Politechnika Wrocławska
Wydział Elektroniki, Fotoniki i Mikrosystemów

KIERUNEK: Automatyka i Robotyka

PRACA DYPLOMOWA INŻYNIERSKA

Tytuł pracy:
Webowy system wykrywania choroby Alzheimera z wykorzystaniem głębokich sieci neuronowych

AUTOR:
Piotr Kuboń

PROMOTOR:
dr inż. Łukasz Jeleń

WROCŁAW 2023

Spis treści

Spis rysunków

Spis tabel

Skróty

MVC (ang. *Model View Controller*)

Rozdział 1

Wstęp

1.1. Cel

Celem pracy jest stworzenie webowego systemu wykrywania i klasyfikacji choroby Alzheimera na podstawie przesłanych zdjęć rentgenowskich mózgu. W tym celu stworzono serwis internetowy z wykorzystaniem frameworku Flask oraz model sieci neuronowej z wykorzystaniem biblioteki TensorFlow. Przeprowadzono również badania nad wpływem parametrów sieci na dokładność klasyfikacji zdjęć oraz wybrano i zaimportowano model, który osiągnął najlepszy wynik.

1.2. Opis choroby Alzheimera

Choroba Alzheimera dotyka z roku na rok coraz większą liczbę osób, szczególnie narażone są osoby starsze, z ograniczoną aktywnością fizyczną i pamięciową[12].
Z chorobą Alzheimera najczęściej związana jest demencja. Według portalu holsamed.pl "jeśli chorujemy na Alzheimera, to zawsze mamy demencję ..." [4]. Chorobę wywołują odkładające się w mózgu białka o patologicznej strukturze, stopniowo uszkadzające neurony, powodując ich obumieranie, w wyniku czego dochodzi do coraz to większych zaników pamięci[2].
Wyróżnia się trzy stadia rozwoju choroby. Każde kolejne stadium charakteryzuje się coraz większym negatywnym wpływem na organizm chorego, stopniowo uniemożliwiając mu samodzielne funkcjonowanie. Obecnie nie jesteśmy w stanie w pełni wyleczyć choroby Alzheimera, a jedynie leczyć jej skutki i objawy oraz opóźniać jej rozwój przez odpowiednie leki i ćwiczenia[2]. Dlatego tak ważne jest wykrycie choroby na jej wczesnym stadium rozwoju.

1.3. Opis systemu

W celu usprawnienia procesu diagnozowania choroby Alzheimera wykonany został internetowy system wspomagający pracę lekarzy w ocenie stopnia zaawansowania choroby bądź jej braku. Po wgraniu zdjęcia rentgenowskiego mózgu pacjenta, system dokonuje analizy, a następnie informuje lekarza o stopniu rozwoju choroby.
Postawienie diagnozy jest możliwe dzięki zastosowaniu modelu sieci neuronowej, wyuczonej na zbiorze tysięcy zdjęć rentgenowskich mózgu, zarówno osób chorych jak i zdrowych. Zdjęcia należące do zbioru uczącego odpowiednio modyfikowano poprzez przesunięcie, powiększenie oraz rotację w celu powiększenia zbioru danych uczących. Uzyskano w ten sposób znacznie

większy zbiór danych, co przełożyło się na zwiększenie dokładności modelu. Dokonano również badania i porównania różnych architektur sieci neuronowej, badaniu podlegał również dobór parametrów sieci.
Wybrano i wdrożono w system serwisu internetowego model osiągający największą dokładność w klasyfikacji stopnia rozwoju choroby. Dodano również do serwisu logikę obróbki przesyłanego zdjęcia przed przekazaniem go do modelu klasyfikującego w celu zachowania spójności z formatem danych, którymi model uczono i sprawdzano.
W serwisie można przeprowadzić takie czynności jak dodawanie, edytowanie i usuwanie danych pacjenta oraz dostęp do historii wcześniej przeprowadzonych badań. System zapewnia również jednoznaczną identyfikację pacjenta dzięki automatycznie inkrementowanemu numerowi id.
Ze względów bezpieczeństwa, system działa w sieci wewnętrznej. Ma to na celu wykluczenie możliwości połączenia się z nim przez osoby z zewnątrz. Dostęp do aplikacji odbywa się za pomocą przeglądarki internetowej. Dzięki takiemu podejściu, nie wymuszamy na lekarzu posiadania konkretnego systemu operacyjnego, zyskujemy również spójność systemu dla każdego użytkownika oraz bezpieczeństwo przechowywania i składowania danych.

1.4. Przebieg pracy nad systemem

Projekt systemu zakładał następujące etapy:

- Zainstalowanie wymaganego oprogramowania na maszynie wirtualnej
- Przeprowadzenie badań nad modelem sieci w celu osiągnięcia jak największej dokładności
- Stworzenie systemu webowego do zarządzania pacjentami
- Wydzielenie modelu i zastosowanie go w aplikacji

Rozdział 2

Projekt systemu

2.1. Opis wykorzystanych narzędzi

Obecnie mamy wiele języków, w których możemy napisać serwis internetowy, należą do nich między innymi PHP, JavaScript, Java, C# czy Python. Biorąc pod uwagę wszechstronność języka oraz wsparcie dla korzystania z modeli sieci neuronowych, wybrany został język Python[1].
Kolejną istotną decyzją jest wybór zestawu modułów (ang. framework) oraz narzędzi wspomagającego naszą pracę w przyjętym języku programowania, przy pomocy którego zostanie zbudowany serwis. Framework jest to zbiór modułów pomagających w pisaniu serwisu internetowego. W szczególności, automatyzuje on działanie podstawowych funkcjonalności, dzięki czemu nie musimy skupiać się na pisaniu powtarzalnych fragmentów kodu[7].
Aktualnie mamy na rynku wiele frameworków wspierających pisanie kodu w języku Python, należą do nich między innymi Django, Flask, Hug czy CherryPy. Najpopularniejsze z nich to Django, który jest wybierany głównie przy tworzeniu dużych rozwiązań webowych, oraz Flask, wybierany częściej do tworzenia mało skomplikowanych serwisów, w których mamy dużą swobodę wyboru modułów rozszerzających jego funkcjonalności[9]. Kierując się swobodą wyboru oraz wsparciem do tworzenia szablonów Jinja2 wybrany został framework Flask.
Aktualnie jednym z najpopularniejszych rozwiązań w tworzeniu sztucznych sieci neuronowych jest korzystanie z biblioteki TensorFlow. Biblioteka ta jest głównie wybierana ze względu na domyślną konfigurację wielu aspektów uczenia maszynowego[17]. Popularną biblioteką wykorzystywaną do tworzenia i uczenia sieci jest również PyTorch, jednak ze względu na wcześniejsze doświadczenie oraz możliwość wdrożenia wytrenowanego modelu w serwisie internetowym, zdecydowano się na wykorzystaniu biblioteki TensorFlow.
W celu wytrenowania modelu oraz przeprowadzenia badań zdecydowano się na skorzystanie z usługi Google Colaboratory (w skrócie Colab). Oferuje ona między innymi darmowe środowisko wykonawcze, posiadające wydajne procesory graficzne, doskonale sprawdzające się w zastosowaniach uczenia sieci neuronowych. Usługa Google Colab udostępnia nam zasoby za pośrednictwem dynamicznie tworzonych maszyn wirtualnych, które zostają usunięte po upływie określonego czasu pracy lub wyczerpaniu zasobów[3]. W tym czasie możemy przeprowadzić badania, wyuczyć sieć oraz wydzielić i pobrać model, który okaże się najlepszy.

2.2. Wykorzystane wzorce projektowe

2.2.1. MVC (Model-View-Controller)

Mając na uwadze późniejszą możliwość skalowania aplikacji jak również komfort pracy i wprowadzania dodatkowych zmian w projekcie, zdecydowano się na stworzenie serwisu internetowego z wykorzystaniem wzorca MVC (Model – Widok - Kontroler). Wzorzec ten zakłada rozdzielenie logiki aplikacji na trzy niezależna, ale powiązane ze sobą bloki[10].
Blok modelu obejmuje zdefiniowanie modeli reprezentacji danych, które następnie będziemy przekazywać oraz którymi będziemy operować. W tym bloku zawiera się również logika połączenia z bazą danych oraz operacje na danych takie jak zapisz, pobranie, modyfikacja oraz usunięcie danych z bazy. Dzięki wydzieleniu logiki modelu jesteśmy w stanie bez ingerencji w pozostałe bloki zmieniać i dostosowywać logikę oraz kod programu, na przykład poprzez zmianę silnika bazy danych na inny[6].
Istotna jest również możliwość rozbudowy już istniejących modeli o nowe parametry dzięki dziedziczeniu już istniejących modeli. Zachowujemy w ten sposób wsteczną kompatybilność z logiką już występującą w programie, dzięki czemu nie musimy przepisywać całego kodu po wprowadzeniu rozszerzenia.
W serwisie wykorzystano następujące modele danych:

- Model użytkownika
- Model pacjenta
- Model badania

Blok widoku reprezentuje interfejs osoby korzystającej ze strony, zawiera warstwę wizualną oraz zajmuje się przedstawieniem danych przekazanych przez kontroler za pośrednictwem modelu[11].
Dzięki zastosowaniu szablonów Jinja2, jesteśmy w stanie dynamicznie zmieniać zawartość strony w zależności od przekazanych danych. Oprócz zmiany poszczególnych wartości na stronie możliwe jest również dziedziczenie szablonów oraz importowanie ich, dzięki czemu jesteśmy w stanie wydzielić widok niektórych elementów do osobnych plików oraz importować ich zawartość wszędzie tam, gdzie jest to konieczne[8].
Zyskujemy w ten sposób możliwość dynamicznych zmian widoku strony, jak również możliwość wydzielenia powtarzalnej logiki w jedno miejsce, gdzie w razie konieczności należy wprowadzić poprawki w kodzie, aby były widoczne we wszystkich widokach na stronie.
W celu prezencji danych użytkownikowi oraz udostępnienia mu możliwości wprowadzenia własnych danych, utworzono następujące widoki:

- widok logowania się użytkownika do serwisu
- widok strony głównej serwisu
- widok listy pacjentów z dodatkowymi funkcjonalnościami
- widok edycji danych pacjenta
- widok historii przeprowadzonych badań na pacjencie
- widok panelu do przeprowadzenia badania

Wydzielono również powtarzające się elementów do następujących pod widoki:

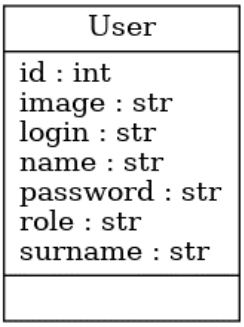
- sekcja menu
- blok pacjenta w liście

Ostatnim i zarazem najważniejszym blokiem modelu MVC jest kontroler. Blok ten odpowiada za przetwarzanie danych modelu i przekazywanie ich do odpowiedniego widoku na podstawie przyjętej logiki aplikacji. Zajmuje się on również przetwarzaniem danych otrzymanych od użytkownika za pośrednictwem widoku[15].
W zależności od stopnia rozbudowania serwisu możliwe jest korzystanie z wielu kontrolerów, odpowiadających na żądania użytkownika wysłane na konkretny adres URL.
W skład bloku kontrolera w serwisie wchodzą kontrolery odpowiedzialne za:

- sprawdzenie danych logowania i blokowanie serwisu przed osobami niezalogowanymi
- dedykowany kontroler dla każdej z operacji dodania, pobrania, edycji i usunięcia pacjenta
- prezentację listy pacjentów
- prezentację listy przeprowadzonych badań na pacjencie
- badanie stopnia zaawansowania choroby na podstawie przesłanego zdjęcia skanu mózgu
- wylogowanie się użytkownika z serwisu

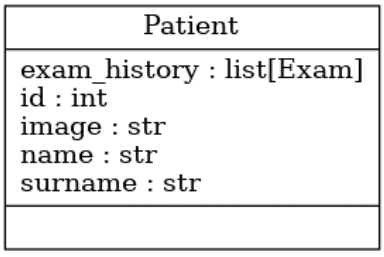
2.3. Modele danych

W projekcie do komunikacji pomiędzy kontrolerem i widokiem wykorzystano model użytkownika, pacjenta oraz badania.
Model użytkownika posiadała następujące pola:



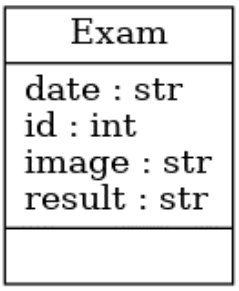
Rys. 2.1: Model użytkownika

Pole id jest typu liczby całkowitej (int). Każdemu użytkownikowi zostaje przydzielony unikalny numer id, w celu jednoznacznej identyfikacji modelu w bazie. Pozostałe pola są typu ciągu znaków (string). Pole name reprezentuje imię użytkownika, surname jego nazwisko, natomiast image stanowi ścieżkę do zdjęcia zapisanego na serwerze. Pola login oraz password są danymi użytkownika którymi loguje się on do serwisu, natomiast role zawiera informację, o funkcji którą pełni w organizacji.
Model pacjenta posiadała następujące pola:



Rys. 2.2: Model pacjęta

Pole id jest typu liczby całkowitej. Każdemu pacjentowi zostaje przydzielony unikalny numer id, w celu jednoznacznej identyfikacji modelu w bazie. Pole exam_history jest listą modeli typu Exam. Pozostałe pola są typu ciągu znaków (string). Pole name reprezentuje imię pacjenta, surname jego nazwisko, natomiast image stanowi ścieżkę do zdjęcia pacjenta zapisanego na serwerze.
Model badania posiadała następujące pola:



Rys. 2.3: Model badania

Pole id jest typu liczby całkowitej. Każde badanie posiada unikalny numer id, w celu jednoznacznej identyfikacji modelu w bazie. Pozostałe pola są typu ciągu znaków (string). Pole result reprezentuje wynik przeprowadzonego badania, pole date zawiera informacje o dacie przeprowadzonego badania, natomiast image stanowi ścieżkę do zdjęcia rendgenowskiego skanu mózgu zapisanego na serwerze.

Rozdział 3

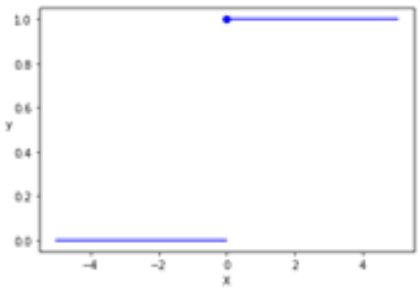
Sztuczne sieci neuronowe

Sztuczne sieci neuronowe stanowią obecnie jeden z najpopularniejszych algorytmów uczenia maszynowego[16]. Wykorzystują one algorytm propagacji wstecznej w celu ustalenia wartości wag połączeń pomiędzy neuronami. Wykorzystują w tym celu zbiór danych uczących oraz testowych, dzięki czemu są w stanie osiągać dużą dokładność np. klasyfikacji[18]. Za wybranie odpowiedniej architektury sieci oraz dobór parametrów takich jak funkcja aktywacji, rozmiar filtrów czy ilość neuronów w warstwach ukrytych odpowiada programista.

3.1. Początki sztucznych sieci neuronowych

Sztuczne sieci neuronowe, podobnie jak inne wielkie odkrycia, były początkowo inspirowane naturą, a w szczególności biologicznymi sieciami neuronowymi. Ich wynalezienie przypisuje się Warrenowi McCullochowi i Walterowi Pittsowi, którzy w 1943 roku w swoim artykule "A Logical Calculus of Ideas Immanent in Nervous Activity"[20] opisali uproszczony model działania zespołów neuronów jak również opisali architekturę pierwszej sztucznej sieci neuronowej. Zaproponowany przez nich model posiadał co najmniej jedno wejście binarne oraz jedno wyjście binarne. Model ten służył głównie do rozwiązywania zadań logicznych.
W 1958 roku Frank Rosenblatt zaproponował model perceptronu[13]. W odróżnieniu od wcześniej zaproponowanego modelu przyjmował on na wejście zbiór liczb oraz odpowiednio przyporządkowanych wag. Na wyjściu natomiast zwracał on wartość wyniku funkcji skokowej dla sumy ważonej liczb podanych na wejście wraz z wagami.
Najpopularniejsze funkcje skokowe to:

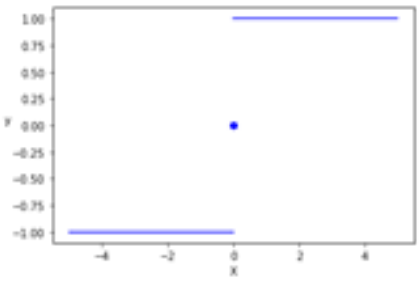
- Funkcja skokowa Heaviside'a:



Rys. 3.1: Funkcja skokowa Heaviside'a

$$y(x) = \begin{cases} 0 & jeśli\ x < 0 \\ 1 & jeśli\ x \geq 0 \end{cases} \tag{3.1}$$

- Funkcja signum:



Rys. 3.2: Funkcja signum

$$y(x) = \begin{cases} -1 & jeśli\ x < 0 \\ 0 & jeśli\ x = 0 \\ 1 & jeśli\ x > 0 \end{cases} \tag{3.2}$$

Pojedynczy perceptron pełni funkcję klasyfikatora binarnego, oblicza on liniową kombinację danych wejściowych i jeżeli uzyskany wynik przekracza wartość progu, zwracany jest wynik klasyfikacji do klasy pierwszej, w przeciwnym przypadku, zwracany jest wynik klasyfikacji do klasy drugiej.
Niestety pojedynczy perceptron nie jest w stanie rozwiązać bardziej skomplikowanych zadań, w szczególności zadania klasyfikacji XOR, co wykazali w 1969 roku Minsky i Papert[21].

3.2. Głębokie sieci neuronowe

Ograniczenia pojedynczego perceptronu są jednak skutecznie niwelowane, przez łączenie wielu perceptronów w warstwy oraz łączenie kolejnych warstw ze sobą. Stworzony w ten sposób model nazywamy najczęściej perceptronem wielowarstwowym[14]. Model ten w szczególności posiada jedną warstwę wejściową, co najmniej jedną warstwę ukrytą oraz jedną warstwę wyjściową. Sieć neuronową nazywamy siecią głęboką, jeżeli posiada wiele warstw ukrytych. Dzięki zastosowaniu dwóch warstw w pełni połączonych jesteśmy w stanie rozwiązać wcześniej wspomniany problem klasyfikacji XOR. Proces uczenia sieci głębokich jest bardzo trudny, dopiero w 1986 roku Rumelhart, Hinton oraz Williams opublikowali algorytm wstecznej propagacji błędów[22]. W wyniku zastosowania wcześniej wspomnianego algorytmu, sieć neuronowa jest w stanie obliczyć różnicę wartości wyjścia oraz oczekiwanej wartości na wyjściu, dzięki czemu jest w stanie wprowadzić odpowiednie modyfikacje wag, aby uzyskać oczekiwany wynik. Początkowe wartości wag połączeń są natomiast inicjowane losowymi, niewielkimi wartościami[18].
W celu poprawienia skuteczności działania algorytmu, a w szczególności w celu zmniejszenia częstotliwości pojawiania się zjawiska eksplodujących i zanikających gradientów wprowadzono dodatkowe funkcje aktywacji, między innymi:

- Funkcję sigmoidalną
- Funkcję tangensa hiperbolicznego
- Funkcję Relu
- Funkcję softmax

Oprócz zadań klasyfikacji, sieci głębokie dobrze sprawdzają się w zadaniach regresji, czyli w zadaniach w których na podstawie danych wejściowych chcemy uzyskać pewną wartość liczbową na wyjściu. W takim przypadku, wystarczy wydzielić jeden neuron wyjściowy, na wyjściu, którego otrzymamy przewidywaną wartość.
Istotnym parametrem, który definiujemy przy pracy z głębokimi sieciami neuronowymi jest funkcja straty. Funkcja ta określa miarę rozbieżności wartości przewidzianej z wartością oczekiwaną.
Najczęściej używa się funkcji:

- Błędu średniokwadratowego
- Błędu bezwzględnego
- Funkcji entropi krzyżowej

3.3. Konwolucyjne sieci neuronowe

Kolejnym przełomowym odkryciem w dziedzinie sztucznych sieci neuronowych był rezultat badań nad korą wzrokową, a dokładniej, odkrycie splotowych (konwolucyjnych) sieci neuronowych. Neurony w warstwach splotowych nie są w pełni połączone, tak jak ma to miejsce w przypadku warstw gęstych, ale łączą się jedynie z wybranymi neuronami wcześniejszej warstwy będącymi w ich polu recepcji. W przypadku pierwszej warstwy, pojedynczy neuron dostaje sygnał jedynie z pikseli będących w jego polu recepcji[18].
Odkrycie warstw splotowych pozwoliło na stworzenie przez LeCuna w 1998 roku sieci LeNet-5[19]. Sieć posługująca się wcześniej wspomnianymi warstwami została następnie wykorzystana do klasyfikacji ręcznie napisanych cyfr, w szczególności była wykorzystywana przez kilka banków, do rozpoznawania numerów na czekach.
Istotną rolę w przypadku sieci konwolucyjnych pełnią warstwy łączące. Zadaniem tych warstw jest zmniejszenie rozmiaru obrazu przekazywanego do dalszych warstw sieci. Redukujemy w ten sposób ilość przeprowadzanych obliczeń a dzięki temu zmniejszamy zużycie pamięci i przyspieszamy pracę sieci. Niestety tracimy w ten sposób też część informacji.
Wyróżniamy obecnie takie warstwy łączące jak warstwa maksymalizująca oraz warstwa uśredniająca. Warstwa maksymalizująca dzieli obraz na okna o zadanym rozmiarze by następnie wyciągnąć z każdego okna maksymalną wartość. W ten sposób zyskujemy zmniejszony obraz składający się z wartości maksymalnych zwracanych przez przyjęte okna. Warstwa uśredniająca działa podobnie do warstwy maksymalizującej, jednak zamiast zwracać maksymalną wartość okna, zwraca ona średnią z wartości znajdujących się w takim oknie. Wiąże się to z większym czasem poświęconym na obliczenia, niż w przypadku warstwy maksymalizującej.

Rozdział 4

Praca badawcza

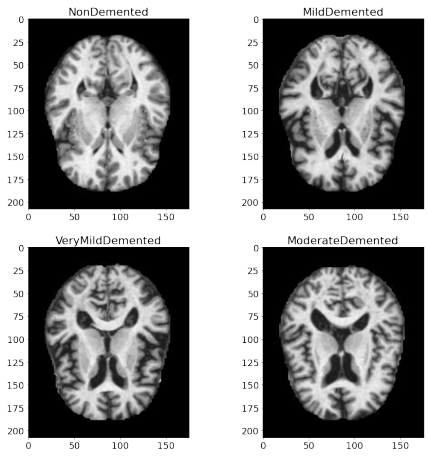
4.1. Zbiór uczący i testowy

W celu wytrenowania modelu wykorzystano zdjęcia ze zbioru Alzheimer's Dataset (4 class of Images). Zbiór ten zawiera zdjęcia mózgu wykonane metodą rezonansu magnetycznego wraz z poprawnie przydzielonymi etykietami. Zdjęcia w zbiorze mają format 176 pikseli szerokości i 208 pikseli wysokości, oraz zawierają trzy składowe kolorów.
Zbiór ten podzielono w sposób losowy uzyskując w zbiorze uczącym 80% obrazów z każdej kategorii, natomiast pozostałem 20% obrazów przydzielono do zbioru testowego.

4.1.1. Przegląd zdjęć

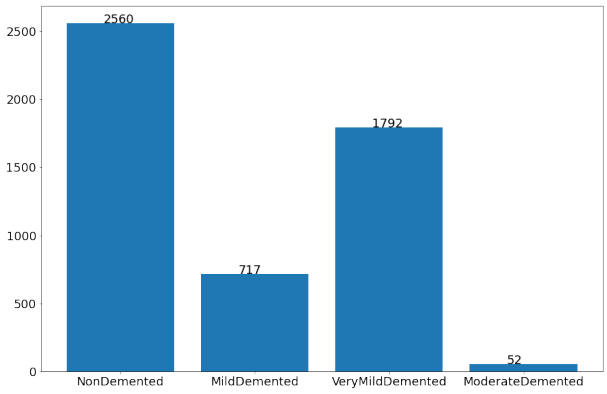
W skład zbioru wchodzą zdjęcia należące do czterech kategorii:

- Brak demencji
- Bardzo łagodna demencja
- Łagodna demencja
- Umiarkowana demencja



Rys. 4.1: Przykładowe zdjęcia rendgenowskie mózgu

Poniższy wykres ilustruje liczbę zdjęć w każdej z kategorii:



Rys. 4.2: Wykres ilustrujący liczbę zdjęć

Można zauważyć, że wśród zdjęć przeważają zdjęcia oznakowane jako "Brak demencji", natomiast zdjęć oznakowanych jako "Umiarkowana demencja"jest najmniej.

4.2. Model referencyjny

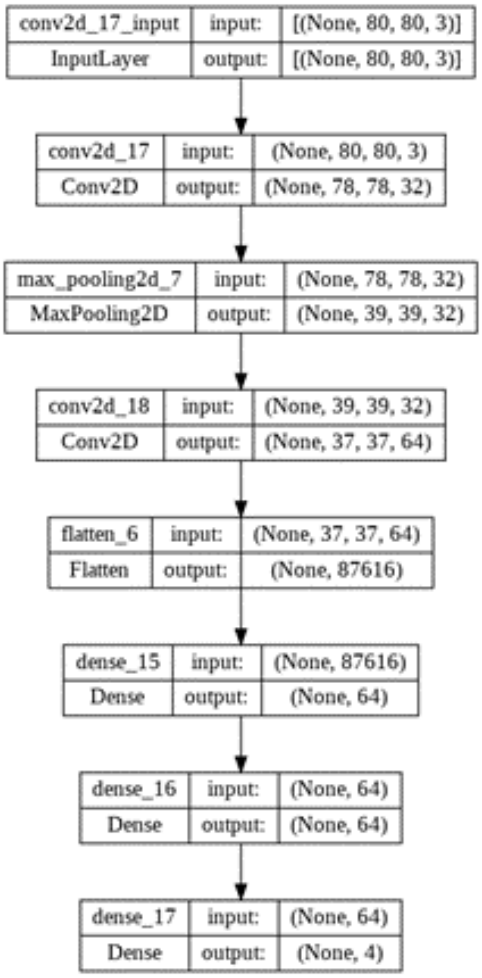
W celu sprawdzenia wpływu różnych parametrów na dokładność klasyfikacji przyjęto model referencyjny składający się z 8 warstw, w tym z:

- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 64 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji relu
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji relu
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Ponadto:

- Za funkcję straty przyjęto funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32

Struktura sieci wygenerowana za pomocą funkcji z biblioteki TensorFlow przedstawia się następująco:



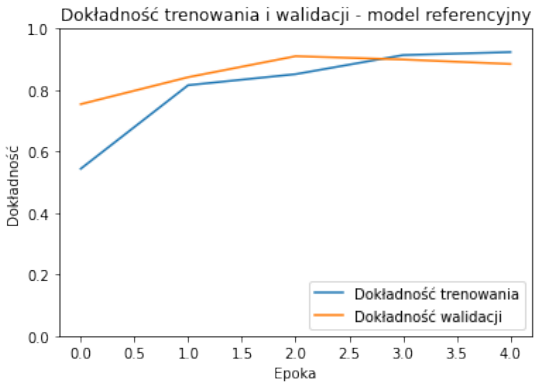
Rys. 4.3: Struktura modelu referencyjnego sieci neuronowej

Model ten zawiera 5.631.300 parametrów, natomiast liczby parametrów w kolejnych warstwach zostały przedstawione na zdjęciu poniżej.

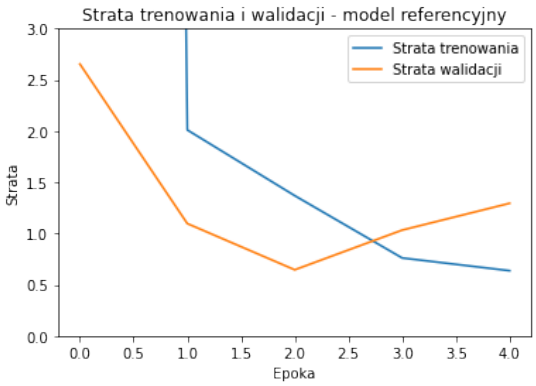
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 78, 78, 32)        896

max_pooling2d_2 (MaxPooling  (None, 39, 39, 32)        0
2D)

conv2d_5 (Conv2D)            (None, 37, 37, 64)        18496

flatten_2 (Flatten)          (None, 87616)             0

dense_6 (Dense)              (None, 64)                5607488

dense_7 (Dense)              (None, 64)                4160

dense_8 (Dense)              (None, 4)                 260

=================================================================
Total params: 5,631,300
Trainable params: 5,631,300
Non-trainable params: 0
```

Rys. 4.4: Liczba parametrów modelu referencyjnego w kolejnych warstwach

Dokładność klasyfikacji jaką jest w stanie osiągnąć model referencyjny, jest na poziomie 88% dla danych walidacyjnych, natomiast proces uczenia modelu został pokazany na poniższym zdjęciu.



Rys. 4.5: Dokładność modelu referencyjnego



Rys. 4.6: Strata modelu referencyjnego

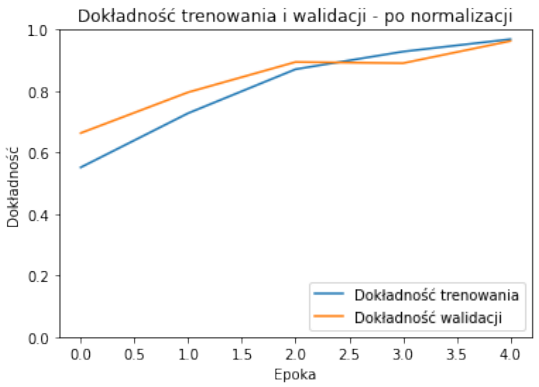
4.3. Przeprowadzone badania

4.3.1. Badanie wpływu normalizacji min-max na dokładność klasyfikacji

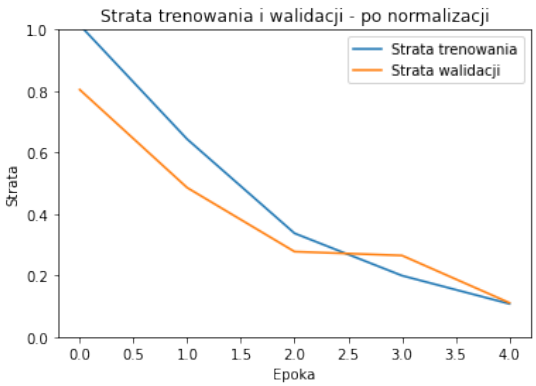
Celem normalizacja min-max jest przekształcenie zbioru danych tak, aby mieściły się w przedziale wartości [0, 1]. Odbywa się to zgodnie ze wzorem:

$$\bar{X} = \frac{X - min(X)}{max(X) - min(X)} \tag{4.1}$$

W przypadku operowania na danych będących obrazami należącymi do wczytanego zbioru, maksymalna wartość X jaką jesteśmy w stanie uzyskać, to wartość 255, natomiast najmniejsza wartość to 0. Oznacza to, że w celu przeprowadzenia procesu normalizacji, wystarczy podzielić wszystkie wartości pikseli przez 255.
W wyniku normalizacji danych metodą min-max oraz wyuczeniu tak zmodyfikowanymi danymi model referencyjny, byliśmy w stanie uzyskać 96% dokładności klasyfikacji.



Rys. 4.7: Dokładność modelu po normalizacji danych



Rys. 4.8: Strata modelu po normalizacji danych

4.3.2. Badanie wpływu funkcji aktywacji

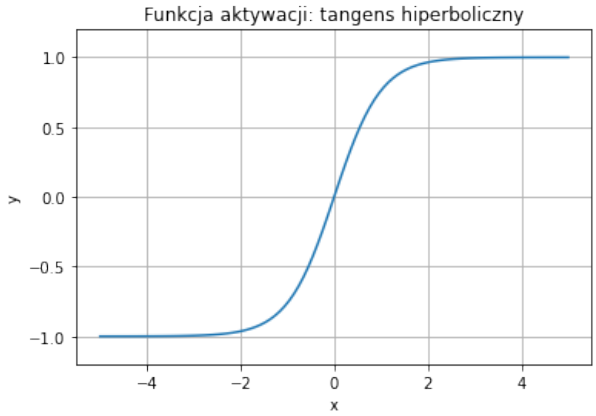
Funkcja aktywacji jest to funkcja której zadaniem jest wprowadzenie nieliniowości do przetwarzania danych przez warstwę gęstą sieci neuronowej. Gdyby model nie posiadał nieliniowej funkcji aktywacji, wykonywał by on jedynie liniowe przekształcenia danych. Nie miało by więc znaczenia jak wiele warstw ukrytych on posiada, ponieważ można by je było zastąpić pojedynczą, odpowiednio rozbudowaną warstwą gęstą. Zastosowanie nieliniowych funkcji aktywacji dla warstw gęstych w znacznym stopniu poprawia ich zdolności uczenia się wzorców, dzięki czemu jesteśmy w stanie uzyskać modele o większej dokładności klasyfikacji.
Badania przeprowadzono dla następujących funkcji aktywacji:

- Tangens hiperboliczny
- Leaky ReLU z parametrem alpha równym 0.1
- Leaky ReLU z parametrem alpha równym 0.3
- Leaky ReLU z parametrem alpha równym 0.5

Badania odbyły się na znormalizowanych danych, natomiast zmianie ulegały funkcje aktywacji warstw gęstych za wyjątkiem ostatniej warstwy, wykorzystującej funkcję softmax.

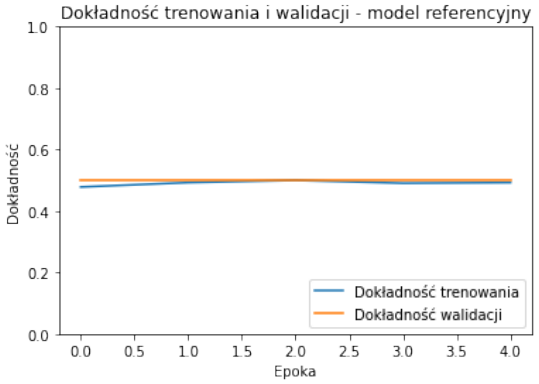
Tangens hiperboliczny

Tangens hiperboliczny jest to różniczkowalna funkcja ciągła, s-kształtna, której zbiór wartości mieści się w przedziale (-1, 1).

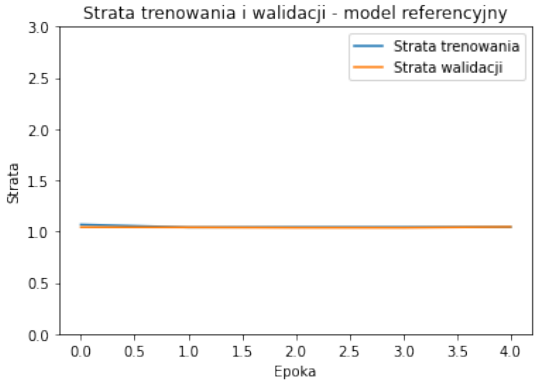


Rys. 4.9: Wykres funkcji aktywacji: tangens hiperboliczny

Wykorzystując wspomnianą funkcję osiągnięto dokładność klasyfikacji, dla znormalizowanych danych, na poziomie 50%
Można zauważyć, że wykorzystując jako funkcję aktywacji tangens hiperboliczny, model uzyskuje gorsze wyniki niż model referencyjny.



Rys. 4.10: Dokładność modelu dla funkcji aktywacji: tangens hiperboliczny



Rys. 4.11: Strata modelu dla funkcji aktywacji: tangens hiperboliczny

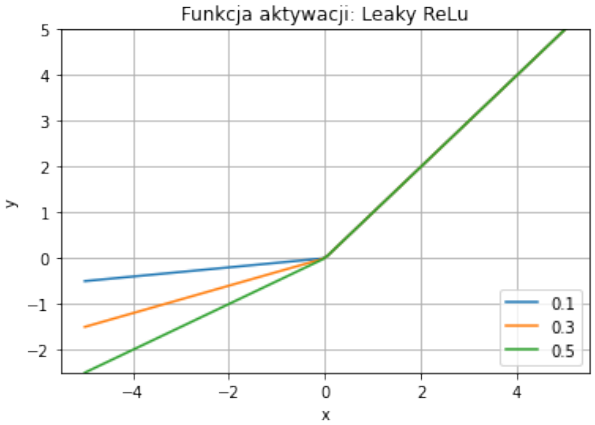
Leaky ReLu

Funkcja ReLu jest obecnie jedną z najpopularniejszych funkcji aktywacji. Posiada ona wartości w przedziale $[0, \infty)$. Funkcja ta przyjmuje dla wartości ujemnych zawsze wartość 0, przez co zmniejsza ona zdolność modelu do uczenia się na podstawie danych.

Rys. 4.12: Wykres funkcji aktywacji: ReLu

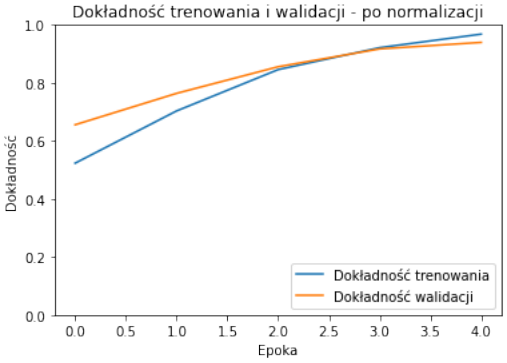
W celu rozwiązania wspomnianego problemu wprowadzona została funkcja Leaky ReLU, posiadająca wartości w przedziale $[-\infty, \infty)$, przy czym kąt nachylenia funkcji w zakresie liczb ujemnych ustalany jest parametrem alpha.
Zbadano wpływ na dokładność klasyfikacji modelu dla parametru alpha wynoszącego odpowiednio:

- 0.1
- 0.3
- 0.5

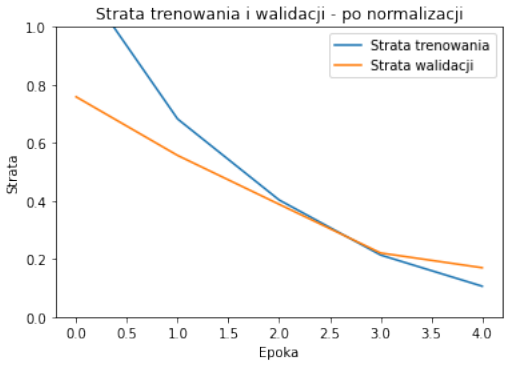


Rys. 4.13: Wykres funkcji aktywacji: Leaky ReLU dla zadanego parametru alpha

Leaky ReLu dla parametru alpha równego 0.1

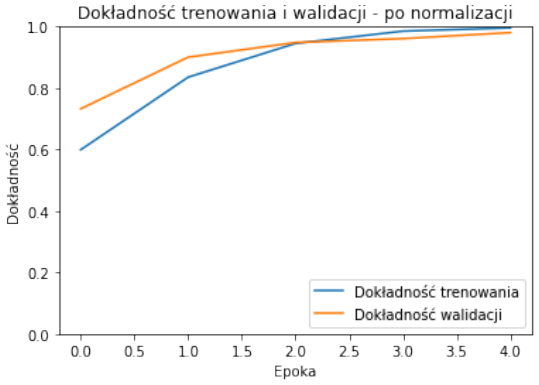


Rys. 4.14: Dokładność klasyfikacji modelu dla funkcji aktywacji Leaky ReLu przy parametrze alpha równym 0.1

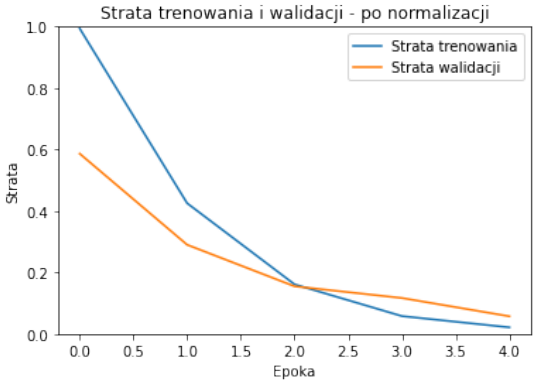


Rys. 4.15: Strata modelu dla funkcji aktywacji Leaky ReLu przy parametrze alpha równym 0.1

Leaky ReLu dla parametru alpha równego 0.3

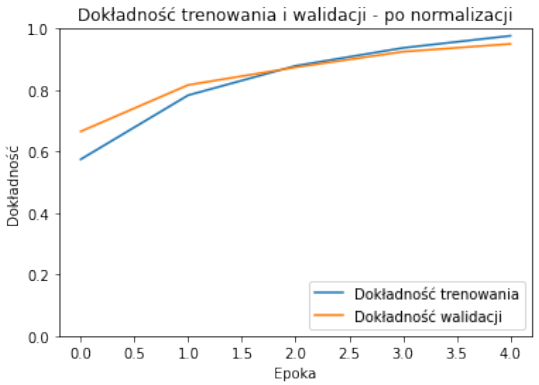


Rys. 4.16: Dokładność klasyfikacji modelu dla funkcji aktywacji Leaky ReLu przy parametrze alpha równym 0.3

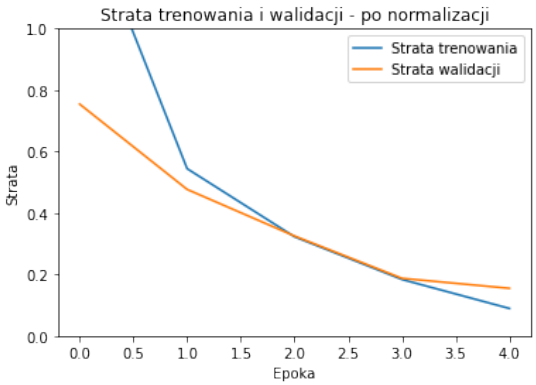


Rys. 4.17: Strata modelu dla funkcji aktywacji Leaky ReLu przy parametrze alpha równym 0.3

Leaky ReLu dla parametru alpha równego 0.5



Rys. 4.18: Dokładność klasyfikacji modelu dla funkcji aktywacji Leaky ReLu przy parametrze alpha równym 0.5



Rys. 4.19: Strata modelu dla funkcji aktywacji Leaky ReLu przy parametrze alpha równym 0.5

Najlepszy wynik model osiągnął dla parametru alpha równego 0,3. Wspomniana wartość jest także wartością domyślną parametru, wykorzystywaną w module TensorFlow[5].
Model wykorzystujący jako funkcję aktywacji Leaky ReLu, dla parametru aplha wynoszącego 0,3 osiągnął dokładność klasyfikacji wynoszącą 97% na znormalizowanych danych, tym samym uzyskując lepszy wynik niż model referencyjny wykorzystujący funkcję ReLu.

4.3.3. Badanie wpływu liczby warstw oraz ilości neuronów w warstwach gęstych

Badanie przeprowadzono dla następujących modeli sieci:

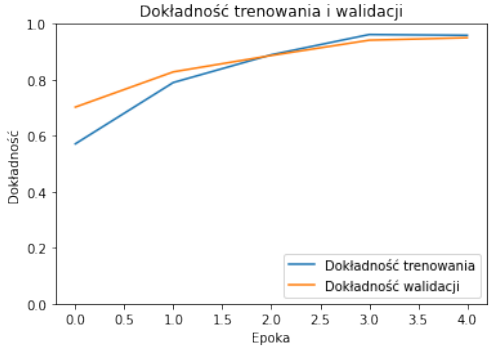
Model 1

W modelu pierwszym usunięto jedną warstwę gęstą, pozostałe warstwy pozostawiono bez zmian w stosunku do modelu referencyjnego. W szczególności pierwszy model składał się z:

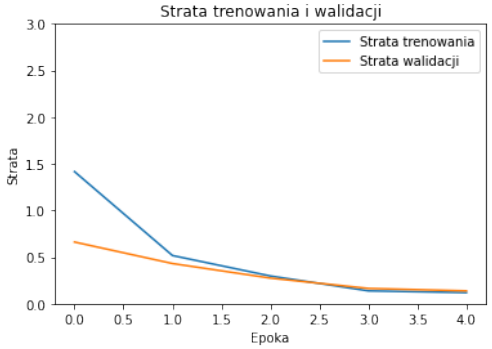
- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 64 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.20: Dokładność klasyfikacji modelu pierwszego na zbiorze treningowym oraz walidacyjnym



Rys. 4.21: Strata modelu pierwszego na zbiorze treningowym oraz walidacyjnym

Model 1 uzyskał dokładność klasyfikacji wynoszącą 95.024%

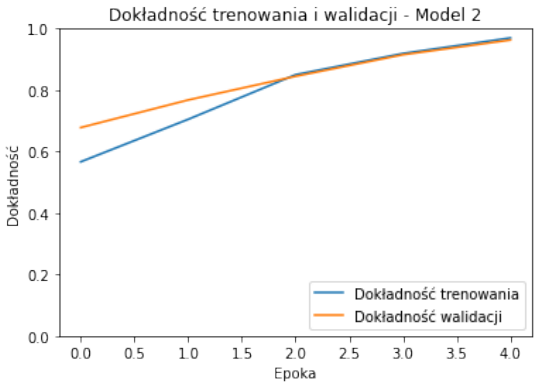
Model 2

W modelu drugim zredukowano liczbę neuronów w drugiej warstwie gęstej z 64 do 16. Model ten w szczególności składał się z:

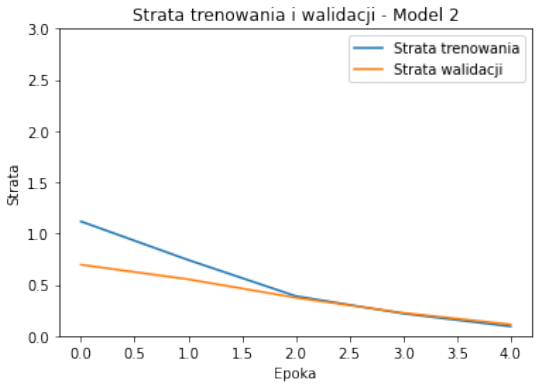
- Warstwy wejściową, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowe składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 64 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 16 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.22: Dokładność klasyfikacji modelu drugiego na zbiorze treningowym oraz walidacyjnym



Rys. 4.23: Strata modelu drugiego na zbiorze treningowym oraz walidacyjnym

Model 2 uzyskał dokładność klasyfikacji wynoszącą 95.99%

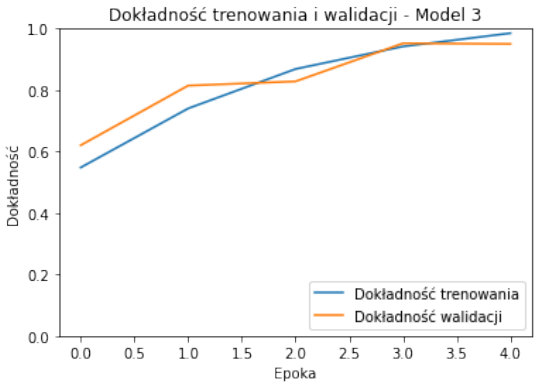
Model 3

W modelu trzecim zwiększono liczbę neuronów w pierwszej warstwie gęstej z 64 do 256. W szczególności model ten składał się z:

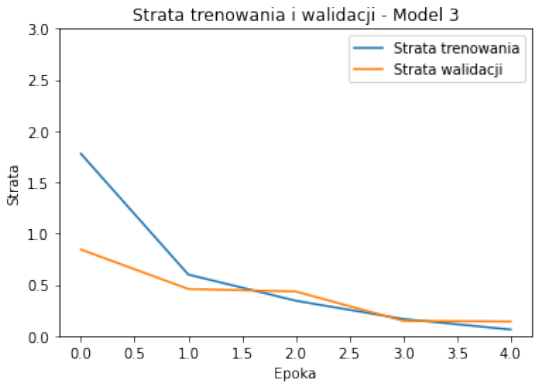
- Warstwy wejściową, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 64 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 256 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.24: Dokładność klasyfikacji modelu trzeciego na zbiorze treningowym oraz walidacyjnym



Rys. 4.25: Strata modelu trzeciego na zbiorze treningowym oraz walidacyjnym

Model 3 uzyskał dokładność klasyfikacji wynoszącą 95.51%

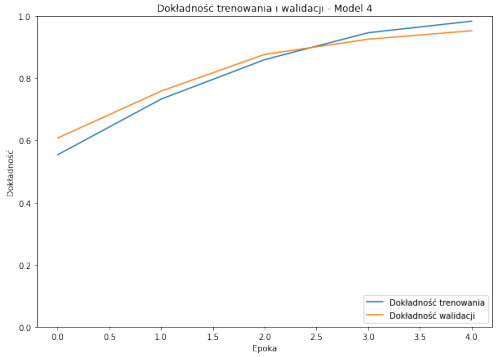
Model 4

W modelu czwartym dodano jedną warstwę gęstą składającą się z 16 neuronów, pozostałe warstwy pozostawiono bez zmian w stosunku do modelu referencyjnego. W szczególności czwarty model składał się z:

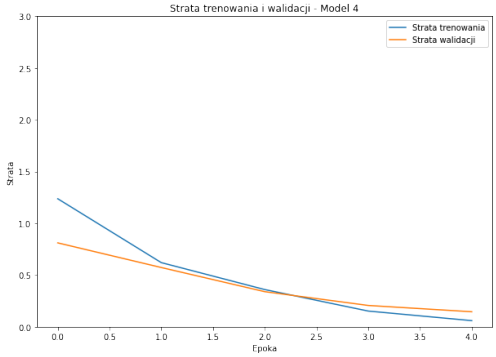
- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 64 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 16 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.26: Dokładność klasyfikacji modelu czwartego na zbiorze treningowym oraz walidacyjnym



Rys. 4.27: Strata modelu czwartego na zbiorze treningowym oraz walidacyjnym

Model 4 uzyskał dokładność klasyfikacji wynoszącą 95.32%.

4.3.4. Badanie wpływu liczby warstw konwolucyjnych oraz liczby filtrów

Badanie przeprowadzono dla następujących modeli sieci.

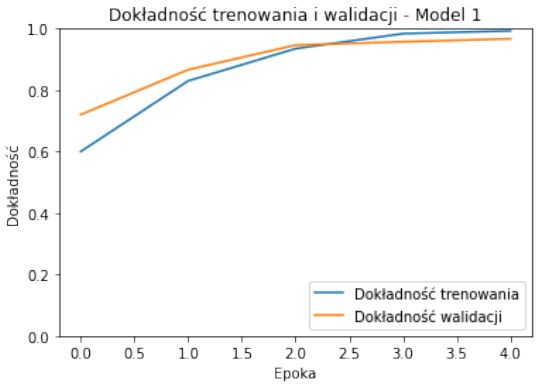
Model 1

W modelu pierwszym usunięto drugą warstwę konwolucyjną oraz warstwę łączącą. Pozostałe warstwy pozostawiono bez zmian w stosunku do modelu referencyjnego. W szczególności pierwszy model składał się z:

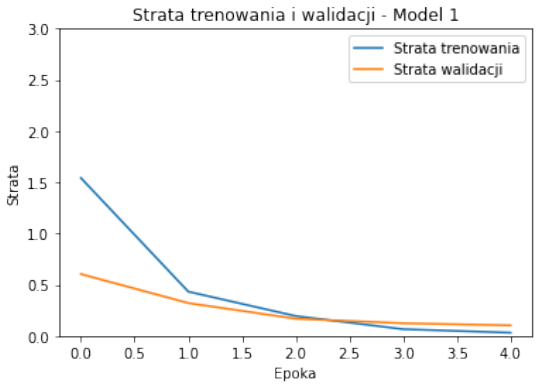
- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax, będącej również warstwą wyjściową modelu

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.28: Dokładność klasyfikacji modelu pierwszego na zbiorze treningowym oraz walidacyjnym



Rys. 4.29: Strata modelu pierwszego na zbiorze treningowym oraz walidacyjnym

Model 1 uzyskał dokładność klasyfikacji wynoszącą 96.68%.

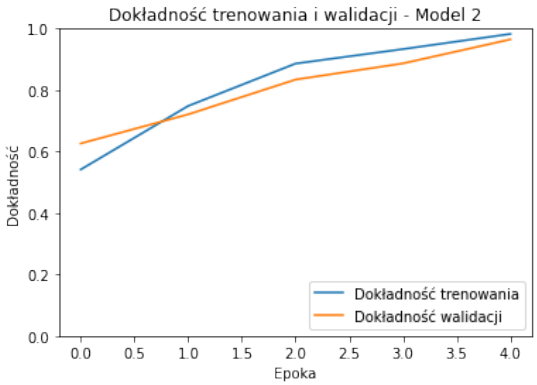
Model 2

W modelu drugim zmniejszono liczbę filtrów w drugiej warstwie splotowej z 64 na 32 filtry. Pozostałe warstwy pozostawiono bez zmian w stosunku do modelu referencyjnego. W szczególności drugi model składał się z:

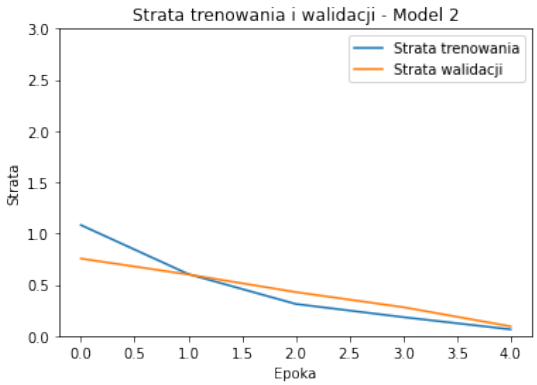
- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.30: Dokładność klasyfikacji modelu drugiego na zbiorze treningowym oraz walidacyjnym



Rys. 4.31: Strata modelu drugiego na zbiorze treningowym oraz walidacyjnym

Model 2 uzyskał dokładność klasyfikacji wynoszącą 94.8%

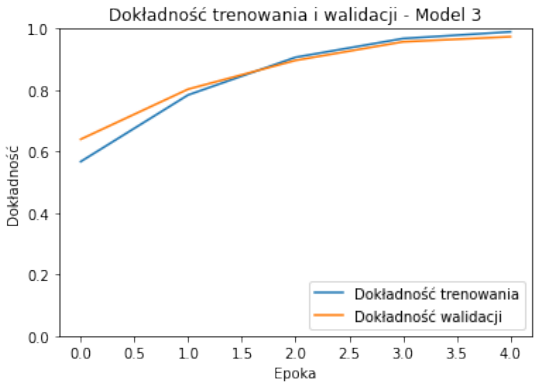
Model 3

W modelu trzecim zwiększono liczbę filtrów drugiej warstwy splotowej z 64 na 128. Pozostałe warstwy pozostawiono bez zmian w stosunku do modelu referencyjnego. W szczególności trzeci model składał się z:

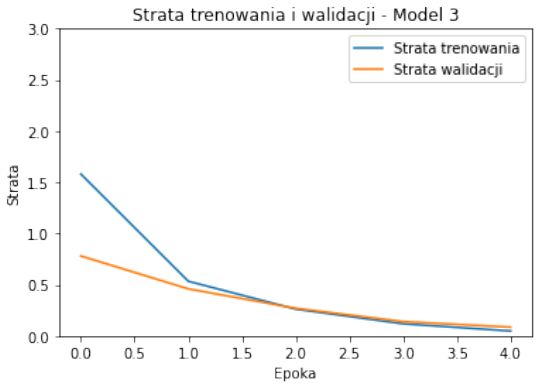
- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 128 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax

Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 5 epok uczących
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.32: Dokładność klasyfikacji modelu trzeciego na zbiorze treningowym oraz walidacyjnym



Rys. 4.33: Strata modelu trzeciego na zbiorze treningowym oraz walidacyjnym

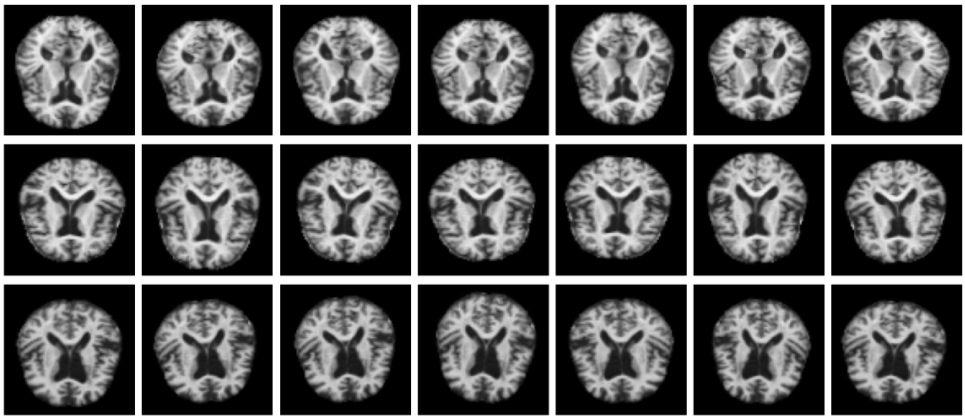
Model 3 uzyskał dokładność klasyfikacji wynoszącą 96.9%.

4.3.5. Badanie wpływu augumentacji danych

Augumentacja danych polega na powiększeniu zbioru danych uczących poprzez niewielkie modyfikacje oryginalnego zbioru. Zyskujemy w ten sposób zbiór podobnych do siebie zdjęć, na podstawie których model jest w stanie nauczyć się ogólnych cech.
Powiększenie zbioru danych jest pomocne w przypadku w którym dysponujemy niewystarczającą ilością danych. Model po zwiększeniu liczby epok uczących może zbytnio dopasować się do danych uczących. Zjawisko to nazywamy przeuczeniem modelu. Wpływa ono negatywnie na zdolności klasyfikacji zbioru danych walidacyjnych, ponieważ przeuczony model traci zdolności generalizacji informacji i dopasowuje się do cech danych uczących.
Należy jednak pamiętać, że zbyt duże modyfikacje oryginalnych obrazów mogą prowadzić do pogorszenia dokładności klasyfikacji. Model zamiast uczyć się cech obiektu na obrazie może zacząć się uczyć przekształceń, przez co osiągnie on gorsze wyniki na zbiorze walidacyjnym, który nie uległ żadnym modyfikacją.

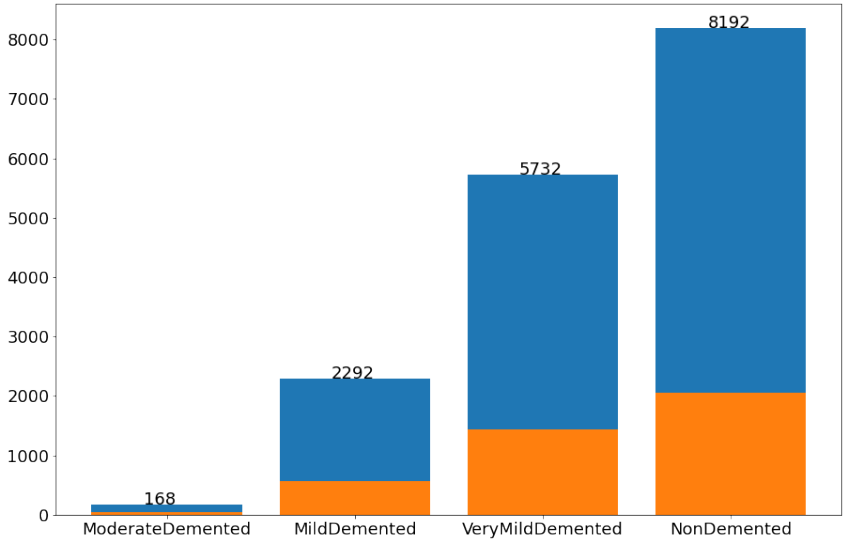
Wykorzystano takie modyfikacje obrazu jak:

- Obrót obrazu o losowy kąt w zakresie [-3, 3] stopni
- Przesunięcie obrazu o losową wartość pikseli w pionie w zakresie [-1, 1]
- Przesunięcie obrazu o losową wartość pikseli w poziomie w zakresie [-1, 1]
- Przybliżenie obrazu o losową wartość w przedziale [-5, 5] procent



Rys. 4.34: Losowe przekształcenia 3 zdjęć rendgenowskich mózgu

W wyniku zastosowania augumentacji danych, zbiór danych uczących został powiększony z 4096 zdjęć do 16384 zdjęć uczących.



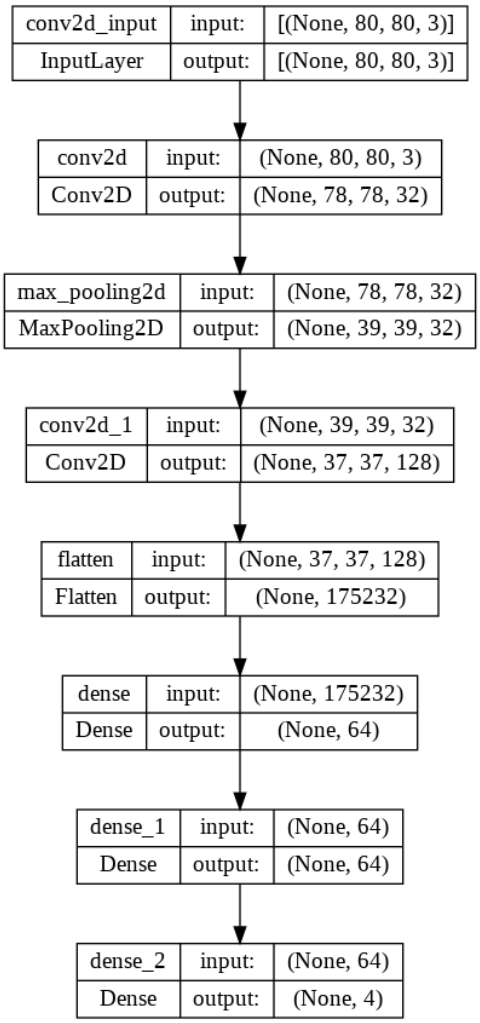
Rys. 4.35: Wykres ilustrujący przyrost danych uczących po zastosowaniu augumentacji

Do sprawdzenia dokładności klasyfikacji wykorzystano wnioski z wcześniej przeprowadzonych badań. Między innymi, wykorzystano:

- znormalizowane dane
- funkcję aktywacji Leaky ReLu z parametrem alpha równym 0.3
- dwie warstwy konwolucyjne o liczbie filtrów odpowiednio 32 w pierwszei i 128 w drugiej warstwie
- dwie warstwy gęste o z 64 neuronami w każdej warstwie

Końcowy model składa się z:

- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy splotowej składającej się z 32 filtrów z oknem splotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy splotowej składającej się z 128 filtrów z oknem splotu o wymiarach 3x3
- Warstwy spłaszczającej dane wejściowe
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy gęstej, w pełni połączonej, składającej się z 64 neuronów oraz funkcji aktywacji Leaky ReLu z parametrem alpha równym 0.3
- Warstwy wyjściowej złożonej z warstwy gęstej, w pełni połączonej, składającej się z 4 neuronów oraz funkcji aktywacji softmax



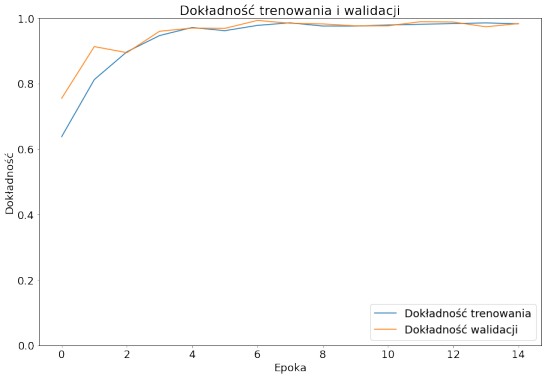
Rys. 4.36: Struktura końcowego modelu sieci neuronowej

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 78, 78, 32)        896

max_pooling2d (MaxPooling2D  (None, 39, 39, 32)       0
)

conv2d_1 (Conv2D)            (None, 37, 37, 128)       36992

flatten (Flatten)            (None, 175232)            0

dense (Dense)                (None, 64)                11214912

dense_1 (Dense)              (None, 64)                4160

dense_2 (Dense)              (None, 4)                 260

=================================================================
Total params: 11,257,220
Trainable params: 11,257,220
Non-trainable params: 0
_________________________________________________________________
```

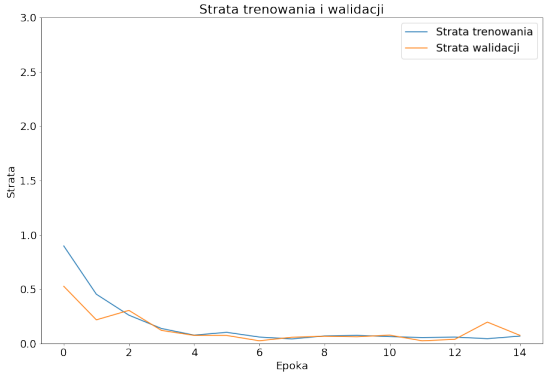
Rys. 4.37: Liczba parametrów końcowego modelu w kolejnych warstwach

Ponadto przyjęto:

- Za funkcję straty przyjęto funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto 15 epok uczących, z czego wybrano model o największej dokładności klasyfikacji danych walidacyjnych
- Za metrykę przyjęto dokładność klasyfikacji (ang. accuracy)
- Wielkość partii (ang. batch size) pozostawiono na domyślnej wartości wynoszącej 32



Rys. 4.38: Dokładność klasyfikacji modelu na zbiorze treningowym oraz walidacyjnym



Rys. 4.39: Strata modelu na zbiorze treningowym oraz walidacyjnym

Dzięki powiększeniu zbioru danych uczących możliwe było zwiększenie liczby epok, co nie spowodowało przeuczenia modelu.
Ostatecznie uzyskany model osiągnął 99.32% dokładności klasyfikacji na zbiorze walidacyjnym. Wynik ten jest znacznie lepszy od wyników wcześniejszych modeli.
Model ten następnie wyeksportowano i wykorzystano w serwisie internetowym.

Rozdział 5

Implementacja systemu

Projekt systemu został rozpoczęty od stworzenia maszyny wirtualnej za pomocą oprogramowania Oracle Virtual Box. W tym celu utworzono nową maszynę wirtualną z systemem operacyjnym Linux Ubuntu, narzucono maksymalne zużycie pamięci RAM i przydzielono ilość miejsca na dysku. Następnie zwiększono ilość procesorów do dwóch oraz przydzielono obraz systemu operacyjnego Linux Ubuntu 22.04. Wybrano minimalną instalację systemu oraz domyślne ustawienia. Następnie przystąpiono do instalacji wymaganych modułów i oprogramowania.

Aplikacja powstała z wykorzystaniem frameworku Flask. W tym celu utworzono plik projektu app.py. W pliku tym umieszczono kontrolery odpowiedzialne za przetwarzanie danych oraz przekazywanie ich do widoku. W tym pliku znajduje się między innymi logika odpowiedzialna za sprawdzanie zalogowania się pracownika jak również wczytanie przesłanego zdjęcia rentgenowskiego mózgu oraz klasyfikacja stopnia zaawansowania choroby za pomocą wcześniej wytrenowanego modelu sieci neuronowej.

W celu prezentacji danych oraz pobierania ich od użytkownika utworzono folder templates w którym umieszczono widoki aplikacji oraz pod widoki, którymi posługiwano się w widokach głównych. W celu zachowania przejrzystości widoków wydzielono style czyli opis graficzny elementów strony do plików formatu css, które następnie umieszczono w folderze static. W folderze tym utworzono również folder def w którym znalazły się wszystkie domyślne zdjęcia, między innymi ikony operacji na pacjencie oraz domyślne zdjęcie pacjenta.

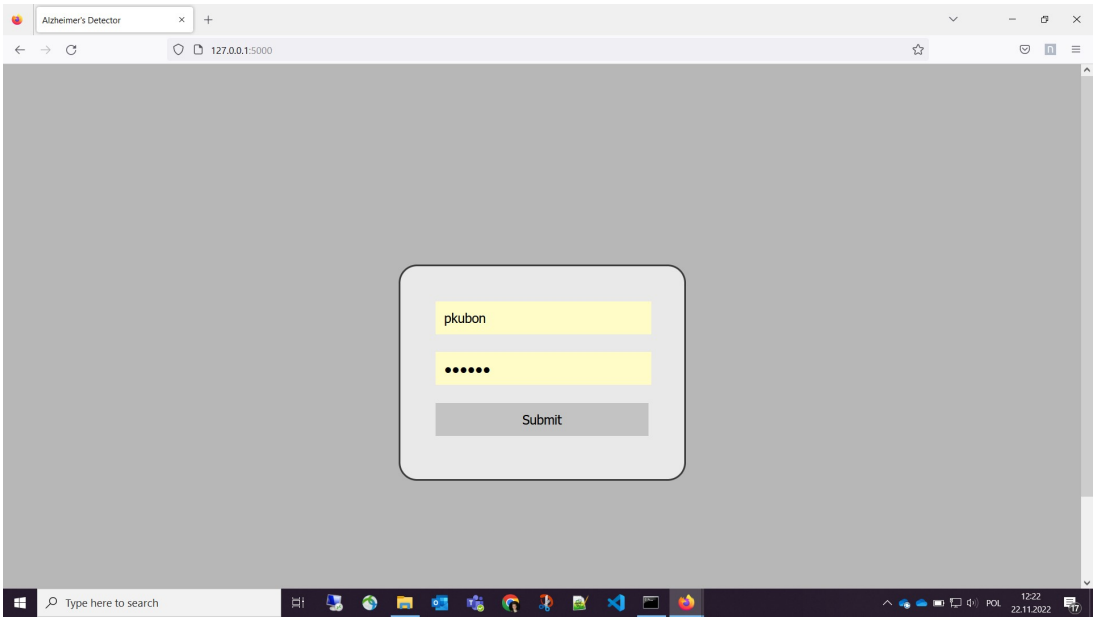
Utworzono również folder models w którym umieszczono moduły z modelami obiektów które wykorzystano w aplikacji do reprezentacji danych, jak również folder db w którym umieszczono pliki formatu json w których zapisywano dane. Przesłane zdjęcia natomiast zostaną zapisane w folderze uploads.

Aby sprawdzać poprawne działanie napisanych funkcjonalności, utworzono testy jednostkowe. Testy te wykonano z wykorzystaniem modułu pytest. Obejmowały one między innymi testy zapisu i odczytu danych jak również działania kontrolerów.

5.1. Implementacja serwisu internetowego

5.1.1. Panel logowania

Panel logowania jest pierwszym panelem serwisu jaki zobaczy użytkownik chcący skorzystać z serwisu. Zapewnia on blokadę treści i danych zawartych w serwisie przed osobami niepowołanymi. Użytkownik chcąc skorzystać z serwisu wprowadza otrzymane od administratora aplikacji dane logowania. Jeżeli dane te są spójne z danymi zapisanymi w serwisie, następuje przekierowanie do strony głównej serwisu, użytkownik ten po zalogowaniu się posiada dostęp do wszystkich funkcjonalności oferowanych przez serwis. W przypadku wprowadzenia niepoprawnych danych logowania, użytkownik nie zostaje przekierowany na stronę główną, ale z powrotem na stronę logowania wraz z uzupełnionymi polami tekstowymi o dane wcześniej wprowadzone. Sprawdzenie danych logowania odbywa w kontrolerze odpowiedzialnym za walidacje przekazanych danych.



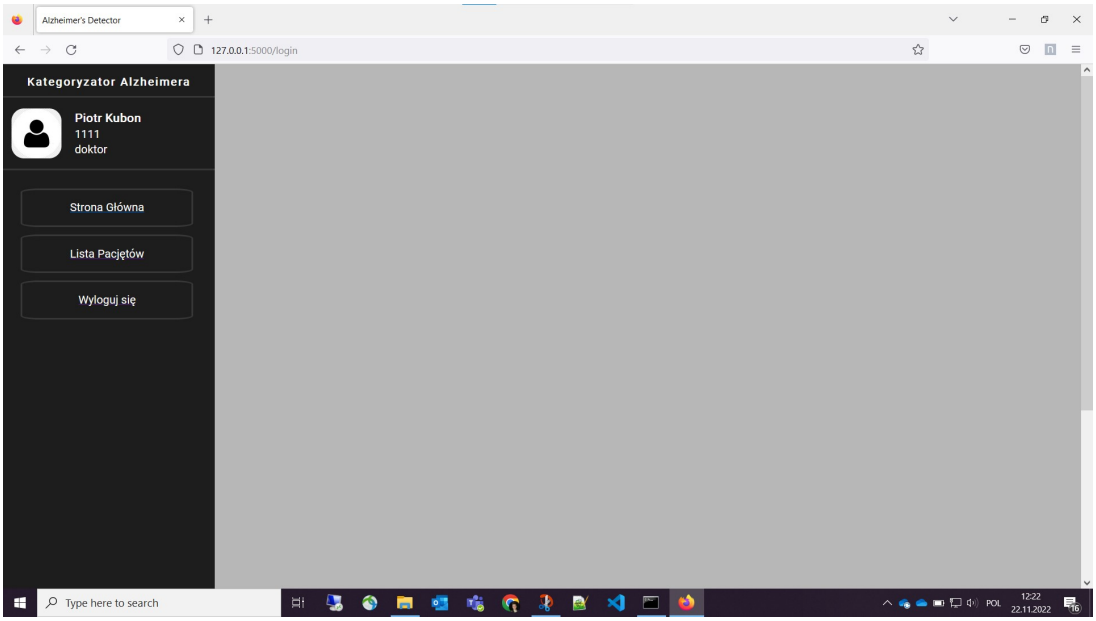
Rys. 5.1: Panel logowania

5.1.2. Panel strony głównej

Zalogowany użytkownik przekierowywany jest na stronę główną. Strona główna składa się z pod widoku menu oraz kontenera na dane. Pod widok menu umieszczany jest również na wszystkich późniejszych panelach w celu łatwiejszego korzystania z serwisu. Możliwe jest to dzięki wykorzystaniu szablonów Jinja2. Szablony te umożliwiają pisanie w widoku kodu zbliżonego w składni do kodu pisanego w języku python, który następnie przetwarza i wyświetla przekazane dane w określony wcześniej sposób.
Menu zapewnia następujące możliwość:

- Przejścia na stronę główną
- Przejścia na panel z listą pacjentów
- Wylogowanie się z serwisu

Menu wyświetla również informację o aktualnie zalogowanym użytkowniku.



Rys. 5.2: Panel strony głównej

5.1.3. Panel listy pacjentów

Panel z listą pacjentów udostępnia możliwość dodania nowego pacjenta do serwisu. Udostępnia on również listę z informacjami na temat pacjentów w serwisie oraz dodatkowe operacje, które można wykonać na wybranym pacjencie.
Do operacji udostępnionych za pośrednictwem serwisu należą:

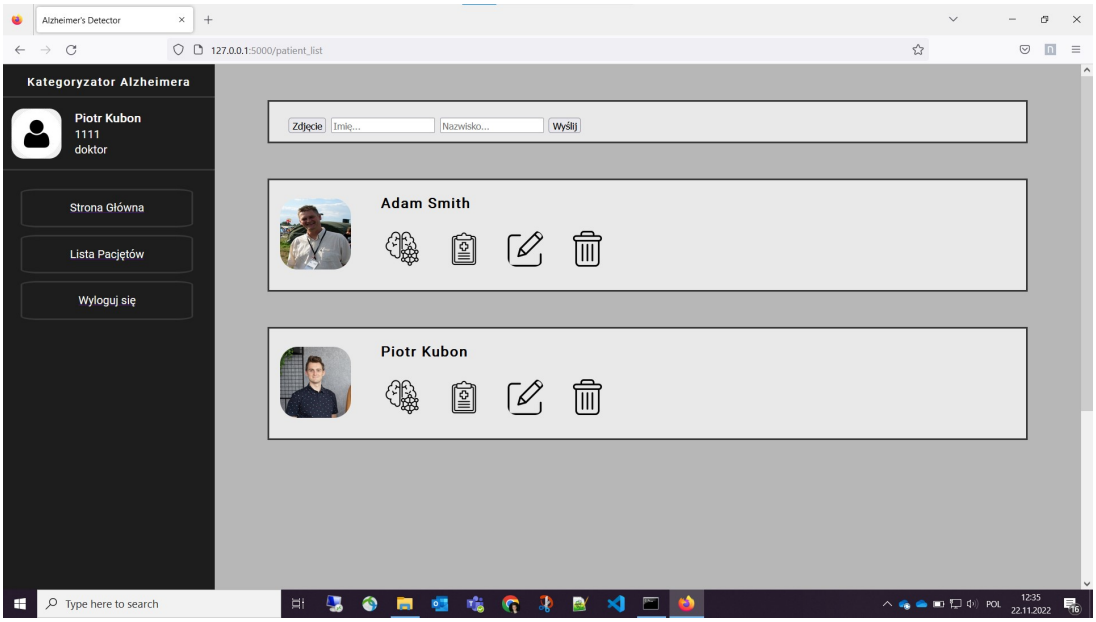
- Przeprowadzenie badań za pomocą wgrania zdjęcia rentgenowskiego skanu mózgu pacjenta
- Wyświetlenie historii badań przeprowadzonych na pacjencie
- Edycja danych pacjenta
- Usunięcie pacjenta z serwisu

Każda funkcjonalność zawiera w serwisie odpowiadający za jej przeprowadzenie kontroler, do którego przekazywane jest id pacjenta.
Pole przeznaczone do dodania pacjenta do serwisu zawiera następujące pola wejściowe:

- Pole do wprowadzenia zdjęcia pacjenta
- Pole do wprowadzenia imienia pacjenta
- Pole do wprowadzenia nazwiska pacjenta

Po wciśnięciu przycisku wyślij, następuje przekazanie wprowadzonych danych do kontrolera odpowiedzialnego za utworzenie modelu pacjenta oraz dodania go do listy pacjentów w serwisie. W przypadku pomyślnego dodania pacjenta następuje przekierowanie do panelu z listą pacjentów.

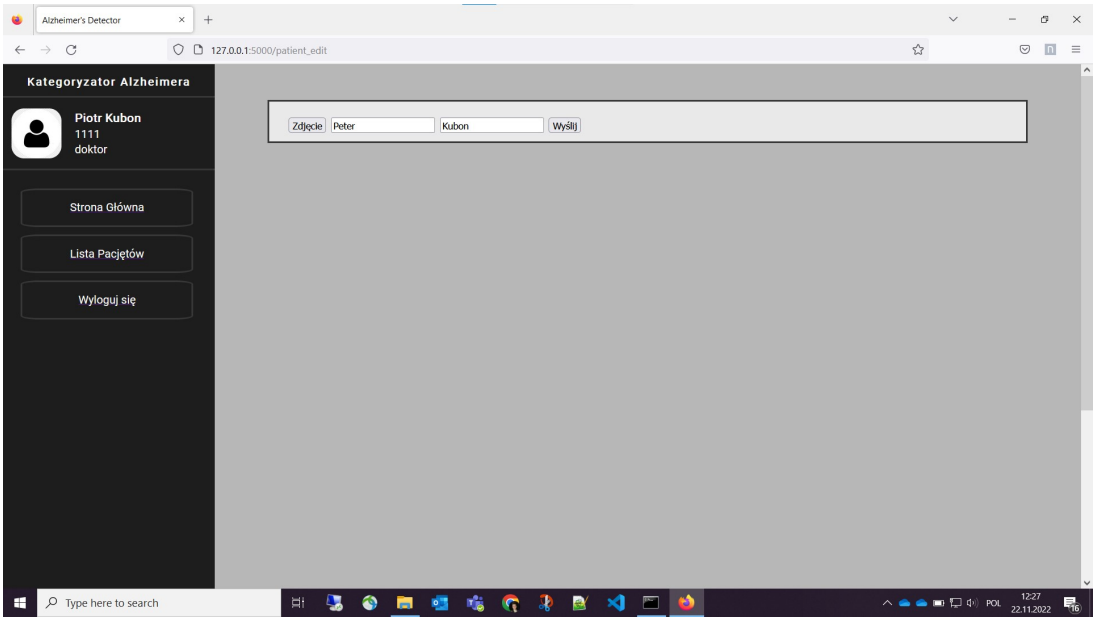
Serwis zapewnia możliwość zostawienia pustego pola danych. W przypadku nie wybrania zdjęcia pacjenta, następuje przydzielenie domyślnego zdjęcia do modelu, natomiast w przypadku nieprzesłania danych o imieniu lub nazwisku, odpowiednie pole nie będzie zawierać żadnej informacji.



Rys. 5.3: Panel listy pacjentów

5.1.4. Panel edycji danych pacjentów

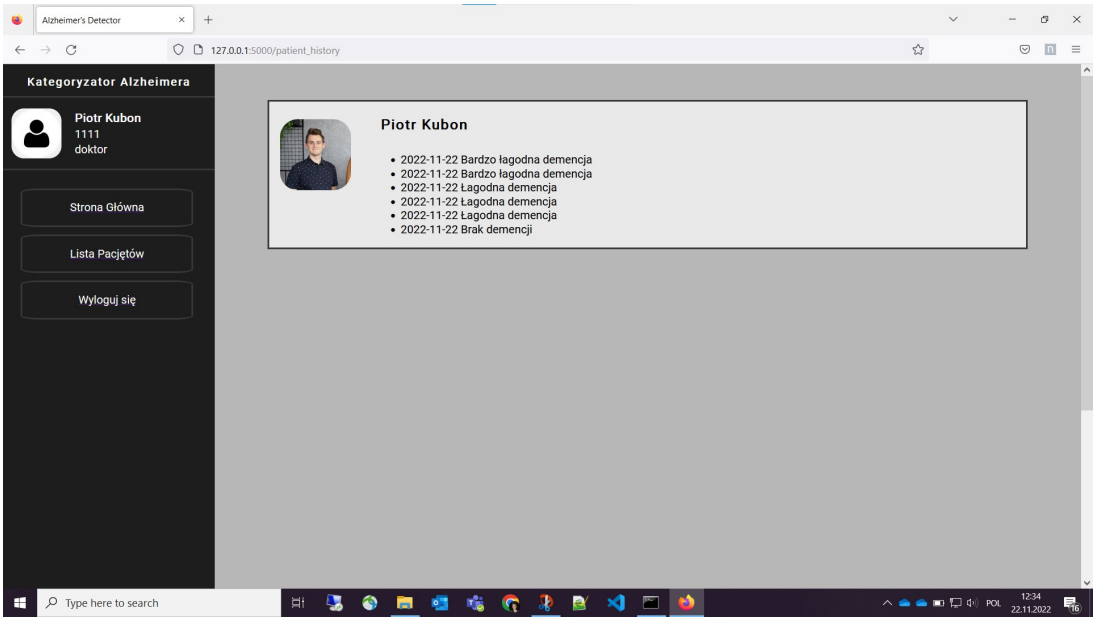
W wyniku wybrania opcji edycji danych pacjenta, następuje przesłanie id pacjenta do dedykowanego kontrolera. W kontrolerze następuje pobranie danych pacjenta oraz przekazanie ich do widoku edycji danych pacjenta, który zostaje wyświetlony użytkownikowi serwisu. Po wyświetleniu się panelu, użytkownik może wprowadzić odpowiednie zmiany danych pacjenta oraz potwierdzić wprowadzone zmiany za pomocą przycisku wyślij. Po ciśnięciu przycisku wyślij następuje przesłanie danych na serwer do odpowiedniego kontrolera, w którym stare dane zostaną zastąpione nowymi. Uaktualnione dane zostaną również natychmiastowo zapisane w bazie. Za pomocą panelu edycji danych pacjenta możemy modyfikować imię, nazwisko oraz zdjęcie pacjenta, jednak pole id oraz historia przeprowadzonych badań pozostają niezmienione. Serwis nie udostępnia możliwości edycji pola id w celu zachowania spójności danych w systemie.



Rys. 5.4: Panel edycji danych pacjentów

5.1.5. Panel historii badań pacjenta

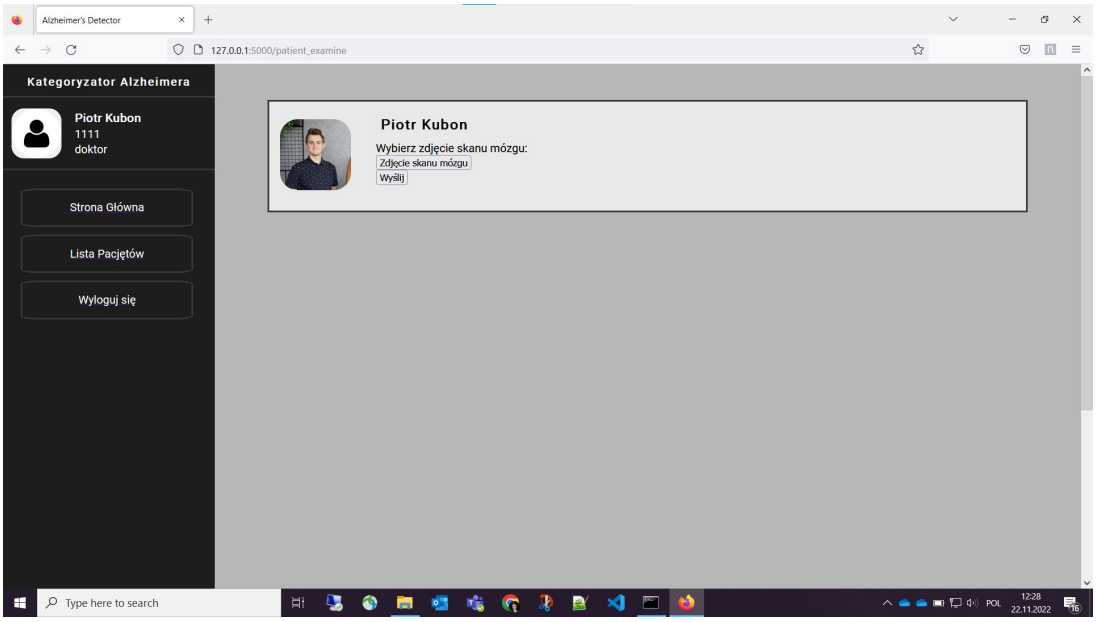
Panel historii badań pacjenta zawiera podstawowe informacje na temat pacjenta wraz z zdjęciem pacjenta oraz historią badań. W skład historii badań wchodzą wyniki wszystkich wcześniej wykonanych badań na pacjencie wraz z datą ich przeprowadzenia. Ostatnie przeprowadzone badanie znajduje się na szczycie listy, natomiast najpóźniej wykonane badanie znajduje się na samym dole listy.



Rys. 5.5: Panel historii badań pacjenta

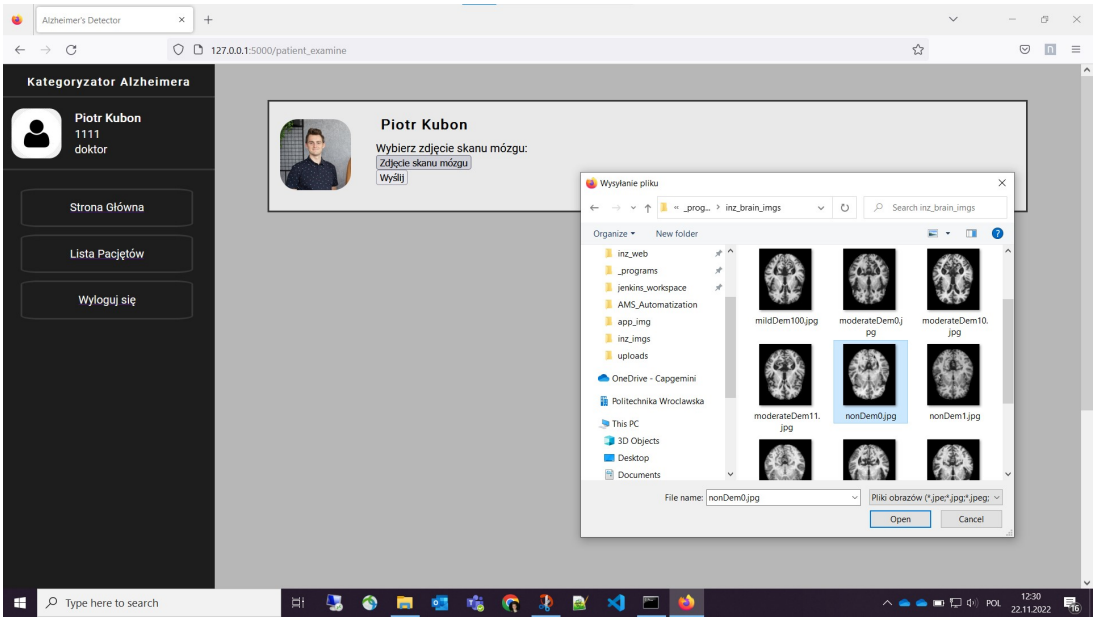
5.1.6. Panel badania pacjenta

Panel badania pacjenta udostępnia możliwość przeprowadzenia badania stopnia rozwoju choroby. Panel ten zawiera podstawowe informacje oraz zdjęcie pacjenta. Panel ten zawiera również przycisk wyboru zdjęcia rentgenowskiego mózgu pacjenta z dysku na komputerze użytkownika oraz przycisk odpowiedzialny za wysłanie zdjęcia mózgu pacjenta.



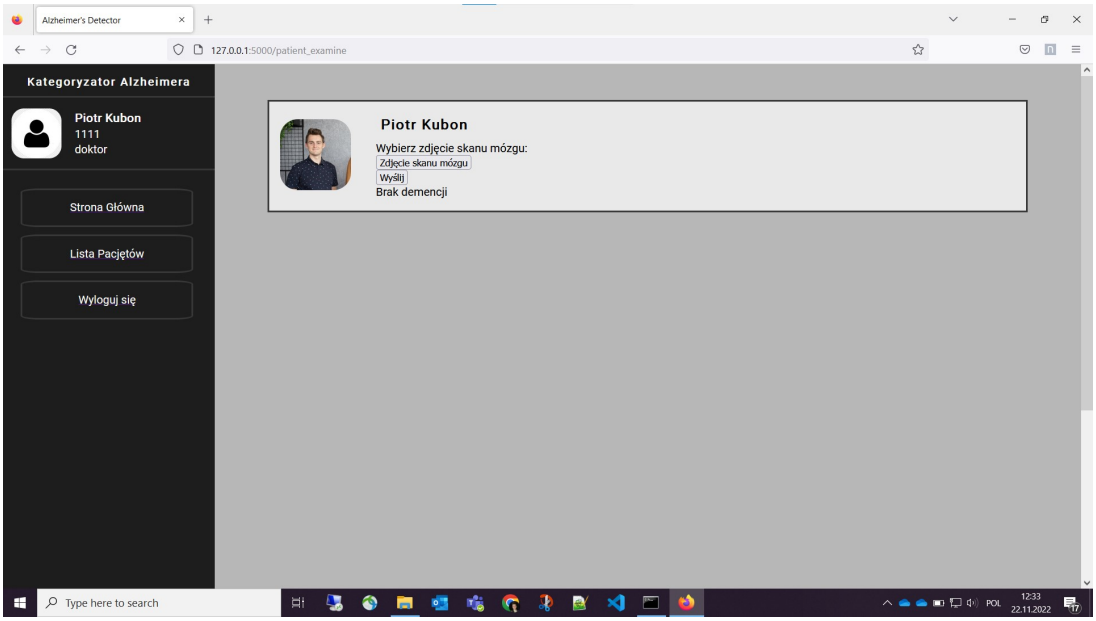
Rys. 5.6: Panel badania pacjenta

Po wciśnięciu przycisku „Zdjęcie skanu mózgu” wyświetla się okienko z podglądem zdjęć na dysku. Użytkownik może wybrać odpowiednie zdjęcie i potwierdzić wybór klikając przycisk „Open”.



Rys. 5.7: Okno wyboru zdjęcia skanu mózgu

Następnie użytkownik może przesłać zdjęcie na serwer w celu wykonania klasyfikacji wciskając przycisk „Wyślij”. Po przesłaniu zdjęcia następuje klasyfikacja z wykorzystaniem wcześniej wytrenowanego i zaimportowanego modelu sieci neuronowej. Wynik klasyfikacji jest następnie przetwarzany i dodany do historii badań pacjenta wraz z aktualną datą wykonania badania. Wynik klasyfikacji jest także przesyłany do widoku w celu prezentacji wyników użytkownikowi.



Rys. 5.8: Panel z wynikiem badania

Rozdział 6

Podsumowanie

Dzięki przeprowadzonym badaniom udało się stworzyć model z wysoką skutecznością klasyfikacji zdjęć rendgenowskich mózgu. Udało się także stworzyć serwis internetowy wykorzystujący wcześniej wspomniany model co w pełni pozwoliło zrealizować temat pracy inżynierskiej.

W wyniku przeprowadzonych badań wyciągnięto także następujące wnioski:

- Normalizacja danych w znacznym stopniu poprawiła wyniki klasyfikacji modelu.
- Funkcja aktywacji jest istotną funkcją wprowadzającą nieliniowe przekształcenia danych do warstwy gęstej, jednak nie wszystkie funkcje aktywacji nadają się do zadań klasyfikacji.
- Najlepsze wyniki klasyfikacji osiągnęła funkcja aktywacji Leaky ReLU z parametrem alpha równym 0.3
- Zmiany liczby neuronów w warstwach ukrytych nie spowodowały znaczącej poprawy klasyfikacji.
- Zmiana liczby filtrów w warstwie konwolucyjnej spowodowała poprawę klasyfikacji badanego modelu.
- Augumentacja danych jest procesem mogącym zarówno poprawić jak i pogorszyć dokładność klasyfikacji. Wprowadzając niewielkie zmiany oryginalnych danych jesteśmy w stanie znacznie zwiększyć zbiór danych uczących.
- Wprowadzając niewielkie zmiany oryginalnych danych byliśmy w stanie poprawić dokładność klasyfikacji modelu. Mogliśmy także zwiększyć ilość epok uczących, nie doprowadzając do przeuczenia modelu.

Literatura

[1] 13 real-world examples of python in web development. https://careerfoundry.com/en/blog/web-development/python-real-examples/ [Dostęp dnia 14 grudnia 2022].

[2] Choroba alzheimera: przyczyny, leczenie, objawy choroby alzheimera. https://www.mp.pl/pacjent/neurologia/choroby/151134,choroba-alzheimera [Dostęp dnia 14 grudnia 2022].

[3] Colaboratory frequently asked questions. https://research.google.com/colaboratory/faq.html [Dostęp dnia 14 grudnia 2022].

[4] Demencja starcza czy już alzheimer – jak to rozróżnić? https://holsamed.pl/demencja-starcza-czy-juz-alzheimer-jak-to-rozroznic/ [Dostęp dnia 14 grudnia 2022].

[5] Dokumentacja tensorflow leakyrelu. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LeakyReLU [Dostęp dnia 14 grudnia 2022].

[6] Everything you need to know about mvc architecture. https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1 [Dostęp dnia 14 grudnia 2022].

[7] Framework. https://pl.wikipedia.org/wiki/Framework [Dostęp dnia 14 grudnia 2022].

[8] Jinja2 introduction. https://jinja.palletsprojects.com/en/3.1.x/intro/ [Dostęp dnia 14 grudnia 2022].

[9] List of 7 best python frameworks to consider for your web project. https://www.monocubed.com/blog/top-python-frameworks/ [Dostęp dnia 14 grudnia 2022].

[10] Model–view–controller. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [Dostęp dnia 14 grudnia 2022].

[11] Mvc - model view controller. https://docs.phalcon.io/4.0/pl-pl/mvc [Dostęp dnia 14 grudnia 2022].

[12] Numbers of people with dementia worldwide. https://www.alzint.org/resource/numbers-of-people-with-dementia-worldwide/ [Dostęp dnia 14 grudnia 2022].

[13] Perceptron. https://en.wikipedia.org/wiki/Perceptron [Dostęp dnia 14 grudnia 2022].

[14] Perceptron wielowarstwowy. https://pl.wikipedia.org/wiki/Perceptron_wielowarstwowy [Dostęp dnia 14 grudnia 2022].

[15] phalcon controllers. https://docs.phalcon.io/4.0/pl-pl/controllers [Dostęp dnia 14 grudnia 2022].

[16] Top 10 machine learning algorithms in 2022. https://www.spiceworks.com/tech/artificial-intelligence/articles/top-ml-algorithms/ [Dostęp dnia 14 grudnia 2022].

[17] Top 7 python neural network libraries for programmers. https://analyticsindiamag.com/top-7-python-neural-network-libraries-for-developers/ [Dostęp dnia 14 grudnia 2022].

[18] A. Geron. Uczenie maszynowe z użyciem scikit-learn i tensorflow, wydanie ii. Helion, 2019.

[19] L. LeCun, Bottou. Gradient-based learning applied to document recognition. 1998.

[20] W. McCulloch. A logical calculus of the ideas immanent in nervous activity. 1943.

[21] M. Minsky, Papert. Perceptrons. 1969.

[22] W. Rumelhart, Hinton. Learning representations by back-propagating errors. 1986.