Politechnika Wrocławska

Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka (AIR)

PRACA DYPLOMOWA INŻYNIERSKA

TYTUŁ PRACY:

Wykorzystanie sieci neuronowej do określenia podobieństwa między dwoma twarzami

AUTOR:

Adam Jankowiak

PROMOTOR:

dr inż. Wojciech Domski

WROCŁAW 2023

Dedykuję tą pracę rodzicom i przyjaciołom, którzy wspierali mnie i pomagali pokonywać wyzwania. Szczególnie chciałbym podziękować mojej babci Uli.

Streszczenie

Głównym celem pracy było stworzenie modelu sieci neuronowej, który umożliwiłaby identyfikacje tej samej osoby na dwóch różnych zdjęciach. Aby zrealizować to zadanie została wykorzystana konwolucyjna sieć neuronowa wraz z zbiorem danych ORL [12], który został wykorzystany do uczenia modelu, jego walidacji oraz testowania. Dodatkowo zastosowano rozszerzenie zbioru w celu zwiększenia liczby próbek, dzięki czemu polepszyła się jakość modelu. Następnie przebadano wpływ dwóch różnych reprezentacji na jakość uzyskanej sieci. W celach testowych wykorzystano reprezentację składającą się z przeplatanych wierszy zdjęć oraz złączonych wierszy obrazów. W efekcie udało się stworzyć sieć neuronową, która była w stanie rozróżnić tożsamość osób na podstawie dwóch zdjęć. Otrzymano wartość precyzji wynoszącą około 98,6%. Następnie przebadano wpływ zwiększonej liczby warstw normalizacji wsadowej na jakość uzyskanych rezultatów. W rezultacie stwierdzono, że zbyt duża liczba warstw normalizacyjnych wpłynęła negatywnie na jakość sieci.

Summary

The main goal of the thesis was to create a neural network model that would enable identification of the same person in two different photos. To accomplish this task, a convolutional neural network was used along with an ORL dataset [12], which was used to train the model, validate it and test it. In addition, the dataset was extended to increase the number of samples, thus improving the quality of the model. The effect of two different representations on the quality of the resulting network was then studied. For testing purposes, a representation consisting of interlaced rows of images and concatenated rows of images was used. As a result, it was possible to create a neural network that was able to distinguish the identity of people from two images. A precision value of 98,6% was obtained. The effect of an increased number of batch normalization values on the quality of the results was then studied. As a result, it was found that too many normalization layers negatively affected the quality of the network.

Słowa kluczowe: sztuczna inteligencja, głębokie sieci neuronowe, keras, rozpoznawanie twarzy, sztuczne sieci neuronowe, uczenie maszynowe, rozszerzanie zbioru

Keywords: artificial intelligence, deep neural networks, keras, face recognition, artificial neural networks, machine learning, dataset augmentation

Spis treści

Rozdział 1

Wstęp

Sztuczna inteligencja jest ogólnym terminem, który można określić jako zbiór metod i algorytmów umożliwiających maszynom na samodzielne uczenie się, adaptację do zmieniających się warunków oraz naśladowanie ludzkiej inteligencji [13]. Początki badań nad SI zaczęły się już w latach pięćdziesiątych XX w. Bardzo istotnym z punktu widzenia rozwoju tej dziedziny wiedzy był Dartmouth workshop, czyli letnie seminarium naukowe w czasie, którego naukowcy dyskutowali na temat sztucznej inteligencji. Wydarzenie jest obecnie uważane za kamień milowy dzięki, któremu zaczęły się pierwsze etapy pracy nad nową technologią.

W dzisiejszych czasach sztuczna inteligencja jest wykorzystywana w praktycznie wszystkich dziedzinach życia. Człowiek korzystając z mediów społecznościowych, wyszukiwarki internetowej, logując się do banku, oglądając filmy, bądź dokonując zakupów online jest stale otoczony sztuczną inteligencją. Mimo wszechobecnego wykorzystania posiada ona także niedoskonałości. Głównymi problemami z którymi boryka się SI jest brak możliwości samodzielnego, kreatywnego myślenia. Drugim znaczącym problemem jest wpływ na dane wejściowe. Celowa zmiana informacji, które są analizowane przez algorytm może wpłynąć negatywnie na jakość uzyskiwanych rezultatów.

Wraz z coraz to większym rozwojem sztucznej inteligencji, Unia Europejska planuje wprowadzenie nowych przepisów regulujących wykorzystanie SI. Nowa europejska ustawa o sztucznej inteligencji (z ang. *EU Artificial Intelligence Act*) dzieli sztuczną inteligencję na trzy kategorie ryzyka. Pierwszą kategorią są programy tworzące nieakceptowalne ryzyko, przykładowo jest to system do oceny społeczeństwa, które będą w pełni zakazane. Drugą kategorią są aplikacje wysokiego ryzyka, takie jak oprogramowanie skanujące CV lub algorytmy do rozpoznawania twarzy. W ich przypadku zostaną wprowadzone odpowiednie regulacje, które będą ograniczały ich wykorzystanie. Trzecią kategorią są aplikacje, które nie zostały zakazane w wcześniejszych klasach dzięki czemu są nieregulowane. Wprowadzenie nowych przepisów może skutkować zakazaniem stosowania algorytmów rozpoznawania twarzy bądź je w znaczący sposób ograniczać. Nowa ustawa o sztucznej inteligencji może stać się światowym standardem przez co będzie obowiązywała na całym świecie tak samo jak ustawa o ochronie danych, RODO.

Uczenie maszynowe składa się z wielu różnych technologii. Głównymi różnicami, które wpływają na podział są wykorzystywane algorytmy oraz odpowiedni sposób dostarczenia danych. Wśród najbardziej popularnych i znaczących można wyróżnić uczenie nadzorowane jak i nienadzorowane (z ang. *supervised learning, unsupervised learnng*). Uczenie nadzorowane polega na dostarczeniu odpowiedniej bazy danych, która pozwala maszynie na określenie odpowiedniego wyniku lub znalezienie pewnej relacji. Umożliwia to stworzenie pewnego rodzaju wzorca pozwalającego na rozwiązywanie analogicznych problemów.

Jest ono wykorzystywane w zadaniach, dla których dane wejściowe nie zmieniają się bądź się powtarzają. Drugim podejściem jest uczenie nienadzorowane. Różni się on tym, że maszyna nie dostaje gotowej bazy danych, ale musi samodzielnie znajdować zależności i wyciągać z nich odpowiednie wnioski. Wraz z zwiększającą się ilością dostępnych danych polepsza się jakość uzyskiwanych rezultatów. Uczenie nienadzorowane bardzo często jest wykorzystywane przez portale ogłoszeniowe w celu szybkiego dostosowywania cen danych produktów w zależności od popytu jak i podaży.

Głębokie uczenie maszynowe opiera się na implementacji sztucznej sieci neuronowej posiadającej jedną warstwę wejściową, wiele warstw ukrytych i jedną warstwę wyjściową. Każda warstwa posiada jednostki (neurony), które przetwarzają dane i propagują je dalej do kolejnej warstwy. Tego typu podejście nazywane jest zadaniem predykcyjnym. Daje ono możliwość do samodzielnego uczenia się algorytmu za pomocą przetworzonych danych.

Rozpoznawanie polega na ustaleniu podobieństwa pomiędzy danym przedmiotem bądź też osobą w oparciu o wcześniejsze doświadczenia. W codziennym życiu ludzki mózg bez problemu jest w stanie rozpoznać w tłumie znajome osoby.

Algorytm rozpoznawania twarzy służy do określenia podobieństwa pomiędzy dostarczonym zdjęciem osoby a bazą danych zawierającą odpowiednie informacje. Porównywanie dwóch zdjęć twarzy może być pomocne przy określaniu tożsamości osób. Rozpoznawanie twarzy z wysoką dokładnością jest uważane za innowacyjne. Należy do kategorii oprogramowania biometrycznego, które skupia się na zmapowaniu rysów twarzy i utworzeniu modelu twarzy. Określanie tożsamości ludzi może być przydatne w celu zapewnienia bezpieczeństwa podczas wydarzeń masowych. Wykorzystywane jest także w poszukiwaniu osób zaginionych lub znajdowaniu przestępców [2]. Jest to możliwe dzięki wykorzystaniu głębokiego uczenia maszynowego, które porównuje otrzymany obraz z wcześniej uzyskanym modelem twarzy.

Celem pracy jest zbadanie wpływu algorytmu na jakość oceny podobieństwa między dwoma osobami. Aby zrealizować to zadanie została wykorzystana konwolucyjna sieć neuronowa wraz z nowo utworzonym zbiorem danych, które zostały wykorzystane do uczenia modelu jak i przetestowania samego algorytmu. Na wejście algorytmu podawane są dwa zdjęcia twarzy po czym zwracana jest wartość podobieństwa pomiędzy osobami widocznymi na zdjęciach.

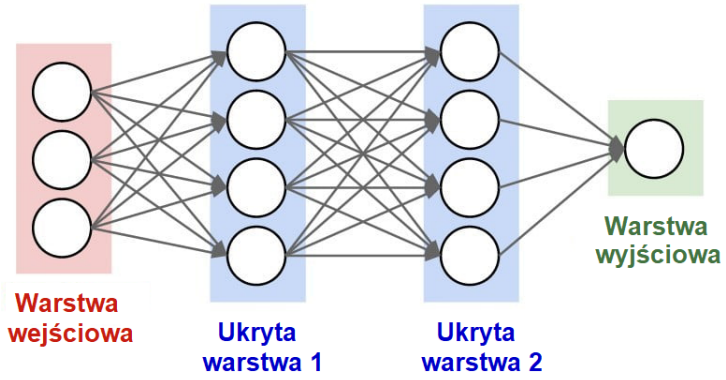
1.1 Teza

Możliwa jest identyfikacja osoby na dwóch różnych zdjęciach wykorzystując metody głębokiego uczenia.

Rozdział 2

Sieć neuronowa

Sieć neuronowa jest odzwierciedleniem budowy ludzkiego mózgu. Umożliwia rozwiązywanie wszelakich problemów sztucznej inteligencji, głębokiego uczenia oraz uczenia maszynowego. Sztuczne sieci neuronowe zwane także symulowanymi sieciami neuronowymi są główną częścią głębokiego uczenia. Ich nazwa jak i budowa nawiązują do mózgu człowieka. Sieć naśladuje naturalne połączenia i przekazuje informacje pomiędzy neuronami [4]. Jak można zauważyć na poniższym rysunku 2.1, sztuczne sieci neuronowe zbudowane są z warstw neuronów, które zawierają warstwę wejściową, jedną lub kilka warstw ukrytych i warstwę wyjściową. W gęstych sieciach wszystkie neurony są połączone pomiędzy sobą oraz posiadają odpowiednie wagi jak i progi. Jeżeli wartość progowa neuronu jest powyżej danego progu wtedy ten neuron zostaje aktywowany przez co dane zostają przesłane do kolejnej warstwy. W przeciwnym wypadku neuron jest nieaktywny.



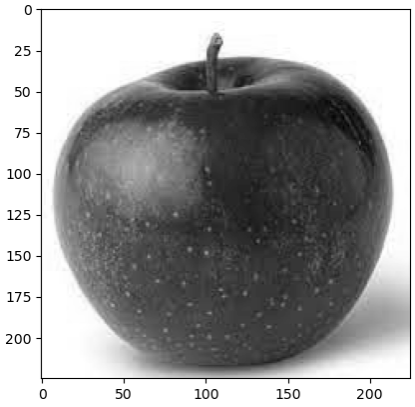
Rysunek 2.1 Przedstawia architekturę sieci neuronowej [9].

2.1 Wymagania sieci neuronowej

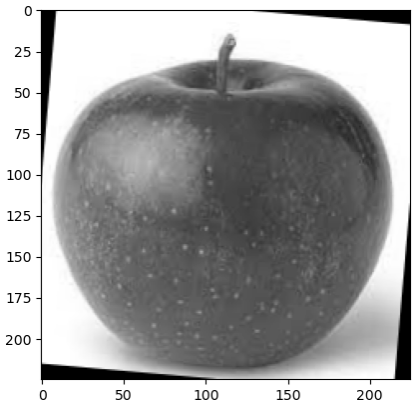
W trakcie tworzenia sieci neuronowej należało pamiętać o spełnieniu odpowiednich warunków. Jednym z bardziej istotnych wymagań było zastosowanie prawidłowego rozmiaru zbioru danych [1]. Zbyt mała baza mogła skutkować uzyskaniem niskiej precyzji. Zależnie od danego problemu, istnieje możliwość uzyskania większej liczby próbek niż była dostępna w początkowym zbiorze. Należy przy tym pamiętać o zachowaniu reprezentatywności zbioru danych. W przypadku braku możliwości uzyskania większej liczby próbek można wykorzystać dwie metody. Pierwsza metoda polega na sztucznym wygenerowaniu danych wykorzystując do tego symulację. Drugą możliwością jest rozszerzenie zbiorów, które polega na wykorzystaniu już istniejącej bazy do stworzenia nowych próbek.

Istnieje wiele różnych technik umożliwiających powiększenie początkowego zbioru danych. Wykorzystanie danej metody zależy w głównej mierze od występującego problemu. Istotnym elementem, w trakcie tworzenia sieci, jest sprawdzenie czy dana strategia może być wykorzystana. Co więcej, niektóre klasy problemów posiadają swoje własne techniki, których nie można wykorzystać dla innych klas. Rozszerzenie zbioru może być wykorzystane dla zbiorów składających się z danych liczbowych, dźwięku, szeregów czasowych bądź też obrazów [3].

W celu rozszerzenia zbioru danych dla bazy zawierającej zdjęcia można wykorzystać wiele różnych technik. Jedną z bardziej powszechnych metod jest rotacja zdjęcia o mały kąt, zwykle stosuje się rotację o $\pm$ 5°. W tym celu oryginalne zdjęcie, rysunek 2.2, jest obracane względem środka dzięki czemu uzyskuje się nowy obraz, rysunek 2.3. Wprowadzenie rotacji powoduje zmianę położenia pikseli dzięki czemu sieć neuronowa widzi obrócony obraz jako nową, unikalną próbkę danych.

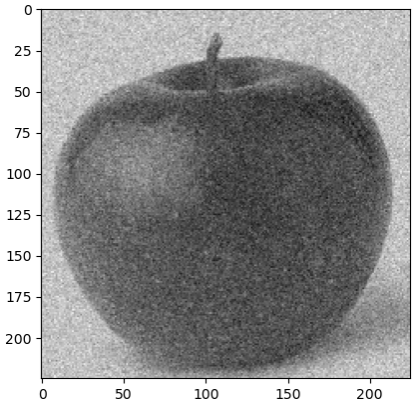


Rysunek 2.2 Przedstawia oryginalne zdjęcie.



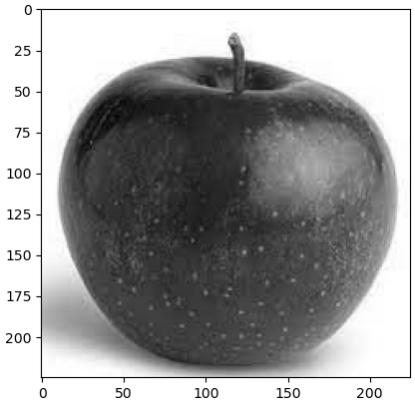
Rysunek 2.3 Przedstawia obrócone zdjęcie.

Druga metoda opiera się na dodaniu szumu. W tym celu można wykorzystać szum Gaussa, który wprowadzi odpowiednie zniekształcenia na obrazie 2.4. Nowy, zaszumiony obraz dla człowieka będzie w znacznym stopniu zbliżony do oryginału, ale dla sieci jest widziany jako odmienna próbka.

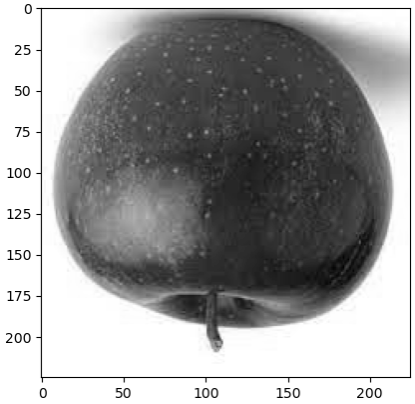


Rysunek 2.4 Przedstawia zdjęcie z szumem Gaussa.

Trzecia technika wykorzystywana do rozszerzenia zbioru polega na stworzeniu lustrzanego odbicia. Można wyróżnić odbicie horyzontalne, rysunek 2.5 jak i wertykalne, rysunek 2.6. Wynikowe próbki różnią się od oryginalnego obrazu, lecz ze względu na problem poruszany w tej pracy nie można było nich wykorzystać. Jest to spowodowane tym, że ludzka twarz jest symetryczna, przez co odbicie horyzontalne nie wpłynęłoby na jakość sieci. Dodatkowo odbicie wertykalne spowodowałoby odwrócenie twarzy o 180° co było nieporządne w tym projekcie.

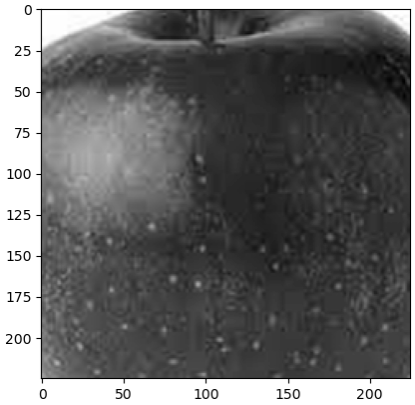


Rysunek 2.5 Przedstawia zdjęcie w odbiciu horyzontalnym.

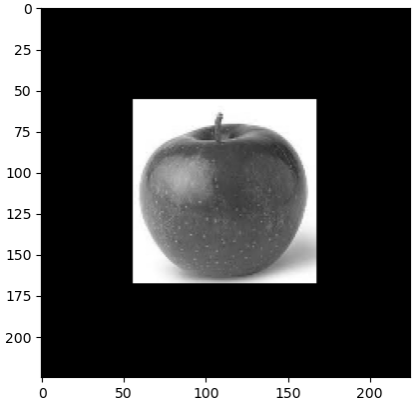


Rysunek 2.6 Przedstawia zdjęcie w odbiciu wertykalnym.

Czwarta metoda umożliwiająca rozszerzenie zbioru opiera się na przybliżeniu lub oddaleniu obrazu. Na poniższym rysunku 2.7 zostało przedstawione powiększenie fotografii przy jednoczesnym zachowaniu rozmiarów. Powiększając obraz należy pamiętać o zachowaniu cech, które będą wykrywane przez sieć. Druga operacja, przestawiona na rysunku 2.8, obrazuje oddalenie fotografii. Wykorzystując oddalenie należy pamiętać o zachowaniu szczegółów obrazu.

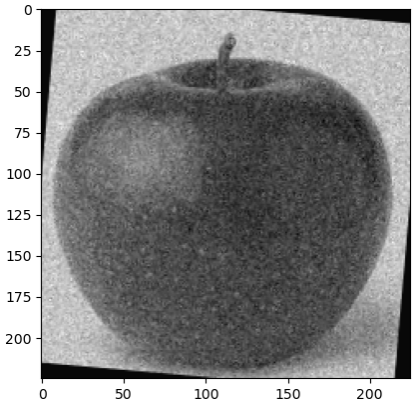


Rysunek 2.7 Przedstawia przybliżone zdjęcie.



Rysunek 2.8 Przedstawia oddalone zdjęcie.

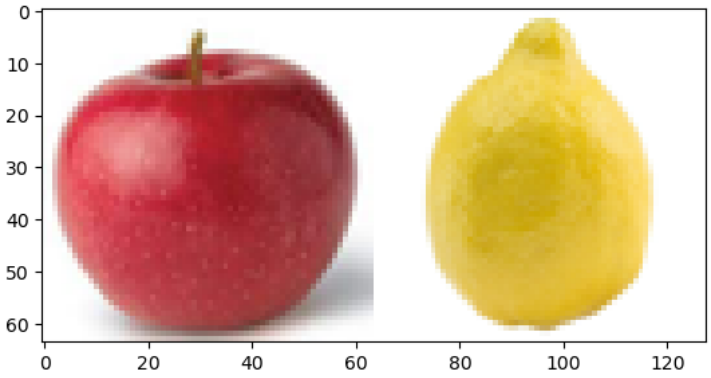
Istnieje także możliwość połączenia ze sobą kilku metod rozszerzenia zbioru. Dzięki zastosowaniu kombinacji różnych metod można w łatwy i szybki sposób zwiększyć liczbę próbek w zbiorze. Jak można zauważyć na poniższym rysunku 2.9 przedstawione jest zdjęcie, do którego dodano szum Gaussa a następnie zostało poddane rotacji o kąt 5°.



Rysunek 2.9 Przedstawia obrócone zdjęcie z szumem Gaussa.

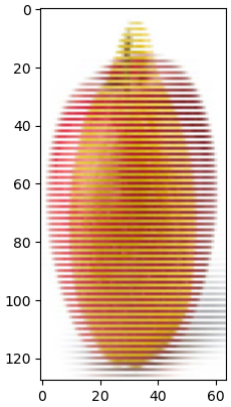
2.2 Reprezentacja danych

Na jakość sieci neuronowej wypływa wiele różnych elementów. Jednym z bardziej istotnych czynników jest sposób dostarczenia danych. W celach demonstracyjnych zostały wykorzystane dwa zdjęcia owoców, jabłka oraz cytryny. Pierwszym sposobem reprezentacji danych jest połączenie ze sobą wierszy z dwóch oddzielnych fotografii. Na rysunku 2.10, obraz wyjściowy składa się z dwóch złączonych ze sobą zdjęć. Metoda ta polega na połączeniu ze sobą wierszy dwóch próbek.



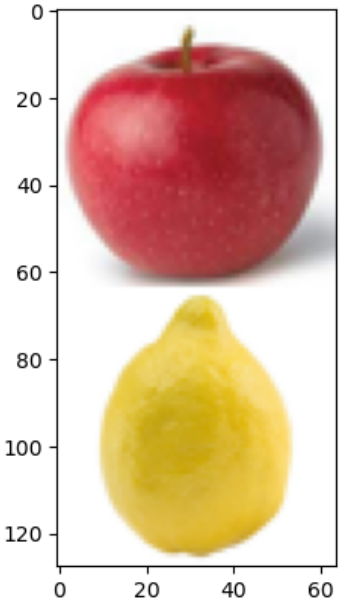
Rysunek 2.10 Przedstawia złączenie fotografii.

Drugim sposobem na reprezentowanie danych jest wzajemne przeplatanie wierszy pomiędzy sobą. Jak można zauważyć na rysunku 2.11, wynikowy obraz poprzez przekładanie się wierszy obrazów, dwukrotnie zwiększa swoją wysokość.



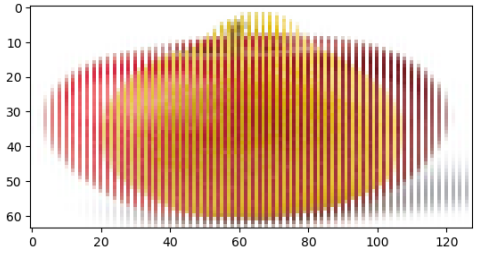
Rysunek 2.11 Przedstawia przeplatanie wierszy obrazów.

Trzecia metoda polega na łączeniu ze sobą kolumn poszczególnych zdjęć. Jak można zauważyć na poniższym rysunku 2.12, wynikowy obraz został zbudowany za pomocą łączenia kolumn dla dwóch różnych obrazów.



Rysunek 2.12 Przedstawia łączenie kolumn obrazów.

Czwarta metoda umożliwiająca reprezentowanie danych polega na wzajemnym przeplataniu się kolumn. Jak można zauważyć na poniższym rysunku 2.13 poszczególne kolumny zostały umieszczone pomiędzy sobą.



Rysunek 2.13 Przedstawia przeplatanie kolumn obrazów.

Rozdział 3

Wykorzystane technologie

Jednym z ważniejszych etapów pracy nad projektem było wybranie odpowiedniego środowiska oraz narzędzi programistycznych, które umożliwiły rozwiązanie problemu. Projekt został zrealizowany w oparciu o język Python w wersji 3.9. Python jest językiem programowania wysokiego poziomu, rozwijanym jako projekt z otwartym oprogramowaniem (z ang. *Open Source*), zawierającym dużą ilość bibliotek. Został on wybrany ze względu na czytelność oraz zwięzłość kodu [8]. Python jest niezwykle łatwy w nauce jaki i codziennym użytkowaniu. Jako język programowania wysokiego poziomu pozwala użytkownikowi skupić się na podstawowych funkcjach tworzonego programu. Program został napisany w oparciu o środowisko Jupyter Notebook, które umożliwia szybkie prototypownie i sprawdzanie działania tylko poszczególnych fragmentów kodu.

3.1 Środowiska programistyczne

Tensorflow jest otwarto źródłową biblioteką, wykorzystywaną jako kompleksowa platforma do uczenia maszynowego [11]. Używana jest do programowania przypływu danych w zależności od różnych zadań. Oferuje wiele poziomów abstrakcji do trenowania i budowania modeli. Tensorflow jest dynamicznie rozwijającą się platformą, posiadającą duży zasób danych, bibliotek oraz narzędzi. Wykorzystanie rozwiniętego środowiska wraz z szerokim ekosystemem narzędzi umożliwiło dynamiczne tworzenie modeli oraz szybkie znajdowanie błędów dzięki przejrzystości kodu.

Keras jest wysokopoziomowym interfejsem programowania aplikacji (z ang.*Application Programming Interface, API*) napisanym w Pythonie [5]. Został on zaprojektowany w celu szybkiego tworzenia nowych, eksperymentalnych sieci neuronowych. Interfejs w głównej mierze skupia się na modułowości, rozszerzalności oraz jest przyjazny dla użytkownika. Biblioteka została wykorzystana w tym projekcie ze względu na możliwość łatwego i szybkiego prototypownia nowych sieci neuronowych, co przy dużej liczbie danych w tej pracy było niezbędne. Keras od 2017 roku jest w pełni zintegrowany z platformą programistyczną (z ang. *framework*) Tensorflow. Użytkownicy mają dostęp do pakietu za pomocą modułu tf.keras, jednak ciągle można korzystać z biblioteki Keras niezależnie.

Wśród innych platform umożliwiających tworzenie sieci neuronowych można wyróżnić PyTorch, który jest stosunkowo nową platformą uczenia maszynowego. Uważany jest za stosunkowo prosty, łatwy w obsłudze, wydajny pamięciowo i elastyczny. W porównaniu do Keras, PyTorch jest znacznie szybszy oraz posiada lepsze możliwość znajdowania błędów w kodzie. Zaletą Kerasa nad PyTorch jest możliwość szybszego budowania, trenowania jak i oceniania sieci. Obie platformy są bardzo popularne, oferują duże zbiory edukacyj-

ne. Głównym powodem wykorzystania w tym projekcie platformy Keras była możliwość szybkiego prototypownia.

Inną biblioteką jest Theano, która niegdyś była jedną z bardziej popularnych otwarto źródłowych bibliotek do głębokiego uczenia. Pozwalała ona na definiowanie, optymalizowanie i ocenianie wyrażeń matematycznych takich jak wielowymiarowe tablice oraz macierze. Theano umożliwia wykonywanie szybkich obliczeń oraz trenowanie algorytmów głębokiego uczenia. Dużą zaletą jest wieloplatformowość jak i możliwość pracy na jednostkach centralnych (CPU) oraz procesorach graficznych (GPU). Tensorflow w porównaniu do Theano również może działać na CPU jaki i GPU, jednak Tensorflow dominuje dzięki wykorzystaniu obliczeń grafowych co umożliwia lepsze wizualizowanie budowanej sieci neuronowej, dzięki czemu można łatwiej znajdować błędy.

3.2 Jednostki obliczeniowe

Tworzenie sieci neuronowej było czasochłonnym procesem, który musiał być w odpowiedni sposób zoptymalizowany. Duży wpływ na czas nauki procesu miało wykorzystane środowisko wykonawcze. Spośród głównych jednostek obliczeniowych można wyróżnić jednostkę centralną (z ang. *central processing unit, CPU*), procesor graficzny (z ang. *graphics processing unit, GPU*) oraz jednostkę do przetwarzania tensorów (z ang. *tensor processing unit, TPU*) [7].

Procesor jest głównym podzespołem zarządzającym funkcjami komputera. Służy on do zarządzania podstawowymi obliczeniami arytmetycznymi, logicznymi oraz operacjami wejścia/wyjścia. Odpowiada także za wykonywanie instrukcji programów komputerowych.

Procesor graficzny został stworzony w celu zarządzania wysokowydajnymi zadaniami, takimi jak wizualizacja oraz renderowanie grafiki, przez co znacząco wpływa na zmniejszenie obciążenia samego procesora. GPU posiada wiele rdzeni dzięki czemu jest w stanie szybko wykonywać kilka równoległych zadań. Wykorzystywany jest także do analizy danych, uczenia maszynowego oraz edycji wideo. Procesor graficzny posiada większą przepustowość oraz mniejsze opóźnienie dzięki równoległym procesom. Głównym powodem wykorzystania w projekcie GPU była możliwość szybszego uczenia sieci neuronowej.

TPU jest niestandardową jednostką służącą do przyspieszenia uczenia głębokich sieci neuronowych i skomplikowanych obliczeń macierzowych. Została opracowana przez firmę Google oraz jest wykorzystywana w środowisku programistycznym Tensorflow. Jednostka przetwarzająca tensory została zaprojektowana z myślą o wysokiej wydajności i elastyczności. Główną przewagą TPU nad innymi jednostkami obliczeniowymi jest znacząco skrócony czas potrzebny na trenowanie sieci.

3.3 Metody rozpoznawania twarzy

Rozpoznawanie twarzy jest techniką polegającą na wykrywaniu twarzy osób, których zdjęcia zostały zapisane w danej bazie. Główną zaletą tej metody identyfikacji tożsamości jest to, że wykorzystuje cechy fizjologiczne. Głównym problemem algorytmów rozpoznawania twarzy jest ich wydajność.

Wśród najbardziej popularnych metod rozpoznawania twarzy można wyróżnić algorytmy wykorzystujące geometrię bądź szablony. Opierają się one na cechach geometrycznych poprzez analizowanie charakterystycznych cech twarzy i ich relacji między sobą [10]. Ideą tego podejścia jest uchwycenie względnego położenia i wielkości charakterystycznych cech twarzy, takich jak oczy, usta, nos. W celu określenia podobieństwa pomiędzy twarzami

wyznaczany jest wynik korelacji pomiędzy wzorcem a nowym obrazem. Modele oparte na szablonach mogą być tworzone za pomocą takich algorytmów jak SVM (z ang. *Support Vector Machines*), PCS (z ang. *Linear Discriminant Analysis*) lub LDA (z ang. *Linear Discriminant Analysis*). Główną wadą tej metody jest niska skuteczność podczas zmiany oświetlenia oraz innych czynników zewnętrznych.

Drugim algorytm umożliwiający rozpoznawanie twarzy wykorzystuje statystyczną analizę głównych składowych [10]. Technika ta polega na wykorzystaniu równań matematycznych, które redukują liczbę wymiarów, poprzez wyodrębnienie najważniejszych cech. Głównym typem, spośród wszystkich metod wykorzystujących statystyczną analizę, jest analiza podprzestrzeni. Ideą tej metody jest kompresja zdjęcia twarzy na obraz o mniejszym wymiarze poprzez wykorzystanie transformacji liniowej lub nieliniowej. Wśród najbardziej popularnych algorytmów można wyróżnić liniową analizę dyskryminacyjną (z ang. *Linear Discriminant Analysis, LDA*) oraz analizę składowych głównych (z ang. *Independent Component Analysis, ICA*).

Kolejną metodą umożliwiającą rozpoznanie twarzy są sztuczne sieci neuronowe. Neurony w sztucznych sieciach neuronowych pełnią funkcję grupy, z których każda jest odpowiedzialna za określone zadanie. Neurony są ze sobą połączone poprzez linie z odpowiednimi wagami. Jako dane wejściowe przyjmują dane przetworzone przez poprzedni neuron. Nich zaletą jest możliwość przechowywania dużej ilośći rozproszonych danych.

3.4 Platformy programistyczne wykorzystujące rozpoznawanie twarzy

InsightFace jest platformą programistyczną opartą o język programowania Python. Platforma wykorzystuje metodę RetinaFace, która służy do wykrywania twarzy oraz SubCenter-ArcFace, służącą do rozpoznawania twarzy. InsightFace jest w stanie uzyskać dokładność na poziomie 99,86% dla zbioru danych LFW [6]. Główną wadą tej platformy jest skomplikowane użytkowanie [14].

CompreFace jest jedną z bardziej intuicyjnych platform służących do rozpoznawania twarzy. Posiada prosty interfejs do zarządzania poleceniami i zbiorami danych. Umożliwia korzystanie spośród dwóch metod rozpoznawania twarzy, FaceNet oraz InsightFace. Posiada interfejs REST API, który umożliwia łatwą integrację z systemem bez znajomości uczenia maszynowego.

FaceNet jest platformą programistyczną wykorzystuje głęboką sieć neuronową do rozpoznawania twarzy. W celu wydobycia najważniejszych cech, obraz jest kompresowany do wektora posiadającego 128 liczb. Podobieństwo jest wyznaczane poprzez porównywanie wartości wektorów dla obu zdjęć.

Rozdział 4

Tworzenie sieci neuronowej

W celu stworzenia sieci neuronowej umożliwiającej określenie podobieństwa pomiędzy dwoma twarzami wykorzystano konwolucyjną sieć neuronową. Została ona wybrana ze względu na wysoką precyzję przy rozwiązywaniu problemów związanych z klasyfikacją obrazów. Konwolucja jest operacją matematyczną, która pozwala wydobyć z obrazu odpowiednie cechy. W tym celu wykorzystywane są przekształcenia macierzowe, nazywane kernelami lub filtrami. Zastosowanie filtru umożliwia wydobycie z zdjęcia nowej charakterystyczne cechy, która zostanie wykorzystana w trakcie nauki sieci. Wartość dla każdego jądra jest dobierana i optymalizowana w trakcie uczenia się sieci.

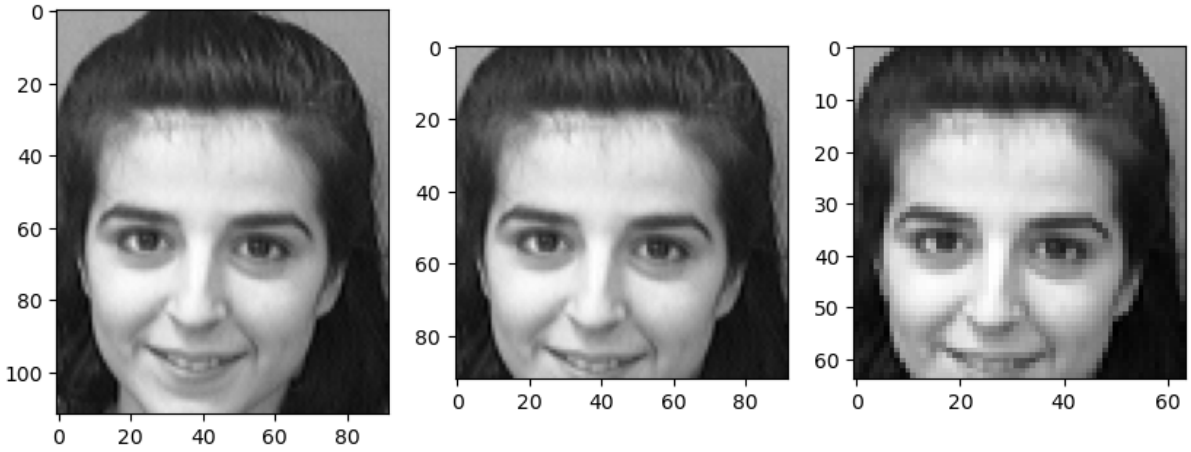
4.1 Wykorzystany zbiór zdjęć

Wśród dostępnych zbiorów, wybrana została baza zdjęć ORL [12]. Jest ona udostępniona przez laboratorium badawcze Olivetti z Wielkiej Brytanii (z ang. *Olivetti Research Laboratory*), czemu zawdzięcza swoją nazwę. Katalog zawiera zdjęcia twarzy, które zostały wykonane 40 różnym osobom, przy czym każda osoba miała zrobione po 10 odmiennych fotografii. Jak można zauważyć na poniższym rysunku 4.1, zdjęcia wykonano w różnym czasie, przy niejednolitym oświetleniu, występują na nich różne mimiki twarzy oraz osoby mają założone lub zdjęte okulary. Wszystkie zdjęcia były zrobione na jednolitym czarnym tle, postacie na środku są wyprostowane, ustawione frontalnie z dopuszczalnym minimalnym obrotem. Wszystkie obrazy są zapisane w 8-bitowych odcieniach szarości oraz posiadają rozmiar 112 na 92 piksele.

Rysunek 4.1 Przykładowe fotografie z bazy ORL.

4.2 Zmiana formatu zdjęć

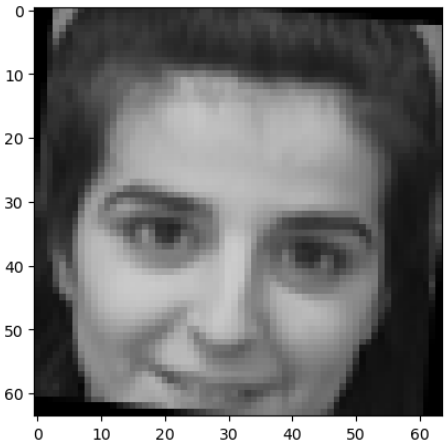
W celu unifikacji bazy została wykonana konwersja formatu zdjęć. Przykład oryginalnego zdjęcia przed zmianami znajduje się po lewej stronie rysunku 4.2. Zbiór danych edytowano poprzez usunięcie wierszy na dole jak i na górze fotografii, dzięki czemu uzyskano próbki o formacie 92 na 92 piksele, rysunek 4.2, środkowe zdjęcie. Bardzo istotnym elementem w tym etapie jest usunięcie nadmiarowych wierszy. W momencie, gdy obraz zostałby poddany samemu przeskalowaniu bez odrzucenia zbędnych linii, wtedy wynikowa próbka byłby zniekształcona poprzez nienaturalne rozszerzenie twarzy. Kolejny etap formatowania zdjęć polegał na przeskalowana zbioru do rozmiaru 64 na 64 piksele w celu zmniejszenia wielkości, rysunek 4.2, zdjęcie po prawej stronie. Obraz o takich wymiarach zawiera wystarczająco dużą ilość informacji potrzebnych do wytrenowania sieci neuronowej. Wykorzystanie obrazów opartych na czworokącie foremnym jest spowodowane możliwością ujęcia nieregularności ludzkiej twarzy.



Rysunek 4.2 Przedstawia zdjęcia o rozmiarze 112x92, 92x92 oraz 64x64 po konwersji .

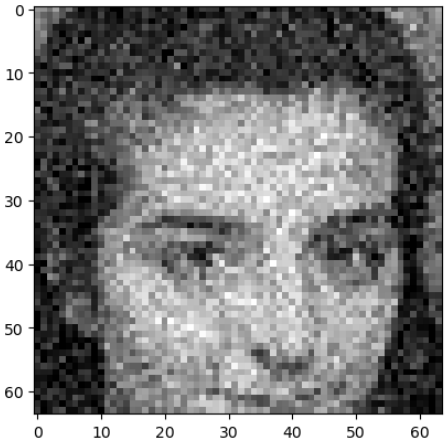
4.3 Rozszerzenie zbioru

Baza ORL posiada w swoim zbiorze 400 zdjęć twarzy, przez co wyuczenie sieci oraz otrzymanie wysokiej precyzji byłoby nie możliwe. W celu zwiększenia ilości próbek zastosowano rozszerzenie zbioru poprzez obrócenie oraz zaszumienie fotografii. Wszystkie zdjęcia zostały poddane obrotowi o kąt 2°, 4° oraz 6° w obu kierunkach, rysunek 4.3, dzięki czemu ilość obrazów na jedną osobę zwiększyła się z 10 do 70. Nowo powstałe próbki, względem oryginalnych są traktowane przez sieć jako odmienne. Jest to spowodowane tym, że algorytm wczytuje zdjęcia wierszami, wprowadzenie rotacji zmienia układ pikseli w macierzy, dzięki czemu można uzyskać większą ilość danych.



Rysunek 4.3 Przedstawia zdjęcie po dodaniu rotacji o kąt 6°.

Drugą wykorzystaną metodą umożliwiającą rozszerzenie zbioru było dodanie zaszumienia. W tym celu do wszystkich zdjęć został dodany szum Gaussa, rysunek 4.4, który wpłynął na wartości poszczególnych pikseli. Dzięki dodaniu zaszumienia liczba dostępnych próbek zwiększyła się z 70 do 140. Tak samo jak poprzednio sieć neuronowa traktuje nowe próbki jako różne względem oryginałów.

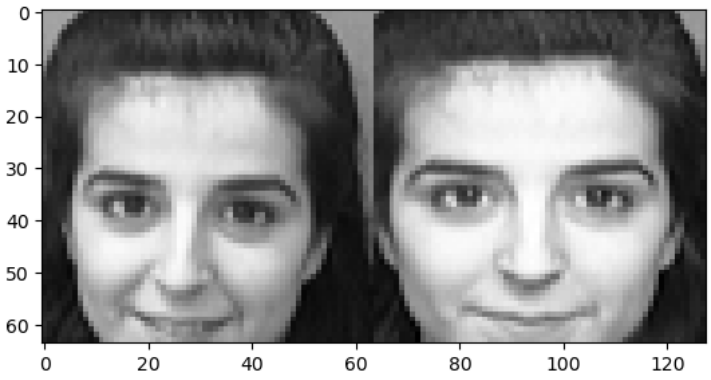


Rysunek 4.4 Przedstawia zdjęcie po dodaniu szumu Gaussa.

Obracanie jak i zaszumienie zdjęć umożliwiło stosunkowo szybkie i łatwe zwiększenie liczby dostępnych obrazów. Dzięki zastosowaniu wyżej wymienionych metod, baza danych zwiększyła wielkość do 5600 zdjęć.

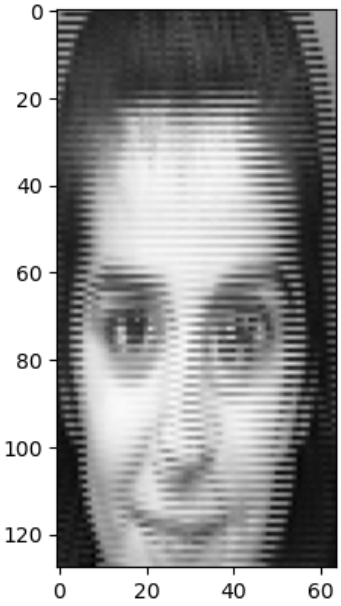
4.4 Reprezentacja danych

Reprezentacja danych była jednym z najbardziej istotnych elementów w trakcie budowania sieci neuronowej. To właśnie ona w głównej mierze wpływała na uzyskaną precyzję oraz jakość sieci w określaniu podobieństwa pomiędzy twarzami. Celem pracy było sprawdzenie wpływu dwóch różnych sposobów reprezentacji danych. Pierwszy sposób polegał na połączeniu ze sobą wierszy dwóch obrazów. Jak można zauważyć na poniższym rysunku 4.5, dołączenie do końca jednego wiersza początku drugiego, powoduje połączenie się z sobą dwóch zdjęć. W ten sposób powstaje próbka o rozmiarze 64 na 128 pikseli.



Rysunek 4.5 Przedstawia złączenie dwóch zdjęć.

Drugi sposób polegał na wzajemnym przeplataniu się wierszy obu fotografii. Jak można zauważyć na poniższym rysunku 4.6, wiersze poszczególnych obrazów zostały poukładane na przemian. W ten sposób otrzymywana jest próbka o rozmiarze 128 na 64 piksele.



Rysunek 4.6 Przedstawia przeplatanie wierszy dwóch zdjęć.

4.5 Tworzenie zbiorów

Kolejny etap tworzenia sieci polegał na odpowiednim połączeniu ze sobą zdjęć. Próbki zostały podzielone na dwa typy, pozytywne oraz negatywne. Pozytywne próbki przedstawiały połączenie dwóch zdjęć tej samej osoby. Próbki negatywne posiadały połączone zdjęcia dwóch różnych osób. W celu dokładnego wyuczenia sieci należało zachować równowagę w ilości próbek pozytywnych jak i negatywnych. Dodatkowym elementem występującym w trakcie tworzenia sieci była etykieta. Przyjmuje ona dwie wartości, 0 oznacza próbkę negatywną, która zawiera fotografie innych postaci. Natomiast 1 oznacza próbkę pozytywną, zawierającą zdjęcia tych samych osób.

W celu poprawnego wyuczenia sieci i sprawdzenia jej jakości należało podzielić cały zbiór na trzy podzbiory. Wśród nich można wyróżnić zbiór treningowy, który służy do

nauki sieci i zawiera 80% zawartości całego zbioru. Drugim zbiorem jest zbiór walidacyjny, który służy do sprawdzania poprawności nauczania sieci. Składa się on z 10% całego zbioru. Trzecim podzbiorem jest zbiór testowy, który służy do określenia jakości sieci po skończeniu uczenia. Zawiera on 10% całego zbioru danych. Dodatkowo wszystkie osoby występujące w zbiorze treningowym, walidacyjnym i testowym się nie powtarzają, dzięki czemu zmniejszone jest ryzyko przeuczenia modelu.

4.6 Architektura sieci

Konwolucyjna sieć neuronowa składa się z wielu warstw. Pierwszym elementem sieci jest warstwa wejściowa (z ang. *input layer*), która służy do reprezentacji obrazu wejściowego. Podawane są w niej wymiary zdjęć oraz liczba kanałów odcieni. W przypadku zdjęć czarno–białych wykorzystywany był tylko jeden kanał [15].

Kolejnym elementem jest warstwa konwolucyjna zwana także splotową. Jest ona główną częścią sieci, ponieważ posiada odpowiednio wyuczone jądra, które umożliwiają wydobycie cech ze zdjęcia. W trakcie tworzenia splotu należy podać odpowiednie parametry. Pierwszym z nich jest liczba filtrów w splocie. Drugim parametrem jest rozmiar jądra (z ang. *kernel size*), który określa wielkość macierzy filtru. Wybranie optymalnego rozmiaru jądra ma istotny wpływ na jakość sieci. Przykładowo mały rozmiar jądra wpływa na zwiększenie wielkości warstw, dzięki czemu można stworzyć głębszą architekturę. Kolejnym parametrem jest wypełnienie (z ang. *padding*), który określa sposób obramowania zdjęcia. Celem wypełnienia jest dodanie dodatkowych wag na brzegach obrazu dzięki czemu zwiększa się rozmiar zdjęcia, aby finalnie po operacji splotu odpowiadał rozmiarowi obrazu wejściowego. Kolejnym parametrem są kroki (z ang. *strides*), które określają o jaką liczbę pikseli zostanie przesunięte okno filtru. Wpływa to na sposób wyszukiwania cech i wielkość kolejnych warstw. Jedną z bardziej istotnych cech sieci konwolucyjnych, która wpływa na wysoką dokładność, jest ich nieliniowość. Jest ona niezbędna ze względu na możliwość stworzenia nieliniowych granic decyzyjnych. W tym celu wykorzystuje się funkcję aktywacji. Rektyfikowana aktywacja liniowa (z ang. *rectified linear activation, ReLU*), przedstawiona wzorem

$$ReLU(x) = max(0, x), \tag{4.1}$$

gdzie x oznacza wartość wejściową. Metoda ta jest wykorzystywana w celu zastosowania nieliniowości. Funkcja ReLU zapobiega wykładniczemu wzrostowi w trakcie przeprowadzania obliczeń, które były wykorzystywane dla prawidłowego działania sieci. Zastosowanie funkcji ReLU wyeliminowało możliwość zbliżenia się gradientu neuronu do zera (z ang. *vanishing gradients*). Kolejną funkcją aktywacyjną jest operacja miękkiego maksimum (z ang. *soft max*) przedstawiona jako

$$SoftMax(x_i) = \frac{e^{x_i}}{\Sigma_j e^{x_j}}, \tag{4.2}$$

gdzie x_i oznacza wektor wejściowy, e^{x_i} określa standardową funkcję wykładniczą oraz wyrażenie $\Sigma_j e^{x_j}$ przedstawia funkcję normalizacyjną. Operacja miękkiego maksimum służy do regulowania wyników modelu przy określaniu prawdopodobieństwa.

Piksele położone blisko siebie zazwyczaj posiadają zbliżone wartości, przez co warstwy konwolucyjne generują bliźniacze rezultaty na wyjściu. W wyniku takiego zjawiska powstaje duża ilość zbędnych informacji. Dla wyeliminowania tego typu zjawiska stosuje się łączenie warstw (z ang. *pooling layers*). Głównym celem wszystkich metod łączenia

warstw jest zmniejszenie wielkości obrazu, co przyczynia się do redukcji parametrów do nauki sieci. Metoda łączenia warstw polega na wycięciu z obrazu macierzy o zadanym rozmiarze i rzutowaniu jej wartości na jeden piksel. Można wyróżnić dwa sposoby łączenia warstw. Pierwszy polega na wybraniu maksymalnej wartości z macierzy, natomiast drugi opera się na obliczaniu średniej wartości wśród wszystkich elementów w macierzy.

Celem trenowania sieci neuronowej jest określenie relacji pomiędzy cechami obiektów a dostarczonymi etykietami. Zjawisko przetrenowania polega na wyuczeniu się zbioru danych co prowadzi do zmniejszenia jakości sieci. W celu wyeliminowania tego zjawiska można wykorzystać warstwę porzucenia (z ang. *dropout layer*). Wykorzystuje ona losowe zerowanie neuronów budujących ukryte warstwy. Metoda ta jest skuteczna, ponieważ uniemożliwia sieci wyuczenie się samego zbioru danych bez uwzględnienia relacji pomiędzy cechami. Wśród dostępnych metod eliminujących zjawisko przetrenowania można także wyróżnić regularyzację wag (z ang. *regularization*), normalizację wsadową (z ang. *batch normalization*) oraz metodę wczesnego zakończenia (z ang. *early stopping*). Normalizacja wsadowa polega na normalizacji danych wejściowych poprzez przeskalowanie i wycentrowanie warstw. Metoda wczesnego zakończenia polega na wstrzymaniu uczenia sieci w wyznaczonym momencie. Bardzo często wykorzystuje się ją do zatrzymywania uczenia, gdy sieć przestaje się uczyć i wartości strat dla próbek walidacyjnych przestają maleć.

Kolejną wykorzystaną warstwą jest warstwa spłaszczająca (z ang. *flatten layer*). Jej głównym zadaniem jest zmiana wielowymiarowej warstwy sieci na wektor. Operację spłaszczania umożliwia dalsze sklasyfikowanie cech. Ostatnią warstwą jest warstwa gęsta (z ang. *Dense*), która łączy ze sobą wszystkie poprzednie warstwy.

Jednym z bardziej istotnych elementów w trakcie tworzenia sieci była optymalizacja procesu uczenia. W tym celu wykorzystano metodę wczesnego zakończenia, która automatycznie przerywała uczenie w momencie, gdy sieć przestawała się uczyć. Głównym wyznacznikiem, który był brany pod uwagę była wartość strat dla próbek walidacyjnych. Drugą wykorzystaną funkcją był *ModelCheckpoint()*, który tak samo jak metoda wczesnego zatrzymania wyszukuje epokę z najmniejszą wartością strat dla zbioru walidacyjnego po czym zapisuje najlepszy model. Podczas nauki samego modelu należało ustawić kilka ważnych parametrów. Wśród nich można wyróżnić epoki (z ang. *epoch*), które określały maksymalną liczbę powtórzeń. Drugim parametrem był rozmiar partii (z ang. *batch size*), który określa liczbę rekordów przetwarzanych przed nastąpieniem aktualizacji.

W celu zbadania wpływu reprezentacji danych na jakość klasyfikacji obrazów zostały stworzone dwie sieci neuronowe. Architektury obu modeli były do siebie zbliżone, jedyną różnicą była wartość parametrów dla warstwy wejściowej. Jest to spowodowane innym rozmiarem danych wejściowych. Poniższa tabela 4.1 przedstawia parametry opisujące architekturę modelu sieci dla pierwszej reprezentacji, która opiera się na złączeniu ze sobą wierszy dwóch obrazów. Z kolei w tabeli 4.2 przedstawiono liczbę parametrów, które podlegają zmianie w procesie nauki dla każdej z warstw.

Rodzaj warstwy	Liczba filtów	Rozmiar jądra	Rozmiar łączenia	Aktywacja
Conv2D	32	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	-
Conv2D	32	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	-
BatchNormalization	-	-	-	-
Conv2D	64	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	
Conv2D	64	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	
BatchNormalization	-	-	-	-
Flatten	-	-	-	-
Dense	64	-	-	ReLU
Dense	1	-	-	ReLU

Tabela 4.1 Przedstawia architekturę sieci dla złączonych zdjęć.

Rodzaj warstwy (typ)	Rozmiar wyjściowy	Parametry
Conv2D	(Brak, 62, 126, 32)	320
MaxPooling2D	(Brak, 31, 63, 32)	0
Conv2D	(Brak, 29, 61, 32)	9248
MaxPooling2D	(Brak, 14, 30, 32)	0
BatchNormalization	(Brak, 14, 30, 32)	128
Conv2D	(Brak, 12, 28, 64)	18496
MaxPooling2D	(Brak, 6, 14, 64)	0
Conv2D	(Brak, 4, 12, 64)	36928
MaxPooling2D	(Brak, 2, 6, 64)	0
BatchNormalization	(Brak, 2, 6, 64)	256
Flatten	(Brak, 768)	0
Dense	(Brak, 64)	49216
Dense	(Brak, 1)	65

Całkowita liczba parametrów: 114 465
Trenowane parametry: 114 657
Nie trenowane parametry: 192

Tabela 4.2 Przedstawia architekturę sieci dla złączonych zdjęć.

Poniższa tabela 4.3 przedstawia parametry opisujące architekturę modelu sieci dla drugiej reprezentacji, która opiera się na przeplataniu ze sobą wierszy dwóch obrazów. Z kolei w tabeli 4.4 przedstawiono liczbę parametrów, które podlegają zmianie w procesie nauki dla każdej z warstw.

Rodzaj warstwy	Liczba filtów	Rozmiar jądra	Rozmiar łączenia	Aktywacja
Conv2D	32	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	-
Conv2D	32	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	-
BatchNormalization	-	-	-	-
Conv2D	64	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	
Conv2D	64	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	
BatchNormalization	-	-	-	-
Flatten	-	-	-	-
Dense	64	-	-	ReLU
Dense	1	-	-	ReLU

Tabela 4.3 Przedstawia architekturę sieci dla przeplatanych zdjęć.

Rodzaj warstwy (typ)	Rozmiar wyjściowy	Parametry
Conv2D	(Brak, 126, 62, 32)	320
MaxPooling2D	(Brak, 63, 31, 32)	0
Conv2D	(Brak, 61, 29, 32)	9248
MaxPooling2D	(Brak, 30, 14, 32)	0
BatchNormalization	(Brak, 30, 14, 32)	128
Conv2D	(Brak, 28, 12, 64)	18496
MaxPooling2D	(Brak, 14, 6, 64)	0
Conv2D	(Brak, 12, 4, 64)	36928
MaxPooling2D	(Brak, 6, 2, 64)	0
BatchNormalization	(Brak, 6, 2, 64)	256
Flatten	(Brak, 768)	0
Dense	(Brak, 64)	49216
Dense	(Brak, 1)	65

Całkowita liczba parametrów: 114 465 Trenowane parametry: 114 657 Nie trenowane parametry: 192

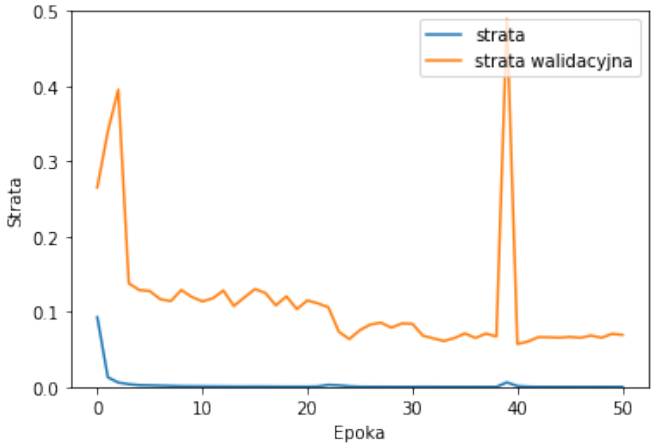
Tabela 4.4 Przedstawia architekturę sieci dla przeplatanych zdjęć.

4.7 Uzyskane rezultaty

Po wytrenowaniu obu modeli przystąpiono do sprawdzenia uzyskanych wyników. W tym celu utworzono wykresy przedstawiające wartość straty oraz wartość precyzji dla danej epoki. Następnie zostały porównane ze sobą oba rezultaty.

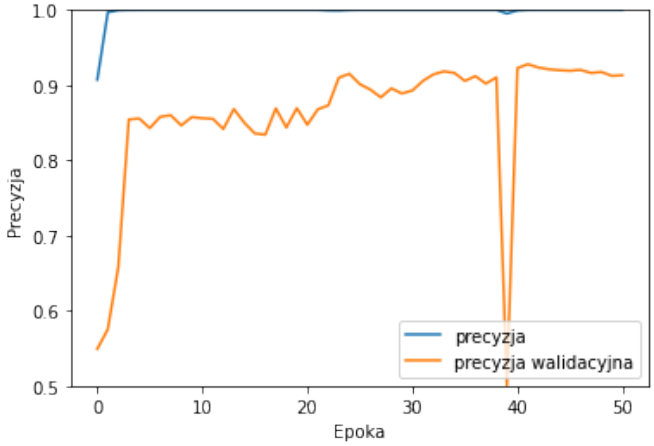
4.7.1 Łączenie wierszy zdjęć

Jako pierwsza sprawdzeniu poddana została sieć zawierająca złączone zdjęcia. Na poniższym rysunku 4.7 został przedstawiony wykres zawierający dwa przebiegi. Pierwszy z nich, zaznaczony jako strata, określa wartość funkcji straty dla zbioru uczącego, natomiast drugi oznaczony jako strata walidacyjna określa wartość funkcji straty dla zbioru walidacyjnego, który służy do sprawdzenia poprawności nauki sieci.



Rysunek 4.7 Wykres funkcji strat dla złączonych zdjęć.

Następnie stworzono drugi wykres, rysunek 4.8, który zawiera funkcję precyzji sieci. Pierwszy przebieg zaznaczony jako precyzja określa wartość funkcji precyzji modelu dla zbioru treningowego. Drugi przebieg oznaczony jako precyzja walidacyjna określa funkcję precyzji uzyskaną dla zbioru walidacyjnego.



Rysunek 4.8 Wykres funkcji precyzji dla złączonych zdjęć.

Po wygenerowaniu wykresów przystąpiono do oceny jakości sieci. W tym celu wykorzystano funkcję *model.evaluate()*, która umożliwiła ewaluację stworzonego modelu za pomocą zbioru testowego. W ten sposób otrzymano wartość precyzji równą 0,8329 oraz wartość straty wynoszącą 0,1278.

Kolejny etap sprawdzania jakości sieci polegał na wykorzystaniu macierzy pomyłek. W tym celu wykorzystano funkcję *probability_ model()*, która dla zbioru testowego zwróciła etykiety wraz z wartością precyzji wyboru. Przyjęto, że dla wartości powyżej lub równej 0,5 osoby na zdjęciach są tej samej tożsamości, natomiast poniżej 0,5 osoby nie są podobne. Dzięki temu warunkowi można było określić czy dane zdjęcia należały do tej samej osoby lub nie. Otrzymane wyniki wraz z prawidłowymi etykietami przedstawiono w formie macierzy pomyłek

$$\begin{bmatrix} T_n & N_n \\ N_p & T_p \end{bmatrix} = \begin{bmatrix} 853 & 407 \\ 14 & 1246 \end{bmatrix}, \tag{4.3}$$

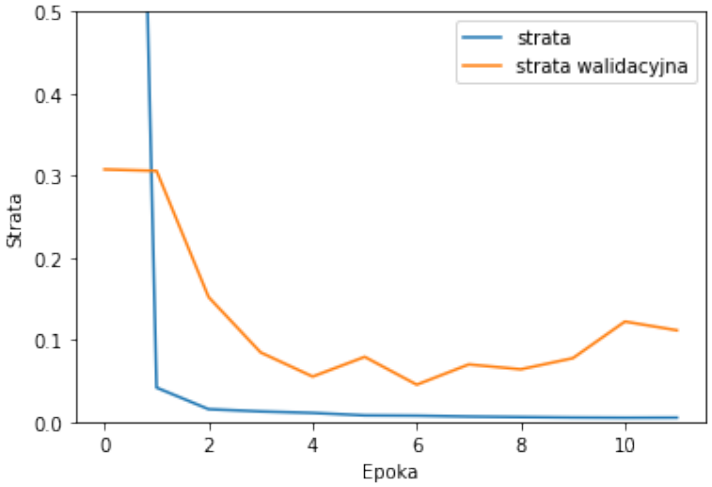
gdzie:

- T_n (z ang. *True negative*) oznacza próbki negatywne, które zostały poprawnie ocenione,
- N_n (z ang. *Negative negative*) oznacza próbki negatywne, które zostały błędnie ocenione,
- N_p (z ang. *Negative positive*) oznacza próbki pozytywne, które zostały błędnie ocenione,
- T_p (z ang. *True positive*) oznacza próbki pozytywne, które zostały poprawnie ocenione.

Jak można zauważyć w powyższej macierzy 4.3, wyuczona sieć w dużej części zwraca poprawną etykietę. Największy błąd występuje podczas oceny zdjęć negatywnych. Liczba błędnie ocenionych negatywnych próbek wynosi 419, co stanowi 16,7% całego zbioru testowego. Warto zauważyć, że sieć dla pozostałych 2099 próbek poprawnie oceniła podobieństwo pomiędzy twarzami, co stanowi aż 83,3% zbioru.

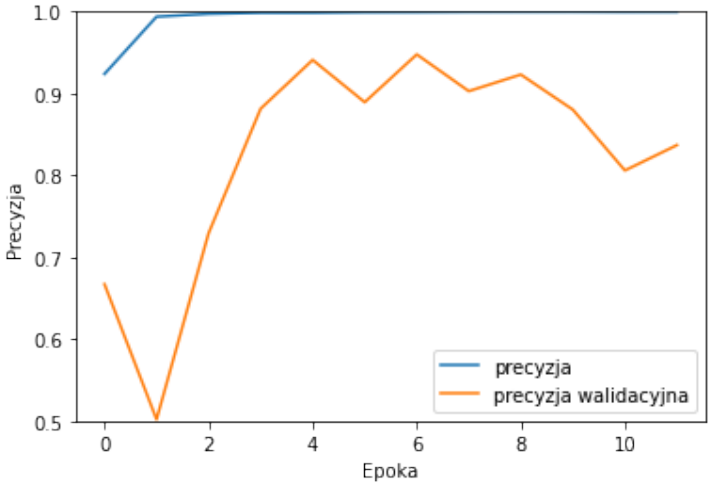
4.7.2 Przeplatanie wierszy zdjęć

Jako druga sprawdzeniu poddana została sieć zawierająca przeplatane wiersze zdjęć. Na poniższym rysunku 4.9 został przedstawiony wykres zawierający dwa przebiegi. Pierwszy z nich, zaznaczony jako strata, określa wartość funkcji strat dla zbioru treningowego, natomiast drugi oznaczony jako strata walidacyjna określa wartość funkcji strat dla zbioru walidacyjnego.



Rysunek 4.9 Wykres wartości funkcji strat dla przeplatanych wierszy zdjęć.

Następnie stworzono drugi wykres, rysunek 4.10 zawierający funkcje precyzji sieci. Pierwszy przebieg oznaczany jako precyzja określa wartość funkcji precyzji sieci dla zbioru treningowego. Drugi przebieg oznaczony jako precyzja walidacyjna określa wartość funkcji precyzji uzyskaną dla zbioru walidacyjnego.



Rysunek 4.10 Wykres wartości funkcji precyzji dla przeplatanych wierszy zdjęć.

Po wygenerowaniu wykresów przystąpiono do oceny jakości modelu. W tym celu wykorzystano funkcję *model.evaluate()*, która umożliwiła ewaluację stworzonego modelu za pomocą zbioru testowego. W ten sposób otrzymano wartość precyzji równą 0,9857 oraz wartość straty wynoszącą 0,0278.

Kolejny etap sprawdzania jakości sieci polegał na wykorzystaniu macierzy pomyłek. W tym celu wykorzystano funkcję *probability_model()*, która dla zbioru testowego zwróciła etykiety z wartością precyzji wyboru. Przyjęto, że dla wartości powyżej lub równej 0,5 osoby na zdjęciach są tej samej tożsamości, natomiast poniżej 0,5 osoby nie są podobne. Następnie otrzymane wyniki wraz z etykietami przedstawiono w formie macierzy pomyłek

$$\begin{bmatrix} T_n & N_n \\ N_p & T_p \end{bmatrix} = \begin{bmatrix} 1251 & 9 \\ 27 & 1233 \end{bmatrix}, \tag{4.4}$$

gdzie wyuczona sieć poprawnie oceniła 2484 próbek co stanowi 98,6% całego zbioru testowego. Warto zwrócić uwagę na błędnie ocenione pozytywne próbki, które stanowią 1,1% próbek testowych. Sieć dla reprezentacji wykorzystującej przeplatanie wierszy bardzo dobrze radzi sobie z określaniem negatywnych próbek. Algorytm łącznie popełnił 9 błędów przy określaniu negatywnych próbek, co stanowi 0,4% zbioru testowego.

4.8 Porównanie rezultatów

W celu porównania wyników uzyskanych dla obu reprezentacji danych należało wybrać odpowiedni parametr, który pozwoliłby na ocenienie jakości obu sieci i porównanie ich między sobą. Jako wskaźnik jakości sieci wykorzystano wartość starty dla zbioru testowego.

Porównanie rezultatów zostało wykonane na zbiorze testowym ze względu na to, że zawiera zdjęcia osób, które nie występowały w zbiorze treningowym i walidacyjnym. Zbiór testowy zawiera łącznie 2520 zdjęć, w tym 1260 próbek pozytywnych przedstawiających te same osoby oraz 1260 próbek negatywnych, które przedstawiały różne postacie. Jako pierwsze zostały porównane wyniki ewaluacji dla zbioru testowego. Reprezentacja składająca się z złączonych zdjęć uzyskała wartość strat równą 0,1278, natomiast wartość precyzji wyniosła 0,8329. W porównaniu wartość strat, dla reprezentacji zawierającej przeplatane wiersze zdjęć wyniosła 0,0278 oraz otrzymano wartość precyzji wynoszącą 0,9857. Uzyskane rezultaty zostały przedstawione w tabeli 4.5.

Rodzaj reprezentacji	Wartość strat	Wartość precyzji
Złączone zdjęcia	0,1278	0,8329
Przeplatane wiersze	0,0278	0,9857

Tabela 4.5 Przedstawia rezultaty ewaluacji obu reprezentacji na zbiorze testowym.

Jak można zauważyć wartość strat, dla reprezentacji zawierającej przeplatane wiersze zdjęć, jest niższa. Podobna zależność występuje dla uzyskanej wartości precyzji. Wynika z tego, że sieć składająca się z przeplatanych wierszy obrazów uzyskała niższą wartość straty oraz wyższą wartość precyzji niż sieć ze złączonymi obrazami.

Drugim parametrem, który został przeanalizowany, była macierz pomyłek. Jak można zauważyć w macierzy (4.3), sieć wykorzystująca złączone zdjęcia poprawnie oceniła 2099 próbek, co stanowi 83,3% całego zbioru testowego. Dodatkowo błędnie zostało ocenione 421 próbek, co stanowi 16,7%. Z macierzy (4.4) wynika, że druga sieć wykorzystująca przeplatanie wierszy zdjęć, poprawnie oceniła 2484 próbek co stanowi 98,6%. Porównując obie macierze pomyłek można wywnioskować, że lepsze rezultaty uzyskano poprzez zastosowanie reprezentacji danych polegającej na przeplataniu wierszy zdjęć.

4.9 Badanie wpływu normalizacji wsadowej na jakość modelu

Kolejny etap badań polegał na sprawdzeniu wpływu zwiększonej liczby warstw normalizacyjnych na jakość oceny sieci. W tym celu zostały stworzone dwa nowe modele dla każdej z reprezentacji, które posiadały dodatkowe dwie warstwy normalizacji wsadowej. W poniższej tabeli 4.6, przedstawiona została nowa architektura sieci z zwiększoną liczbą warstw

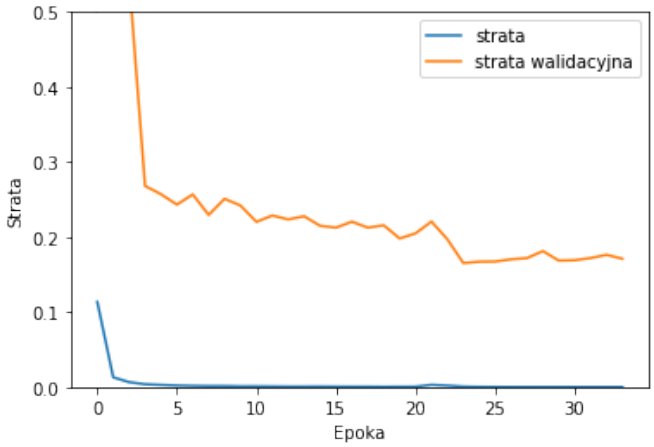
BatchNormalization. Jak można zauważyć, warstwy normalizacji wsadowej występują po każdej warstwie łączeniowej.

Rodzaj warstwy	Liczba filtów	Rozmiar jądra	Rozmiar łączenia	Aktywacja
Conv2D	32	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	-
BatchNormalization	-	-	-	-
Conv2D	32	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	-
BatchNormalization	-	-	-	-
Conv2D	64	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	
BatchNormalization	-	-	-	-
Conv2D	64	3 x 3	-	ReLU
MaxPooling2D	-	-	2 x 2	
BatchNormalization	-	-	-	-
Flatten	-	-	-	-
Dense	64	-	-	ReLU
Dense	1	-	-	ReLU

Tabela 4.6 Przedstawia architekturę sieci dla zwiększonej liczby warstw normalizacji wsadowej.

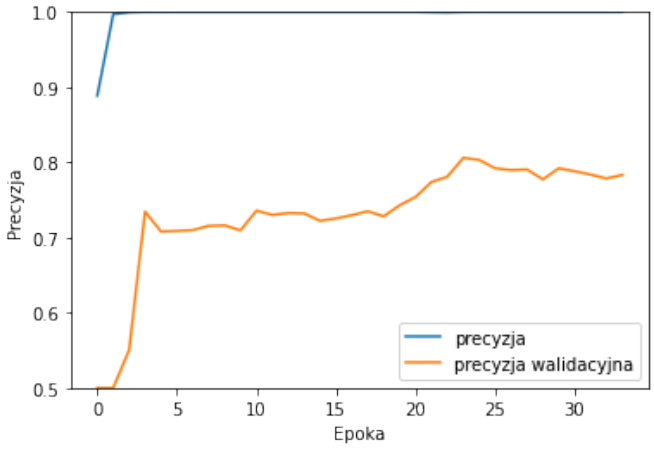
4.9.1 Łączenie wierszy zdjęć

Jako pierwsza sprawdzeniu została poddana sieć zawierająca złączone zdjęcia. Na poniższym rysunku 4.11 został przedstawiony wykres zawierający dwa przebiegi. Pierwszy z nich, zaznaczony jako strata, określa wartość funkcji straty dla zbioru uczącego, natomiast drugi oznaczony jako strata walidacyjna określa wartość funkcji straty dla zbioru walidacyjnego.



Rysunek 4.11 Wykres funkcji straty dla złączonych zdjęć.

Następnie stworzono drugi wykres, rysunek 4.12, zawierający wartość funkcji precyzji modelu. Pierwszy przebieg oznaczany jako precyzja określa wartość funkcji precyzji sieci dla zbioru treningowego. Drugi przebieg zaznaczony jako precyzja walidacyjna określa wartość funkcji precyzji uzyskaną dla zbioru walidacyjnego.



Rysunek 4.12 Wykres funkcji precyzji dla złączonych zdjęć.

Po wygenerowaniu wykresów przystąpiono do oceny jakości sieci. W tym celu wykonano ewaluację otrzymanego modelu na zbiorze testowym. W ten sposób otrzymano wartość precyzji równą 0,7127 oraz wartość straty wynoszącą 0,2398.

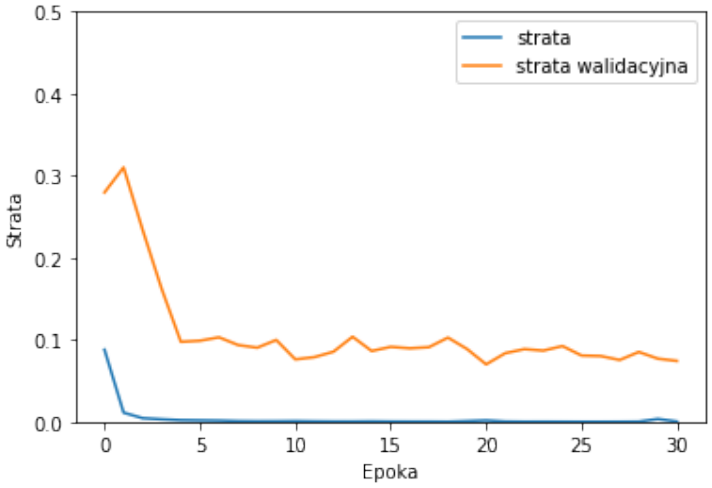
Kolejny etap sprawdzania jakości sieci polegał na wykorzystaniu macierzy pomyłek. W tym celu wykorzystano funkcję *probability_model()*, która dla zbioru testowego zwróciła etykiety z wartością precyzji wyboru. Przyjęto, że dla wartości powyżej lub równej 0,5 osoby na zdjęciach są tej samej tożsamości, natomiast poniżej 0,5 osoby nie są podobne. Następnie otrzymane wyniki wraz z etykietami przedstawiono w formie macierzy pomyłek

$$\begin{bmatrix} T_n & N_n \\ N_p & T_p \end{bmatrix} = \begin{bmatrix} 605 & 655 \\ 69 & 1191 \end{bmatrix}, \tag{4.5}$$

gdzie wyuczona sieć poprawnie oceniła 1796 próbek co stanowi 71,3% całego zbioru testowego. Warto zwrócić uwagę na błędnie ocenione próbki, które stanowią 28,7% próbek testowych. Sieć dla reprezentacji wykorzystującej złączone zdjęcia bardzo dobrze radzi sobie z określaniem pozytywnych próbek. Algorytm łącznie popełnił 69 błędów przy określaniu negatywnych próbek, co stanowi 2,7% zbioru testowego.

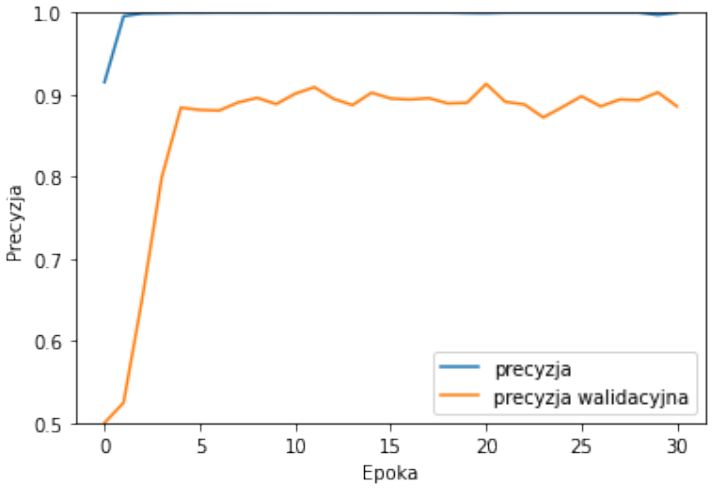
4.9.2 Przeplatanie wierszy zdjęć

Jako druga sprawdzeniu poddana została sieć zawierająca przeplatane wiersze zdjęć. Na rysunku 4.13 został przedstawiony wykres zawierający dwa przebiegi. Pierwszy z nich, zaznaczony jako strata, określa wartość funkcji straty dla zbioru uczącego, natomiast drugi oznaczony jako strata walidacyjna określa wartość funkcji straty dla zbioru walidacyjnego.



Rysunek 4.13 Wykres wartości funkcji straty dla przeplatanych wierszy zdjęć.

Następnie stworzono drugi wykres, rysunek 4.14, zawierający wartość funkcji precyzji sieci. Pierwszy przebieg oznaczany jako precyzja określa wartość funkcji precyzji sieci dla zbioru treningowego. Drugi przebieg oznaczony jako precyzja walidacyjna określa wartość funkcji precyzji uzyskaną dla zbioru walidacyjnego.



Rysunek 4.14 Wykres wartości funkcji precyzji dla przeplatanych wierszy zdjęć.

Po wyrysowaniu wykresów przystąpiono do oceny jakości sieci. W tym celu wykonano ewaluację stworzonego modelu za pomocą zbioru testowego. Otrzymano wartość precyzji równą 0,9587 oraz wartość straty wynoszącą 0,0313.

Uzyskane rezultaty, dla architektury zawierającej zwiększoną liczbę warstw normalizacji wsadowej, zostały przedstawione w tabeli 4.7.

Rodzaj reprezentacji	Wartość strat	Wartość precyzji
Złączone zdjęcia	0,2398	0,7127
Przeplatane wiersze	0,0313	0,9587

Tabela 4.7 Przedstawia rezultaty ewaluacji obu reprezentacji na zbiorze testowym.

Kolejny etap sprawdzania jakości sieci polegał na wykorzystaniu macierzy pomyłek. W tym celu wykorzystano funkcję *probability_ model()*, która dla zbioru testowego zwróciła

etykiety z wartością precyzji wyboru. Przyjęto, że dla wartości powyżej lub równej 0,5 osoby na zdjęciach są tej samej tożsamości, natomiast poniżej 0,5 osoby nie są podobne. Następnie otrzymane wyniki wraz z etykietami przedstawiono w formie macierzy pomyłek.

$$\begin{bmatrix} T_n & N_n \\ N_p & T_p \end{bmatrix} = \begin{bmatrix} 1215 & 45 \\ 22 & 1238 \end{bmatrix}, \tag{4.6}$$

gdzie wyuczona sieć poprawnie oceniła 2453 próbek co stanowi 97,3% całego zbioru testowego. Warto zwrócić uwagę na błędnie ocenione próbki, które stanowią 2,7% próbek testowych.

4.9.3 Porównanie rezultatów dla dwóch różnych architektur sieci

W celu porównania wyników uzyskanych dla obu reprezentacji danych należało wybrać odpowiedni parametr, który pozwoliłby na ocenienie jakości obu sieci i porównanie ich między sobą. Jako wskaźnik jakości sieci wykorzystano wartość starty dla zbioru testowego. Porównanie rezultatów zostało wykonane na zbiorze testowym ze względu na to, że zawiera zdjęcia osób, które nie występowały w zbiorze treningowym i walidacyjnym. Zbiór testowy zawiera łącznie 2520 zdjęć, w tym 1260 próbek pozytywnych przedstawiających te same osoby oraz 1260 próbek negatywnych, które przedstawiały różne postacie.

Porównując uzyskane rezultaty ewaluacji na zbiorze testowym dla pierwszej architektury, tabela 4.5 z otrzymanymi wynikami dla drugiej, tabela 4.7 można wywnioskować, że zwiększenie liczby warstw normalizacji wsadowej, dla przeplatanych zdjęć, pogarsza jakość oceny.

Drugim parametrem, który został przeanalizowany, była macierz pomyłek. Jak można zauważyć w macierzy (4.3), sieć wykorzystująca złączone zdjęcia, dla pierwszej architektury, poprawnie oceniła 2099 próbek, co stanowi 83,3% całego zbioru testowego. Dodatkowo błędnie zostało ocenione 421 próbek, co stanowi 16,7%. Dla porównania, sieć wykorzystująca drugą architekturę dla tej samej reprezentacji poprawnie oceniła 1796 próbek co stanowi 71,3% oraz negatywnie zostało ocenione 724 próbek co stanowi 27,8% całego zbioru testowego. Wyniki dla drugiej architektury zostały przedstawione w macierzy (4.5).

Jak można zauważyć w macierzy (4.4), sieć wykorzystująca przeplatane zdjęcia, dla pierwszej architektury, poprawnie oceniła 2484 próbek, co stanowi 98,6% całego zbioru testowego. Dodatkowo błędnie zostało ocenione 36 próbek, co stanowi 1,4%. Dla porównania, sieć wykorzystująca drugą architekturę dla tej samej reprezentacji poprawnie oceniła 2453 próbek co stanowi 97,3% oraz negatywnie zostało ocenione 67 próbek co stanowi 2,7% całego zbioru testowego. Wyniki dla drugiej architektury zostały przedstawione w macierzy (4.6).

Uwzględniając wyniki uzyskane z ewaluacji zbioru testowego i rezultatów uzyskanych z macierzy pomyłek można wywnioskować, że zwiększenie liczby warstw normalizacyjnych pogarsza jakość sieci.

Rozdział 5

Podsumowanie

Główne założenia, przyjęte w tej pracy, zostały spełnione. W efekcie udało się stworzyć sieć neuronową, która była w stanie rozróżnić tożsamość na podstawie dwóch zdjęć. Model, wykorzystujący reprezentację składającą się z przeplatanych wierszy, uzyskał precyzję o wartości 0,9857. Istotnym elementem w trakcie tworzenia sieci było stworzenie próbek negatywnych jak i pozytywnych, które umożliwiły naukę sieci. W celu stworzenia modelu z wyższą precyzją i niższą stratą, należałoby zwiększyć liczbę próbek poprzez dołączenie nowych zdjęć z innej bazy. Podczas dodawania nowych próbek należy pamiętać o zachowaniu odpowiednich proporcji i jakości obróbki zdjęć podczas rozszerzenia zbioru.

Praca dowiodła, że reprezentacja danych wejściowych ma wpływ na jakość uzyskanych rezultatów. Wśród testowanych dwóch reprezentacji, lepsze rezultaty osiągnięto dla próbek składających się z przeplatanych wierszy zdjęć. Jakość uzyskanej precyzji jest wysoka, uwzględniając, że początkowo baza zawierała czterysta zdjęć twarzy. Istotny wpływ na otrzymane rezultaty miało rozszerzenie zbioru danych, dzięki któremu uzyskano aż 5600 próbek. Następnie zostały stworzone pozytywne jak i negatywne próbki, których kombinacji było łącznie 25200. Dodatkowo przebadano wpływ normalizacji wsadowej na jakość otrzymywanych wyników. Zbyt duża liczba warstw *BatchNormalization* wpłynęła negatywnie na jakość sieci.

Jednym z bardziej istotnych problemów podczas tworzenia sieci neuronowej było wybranie odpowiedniej bazy zdjęć, która posiadała odpowiednio dużą liczbę reprezentatywnych próbek. Wśród dużej ilości zbiorów można znaleźć wiele baz niskiej jakości, które zawierały niereprezentatywne zdjęcia twarzy. Wśród nich, można było zlokalizować zdjęcia o niskiej jakości, których obraz był rozmazany. Drugim ważnym elementem podczas wybierania odpowiedniej bazy zdjęć był jej rozmiar. Większa liczba próbek umożliwiła polepszenie jakości sieci, a co za tym idzie zwiększenie jej precyzji.

Praca mogłaby być dalej rozwijana w kierunku obsłużenia występujących wyjątków. Jednym z nich jest określanie podobieństwa przy zdjęciach bliźniaków jednojajowych. Jest to stosunkowo nietrywialny problem ze względu na dużą liczbę cech wspólnych, które występują pomiędzy takimi osobami. W tym celu należałoby odszukać lub stworzyć nową bazę zdjęć, która zawierałaby fotografie takiego rodzeństwa. Druga droga rozwoju mogłaby polegać na stworzeniu aplikacji, która automatycznie wyszukiwałaby twarze osób. Następnie program określałby ich tożsamość na podstawie już zapisanych wcześniej zdjęć. W tym celu należałoby stworzyć drugą sieć neuronową, która byłaby w stanie wyszukiwać obrys twarzy. Następnie znaleziona twarzy byłaby wycinana i przetwarzana. Dzięki takiemu połączeniu powstałby algorytm, który mógłby wyszukiwać osoby i określać ich tożsamość na podstawie zapisanej bazy zdjęć.

Załącznik A

Do pracy załączono płytę DVD zawierającą w poszczególnych katalogach:

`/W12N_252919_2023_praca_inżynierska.pdf` — wersja cyfrowa pracy,

`/Rozpoznawanie_twarzy.zip` — archiwum zip zawierające skrypty wykorzystane w trakcie projektu wraz z bazą zdjęć.

Literatura

[1] R. M. Cichy, D. Kaiser. Deep neural networks as scientific models. *Trends in cognitive sciences*, 23(4):305–317, 2019.

[2] J. Dhamija, T. Choudhury, P. Kumar, Y. S. Rathore. An advancement towards efficient face recognition using live video feed:" for the future". *2017 3rd international conference on computational intelligence and networks (CINE)*, strony 53–56. IEEE, 2017.

[3] W. Domski. Selected topics in Artificial Intelligence Data augmentation. 2017.

[4] I. Goodfellow, Y. Bengio, A. Courville. *Deep learning*. MIT Press, 2016.

[5] A. Gulli, S. Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[6] A. Jalal, U. Tariq. The LFW-gender dataset. *Asian Conference on Computer Vision*, strony 531–540. Springer, 2017.

[7] H. Kimm, I. Paik, H. Kimm. Performance comparision of tpu, gpu, cpu on google colaboratory over distributed deep learning. *2021 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, strony 312–319. IEEE, 2021.

[8] E. Matthes. *Python crash course: A hands-on, project-based introduction to programming*. No starch press, 2019.

[9] B. Matuszewski. Wizualizacja koncepcji budowy sieci neuronowej. `https://aivision.pl/binarny-classification/`.

[10] D. N. Parmar, B. B. Mehta. Face recognition methods & applications. *International Journal of Computer Applications in Technology*, 4(1):84–86, 2014.

[11] S. Raschka, V. Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[12] F. Samaria, A. Harter. Parameterisation of a stochastic model for human face identification. *Applications of Computer Vision*, strony 138 – 142. Workshop on Applications of Computer Vision - Proceedings, 1995.

[13] R. C. Schank. Where's the ai? *AI magazine*, 12(4):38–38, 1991.

[14] S. I. Serengil, A. Ozpinar. Lightface: A hybrid deep face recognition framework. *Innovations in intelligent systems and applications conference*, strony 1–5, 2020.

[15] B. M. Wilamowski. Neural network architectures and learning algorithms. *Industrial Electronics Magazine*, 3(4):56–63, 2009.