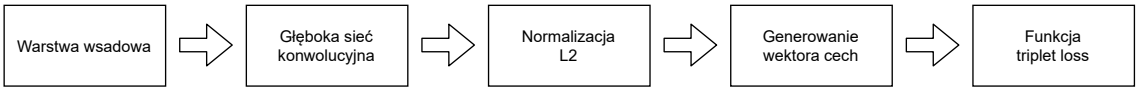
α - margines odległości pomiędzy wektorami (3.1)



Rysunek 3.2: Uproszczony schemat obrazujący zasadę działania funkcji strat *triplet loss*
Źródło własne

Poszczególne etapy trenowania sieci o architekturze FaceNet zostały przedstawione na rysunku 3.3. Całość procesu uczenia się sieci FaceNet, można w uproszczeniu podsumować w następujących krokach:

1. zdefiniowanie parametrów początkowych sieci,
2. wygenerowanie wektorów cech twarzy,
3. użycie funkcji strat *triplet loss*,
4. dostosowanie się parametrów sieci tak, aby przykład pozytywny był bliżej zdjęcia wzorcowego, niż przykład negatywny,
5. powrót do kroku drugiego tak długo, aż zwracane wektory osiągną zadaną dokładność.



Rysunek 3.3: Proces uczenia się sieci o architekturze FaceNet z podziałem na poszczególne etapy
Źródło własne

3.2.3 Wpływ wielkości zdjęcia na działanie sieci

W pracy pod tytułem *Facenet: A unified embedding for face recognition and clustering* [16] została wytrenowana sieć o architekturze FaceNet na zdjęciach w formacie JPEG o rozmiarach 220 na 220 pikseli. Podczas testowania wytrenowanej sieci, po dostarczeniu zdjęć o mniejszych rozmiarach (względem rozmiarów zdjęć, na których sieć była trenowana) okazało się, że sieć ta jest również bardzo skuteczna. Sieć miała akceptowalną skuteczność nawet po dostarczeniu zdjęć o rozmiarach 80 na 80 pikseli. Trenowanie sieci na zdjęciach o mniejszej rozdzielczości mogłoby jeszcze poprawić ten wynik. Dokładne informacje na temat wpływu liczby pikseli na skuteczność sieci zostały przedstawione w tabeli 3.1

Tablica 3.1: Wpływ jakości zdjęcia oraz liczby pikseli na skuteczność sieci FaceNet. Źródło: [16]

Liczba pikseli		Skuteczność
1 600	(40x40)	37.8%
6 400	(80x80)	79.5%
14 400	(120x120)	84.5%
25 600	(160x160)	85.7%
65 536	(256x256)	86.4%

Jakość JPEG	Skuteczność
10	67.3%
20	81.4%
30	83.9%
50	85.5%
70	86.1%
90	86.5%

3.2.4 Użycie wytrenowanej sieci

W projektowanym systemie zostanie użyty wytrenowany model *Keras FaceNet* [17], który został wyszkolony na bazie zdjęć dostarczonych przez MS-Celeb-1M [13] i udostępniony przez *Hiroki Taniai* [18]. Dostarczane zdjęcia twarzy powinny być kolorowe, w formacie RGB oraz o rozmiarach 160 pikseli na 160 pikseli [1].

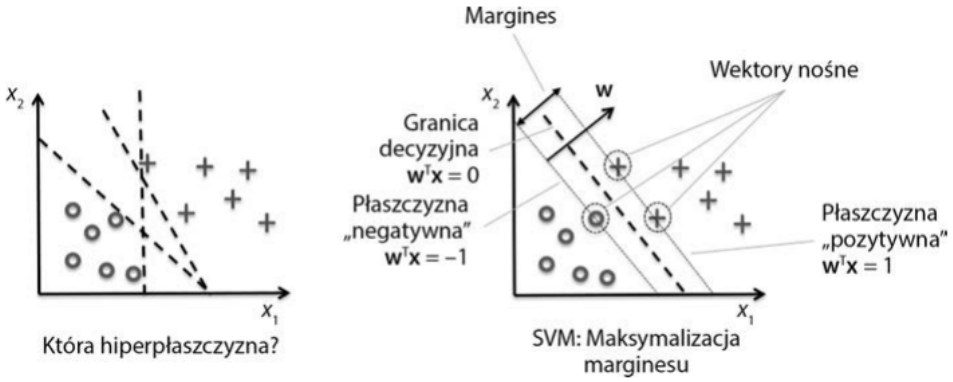
3.3 Kadrowanie i standaryzacja

Do przetwarzania plików graficznych zostanie wykorzystana biblioteka *PILLOW* napisana w języku python. Biblioteka wspiera wiele formatów graficznych, w tym te najpopularniejsze jak *PNG*, *GIF*, *JPEG* oraz BMP [4], dzięki czemu nie będzie wymagane, aby użytkownik sam konwertował plików graficznych do odpowiedniego formatu. Biblioteka *PILLOW* zostanie również wykorzystana do wycinania zdjęcia twarzy.

3.4 Klasyfikator SVM

Ostatnim krokiem w systemie jest dokonanie klasyfikacji wygenerowanego wektora cech, czyli określenie, do jakiej grupy należy wektor na podstawie próbek uczących. Innymi słowy, etap ten polega na zwróceniu informacji, do kogo najbardziej pasuje przesłane zdjęcie, opierając się na fotografiach dostępnych w systemie.

Jednym z szeroko stosowanych algorytmów klasyfikacji jest **maszyna wektorów nośnych** (ang. support vector machine — SVM). Klasyfikator ten konstruuje hiperpłaszczyznę lub ich zbiór w przestrzeni wielowymiarowej, na podstawie otrzymanych próbek, która oddziela poszczególne klasy. Optymalizowanie modelu SVM polega na maksymalizacji marginesu pomiędzy płaszczyzną a poszczególnymi klasami, ponieważ na ogół nim większy margines, tym mniejszy jest błąd generalizacji [9]. Koncepcja klasyfikatora została zaprezentowana na rysunku 3.4.



Rysunek 3.4: Model maszyny wektorów nośnych. Po lewej możliwe położenia hiperpłaszczyzny, po prawej zoptymalizowany model SVM. Źródło: [15]

3.4.1 Margines twardy

Jeśli dane uczące są liniowo separowalne, to można wybrać dwie równoległe hiperpłaszczyzny, które oddzielają dwie klasy danych, tak aby odległość między nimi była maksymalna. Obszar ograniczony przez te dwie hiperpłaszczyzny nazywany jest "marginesem", a hiperpłaszczyzna maksymalnego marginesu to hiperpłaszczyzna leżąca w połowie odległości między nimi. Problem optymalizacji algorytmu został przedstawiony równaniem 3.2 [15]. Lewą stroną tego równania jest odległość pomiędzy hiperpłaszczyzną "pozytywną" a "negatywną", zatem, w celu maksymalizacji marginesu, należy zmaksymalizować $\frac{2}{||w||}$ lub minimalizować odwrotność wyrażenia, czyli $\frac{1}{2}||w||^2$.

$$\frac{w^T(x_{poz} - x_{neg})}{||w||} = \frac{2}{||w||}$$

gdzie:

w - wektor normalny do granicy decyzyjności

x_{poz} - "pozytywny" wektor nośny

x_{neg} - "negatywny" wektor nośny

(3.2)

W przypadku danych liniowo separowalnych równanie 3.2 musi spełniać warunki przedstawione w równaniu 3.3 [15]. Równania te oznaczają, że wszystkie negatywne próbki powinny wylądować po stronie negatywnej hiperpłaszczyzny, a wszystkie próbki pozytywne po stronie hiperpłaszczyzny pozytywnej.

$$\begin{aligned} w_0 + w^T x^{(i)} &\geq 1 && \text{jeśli } y^{(i)} = 1 \\ w_0 + w^T x^{(i)} &\leq -1 && \text{jeśli } y^{(i)} = -1 \end{aligned}$$

dla $i = 1 \ldots N$

gdzie:

N - liczba próbek

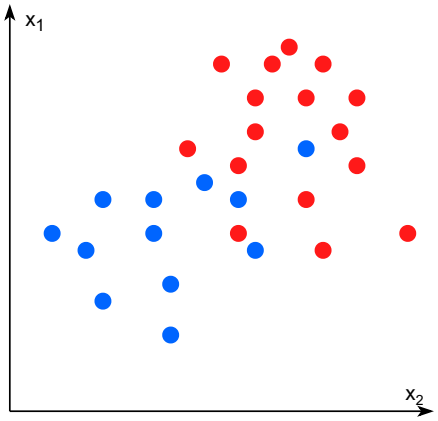
$w_0 + w^T x^{(i)}$ - odległość od granicy decyzyjności

(3.3)

3.4.2 Margines miękki

W przypadku, gdy dane nie są liniowo separowalne (rysunek 3.5) wprowadza się dodatkową zmienną ξ. Motywacją wprowadzenia tej zmiennej jest potrzeba "uelastycznienia" liniowych ograniczeń (równanie 3.3) podczas analizowania nieliniowo rozdzielnych danych, co pozwala na uzyskanie zbieżności algorytmu uczącego w obecności nieprawidłowych klasyfikacji podczas stosowania odpowiedniej funkcji strat. Po wprowadzeniu zmiennej ξ do równiania 3.3, równanie to przybiera postać opisaną wzorami 3.4 [15].

$$
\begin{aligned}
w_0 + w^T x^{(i)} &\geq 1 - \xi^{(i)} && \text{jeśli } y^{(i)} = 1 \\
w_0 + w^T x^{(i)} &\leq -1 + \xi^{(i)} && \text{jeśli } y^{(i)} = -1 \\
&\text{dla } i = 1 \ldots N \\
&\text{gdzie:} \\
&N \text{ - liczba próbek} \\
&\xi \text{ - wartość przesunięcia granicy przynależności} \\
w_0 + w^T x^{(i)} &\text{ - odległość od granicy decyzyjności}
\end{aligned}
\tag{3.4}
$$



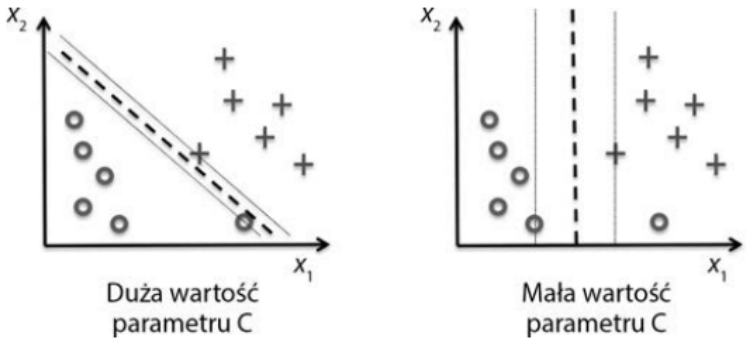
Rysunek 3.5: Przykład próbek nieliniowo rozdzielnych
Źródło własne

Nowym celem optymalizacji staje wtedy się równanie 3.5 [15]. Za pomocą zmiennej C kontroluje się wagę kary za niewłaściwą klasyfikację. Duża wartość parametru C odpowiada wysokim karom za błędy, z kolei przy niskich wartościach kara nie będzie mocno wpływać na szerokość marginesu. Dzięki temu parametrowi jest się w stanie regulować kompromis pomiędzy obciążeniem a wariancją [15] (rysunek 3.6).

$$\frac{1}{2}||w||^2 + C(\sum_{i}^{N} \xi^{(i)}) \tag{3.5}$$

gdzie:

C - mnożnik kary za złą klasyfikację

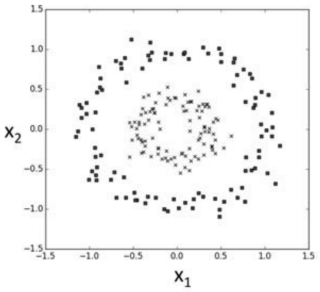


Rysunek 3.6: Wpływ zmiennej C na szerokość marginesu
Źródło: [15]

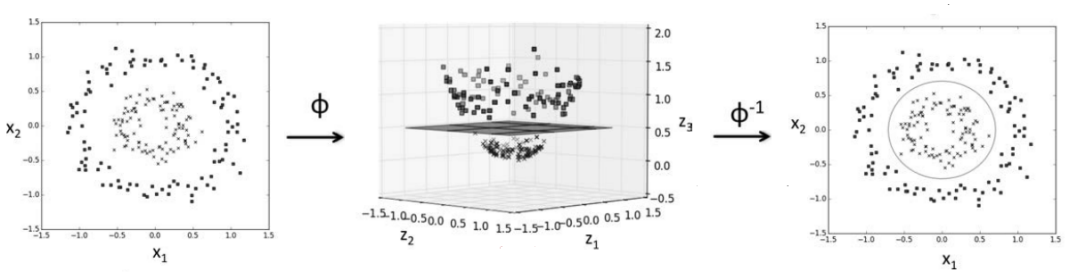
3.4.3 Klasyfikacja nieliniowa

W przypadku, gdy dostarczone dane nie są separowalne za pomocą hiperpłaszczyzny w N wymiarach (rysunek 3.7) wprowadza się dodatkowe wymiary za pomocą funkcji mapującej ϕ. Dodatkowe wymiary wprowadza się tak długo, aż dane staną się liniowo separowalne. Przykładowo, w celu wyznaczenia granicy decyzyjności danych przedstawionych na rysunku 3.7, funkcja mapująca, za pomocą której będzie możliwe wyznaczenie hiperpłaszczyzny, została przedstawiona równaniem 3.6. Rezultat mapowania danych z rysunku 3.7 przez funkcję 3.6 został przedstawiony na rysunku 3.8. Po wykonaniu mapowania oraz wyznaczeniu hiperpłaszczyzny dokonuje się rzutowania do pierwotnej przestrzeni cech (ϕ^{-1}).

$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2) \tag{3.6}$$



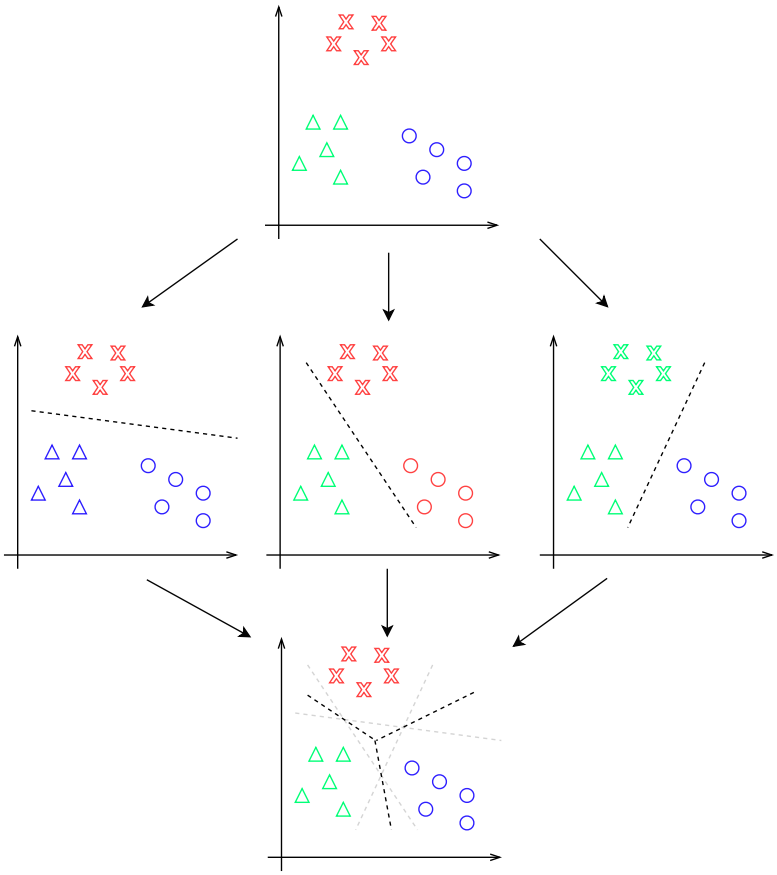
Rysunek 3.7: Zestaw danych nie separowalnych liniowo
Źródło: [15]



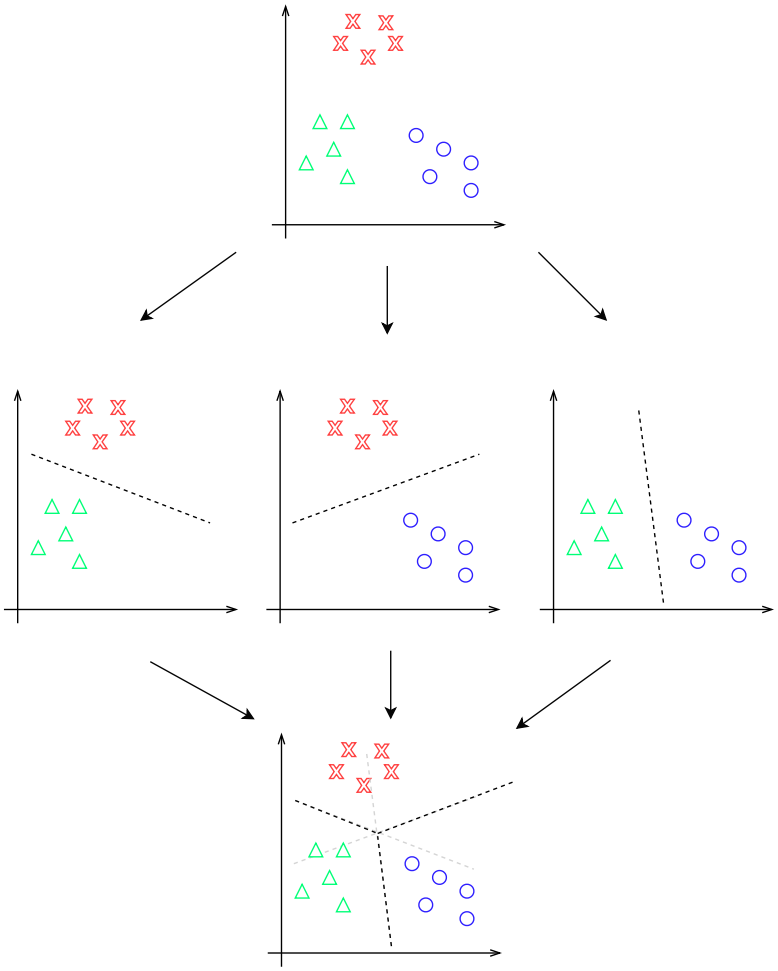
Rysunek 3.8: Sposób wyznaczania nieliniowej granicy decyzyjności
Źródło: [15]

3.4.4 Klasyfikacja wieloklasowa

Wszystkie opisane wyżej przypadki dotyczą problemów binarnych, czyli takich gdzie dostarczona próbka należy do jednej z dwóch klas. W związku z tym, że SVM obsługuje tylko klasyfikację binarną, dlatego w przypadku gdy dana próbka może należeć do jeden z N klas (klasyfikacja wieloklasowa) dokonuje się rozbicia jednego problemu klasyfikacji wieloklasowej na wiele problemów klasyfikacji binarnej. W tym celu stosuje się jedną z dwóch strategii - jeden kontra jeden (ang. one vs one — OvO) lub jeden przeciwko wszystkim (ang. one vs all — OvA). Obie te strategie zostały przedstawione na rysunku 3.9 oraz 3.10.



Rysunek 3.9: Strategia jeden przeciwko wszystkim
Źródło własne



Rysunek 3.10: Strategia jeden kontra jeden
Źródło własne

3.5 Obsługa systemu

Całość zaprojektowanego systemu ma być dostępna z przeglądarki internetowej. Aby sprostać temu wymaganiu, należy napisać usługę, która jest w stanie obsługiwać połączenia HTTP (ang. Hypertext Transfer Protocol). Z powodu, że narzędzia do wykrywania twarzy, obróbki zdjęć oraz FaceNet są udostępnione w języku python, program zostanie również napisany w tym języku. Oprócz samej strony internetowej potrzebna jest również baza danych, w której będą przechowywane informacje o dostępnych zdjęciach. Same zdjęcia będą przechowywane na serwerze plików. Sporym ułatwieniem będzie również panel umożliwiający zarządzanie dostępnymi zdjęciami w systemie.

Narzędziem, które spełni wyżej postawione wymagania jest platforma programistyczna *Django*, która umożliwia w łatwy sposób zarządzanie bazą danych za pomocą mapowania obiektowo-relacyjnego, pozwala na zarządzanie danymi dostępnymi w bazie danych poprzez panel administratora, dostępny z poziomu przeglądarki oraz co najważniejsze, jest napisana w języku python [5].

Rozdział 4

Implementacja systemu

4.1 Kadrowanie twarzy

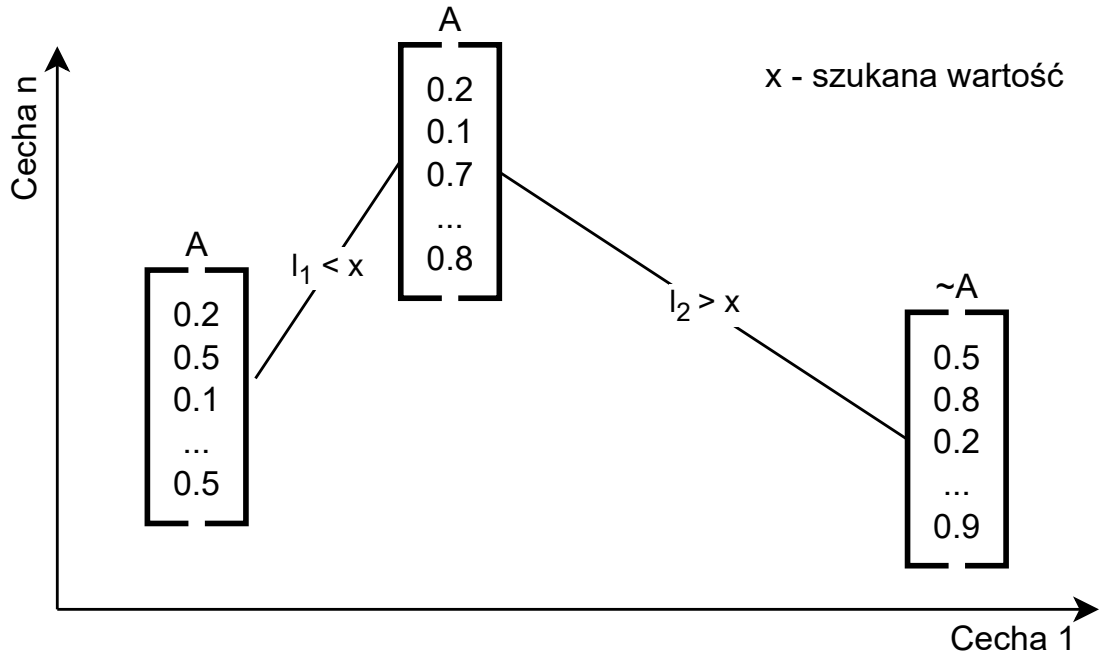
Zdjęcia, które zostaną przesłane na serwer, należy w pierwszej kolejności odpowiednio przygotować. W tym celu została wykorzystana wcześniej wspomniana biblioteka PILLOW, która ma za zadanie wykonać przekształcenie otrzymanego pliku do tablicy pikseli. Otrzymana tablica następnie trafia do sieci MTCNN, która zwraca informacje na temat wykrytych twarzy, po czym następuje wycinanie fragmentu zdjęcia przedstawiającego samą twarz. Jeżeli nie zostaną znalezione żadne twarze, zostanie wygenerowany wyjątek.

4.2 Generowanie wektora cech

Po wykadrowaniu twarzy, kolejnym krokiem jest wygenerowanie wektora cech. Odpowiednio przygotowana wcześniej tablica pikseli jest standaryzowana, ponieważ wymaga tego używana implementacja architektury FaceNet, a następnie wykonywana jest predykcja. Jako wynik działa funkcji zwracany jest 128 wymiarowy wektor cech.

4.3 Ocena podobieństwa twarzy

FaceNet został nauczony tak, aby mapować zdjęcia twarzy do n-wymiarowych wektorów, gdzie odległości mierzone za pomocą metryki euklidesowej odpowiadają mierze podobieństwa twarzy. Oznacza to, że jeżeli odległość pomiędzy dwoma punktami w przestrzeni jest względnie mała, to jest wysoce prawdopodobne, że zdjęcia twarzy przedstawiają tę samą osobę. Mając to na uwadze, można znaleźć odległość, poniżej której zostałoby uznane, że zdjęcia dotyczą tej samej osoby (rysunek 4.1).



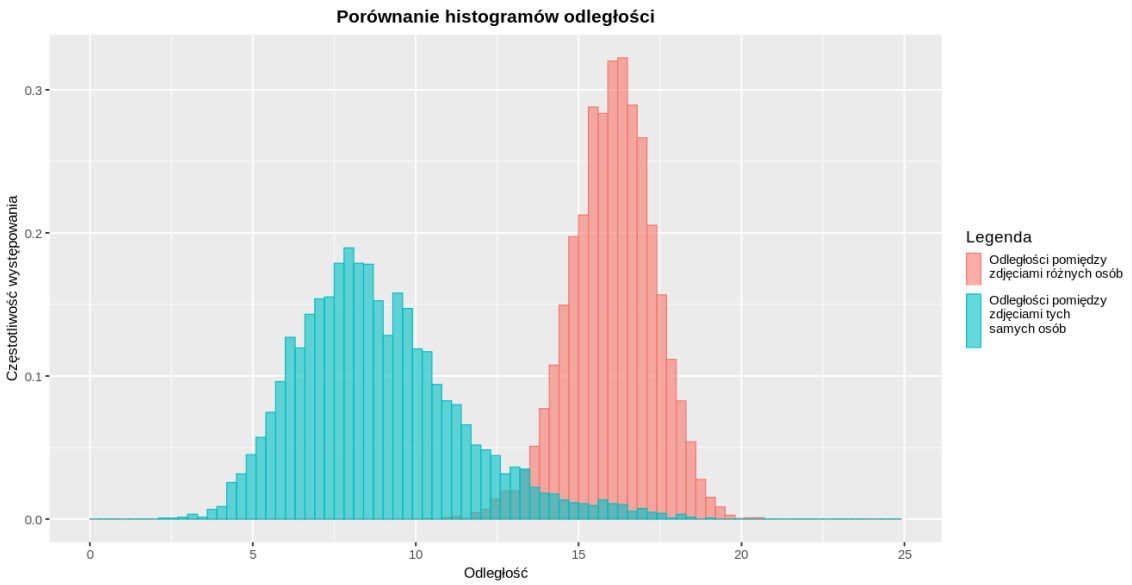
Rysunek 4.1: Wizualizacja szukanego progu podobieństwa. Jeżeli odległość pomiędzy punktami jest zmniejsza niż wartość "x" to program powinien uznać, że wektory opisują tę samą osobę.
Źródło własne

4.3.1 Poszukiwanie odległości

Do poszukiwania wartości progu zostało wykorzystanych 5000 losowo wybranych zdjęć, przedstawiających 500 różnych osób, po 10 zdjęć na osobę, z bazy CelebA [12]. Na każdym zdjęciu zostało następnie wykonane kadrowanie twarzy oraz generowanie wektora cech. Ostatnim elementem było zbadanie odległości pomiędzy wszystkimi wektorami tych samych oraz różnych osób. Całość badania można podsumować w następujących krokach:

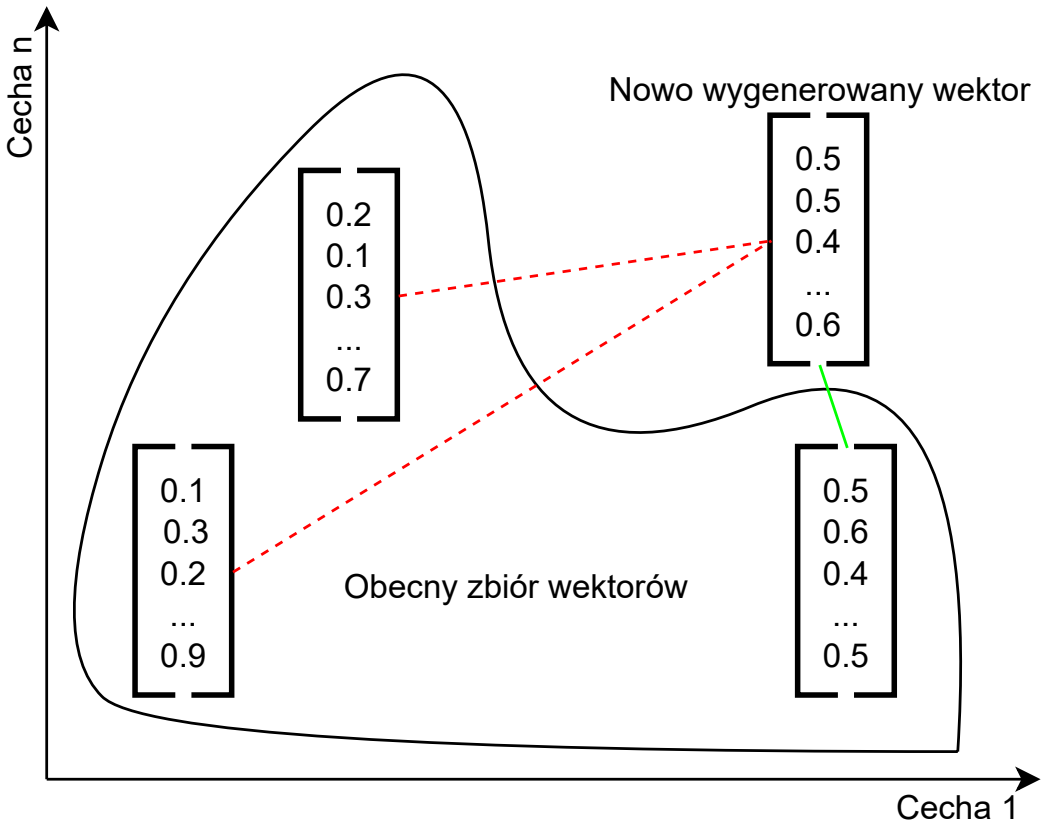
1. wygenerowania wektora cech dla każdego z dostępnych zdjęć,
2. wybranie wektora cech ze zbioru,
3. zmierzenie odległości pomiędzy wybranym wektorem i wszystkimi pozostałymi wektorami z uwzględnieniem, czy wektory cech reprezentują tę samą osobę, czy dwie różne osoby,
4. powrót do kroku drugiego do momentu, aż zostaną zbadane wszystkie odgległości (każdy z każdym).

Wyniki przeprowadzonych kroków zostały przedstawione w postaci histogramów na rysunku 4.2.



Rysunek 4.2: Porównanie histogramów odległości pomiędzy wektorami twarzy tych samych oraz różnych osób
Źródło własne

Na podstawie histogramów (rysunek 4.2) można zauważyć, że nie istnieje taka wartość, która jest w stanie odseparować przedstawione zbiory. Należy jednak zwrócić uwagę na to, że w badaniu zostały wzięte odległości pomiędzy **każdym** zdjęciem z grupy. Przy ocenie podobieństwa pomiędzy twarzami nie ma potrzeby patrzenia na odległości na całym zbiorze zdjęć. Po wygenerowaniu wektora cech dla danego zdjęcia interesujący jest tylko ten wektor, który leży **najbliżej** wygenerowanego (rysunek 4.3).

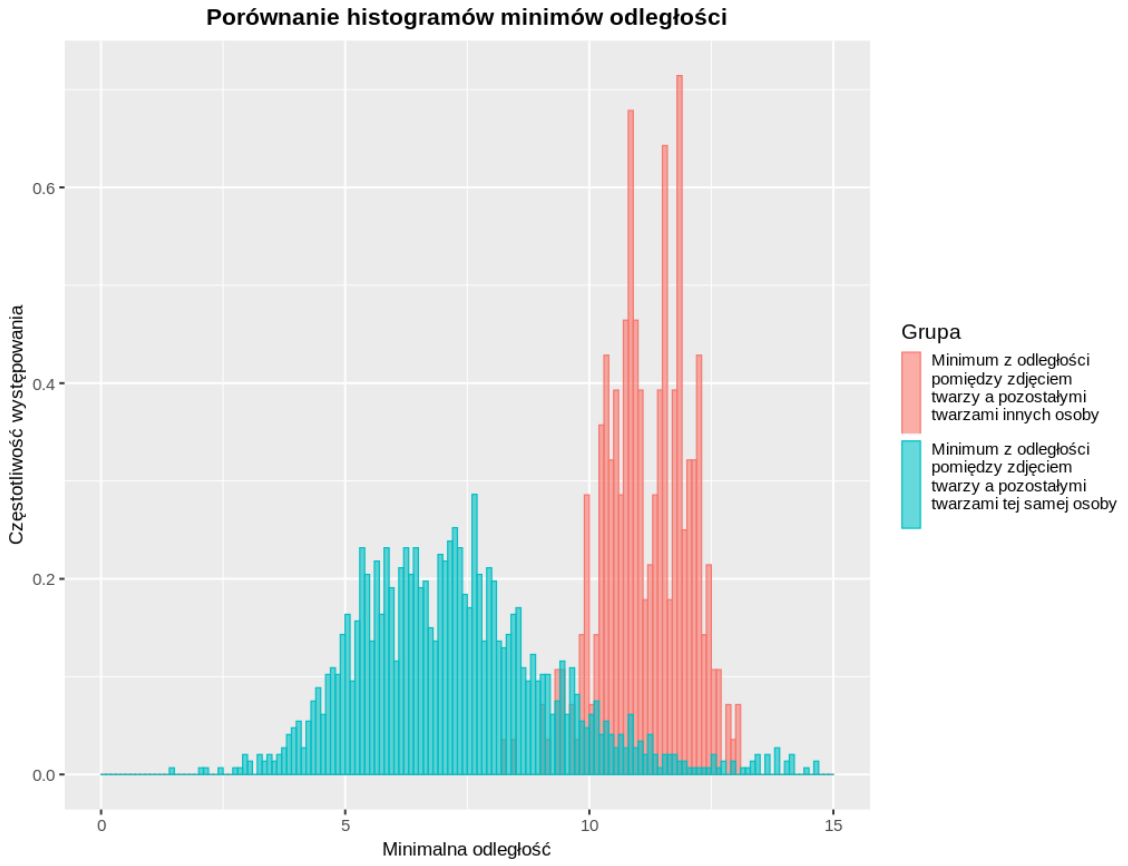


Rysunek 4.3: Schemat obrazujący najbliższą odległość pomiędzy nowym wektorem a zbiorem. Na czerwono zostały oznaczone odległości do dalszych wektorów. Na zielono została oznaczona odległość do najbliższego wektora i to względem niego system powinien oceniać podobieństwo.
Źródło własne

Biorą pod uwagę opisany problem, wyżej opisane badanie zostało powtórzone, jednak w tym przypadku zostały zapisane tylko odległość do najbliższego wektora. Całość zatem można podsumować w następujących krokach:

1. wygenerowania wektora cech dla każdego z dostępnych zdjęć,
2. wybranie wektora cech ze zbioru,
3. znalezienie wektora, który leży najbliżej względem wybranego i zapisanie odległości z uwzględnieniem czy znaleziony wektor reprezentuje tę samą osobę,
4. powrót do kroku drugiego do momentu, aż zostaną wybrane wszystkie wektory.

Wyniki powtórzonego badania zostały ponownie przedstawione w postaci histogramów na rysunku 4.4.

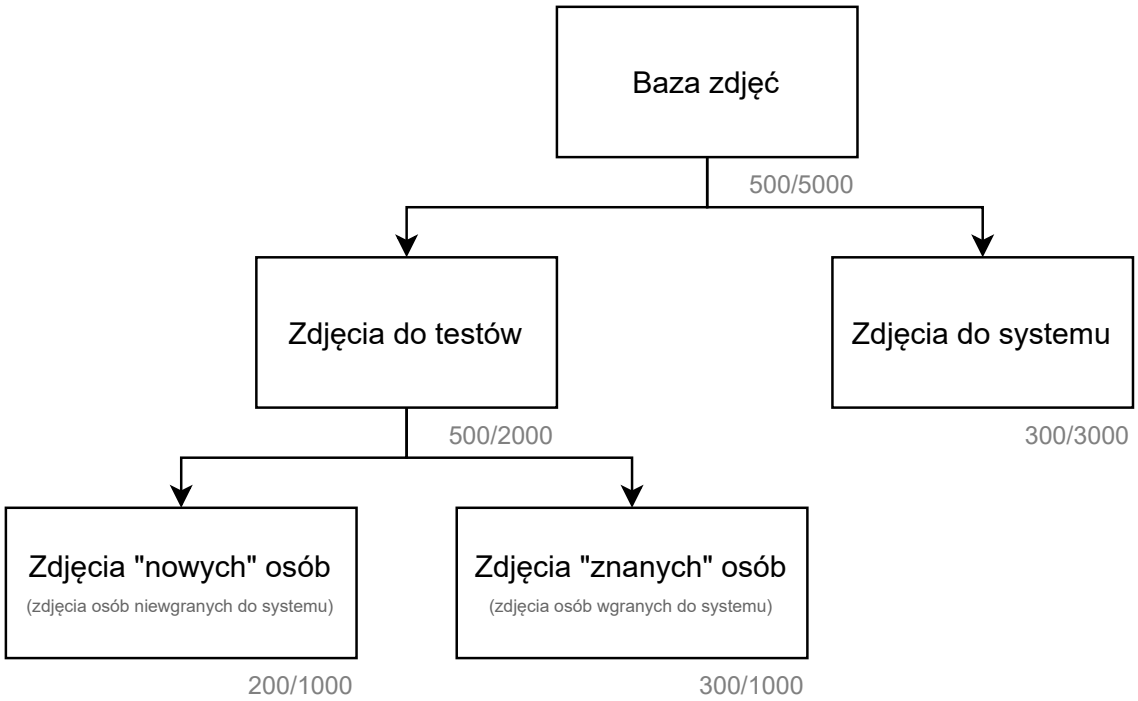


Rysunek 4.4: Porównanie histogramów najmniejszych odległości pomiędzy zdjęciami twarzy tych samych oraz różnych osób.
Źródło własne

Z histogramów (rysunek 4.4) wykonanego badania również wynika, że nie istnieje taka wartość, która będzie w stanie odseparować przedstawione zbiory. Konsekwencją tego jest, że dla dowolnego "x" będą istnieć przypadki, dla których zdjęcia błędnie zostaną uznane, że przedstawiają tę samą osobę. Należy zatem znaleźć taką wartość "x", dla której walidator, czyli funkcja oceniająca, czy podane wektory dotyczą tej samej osoby będzie miał największą skuteczność. Przedział, w jakim należy spodziewać się największej skuteczności, powinien (jak wynika z rysunku 4.4) znaleźć się w przedziale od 8 do 12, ponieważ na tym odcinku oba histogramy pokrywają się na największym obszarze.

4.3.2 Odległość o największej skuteczności

W celu sprawdzania dla jakiego "x" zostanie zmaksymalizowana liczba prawidłowo sklasyfikowanych twarzy, wspomniany wcześniej zbiór zdjęć, został podzielony na dwie grupy. Do pierwszej grupy trafiły zdjęcia, które zasiliły system i stanowiły bazę w weryfikowaniu, czy nowe zdjęcie, które zostanie dostarczone, zostanie uznane za podobne do jakiegoś zdjęcia twarzy, które znajduje się już w systemie. Do drugiej grupy trafiły zdjęcia, które podlegały ocenie przez system, czy owo zdjęcie jest znane systemowi. Dokładny podział zdjęć został przedstawiony na rysunku 4.5.



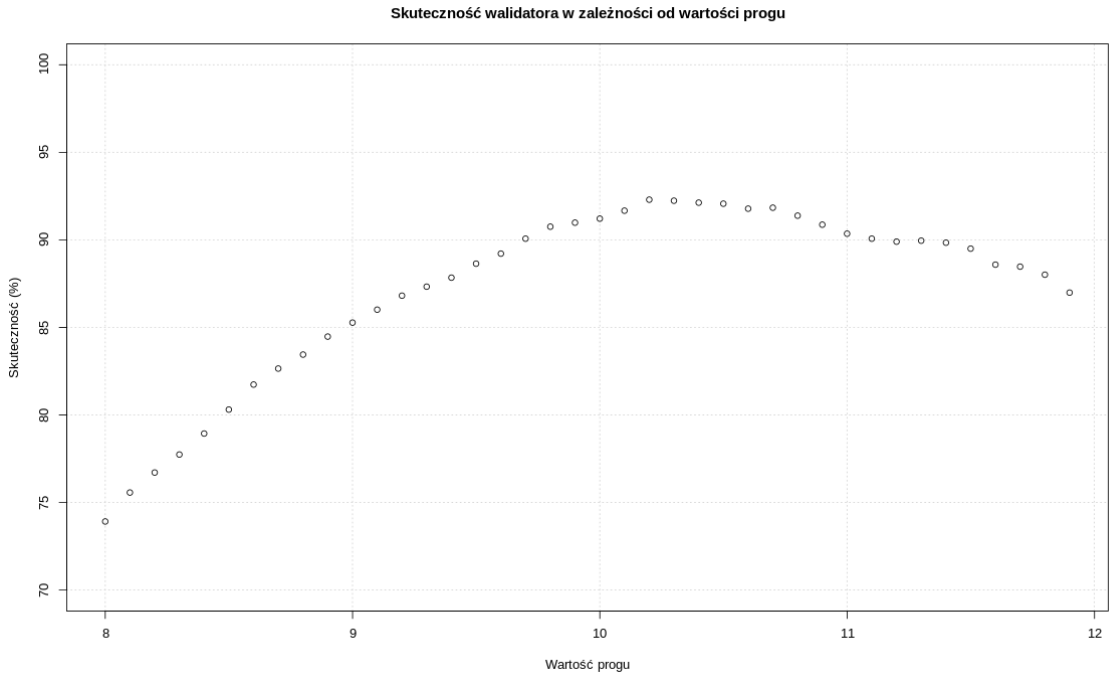
Rysunek 4.5: Podział zdjęć na grupy. Obok każdej z grupy widnieje liczba osób oraz suma wszystkich zdjęć należąca do owej grupy.
Źródło własne

Po podzieleniu zdjęć na grupy kolejnym krokiem jest zbadanie skuteczności oceny podobieństwa dla poszczególnych progów z zakresu od 8 do 12. Zakres ten będzie badany z dokładnością do 0.1. Kroki, jakie zostały wykonane w badaniu, są następujące:

1. wygenerowanie wektora cech dla wszystkich zdjęć z każdej grupy,
2. wgranie wygenerowanych wektorów z grupy "Zdjęcia do systemu" do systemu,
3. pobranie zdjęć z grupy "Zdjęcia nowych osób" i przepuszczenie ich przez funkcję oceniająca podobieństwo na podstawie wgranych zdjęć do systemu dla danego "x",
4. powtórzenie kroku trzeciego dla zdjęć z grupy "Zdjęcia znanych osób",
5. potwórzenie kroku trzeciego oraz czwartek dla nowego parametru "x".

Dla kroku trzeciego, przy założeniu, że skuteczność oceny podobieństwa była by stuprocentowa, funkcja powinna zwrócić zawsze wartość *False*, ponieważ **żadna** z osób przedstawiona na zdjęciach z grupy "Zdjęcia nowych osób" nie znajduje się w systemie. Dla kroku czwartego, w przeciwieństwie do kroku trzeciego, funkcja ta powinna zawsze zwrócić *True*, ponieważ **każda** osoba przedstawiona na zdjęciu z grupy "Zdjęcia znanych osób" znajduje się w systemie. Niestety z przeprowadzonych wcześniej badań (rysunek 4.4) wynika, że taki scenariusz nie jest możliwy i z tego powodu należy znaleźć taką wartość parametru "x", dla którego skuteczność walidatora oceniającego podobieństwo będzie maksymalne.

Wyniki opisanego wcześniej badania zostały przedstawione na rysunku 4.6. Na jego podstawie można zauważyć, że skuteczność walidatora rośnie wraz ze wzrostem wartości progu aż do momentu osiągnięcia przez "x" wartości 10.2 po czym nastepuje spadek. Dla wartości **10.2** skuteczność funkcji została oceniona na poziomie **92.3%**. Obliczona wartość oznacza, że jeżeli odległość pomiędzy dwoma wektorami, które leżą najbliżej siebie, jest mniejsza niż 10.2 to system uzna, że owe wektory dotyczą tej samej osoby, przez co proces będzie kontynuowany. W przeciwnym wypadku zostanie wygenerowany błąd z informacją, że osoba przedstawiona na zdjęciu, która podlega weryfikacji, nie jest znana systemowi.



Rysunek 4.6: Skuteczność walidatora (funkcji oceniającej, czy podane wektory dotyczą tej samej osoby) w zależności od wartości progu (parametru “x”)
Źródło własne

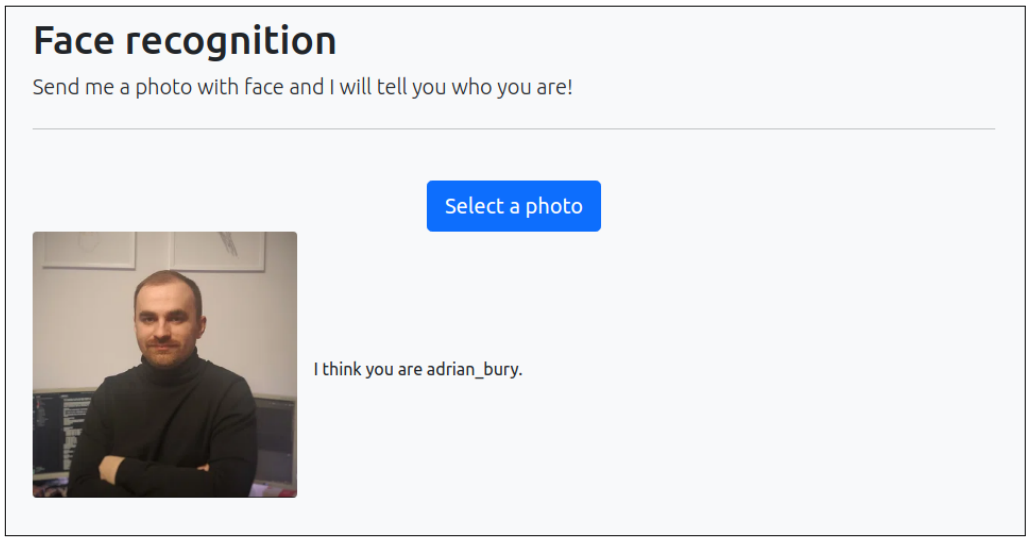
4.4 Klasyfikator

Ostatnim krokiem w procesie działającym po stronie serwera (rysunek 2.1) jest wykonanie klasyfikacji. Na tym etapie wiadome już jest, że wektor cech, który trafił do klasyfikatora, reprezentuje osobę, która jest znana systemowi. Jest to bardzo istotna informacja, ponieważ w przeciwnym wypadku klasyfikator mógłby dokonać klasyfikacji pomimo tego, że dana klasa (w tym wypadku osoba) nie istnieje w systemie, co oznaczałoby zawsze błędną klasyfikację.

Do wykonywania klasyfikacji został wykorzystany klasyfikator SVM z liniowym jądrem, z parametrem C równym 1.0 (równanie 3.5) oraz strategią jeden na jednego (rysunek 3.10). Jego skuteczność na opisanym wcześniej zbiorze (rysunek 4.5) wyniosła **96.1%**. Z powodu tak wysokiej skuteczności nie były testowane inne klasyfikatory. Implementacja klasyfikatora SVM została zaczerpnięta z biblioteki *sklearn* [2].

4.5 Strona internetowa

Zaprojektowana strona internetowa składa z prostego formularza. Na stronie został umieszczony przycisk, który umożliwia wybranie zdjęć z dysku urządzenia klienta. Po wybraniu zdjęcia pojawi się jego podgląd, po czym plik jest wysyłany na serwer. Po uzyskaniu odpowiedzi z serwera zostanie wyświetlony otrzymany komunikat. Wygląd strony zostaw przedstawiony na zdjęciu 4.7.



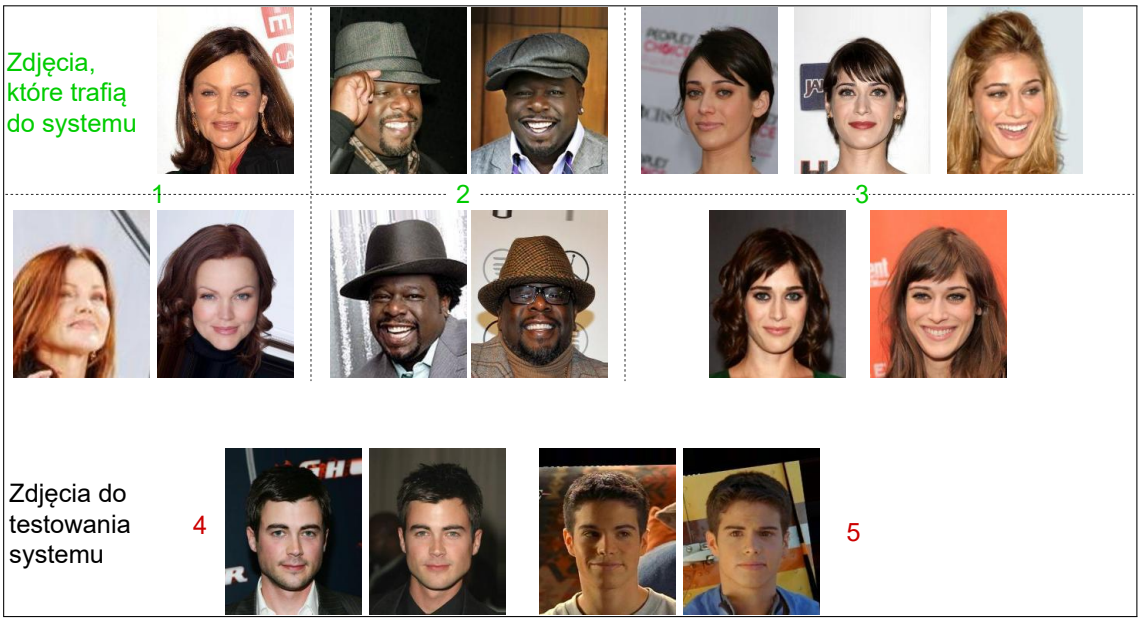
Rysunek 4.7: Wygląd strony internetowej po wybraniu zdjęcia z dysku i uzyskaniu odpowiedzi z serwera
Źródło własne

Rozdział 5

Testowanie systemu

Testowanie napisanej aplikacji będzie polegać na ręcznym wprowadzeniu wybranych zdjęć do systemu. Wszystkie zdjęcia twarzy, użyte do testowania aplikacji, zostały pobrane ze zbioru CelebA [13]. Wybrane zdjęcia (rysunek 5.1) zostały podzielone na dwie kategorie:

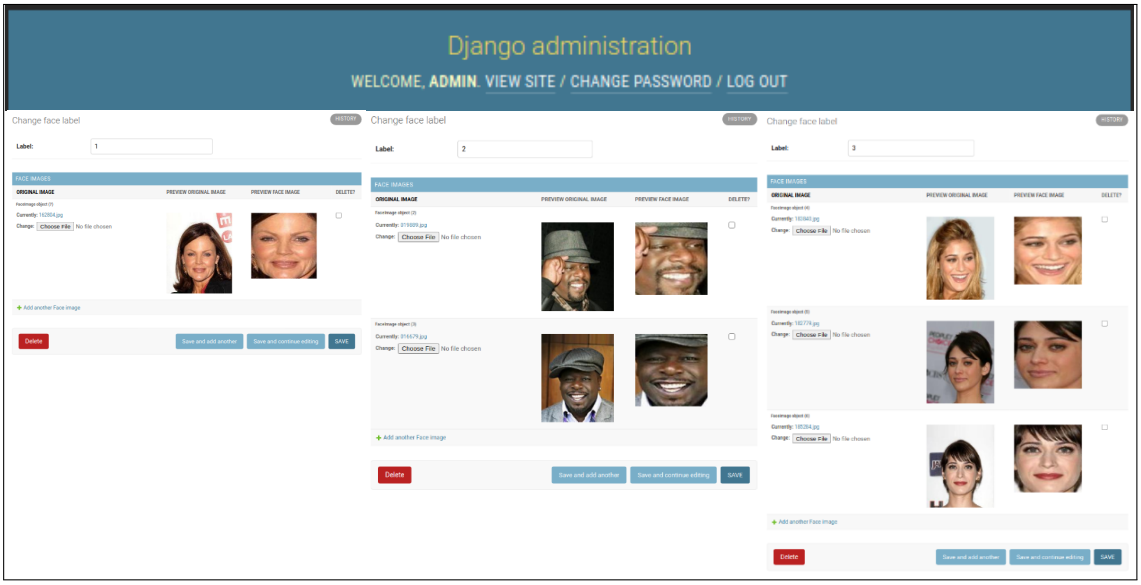
- zdjęcia, które trafią do systemu jako wzorzec,
- zdjęcia, które zostaną wykorzystane w celu zbadania zachowania oraz poprawności komunikatów zwracanych przez system.



Rysunek 5.1: Użyte zdjęcia do przetestowania systemu. Obok każdej grupy przedstawiona jest etykieta zdjęć.
Źródło własne

5.1 Wgranie zdjęć do systemu

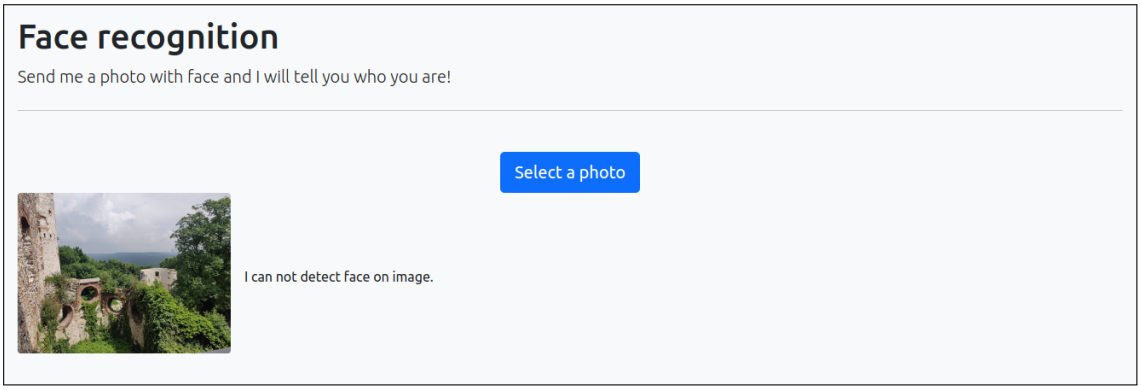
Aplikacja udostępnia dla administratorów panel, za pomocą którego jest możliwe wprowadzenie zdjęć do systemu z poziomu przeglądarki. Po zalogowaniu się zostały wprowadzone wyżej przedstawione zdjęcia (rysunek 5.1). Dla etykiety oznaczonej numerem jeden zostało wprowadzone jedno zdjęcie, oznaczonej numerem dwa - dwa zdjęcia przedstawiające tę samą osobę, dla kolejnej trzy. Podgląd wprowadzonych zdjęć oraz wykadrowanych twarzy został przedstawiony na rysunku 5.2.



Rysunek 5.2: Podgląd wprowadzonych zdjęć do systemu.
Źródło własne

5.2 Zdjęcie niezawierające twarzy

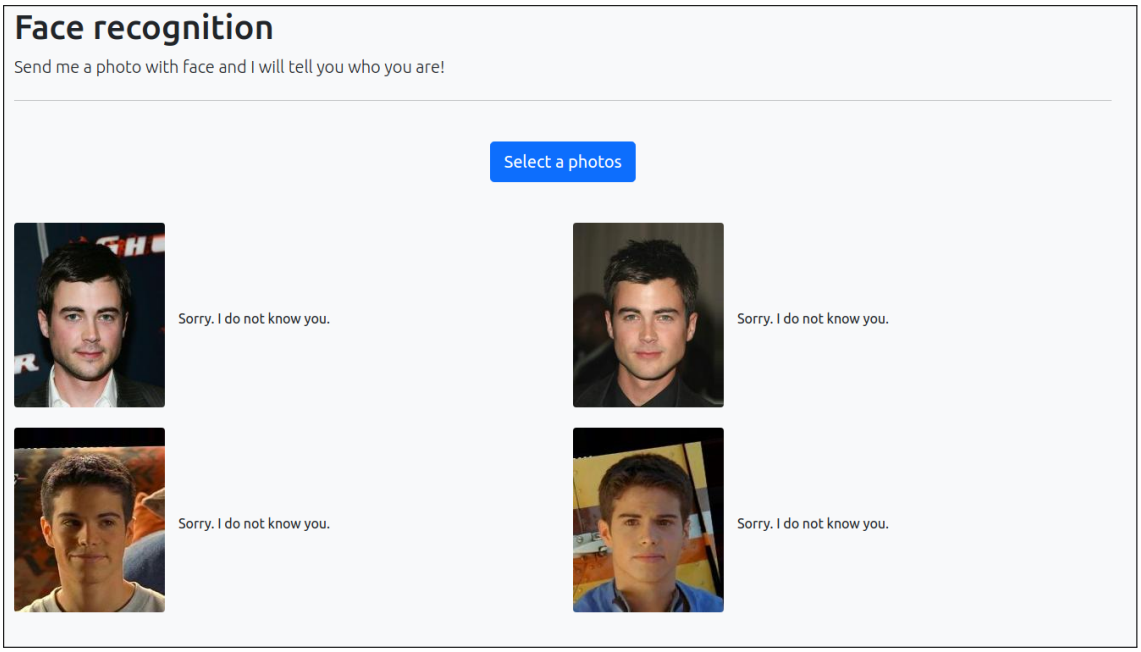
Po wysłaniu zdjęcia na serwer, zgodnie ze schematem przedstawionym na rysunku 2.1, pierwszym krokiem jest kadrowanie twarzy. Informacja na temat samego kadrowania nie jest znana użytkownikowi końcowemu, natomiast w przypadku, gdy zostanie wysłane zdjęcie niezawierające twarzy, to użytkownik powinien dostać informację zwrotną. W celu sprawdzenia, czy funkcjonalność zadziała prawidłowo, zostało wysłane na serwer zdjęcie, które zawiera krajobraz. Z powodu, że na zdjęciu nie ma przedstawionych żadnych osób, system powinien zwrócić informację o błędzie. Wynik testu został pokazany na rysunku 5.3. Komunikat o braku twarzy został zwrócony, co oznacza, że system zachował się poprawnie.



Rysunek 5.3: Zachowanie systemu po wprowadzeniu zdjęcia niezawierającego twarzy.
Źródło własne

5.3 Zdjęcie osoby nieznanej systemowi

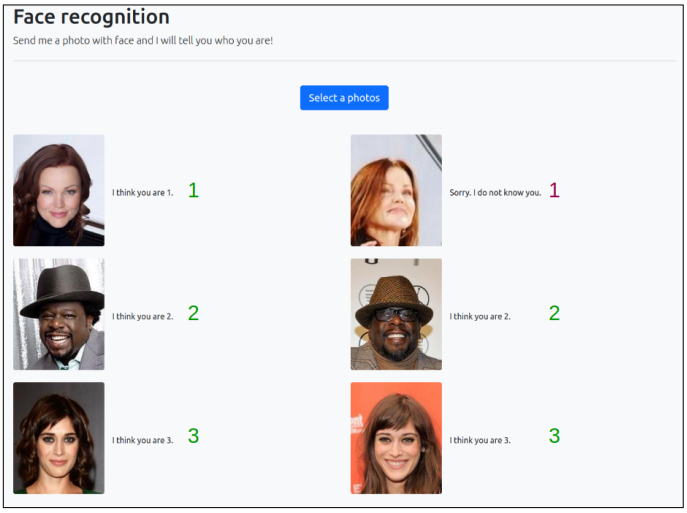
Kolejnym krokiem do przetestowania jest sprawdzenie czy system zwróci informację, gdy zostanie wysłane zdjęcie osoby, która nie została wcześniej zaimportowana. Z tego powodu system powinien zwrócić informację, że nie rozpoznano osoby na zdjęciu. W tym celu zostały przygotowane zdjęcia mężczyzn oznaczone etykietą "4" oraz "5" (rysunek 5.1). Wyniki testu zostały przedstawione na rysunku 5.4. Dla każdego ze zdjęć został zwrócony komunikat z informacją, że nie rozpoznano osoby na zdjęciu. Oznacza to, że system zachował się poprawnie i test można uznać za zaliczony.



Rysunek 5.4: Zachowanie systemu po wybraniu zdjęć nieznanych systemowi
Źródło własne

5.4 Zdjęcie osoby znanej systemowi

Ostatnim krokiem w procesie jest klasyfikacja, czyli zwrócenie informacji kto został przedstawiony na przesłanym zdjęciu. W celu dokładniejszego przetestowania napisanej funkcjonalności specjalnie w tym celu zostały wgrane do systemu zdjęcia osób o różnej liczbie (rysunek 5.2), aby sprawdzić, czy wpłynie to na otrzymane wyniki. Wyniki testu zostały przedstawione na rysunku 5.5. Komunikaty zwrócone przez system dla zdjęć reprezentujących etykiety "2" oraz "3" są poprawne. Dla jednego ze zdjęć przedstawiających osobę o etykiecie "1" system zwrócił informacje, że nie rozpoznano osoby na zdjęciu. **Najprawdopodobniej** spowodowane jest to tym, że w systemie znajduje się tylko **jedno** zdjęcie reprezentujące etykietę "1" przedstawiające kobietę o ciemnym kolorze włosów, a zdjęcie, dla którego system zwrócił błędny komunikat, przedstawia tę samą kobietę, tylko że w innym kolorze włosów. Dodatkowo zdjęcie twarzy zrobione jest z inne profilu, częściowo zasłoniętego przez włosy (rysunek 5.6).



Rysunek 5.5: Zwrócone komunikaty przez system dla wybranych zdjęć. Numer obok komunikatu przedstawia ID osoby na zdjęciu w systemie. Źródło własne

Rysunek 5.6: Porównanie zdjęcia wgranego do systemu (po lewej) ze zdjęciem źle zakwalifikowanym przez system (po prawej). Źródło własne

5.5 Skuteczność algorytmu rozpoznawania twarzy

Przedstawione dotychczas testy zostały wykonane na bardzo małym zbiorze twarzy. Do aplikacji zostało wgranych tylko 6 zdjęć przedstawiających twarze 3 osób. Wykonanie statystyk na tak małym zbiorze jest niereprezentatywne, dlatego tym razem został przygotowany zestaw zdjęć składający się z 2 000 różnych osób, łącznie 10 000 zdjęć twarzy. Do samej aplikacji, z przygotowanego zbioru, zostało wgranych 5 000 zdjęć twarzy, przedstawiających 1 000 różnych osób (po 5 zdjęć na jedną osobę). Pozostałe 5 000 zdjęć zostało wykorzystanych do testowania zwracanych komunikatów przez aplikację. Wyniki z testów zostały przedstawione w tabeli 5.1. Z przeprowadzonych testów wynika, że skuteczność aplikacji wynosi 85,94%, a najbardziej newralgicznym punktem w całym zaprojektowanym systemie jest walidator, który wygenerował aż 639 z 703 wszystkich błędnych komunikatów.

Tablica 5.1: Wyniki przeprowadzonych testów

Liczba wykonanych testów	5 000
Suma błędnych komunikatów	703
Liczba zdjęć, na których nie znaleziono twarzy	12
Liczba błędnych komunikatów walidatora	639
Liczba błędnych komunikatów klasyfikatora	52

Podsumowanie

Użytkownik, dzięki temu, że komunikacja z aplikacją odbywa się przez protokół HTTP, może połączyć się z serwerem, z dowolnego urządzenia mającego dostęp do internetu oraz do przeglądarki internetowej, w celu wysłania zdjęcia na serwer. Jeżeli to urządzenie posiada również dostęp do kamery, to jest możliwość bezpośredniego wysłania zrobionego zdjęcia od razu do systemu. W odpowiedzi na wysłane zdjęcie zostanie zwrócony komunikat, zawierający kim jest osoba przedstawiona na zdjęciu, pod warunkiem, że ta osoba została wcześniej wgrana do systemu.

Dużym ułatwieniem w zarządzaniu bazą zdjęć jest przygotowany panel administratora, który pozwala na łatwe zarządzanie zdjęciami, które znajdują się w systemie. Podgląd aktualnych lub dodawanie kolejnych zdjęć ogranicza się tylko do paru kliknięć. Niestety ręczne wysyłanie bardzo dużej liczby zdjęć za pomocą panelu może wymagać sporo czasu, dlatego w takim przypadku lepiej jest przygotować skrypt, który zaimportuje bezpośrednio zdjęcia z dysku do systemu.

Próg, po którego przekroczeniu, walidator oceniający podobieństwo uzna, że dane zdjęcia nie są do siebie podobne został wyliczony na 10.2. Przy tej wartości, na badanym zbiorze, została osiągnięta maksymalna skuteczność wynosząca 92.3%. W zależności od tego, gdzie napisany system zostałby użyty, wartość tą można zmodyfikować. W przypadku, gdy system zostałby użyty w miejscu, gdzie błędy fałszywie pozytywne są niedopuszczalne, wartość ta powinna zostać zmniejszona. Spowoduje to, że ogólna skuteczność walidatora spadnie, natomiast częstotliwość występowania wyników fałszywie pozytywnych zostanie **najprawdopodobniej** ograniczona.

Wymagania, jakie zostały podstawione systemowi w rozdziale pierwszym, zostały w pełni zrealizowane. Wyniki testów pokazały, że aplikacja poprawnie rozpoznaje zdjęcia niezawierające twarzy, jest w stanie również ocenić z wysokim prawdopodobieństwem czy osoba rozpoznana na zdjęciu jest znana systemowi oraz zwrócić informację na temat osoby przesłanej do systemu po jej rozpoznaiu

Literatura

[1] Jason Brownlee. *How to Develop a Face Recognition System Using FaceNet in Keras*. Amerykański. 7 czer. 2019. URL: https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier.

[2] Lars Buitinck i in. "API design for machine learning software: experiences from the scikit-learn project". W: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, s. 108–122.

[3] Gal Chechik i in. "Large Scale Online Learning of Image Similarity Through Ranking." W: *Journal of Machine Learning Research* 11.3 (2010).

[4] Alex Clark. *Image file formats — Pillow (PIL Fork) 8.2.0 documentation*. Angielski. URL: https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html#fully-supported-formats.

[5] *Django documentation — Django*. Angielski. URL: https://docs.djangoproject.com/en/3.2/.

[6] Facebook". *What is the face recognition setting on Facebook and how does it work?* Angielski. URL: https://www.facebook.com/help/122175507864081.

[7] Vladimir Goncharov. *Face detection and alignment with mtcnn*. Angielski. URL: https://github.com/open-face/mtcnn/blob/master/README.md.

[8] Google. *Wyszukiwanie osób, rzeczy i miejsc na zdjęciach - Komputer - Zdjęcia Google - Pomoc*. Angielski. URL: https://support.google.com/photos/answer/6128838.

[9] T. Hastie, R. Tibshirani i J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846. URL: https://books.google.pl/books?id=eBSgoAEACAAJ.

[10] Erik Hjelmås i Boon Kee Low. "Face Detection: A Survey". W: *Computer Vision and Image Understanding* 83.3 (2001), s. 236–274. ISSN: 1077-3142. DOI: https://doi.org/10.1006/cviu.2001.0921. URL: https://www.sciencedirect.com/science/article/pii/S107731420190921X.

[11] Iván de Paz Centeno. *ipazc/mtcnn*. Angielski. URL: https://github.com/ipazc/mtcnn.

[12] Ziwei Liu i in. "Deep Learning Face Attributes in the Wild". W: *Proceedings of International Conference on Computer Vision (ICCV)*. Grud. 2015.

[13] Microsoft". *MS-Celeb-1M: Challenge of Recognizing One Million Celebrities in the Real World*. Amerykański. 9 grud. 2020. URL: https://www.microsoft.com/en-us/research/project/ms-celeb-1m-challenge-recognizing-one-million-celebrities-real-world/.

[14] PimEyes". *Face Recognition Search Engine and Reverse Image Search*. Angielski. URL: https://pimeyes.com/en.

[15] S. Raschka i V. Mirjalili. *Python Machine Learning*. Packt Publishing, 2017. ISBN: 9781787126022. URL: https://books.google.pl/books?id=%5C_plGDwAAQBAJ.

[16] Florian Schroff, Dmitry Kalenichenko i James Philbin. "Facenet: A unified embedding for face recognition and clustering". W: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, s. 815–823.

[17] Hiroki Taniai. *model - Google Drive*. Holenderski. 21 sty. 2018. URL: https://drive.google.com/drive/folders/12aMYASGCKvDdkygSv1yQq8ns03AStDO_.

[18] Hiroki Taniai. *nyoki-mtl/keras-facenet*. Angielski. URL: https://github.com/nyoki-mtl/keras-facenet.

[19] Ivan William i in. "Face recognition using FaceNet (survey, performance test, and comparison)". W: *2019 Fourth International Conference on Informatics and Computing (ICIC)*. IEEE. 2019, s. 1–6.

[20] P. Zbiorowa. *Slownik jezyka polskiego PWN, wydanie III, dodruk*. Wydawnictwo Naukowe PWN, 1996.

[21] Kaipeng Zhang i in. "Joint face detection and alignment using multitask cascaded convolutional networks". W: *IEEE Signal Processing Letters* 23.10 (2016), s. 1499–1503.

Spis rysunków

Imię i Nazwisko Kraków, dnia .....

O Ś W I A D C Z E N I E

Świadomy(a) odpowiedzialności oświadczam, że przedłożona praca pt.:

Zastosowanie głębokiej sieci neuronowej o architekturze FaceNet do rozpoznawania twarzy

została napisana przeze mnie samodzielnie.

Jednocześnie oświadczam że w/w praca nie narusza praw autorskich w rozumieniu Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 nr 90, poz. 631 z późn. zmianami) oraz dóbr osobistych chronionych prawem cywilnym.

Przedłożona praca nie zawiera danych empirycznych ani też informacji, które uzyskałem(am) w sposób niedozwolony. Stwierdzam, iż przedstawiona praca w całości ani też w części nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z uzyskaniem dyplomu ani też nadania tytułów zawodowych.

.....

(podpis)