

# Spis treści

<b>1. Wstęp</b>	<b>7</b>
1.1. Cel	7
1.2. Opis choroby Alzheimera	7
1.3. Opis systemu	7
1.4. Przebieg pracy nad systemem	8
<b>2. Projekt systemu</b>	<b>9</b>
2.1. Opis wykorzystanych narzędzi	9
2.2. Wykorzystane wzorce projektowe	10
2.2.1. MVC (Model-View-Controller)	10
2.3. Modele danych	11
<b>3. Sztuczne sieci neuronowe</b>	<b>13</b>
3.1. Początki sztucznych sieci neuronowych	13
3.2. Głębokie sieci neuronowe	14
3.3. Konwolucyjne sieci neuronowe	15
<b>4. Praca badawcza</b>	<b>16</b>
4.1. Zbiór uczący i testowy	16
4.1.1. Przegląd zdjęć	16
4.2. Opis badanych metryk klasyfikacji	17
4.2.1. Precyzja (ang. Precision)	18
4.2.2. Odwołanie (ang. Recall)	18
4.2.3. Metryka F1	18
4.3. Model referencyjny	18
4.4. Przeprowadzone badania	21
4.4.1. Badanie wpływu normalizacji min-max na dokładność klasyfikacji	21
4.4.2. Badanie wpływu funkcji aktywacji	22
4.4.3. Badanie wpływu liczby warstw oraz ilości neuronów w warstwach gęstych	28
4.4.4. Badanie wpływu liczby warstw konwolucyjnych oraz liczby filtrów	34
4.4.5. Badanie wpływu augmentacji danych	39
<b>5. Implementacja systemu</b>	<b>44</b>
5.1. Implementacja serwisu internetowego	45
5.1.1. Panel logowania	45
5.1.2. Panel strony głównej	46
5.1.3. Panel listy pacjentów	47
5.1.4. Panel edycji danych pacjentów	48
5.1.5. Panel historii badań pacjenta	49
5.1.6. Panel badania pacjenta	50

<b>6. Podsumowanie . . . . .</b>	<b>52</b>
<b>Literatura . . . . .</b>	<b>53</b>

# Spis rysunków

2.1. Model użytkownika . . . . .	11
2.2. Model pacjenta . . . . .	11
2.3. Model badania . . . . .	12
3.1. Funkcja skokowa Heaviside'a . . . . .	13
3.2. Funkcja signum . . . . .	14
4.1. Przykładowe zdjęcia rentgenowskie mózgu ze zbioru Kaggle, Alzheimer's Dataset (4 class of Images) . . . . .	16
4.2. Wykres ilustrujący liczbę zdjęć w każdej kategorii . . . . .	17
4.3. Struktura modelu referencyjnego sieci neuronowej . . . . .	19
4.4. Liczba parametrów modelu referencyjnego w kolejnych warstwach . . . . .	19
4.5. Dokładność modelu referencyjnego . . . . .	20
4.6. Strata modelu referencyjnego . . . . .	20
4.7. Macierz pomyłek modelu referencyjnego . . . . .	20
4.8. Dokładność modelu referencyjnego po normalizacji danych . . . . .	21
4.9. Strata modelu referencyjnego po normalizacji danych . . . . .	21
4.10. Macierz pomyłek modelu referencyjnego po normalizacji danych . . . . .	21
4.11. Wykres funkcji aktywacji: tangens hiperboliczny . . . . .	22
4.12. Dokładność klasyfikacji modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji tangens hiperboliczny . . . . .	23
4.13. Strata modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji tan- gens hiperboliczny . . . . .	23
4.14. Macierz pomyłek modelu . . . . .	23
4.15. Wykres funkcji aktywacji: ReLu . . . . .	24
4.16. Wykres funkcji aktywacji: Leaky ReLU dla zadanego parametru alpha . . . . .	24
4.17. Dokładność klasyfikacji modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji Leaky ReLu z parametrem alpha równym 0.1 . . . . .	25
4.18. Strata modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji Leaky ReLu z parametrem alpha równym 0.1 . . . . .	25
4.19. Macierz pomyłek modelu . . . . .	25
4.20. Dokładność klasyfikacji modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji Leaky ReLu z parametrem alpha równym 0.3 . . . . .	26
4.21. Strata modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji Leaky ReLu z parametrem alpha równym 0.3 . . . . .	26
4.22. Macierz pomyłek modelu . . . . .	26
4.23. Dokładność klasyfikacji modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji Leaky ReLu z parametrem alpha równym 0.5 . . . . .	27
4.24. Strata modelu referencyjnego po zastosowaniu jako funkcja aktywacji funkcji Leaky ReLu z parametrem alpha równym 0.5 . . . . .	27
4.25. Macierz pomyłek modelu . . . . .	27

4.26. Dokładność klasyfikacji modelu pierwszego na zbiorze treningowym oraz walidacyjnym . . . . .	28
4.27. Strata modelu pierwszego na zbiorze treningowym oraz walidacyjnym . . . . .	28
4.28. Macierz pomyłek modelu . . . . .	29
4.29. Dokładność klasyfikacji modelu drugiego na zbiorze treningowym oraz walidacyjnym	30
4.30. Strata modelu drugiego na zbiorze treningowym oraz walidacyjnym . . . . .	30
4.31. Macierz pomyłek modelu . . . . .	30
4.32. Dokładność klasyfikacji modelu trzeciego na zbiorze treningowym oraz walidacyjnym	31
4.33. Strata modelu trzeciego na zbiorze treningowym oraz walidacyjnym . . . . .	31
4.34. Macierz pomyłek modelu . . . . .	32
4.35. Dokładność klasyfikacji modelu czwartego na zbiorze treningowym oraz walidacyjnym . . . . .	33
4.36. Strata modelu czwartego na zbiorze treningowym oraz walidacyjnym . . . . .	33
4.37. Macierz pomyłek modelu . . . . .	33
4.38. Dokładność klasyfikacji modelu pierwszego na zbiorze treningowym oraz walidacyjnym . . . . .	34
4.39. Strata modelu pierwszego na zbiorze treningowym oraz walidacyjnym . . . . .	34
4.40. Macierz pomyłek modelu . . . . .	35
4.41. Dokładność klasyfikacji modelu drugiego na zbiorze treningowym oraz walidacyjnym	36
4.42. Strata modelu drugiego na zbiorze treningowym oraz walidacyjnym . . . . .	36
4.43. Macierz pomyłek modelu . . . . .	37
4.44. Dokładność klasyfikacji modelu trzeciego na zbiorze treningowym oraz walidacyjnym	38
4.45. Strata modelu trzeciego na zbiorze treningowym oraz walidacyjnym . . . . .	38
4.46. Macierz pomyłek modelu . . . . .	38
4.47. Losowe przekształcenia 3 zdjęć rentgenowskich mózgu ze zbioru Kaggle, Alzheimer's Dataset (4 class of Images) . . . . .	39
4.48. Wykres ilustrujący przyrost danych uczących po zastosowaniu augmentacji . . . . .	40
4.49. Struktura końcowego modelu sieci neuronowej . . . . .	41
4.50. Liczba parametrów końcowego modelu w kolejnych warstwach . . . . .	41
4.51. Dokładność klasyfikacji modelu . . . . .	42
4.52. Strata modelu . . . . .	42
4.53. Macierz pomyłek modelu . . . . .	43
5.1. Panel logowania . . . . .	45
5.2. Panel strony głównej . . . . .	46
5.3. Panel listy pacjentów . . . . .	47
5.4. Panel edycji danych pacjentów . . . . .	48
5.5. Panel historii badań pacjenta . . . . .	49
5.6. Panel badania pacjenta . . . . .	50
5.7. Okno wyboru zdjęcia skanu mózgu . . . . .	50
5.8. Panel z wynikiem badania . . . . .	51

# Spis tabel

4.1. Tabela wartości metryk klasyfikacji modelu referencyjnego . . . . .	20
4.2. Tabela wartości metryk klasyfikacji modelu referencyjnego dla znormalizowanych danych . . . . .	22
4.3. Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: tangens hiperboliczny	23
4.4. Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: Leaky ReLu z parametrem alpha wynoszącym 0.1 . . . . .	25
4.5. Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: Leaky ReLu z parametrem alpha wynoszącym 0.3 . . . . .	26
4.6. Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: Leaky ReLu z parametrem alpha wynoszącym 0.5 . . . . .	27
4.7. Tabela wartości metryk klasyfikacji modelu pierwszego . . . . .	28
4.8. Tabela wartości metryk klasyfikacji modelu drugiego . . . . .	30
4.9. Tabela wartości metryk klasyfikacji modelu trzeciego . . . . .	31
4.10. Tabela wartości metryk klasyfikacji modelu czwartego . . . . .	33
4.11. Tabela wartości metryk klasyfikacji modelu pierwszego . . . . .	35
4.12. Tabela wartości metryk klasyfikacji modelu drugiego . . . . .	36
4.13. Tabela wartości metryk klasyfikacji modelu trzeciego . . . . .	38
4.14. Tabela wartości metryk klasyfikacji modelu wykorzystującego wnioski z wcześniej przeprowadzonych badań . . . . .	42

# 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[11].

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[6].

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[8]. 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[16]. 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 niezależne, ale powiązane ze sobą bloki[9].

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[5].

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[10].

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[7].

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 elementy 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[14].

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 wchodzi 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:

User
id : int image : str login : str name : str password : str role : str surname : str

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:

Patient
exam_history : list[Exam] id : int image : str name : str surname : str

Rys. 2.2: Model pacjenta

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:

Exam
date : str id : int image : str result : str

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 rentgenowskiego skanu mózgu zapisanego na serwerze.

## Rozdział 3

# Sztuczne sieci neuronowe

Sztuczne sieci neuronowe stanowią obecnie jeden z najpopularniejszych algorytmów uczenia maszynowego[15]. 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[17]. Za wybranie odpowiedniej architektury sieci oraz dobór parametrów takich jak funkcja aktywacji, rozmiar filtrów czy ilość neuronów w warstwach ukrytych odpowiada programista.

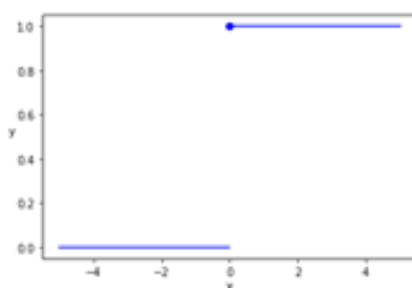
### 3.1. Początki sztucznych sieci neuronowych

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”[19] 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[12]. 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:

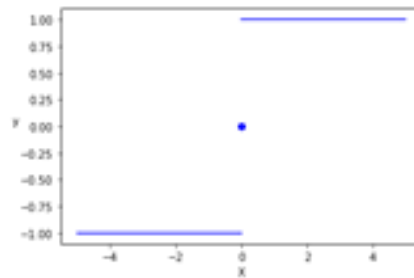
- Funkcja skokowa Heaviside’a:



Rys. 3.1: Funkcja skokowa Heaviside’a

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

- Funkcja signum:



Rys. 3.2: Funkcja signum

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

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[20].

## 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[13]. 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[21]. 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[17].

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 średnio-kwadratowego
- Błędu bezwzględnego
- Funkcji entropii 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[17].

Odkrycie warstw splotowych pozwoliło na stworzenie przez LeCuna w 1998 roku sieci LeNet-5[18]. 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

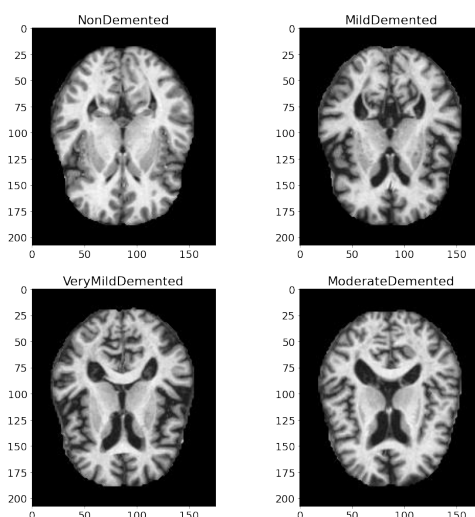
W celu wytrenowania modelu wykorzystano zdjęcia ze zbioru Alzheimer's Dataset (4 class of Images). Zbiór ten zawiera zdjęcia mózgu wykonane metodą rezonansu magnetycznego wraz z poprawnie przydzielonymi etykietami. Zdjęcia w zbiorze mają format 176 pikseli szerokości i 208 pikseli wysokości, oraz zawierają trzy składowe kolorów.

Zbiór ten podzielono w sposób losowy uzyskując w zbiorze uczącym 80% obrazów z każdej kategorii, natomiast pozostałym 20% obrazów przydzielono do zbioru testowego.

#### 4.1.1. Przegląd zdjęć

W skład zbioru wchodzi zdjęcia należące do czterech kategorii:

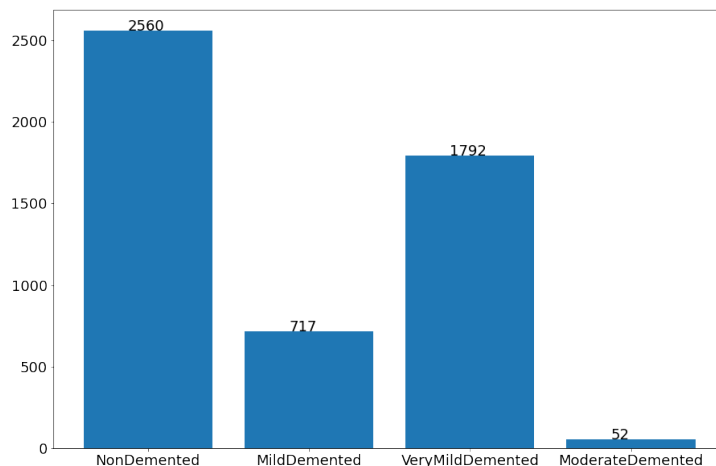
- Brak demencji
- Bardzo łagodna demencja
- Łagodna demencja
- Umiarkowana demencja



Rys. 4.1: Przykładowe zdjęcia rentgenowskie mózgu ze zbioru Kaggle, Alzheimer's Dataset (4 class of Images)



Poniższy wykres ilustruje liczbę zdjęć w każdej z kategorii:



Rys. 4.2: Wykres ilustrujący liczbę zdjęć w każdej kategorii

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

## 4.2. Opis badanych metryk klasyfikacji

Przy badaniu jakości klasyfikacji modelu istotne jest zbadanie nie tylko dokładności (ang. accuracy) klasyfikacji, ale także takich metryk jak:

- Odwołanie (ang. Recall)
- Precyzja (ang. Precision)
- Metryka F1

Dokonując klasyfikacji, możemy poprawnie zaklasyfikować obiekt do właściwej klasy, na przykład do klasy A. Możemy również zaklasyfikować obiekt który powinien być oznaczony jako klasa A, do jednej z niewłaściwych klas, na przykład do klasy B. Możliwe jest także niepoprawne zaklasyfikowanie obiektu klasy, na przykład C, do klasy A. Istnieje również możliwość poprawnego wskazania braku przynależności obiektu do klasy klasą A.

Wspomniana niewłaściwa klasyfikacja może świadczyć o błędnym funkcjonowaniu modelu. Dokładność klasyfikacji wskazuje nam jedynie jak dużo danych testowych zostało poprawnie zaklasyfikowanych. Nie jesteśmy jednak na tej podstawie wykryć na przykład pominięcia pewnej klasy, którą model uznał za zbyt mało prawdopodobną.

Wspomniane wcześniej wyniki klasyfikacji można przedstawić w tabeli zwanej macierzą pomyłek (ang. Confusion matrix). Wyróżnia się w niej następujące przypadki wyników:

- Wynik prawdziwie pozytywny (ang. True positive - w skrócie TP)
- Wynik fałszywie pozytywny (ang. False positive - w skrócie FP)
- Wynik fałszywie negatywny (ang. False negative - w skrócie FN)
- Wynik prawdziwie negatywny (ang. True negative - w skrócie TN)

Wartości otrzymanych wyników będą wykorzystywane do obliczania odpowiednich metryk klasyfikacji.

### 4.2.1. Precyzja (ang. Precision)

Miara precyzji określa ilość poprawnie wskazanych elementów klasy, w stosunku do wszystkich elementów wskazanych jako dana klasa. Obliczą się ją za pomocą wzoru:

$$Precision = \frac{TP}{TP + FP}$$

### 4.2.2. Odwołanie (ang. Recall)

Miara recall określa ilość poprawnie wskazanych elementów klasy, w stosunku do całkowitej liczby elementów w danej klasie. Obliczą się ją za pomocą wzoru:

$$Recall = \frac{TP}{TP + FN}$$

### 4.2.3. Metryka F1

Miara F1 jest to miara bazująca na wartościach miar recall i precyzji. Jest ona pomocna w znalezieniu równowagi między wcześniej wspomnianymi miarami. Oblicza się ją za pomocą wzoru:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

## 4.3. Model referencyjny

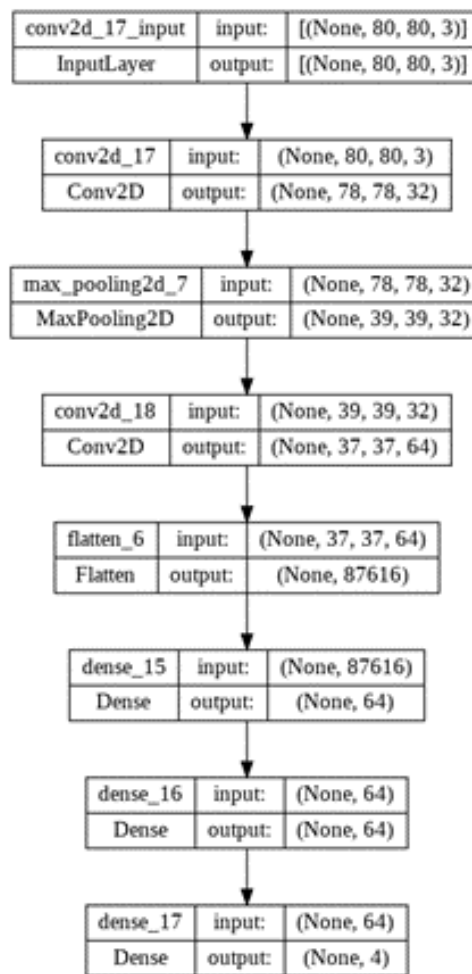
W celu sprawdzenia wpływu różnych parametrów na dokładność klasyfikacji przyjęto model referencyjny składający się z 8 warstw, w tym z:

- 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 entropii 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:



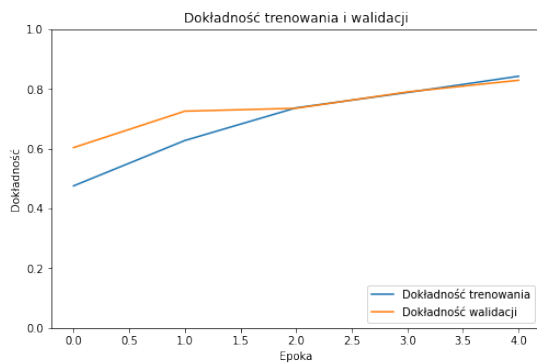
Rys. 4.3: Struktura modelu referencyjnego sieci neuronowej

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

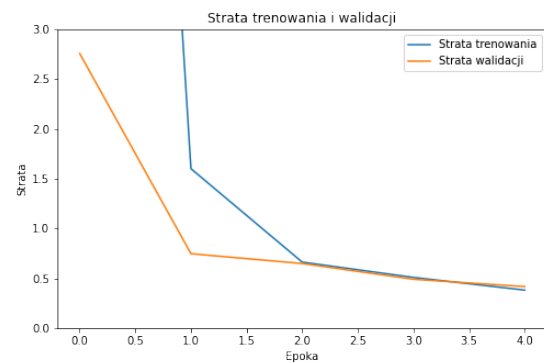
Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 78, 78, 32)	896
max_pooling2d_2 (MaxPooling 2D)	(None, 39, 39, 32)	0
conv2d_5 (Conv2D)	(None, 37, 37, 64)	18496
flatten_2 (Flatten)	(None, 87616)	0
dense_6 (Dense)	(None, 64)	5607488
dense_7 (Dense)	(None, 64)	4160
dense_8 (Dense)	(None, 4)	260
=====		
Total params: 5,631,300		
Trainable params: 5,631,300		
Non-trainable params: 0		

Rys. 4.4: Liczba parametrów modelu referencyjnego w kolejnych warstwach

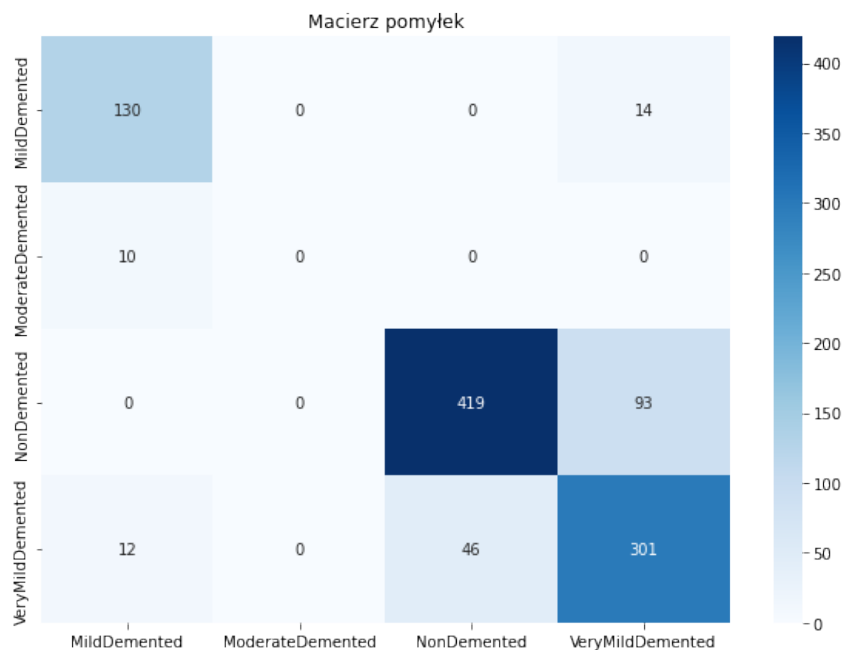
Dokładność klasyfikacji jaką jest w stanie osiągnąć model referencyjny wynosi 82.93% dla danych walidacyjnych. Pozostałe metryki klasyfikacji zostały zawarte w tabeli 4.1. Proces uczenia modelu został pokazany na poniższym zdjęciu 4.3.



Rys. 4.5: Dokładność modelu referencyjnego



Rys. 4.6: Strata modelu referencyjnego



Rys. 4.7: Macierz pomyłek modelu referencyjnego

Tab. 4.1: Tabela wartości metryk klasyfikacji modelu referencyjnego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.9028	0	0.818	0.838
Precision	0.856	0	0.901	0.738
F1	0.878	0	0.856	0.785

## 4.4. Przeprowadzone badania

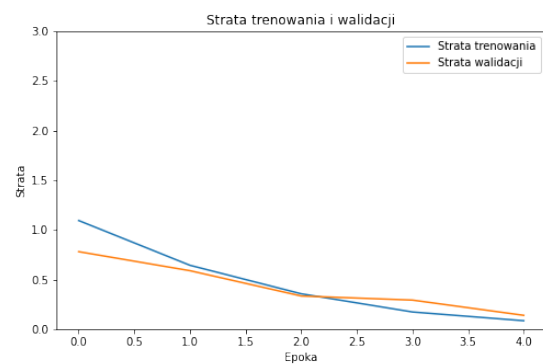
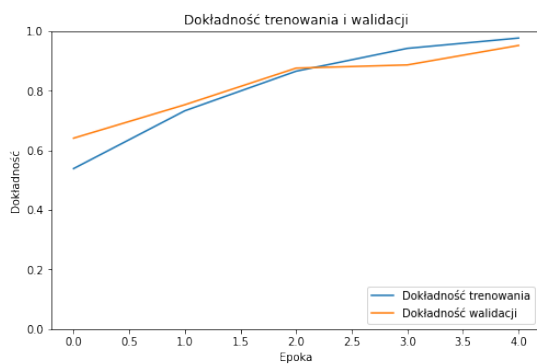
### 4.4.1. Badanie wpływu normalizacji min-max na dokładność klasyfikacji

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

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

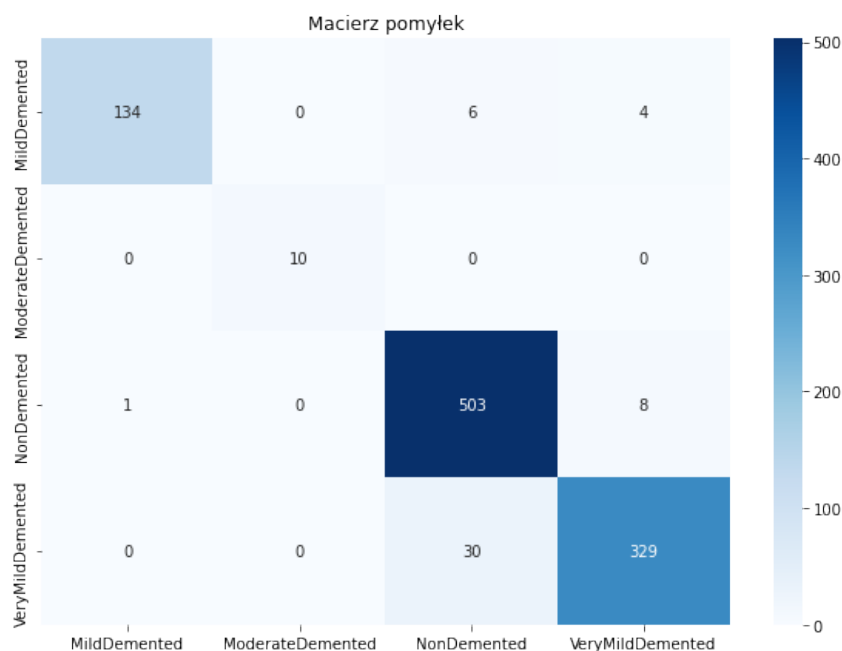
W przypadku operowania na danych będących obrazami należącymi do wczytanego zbioru, maksymalna wartość  $X$  jaką jesteśmy w stanie uzyskać, to wartość 255, natomiast najmniejsza wartość to 0. Oznacza to, że w celu przeprowadzenia procesu normalizacji, wystarczy podzielić wszystkie wartości pikseli przez 255.

W wyniku normalizacji danych metodą min-max oraz wyuczeniu tak zmodyfikowanymi danymi model referencyjny, byliśmy w stanie uzyskać 95.22% dokładności klasyfikacji.



Rys. 4.8: Dokładność modelu referencyjnego po normalizacji danych

Rys. 4.9: Strata modelu referencyjnego po normalizacji danych



Rys. 4.10: Macierz pomyłek modelu referencyjnego po normalizacji danych

Tab. 4.2: Tabela wartości metryk klasyfikacji modelu referencyjnego dla znormalizowanych danych

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.931	1.0	0.982	0.916
Precision	0.993	1.0	0.957	0.940
F1	0.961	1.0	0.957	0.940

#### 4.4.2. 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.

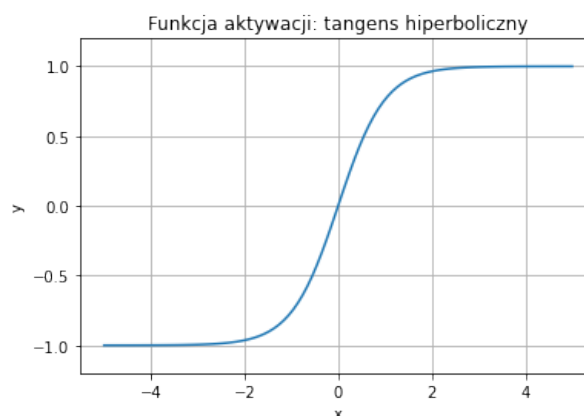
Badania przeprowadzono dla następujących funkcji aktywacji:

- 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

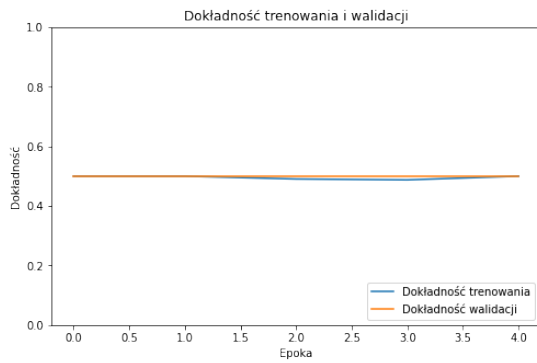
Badania odbyły się na znormalizowanych danych, natomiast zmianie ulegały funkcje aktywacji warstw gęstych za wyjątkiem ostatniej warstwy, wykorzystującej funkcję softmax.

##### Tangens hiperboliczny

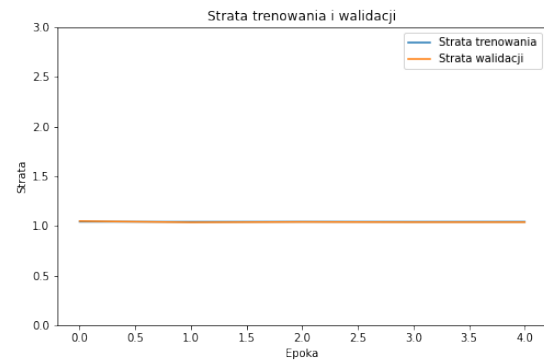
Tangens hiperboliczny jest to różniczkowalna funkcja ciągła, s-kształtna, której zbiór wartości mieści się w przedziale  $(-1, 1)$ .



Rys. 4.11: Wykres funkcji aktywacji: tangens hiperboliczny



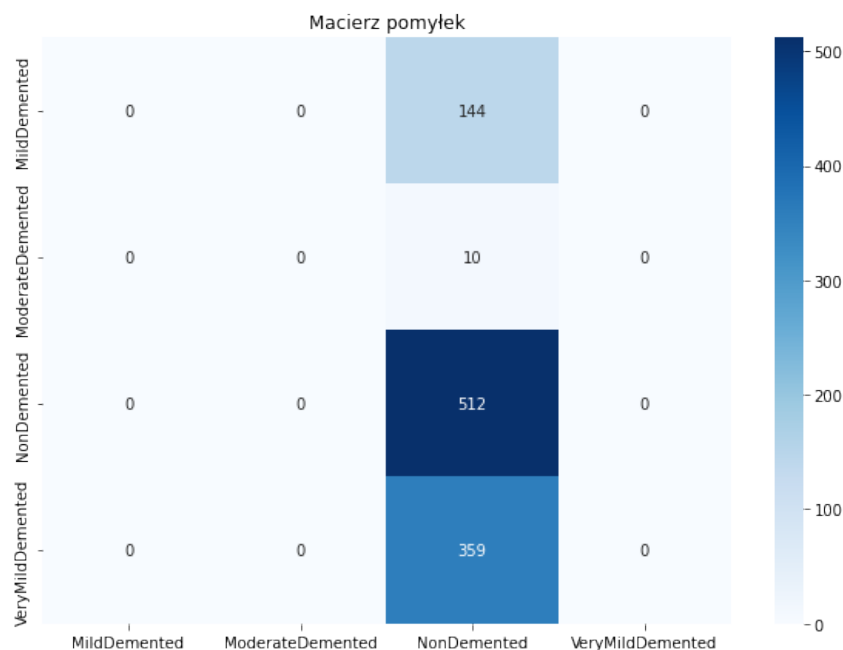
Rys. 4.12: Dokładność klasyfikacji modelu referen-  
cyjnego po zastosowaniu jako funkcja aktywacji  
funkcji tangens hiperboliczny



Rys. 4.13: Strata modelu referencyjnego po zastosowa-  
niu jako funkcja aktywacji funkcji tangens hiper-  
boliczny

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.



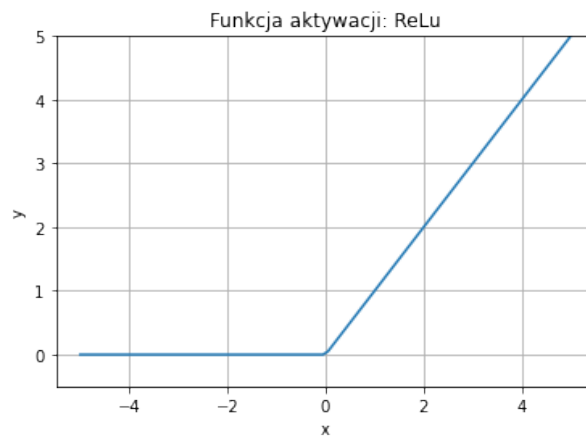
Rys. 4.14: Macierz pomyłek modelu

Tab. 4.3: Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: tangens hiperboliczny

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.0	0.0	1.0	0.0
Precision	0.0	0.0	0.5	0.0
F1	0.0	0.0	0.666	0.0

## 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.

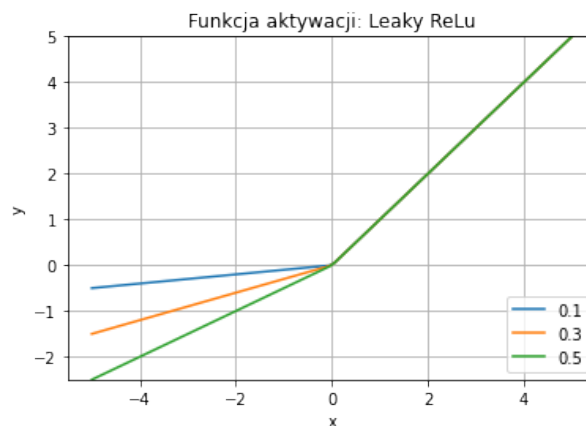


Rys. 4.15: Wykres funkcji aktywacji: ReLu

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:

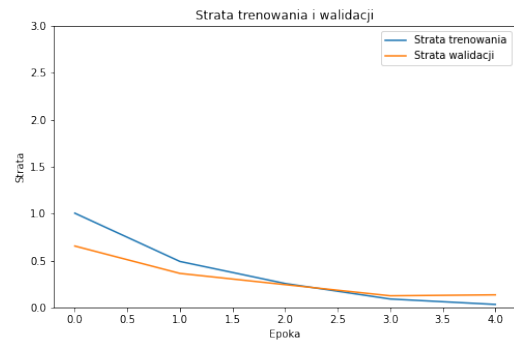
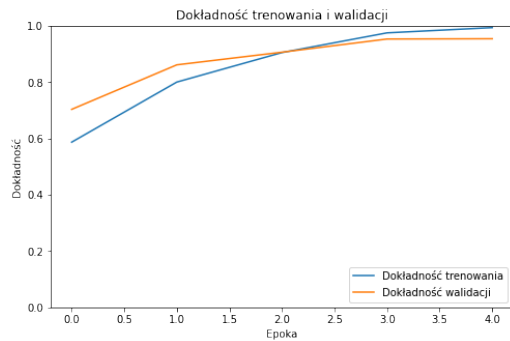
- 0.1
- 0.3
- 0.5



Rys. 4.16: Wykres funkcji aktywacji: Leaky ReLU dla zadanego parametru alpha

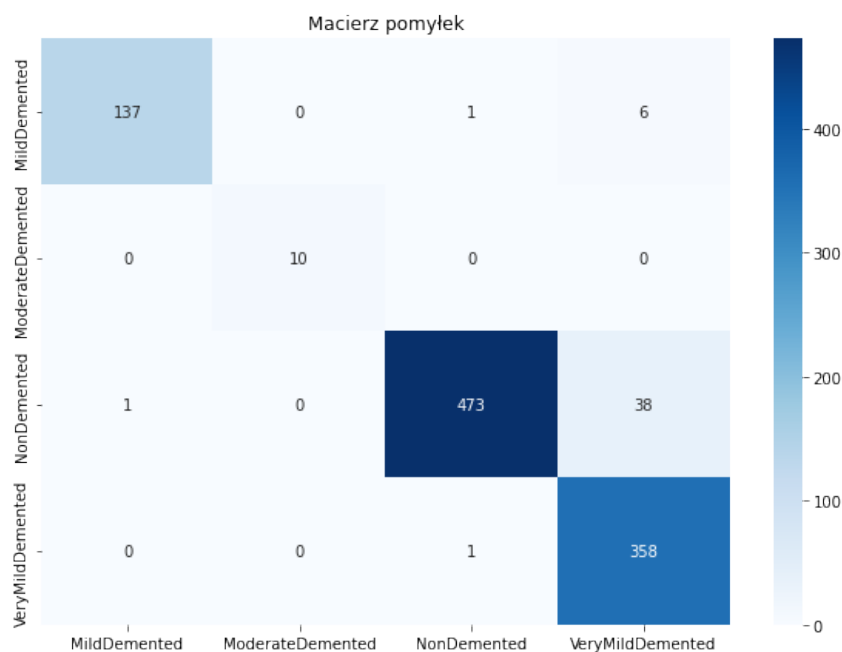


### Leaky ReLu dla parametru alpha równego 0.1



Rys. 4.17: Dokładność klasyfikacji modelu referencyjnego po zastosowaniu funkcji Leaky ReLu z parametrem alpha równym 0.1  
 Rys. 4.18: Strata modelu referencyjnego po zastosowaniu funkcji Leaky ReLu z parametrem alpha równym 0.1

Wykorzystując jako funkcję aktywacji funkcję Leaky ReLu z parametrem alpha równym 0.1 osiągnięto dokładność klasyfikacji, dla znormalizowanych danych, na poziomie 95.42%.

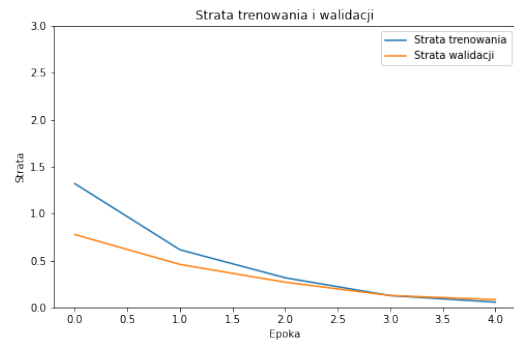
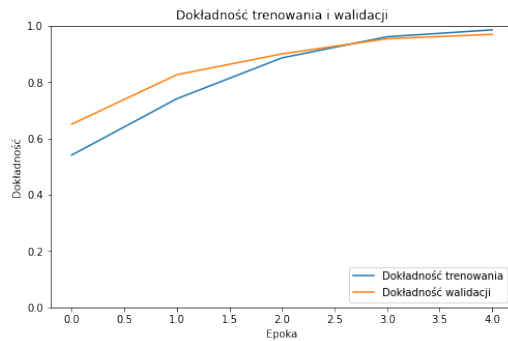


Rys. 4.19: Macierz pomyłek modelu

Tab. 4.4: Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: Leaky ReLu z parametrem alpha wynoszącym 0.1

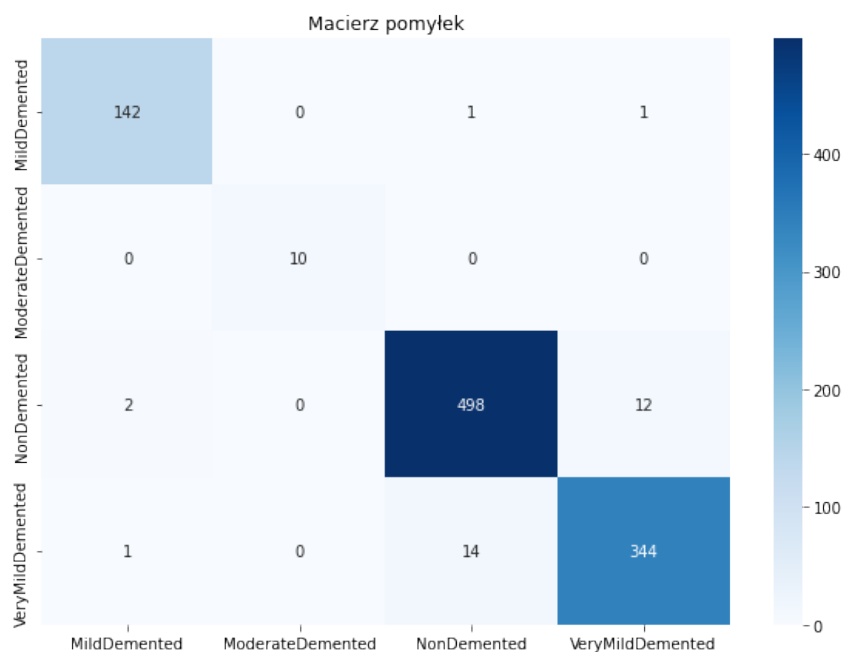
	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.951	1.0	0.924	0.997
Precision	0.993	1.0	0.996	0.891
F1	0.972	1.0	0.958	0.941

### Leaky ReLu dla parametru alpha równego 0.3



Rys. 4.20: Dokładność klasyfikacji modelu referencyjnego po zastosowaniu funkcji Leaky ReLu z parametrem alpha równym 0.3

Wykorzystując jako funkcję aktywacji funkcję Leaky ReLu z parametrem alpha równym 0.3 osiągnięto dokładność klasyfikacji, dla znormalizowanych danych, na poziomie 96.98%.

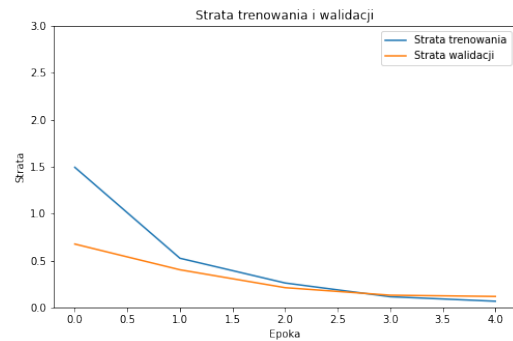
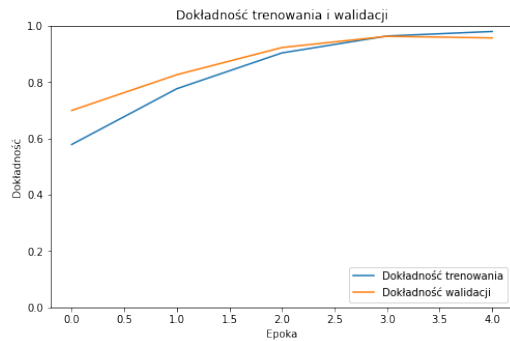


Rys. 4.22: Macierz pomyłek modelu

Tab. 4.5: Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: Leaky ReLu z parametrem alpha wynoszącym 0.3

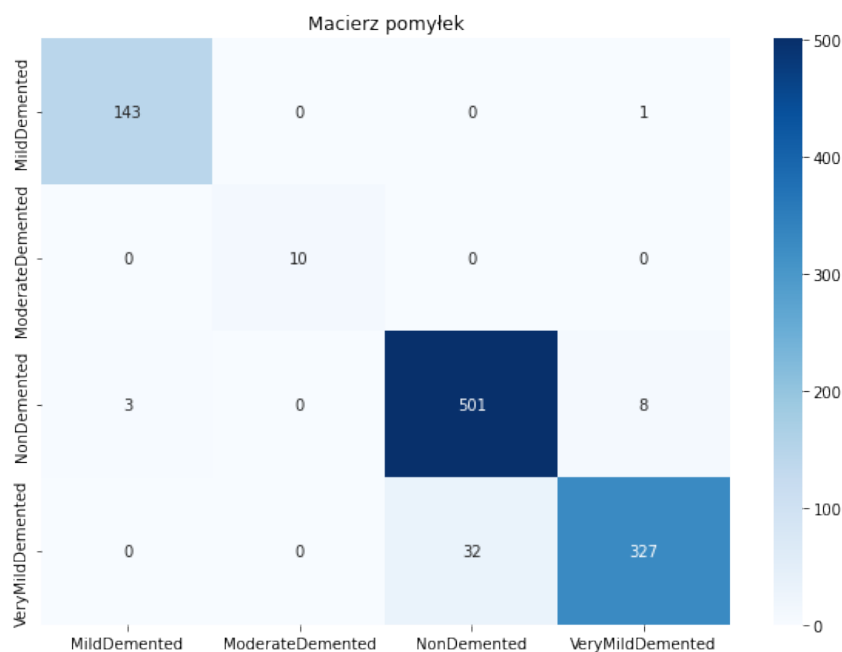
	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.986	1.0	0.973	0.958
Precision	0.979	1.0	0.971	0.964
F1	0.983	1.0	0.972	0.961

### Leaky ReLu dla parametru alpha równego 0.5



Rys. 4.23: Dokładność klasyfikacji modelu referencyjnego po zastosowaniu funkcji Leaky ReLu z parametrem alpha równym 0.5  
Rys. 4.24: Strata modelu referencyjnego po zastosowaniu funkcji Leaky ReLu z parametrem alpha równym 0.5

Wykorzystując jako funkcję aktywacji funkcję Leaky ReLu z parametrem alpha równym 0.5 osiągnięto dokładność klasyfikacji, dla znormalizowanych danych, na poziomie 95.7%.



Rys. 4.25: Macierz pomyłek modelu

Tab. 4.6: Tabela wartości metryk klasyfikacji modelu z funkcją aktywacji: Leaky ReLu z parametrem alpha wynoszącym 0.5

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.993	1.0	0.979	0.911
Precision	0.979	1.0	0.94	0.973
F1	0.986	1.0	0.959	0.941

### 4.4.3. Badanie wpływu liczby warstw oraz ilości neuronów w warstwach gęstych

Badanie przeprowadzono dla następujących modeli sieci:

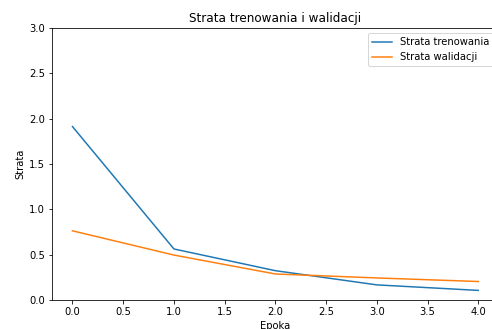
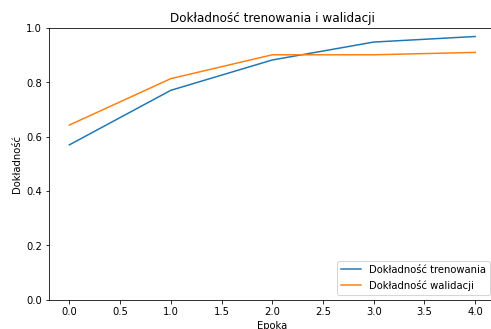
#### Model 1

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 spłotowej składającej się z 32 filtrów z oknem spłotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy spłotowej składającej się z 64 filtrów z oknem spłotu 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 entropii 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

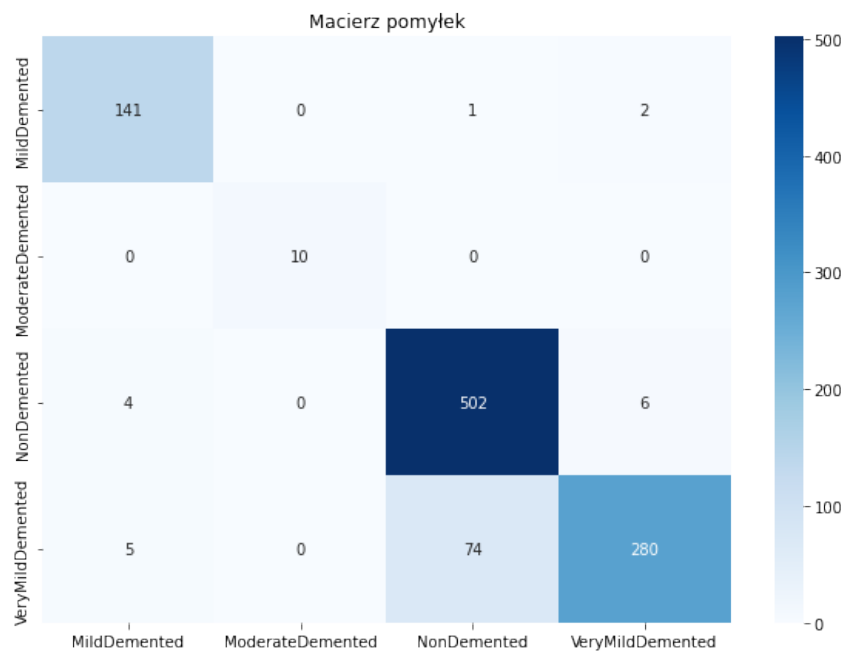


Rys. 4.26: Dokładność klasyfikacji modelu pierwszego na zbiorze treningowym oraz walidacyjnym Rys. 4.27: Strata modelu pierwszego na zbiorze treningowym oraz walidacyjnym

Model 1 uzyskał dokładność klasyfikacji wynoszącą 91.024%

Tab. 4.7: Tabela wartości metryk klasyfikacji modelu pierwszego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.979	1.0	0.98	0.78
Precision	0.94	1.0	0.87	0.972
F1	0.959	1.0	0.922	0.866



Rys. 4.28: Macierz pomyłek modelu

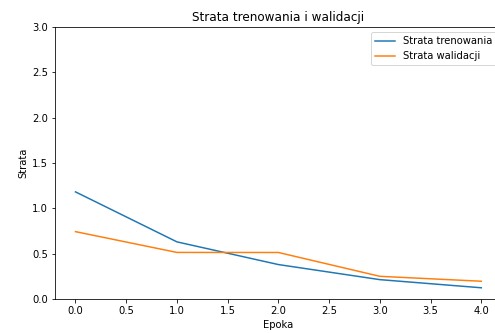
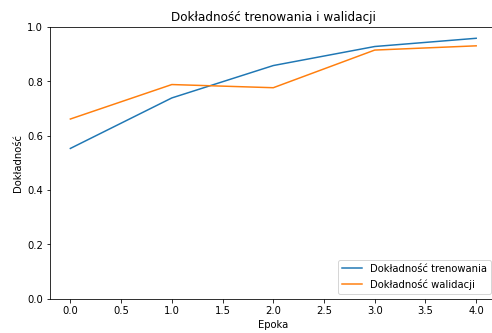
## Model 2

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

- Warstwy wejściową, przyjmującą obraz RGB o wymiarach 80x80
- Warstwy spłotowej składającej się z 32 filtrów z oknem spłotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy spłotowej składającej się z 64 filtrów z oknem spłotu 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

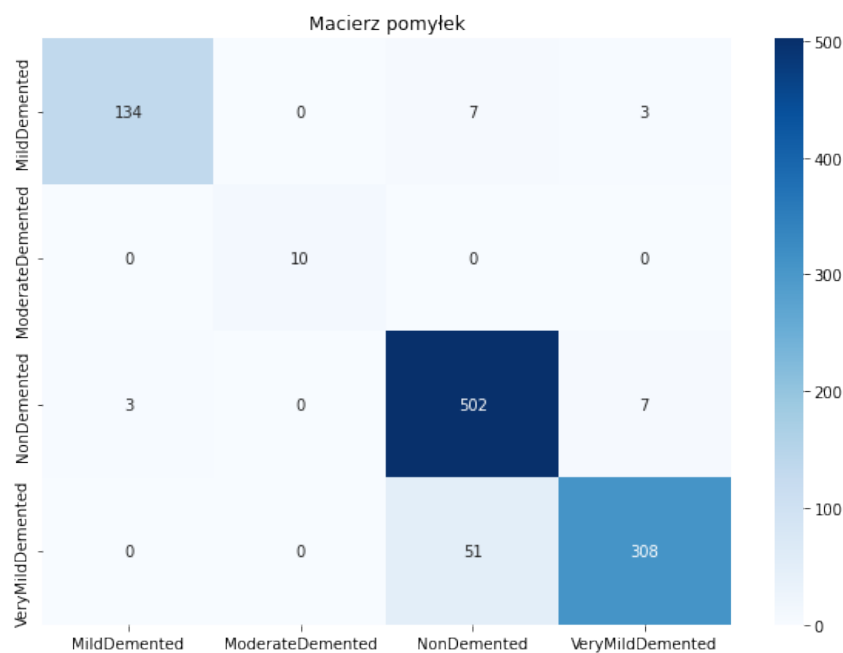
Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropii 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. 4.29: Dokładność klasyfikacji modelu drugiego na zbiorze treningowym oraz walidacyjnym Rys. 4.30: Strata modelu drugiego na zbiorze treningowym oraz walidacyjnym

Model 2 uzyskał dokładność klasyfikacji wynoszącą 93.07%



Rys. 4.31: Macierz pomyłek modelu

Tab. 4.8: Tabela wartości metryk klasyfikacji modelu drugiego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.931	1.0	0.98	0.858
Precision	0.978	1.0	0.896	0.969
F1	0.954	1.0	0.937	0.91

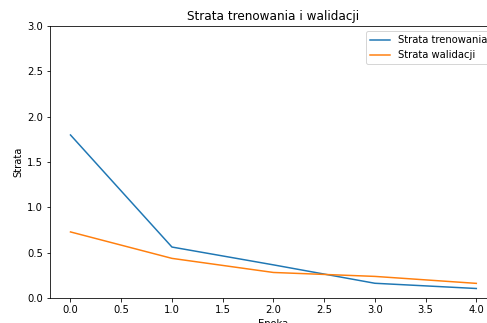
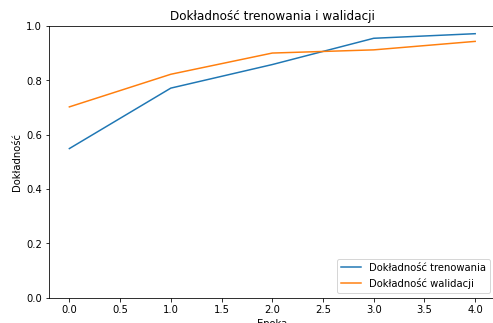
### Model 3

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



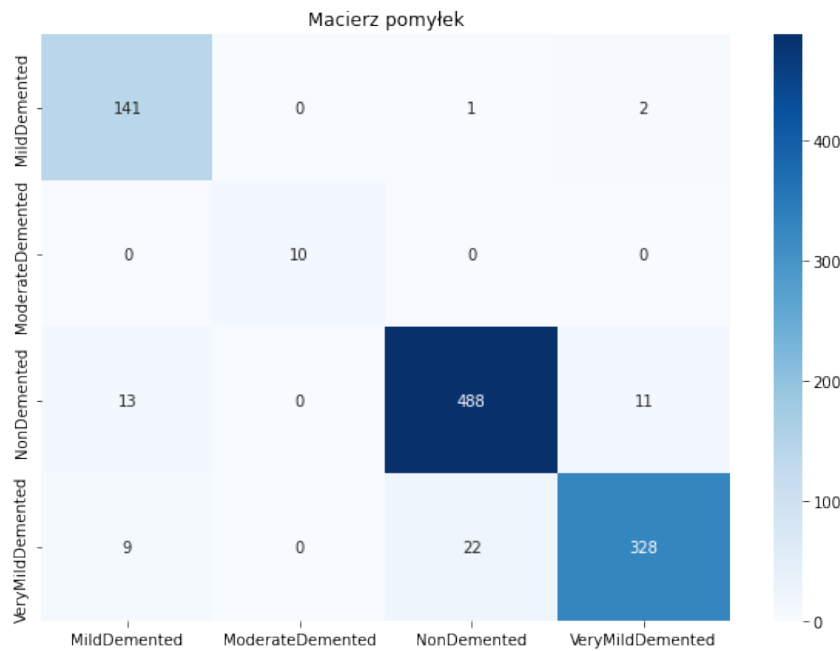
Rys. 4.32: Dokładność klasyfikacji modelu trzeciego na zbiorze treningowym oraz walidacyjnym

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

Model 3 uzyskał dokładność klasyfikacji wynoszącą 94.34%

Tab. 4.9: Tabela wartości metryk klasyfikacji modelu trzeciego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.979	1.0	0.953	0.914
Precision	0.865	1.0	0.955	0.962
F1	0.919	1.0	0.954	0.937



Rys. 4.34: Macierz pomyłek modelu

## Model 4

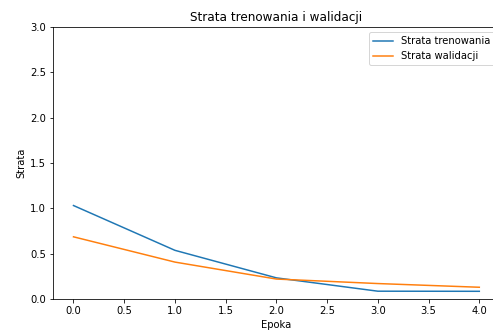
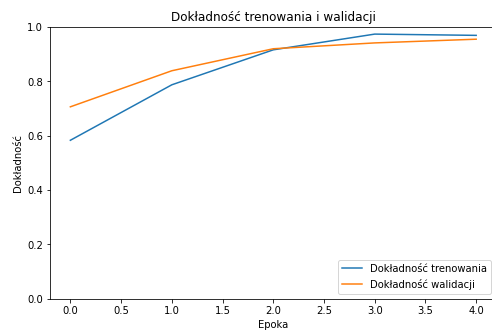
W modelu czwartym dodano jedną warstwę gęstą składającą się z 16 neuronów, pozostałe warstwy pozostawiono bez zmian w stosunku do modelu referencyjnego. W szczególności czwarty model składał się z:

- 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:

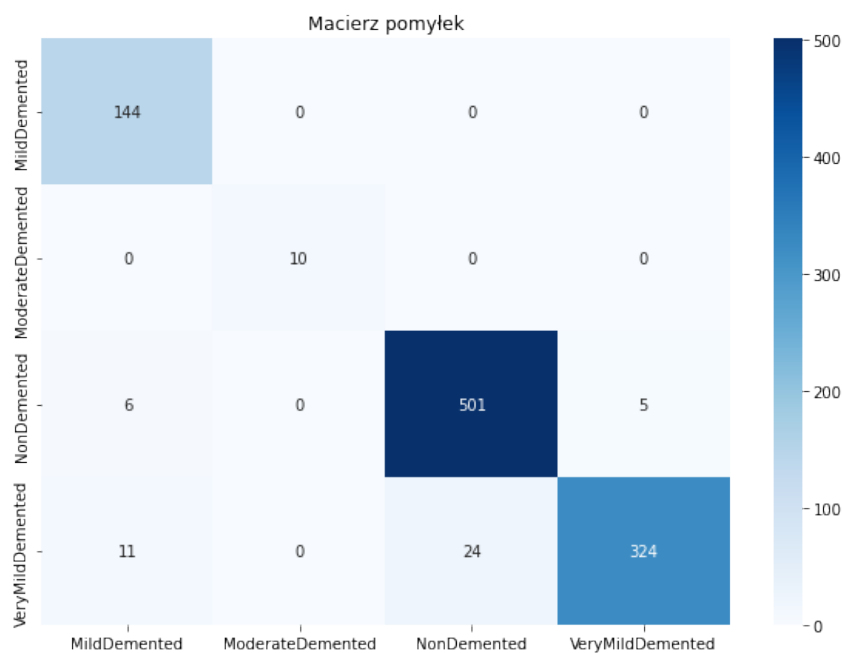
- Za funkcję straty, funkcję entropi krzyżowej (ang. Sparse Categorical Crossentropy)
- Minimalizowano ją przy pomocy optymalizatora Adam (ang. Adaptive moment estimation)
- Parametr reprezentujący szybkość uczenia się (ang. learning\_rate) pozostawiono na domyślnej wartości wynoszącej 0.001
- Przyjęto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





Rys. 4.35: Dokładność klasyfikacji modelu czwartego na zbiorze treningowym oraz walidacyjnym Rys. 4.36: Strata modelu czwartego na zbiorze treningowym oraz walidacyjnym

Model 4 uzyskał dokładność klasyfikacji wynoszącą 95.51%



Rys. 4.37: Macierz pomyłek modelu

Tab. 4.10: Tabela wartości metryk klasyfikacji modelu czwartego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	1.0	1.0	0.979	0.903
Precision	0.894	1.0	0.954	0.985
F1	0.944	1.0	0.966	0.94

#### 4.4.4. Badanie wpływu liczby warstw konwolucyjnych oraz liczby filtrów

Badanie przeprowadzono dla następujących modeli sieci.

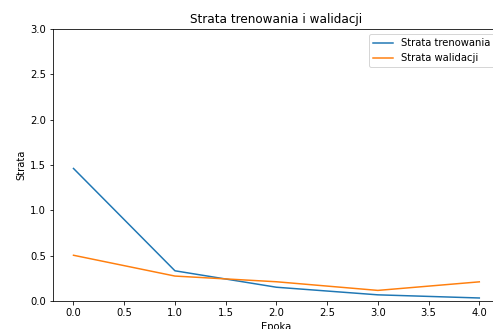
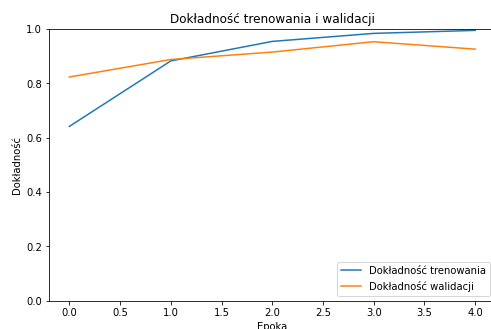
##### Model 1

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

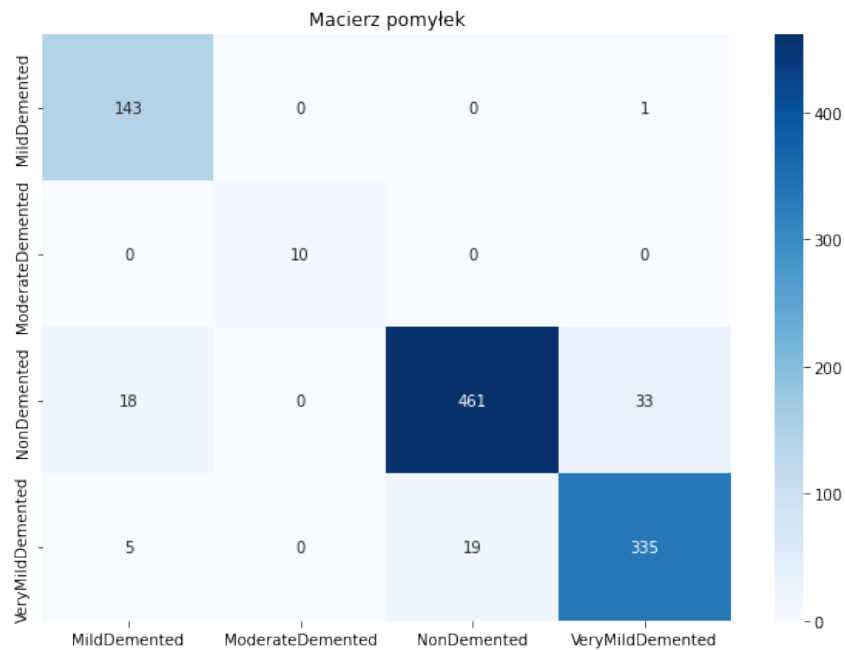
Pozostałe parametry zostały niezmienione i przyjmowały odpowiednio:

- Za funkcję straty, funkcję entropii 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. 4.38: Dokładność klasyfikacji modelu pierwszego na zbiorze treningowym oraz walidacyjnym Rys. 4.39: Strata modelu pierwszego na zbiorze treningowym oraz walidacyjnym

Model 1 uzyskał dokładność klasyfikacji wynoszącą 92.59%



Rys. 4.40: Macierz pomyłek modelu

Tab. 4.11: Tabela wartości metryk klasyfikacji modelu pierwszego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.993	1.0	0.9	0.933
Precision	0.861	1.0	0.96	0.908
F1	0.923	1.0	0.929	0.92

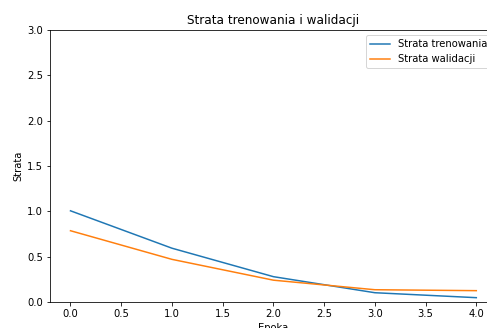
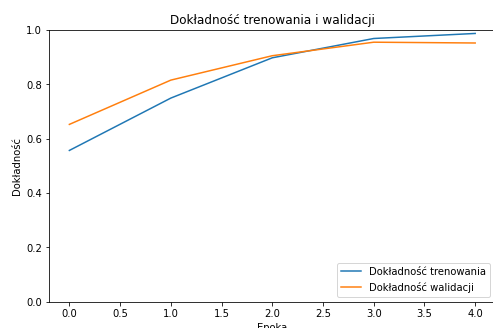
## Model 2

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 entropii 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



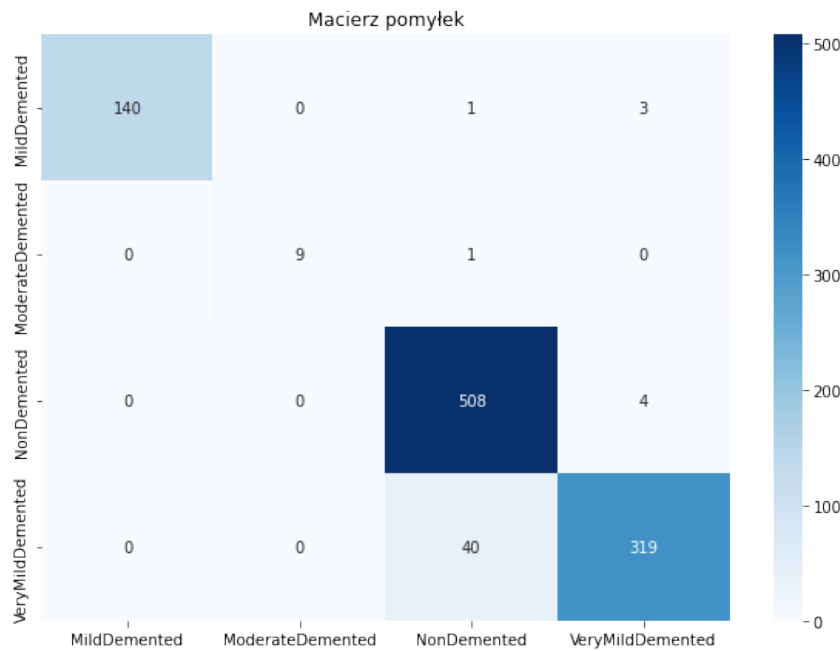
Rys. 4.41: Dokładność klasyfikacji modelu drugiego na zbiorze treningowym oraz walidacyjnym

Rys. 4.42: Strata modelu drugiego na zbiorze treningowym oraz walidacyjnym

Model 2 uzyskał dokładność klasyfikacji wynoszącą 95.22%

Tab. 4.12: Tabela wartości metryk klasyfikacji modelu drugiego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.972	0.9	0.992	0.889
Precision	1.0	1.0	0.924	0.979
F1	0.986	0.947	0.957	0.931



Rys. 4.43: Macierz pomyłek modelu

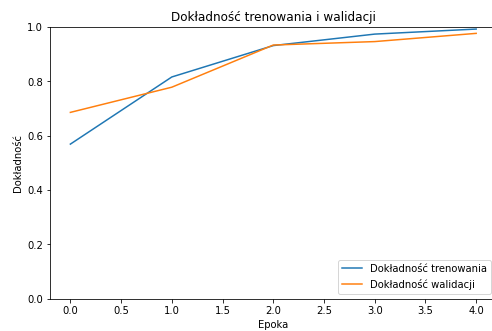
### Model 3

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

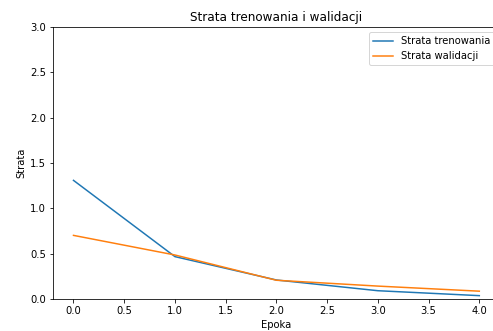
- Warstwy wejściowej, przyjmującej obraz RGB o wymiarach 80x80
- Warstwy spłotowej składającej się z 32 filtrów z oknem spłotu o wymiarach 3x3
- Warstwy łączącej, zwracającej maksymalną wartość w oknie (max-pooling), z oknem o wymiarach 2x2
- Warstwy spłotowej składającej się z 128 filtrów z oknem spłotu 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 entropii 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

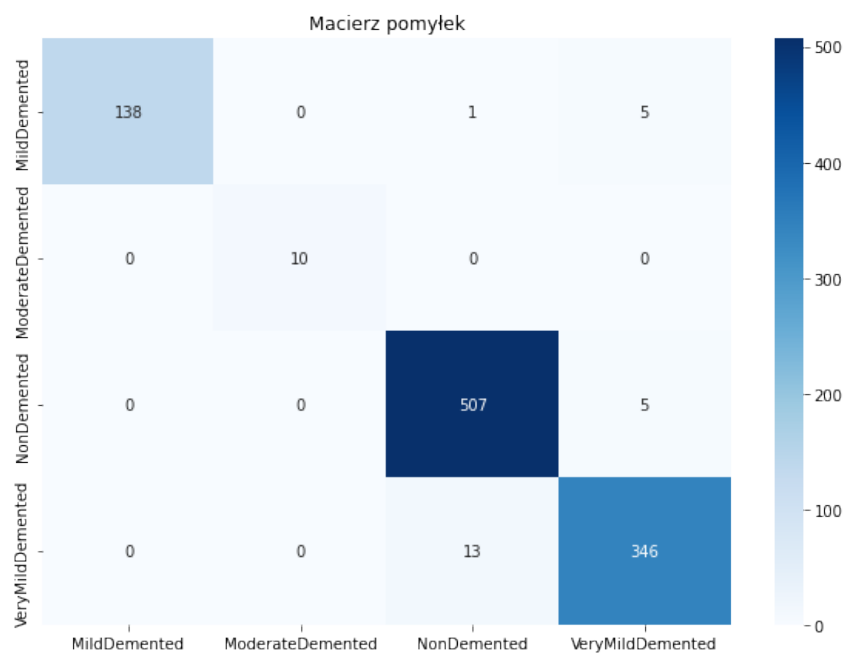


Rys. 4.44: Dokładność klasyfikacji modelu trzeciego na zbiorze treningowym oraz walidacyjnym



Rys. 4.45: Strata modelu trzeciego na zbiorze treningowym oraz walidacyjnym

Model 3 uzyskał dokładność klasyfikacji wynoszącą 97.66%



Rys. 4.46: Macierz pomyłek modelu

Tab. 4.13: Tabela wartości metryk klasyfikacji modelu trzeciego

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.958	1.0	0.99	0.964
Precision	1.0	1.0	0.973	0.972
F1	0.979	1.0	0.982	0.968

#### 4.4.5. Badanie wpływu augmentacji danych

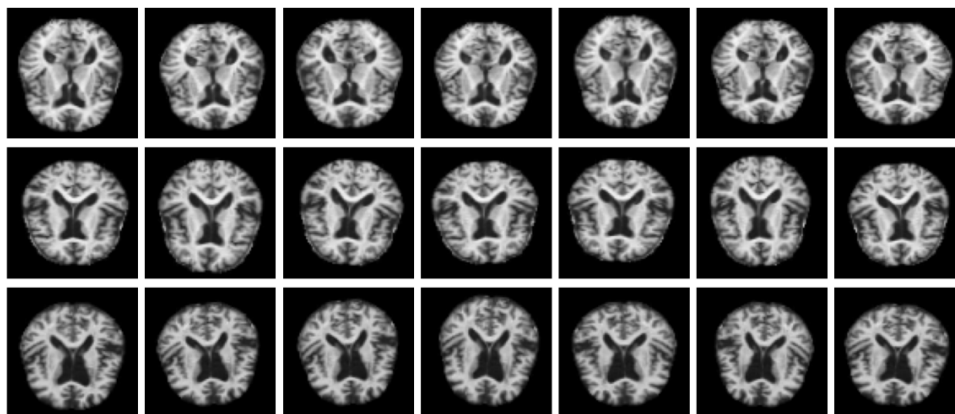
Augmentacja danych polega na powiększeniu zbioru danych uczących poprzez niewielkie modyfikacje oryginalnego zbioru. Zyskujemy w ten sposób zbiór podobnych do siebie zdjęć, na podstawie których model jest w stanie nauczyć się ogólnych cech.

Powiększenie zbioru danych jest pomocne w przypadku w którym dysponujemy niewystarczającą ilością danych. Model po zwiększeniu liczby epok uczących może zbyt dokładnie 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 modyfikacjom.

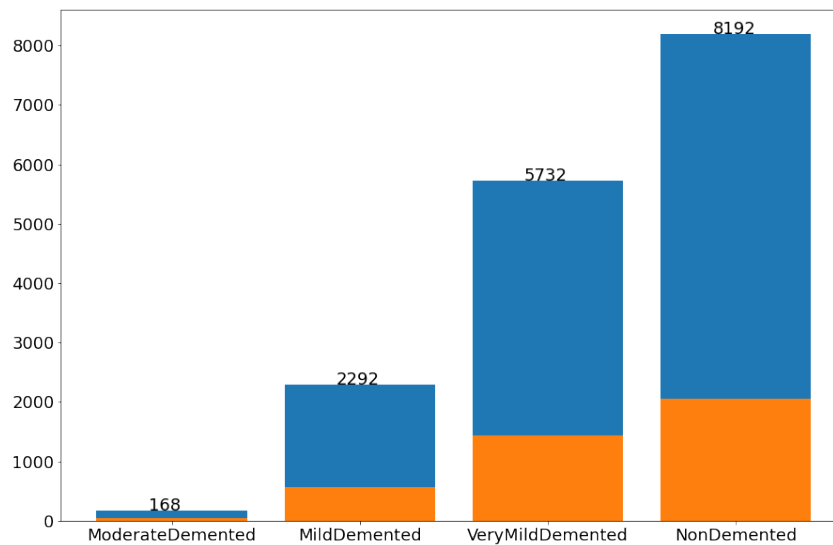
Wykorzystano takie modyfikacje obrazu jak:

- 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. 4.47: Losowe przekształcenia 3 zdjęć rentgenowskich mózgu ze zbioru Kaggle, Alzheimer's Dataset (4 class of Images)

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



Rys. 4.48: Wykres ilustrujący przyrost danych uczących po zastosowaniu augmentacji

Na powyższym wykresie 4.4.5 kolorem pomarańczowym oznaczono wielkości zbiorów przed augmentacją, natomiast kolorem niebieskim oznaczono wielkości zbiorów po augmentacji.

Sprawdzenie dokładności klasyfikacji z wykorzystaniem poszerzonego zbioru danych uczących odbyło się z wykorzystaniem wniosków z wcześniej przeprowadzonych badań.

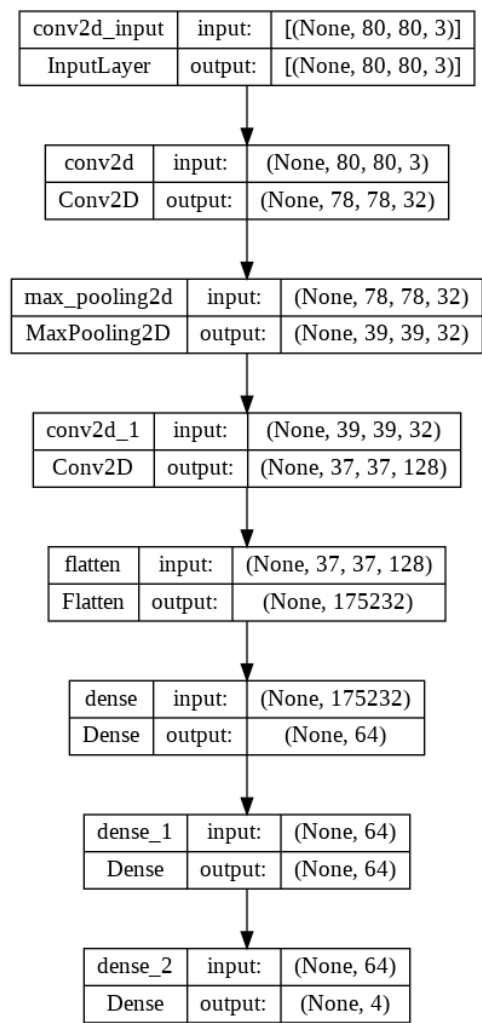
Między innymi, wykorzystano:

- znormalizowane dane
- funkcję aktywacji Leaky ReLu z parametrem alpha równym 0.3
- dwie warstwy konwolucyjne o liczbie filtrów odpowiednio 32 w pierwszej 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





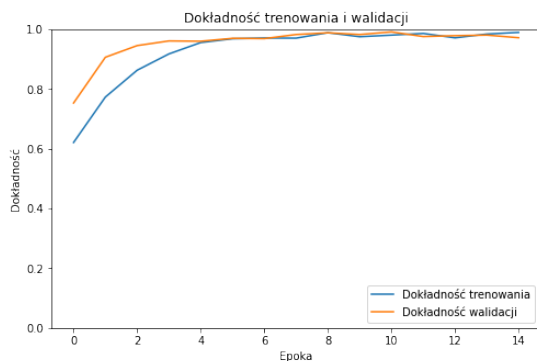
Rys. 4.49: Struktura końcowego modelu sieci neuronowej

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		

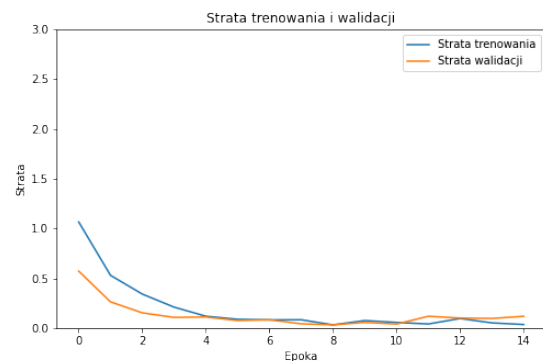
Rys. 4.50: Liczba parametrów końcowego modelu w kolejnych warstwach

Ponadto przyjęto:

- Za funkcję straty przyjęto funkcję entropii 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. 4.51: Dokładność klasyfikacji modelu



Rys. 4.52: Strata modelu

Dzięki powiększeniu zbioru danych uczących możliwe było zwiększenie liczby epok, co nie spowodowało przeuczenia modelu.

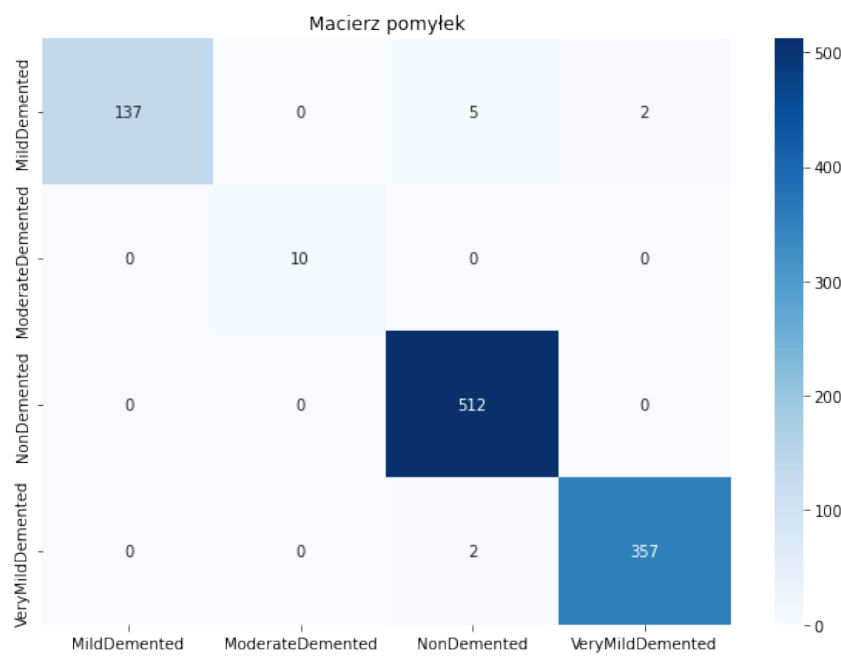
Ostatecznie uzyskany model osiągnął 99.12% dokładności klasyfikacji na zbiorze walidacyjnym.

Tab. 4.14: Tabela wartości metryk klasyfikacji modelu wykorzystującego wnioski z wcześniej przeprowadzonych badań

	Łagodna demencja	Umiarkowana demencja	Brak demencji	Bardzo łagodna demencja
Recall	0.951	1.0	1.0	0.994
Precision	1.0	1.0	0.987	0.994
F1	0.975	1.0	0.993	0.994

Wynik ten jest znacznie lepszy od wyników wcześniejszych modeli.

Model ten następnie wyeksportowano i wykorzystano w serwisie internetowym.



Rys. 4.53: Macierz pomyłek modelu

## 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.

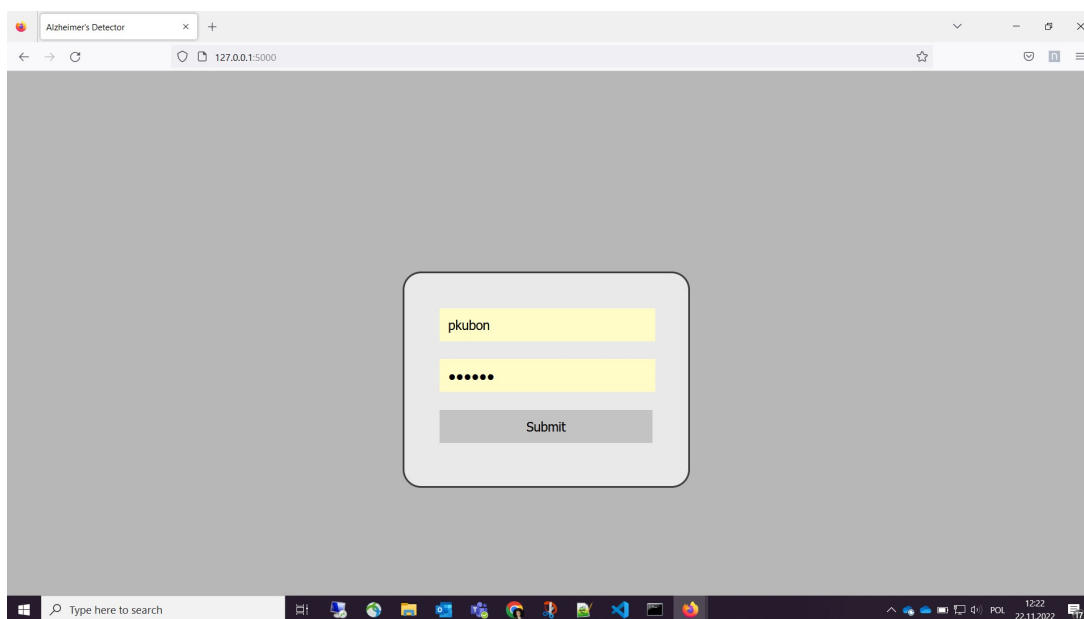
Utworzono również folder models w którym umieszczono moduły z modelami obiektów które wykorzystano w aplikacji do reprezentacji danych, jak również folder db w którym umieszczono pliki formatu json w których zapisywano dane. Przesłane zdjęcia natomiast zostaną zapisane w folderze uploads.

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 walidację przekazywanych danych.



Rys. 5.1: Panel logowania

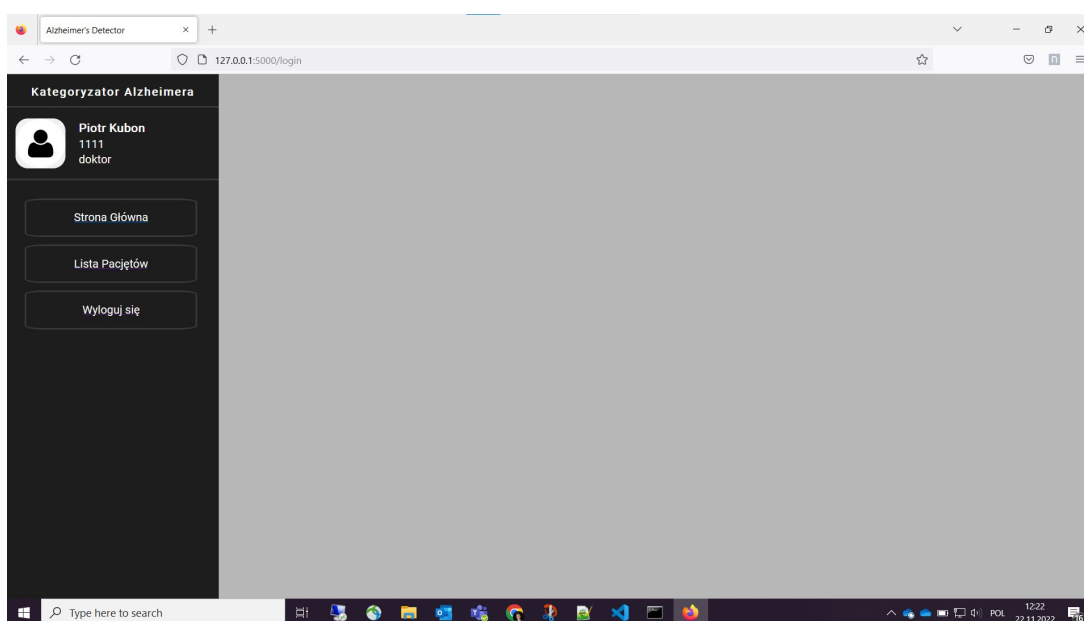
### 5.1.2. Panel strony głównej

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 możliwości:

- 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

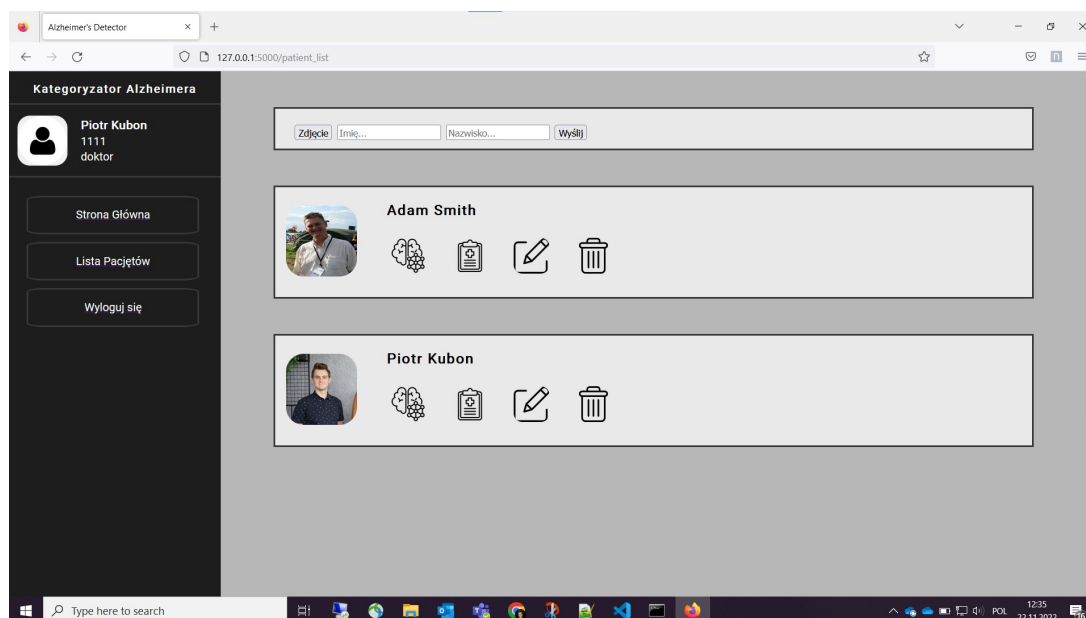
Każda funkcjonalność zawiera w serwisie odpowiadający za jej przeprowadzenie kontroler, do którego przekazywane jest id pacjenta.

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.

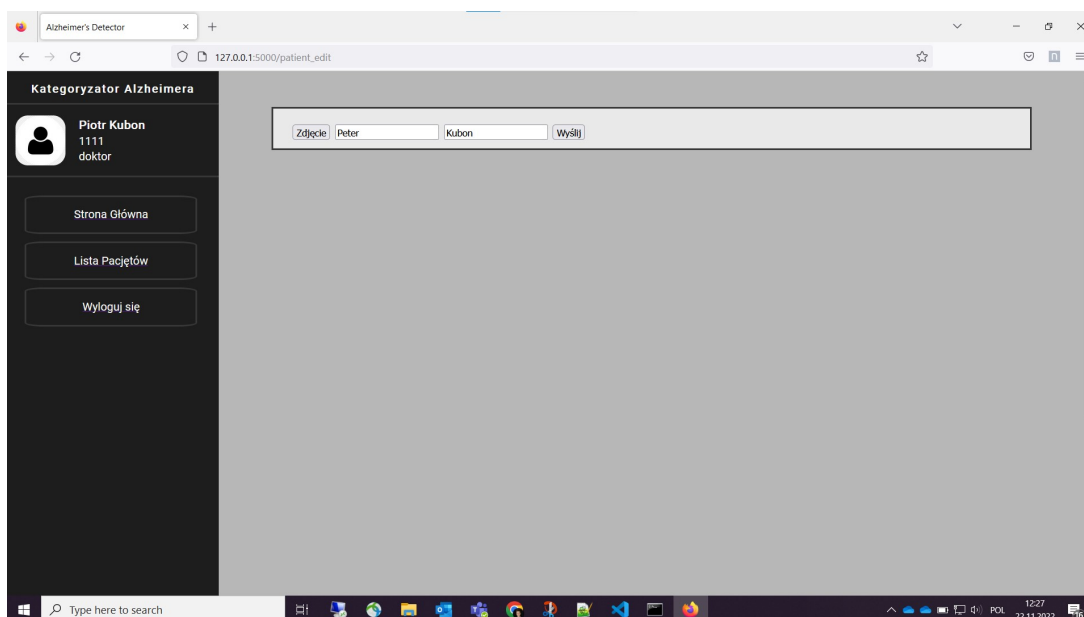
Serwis zapewnia możliwość zostawienia pustego pola danych. W przypadku nie wybrania zdjęcia pacjenta, następuje przydzielenie domyślnego zdjęcia do modelu, natomiast w przypadku nieprzesłania danych o imieniu lub nazwisku, odpowiednie pole nie będzie zawierać żadnej informacji.



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.

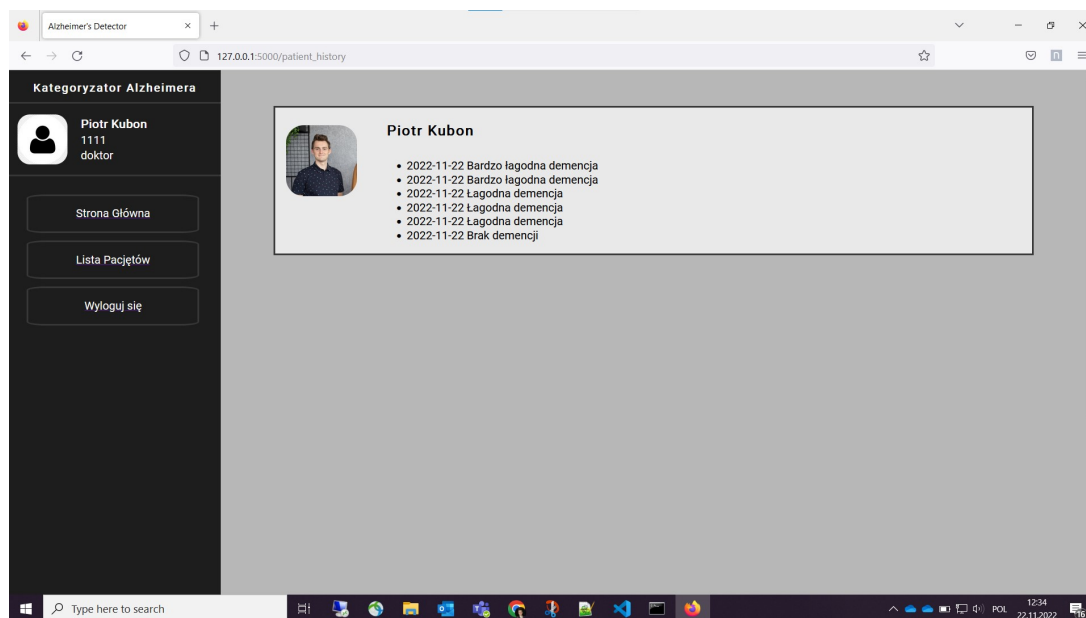


Rys. 5.4: Panel edycji danych pacjentów



### 5.1.5. Panel historii badań pacjenta

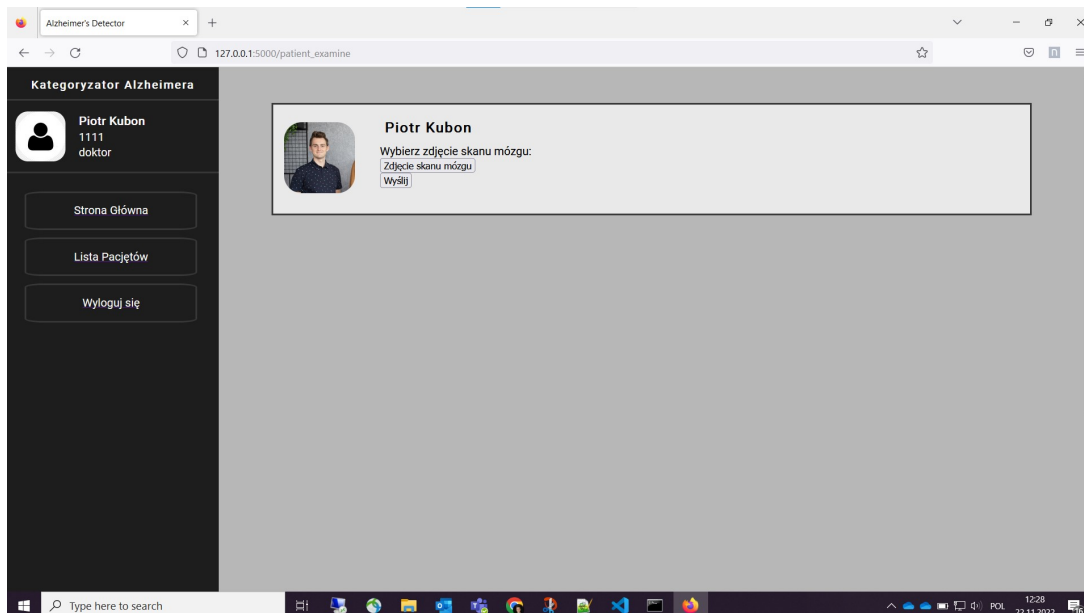
Panel historii badań pacjenta zawiera podstawowe informacje na temat pacjenta wraz z zdjęciem pacjenta oraz historią badań. W skład historii badań wchodzi wyniki wszystkich wcześniej wykonanych badań na pacjencie wraz z datą ich przeprowadzenia. Ostatnie przeprowadzone badanie znajduje się na szczycie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



Rys. 5.5: Panel historii badań pacjenta

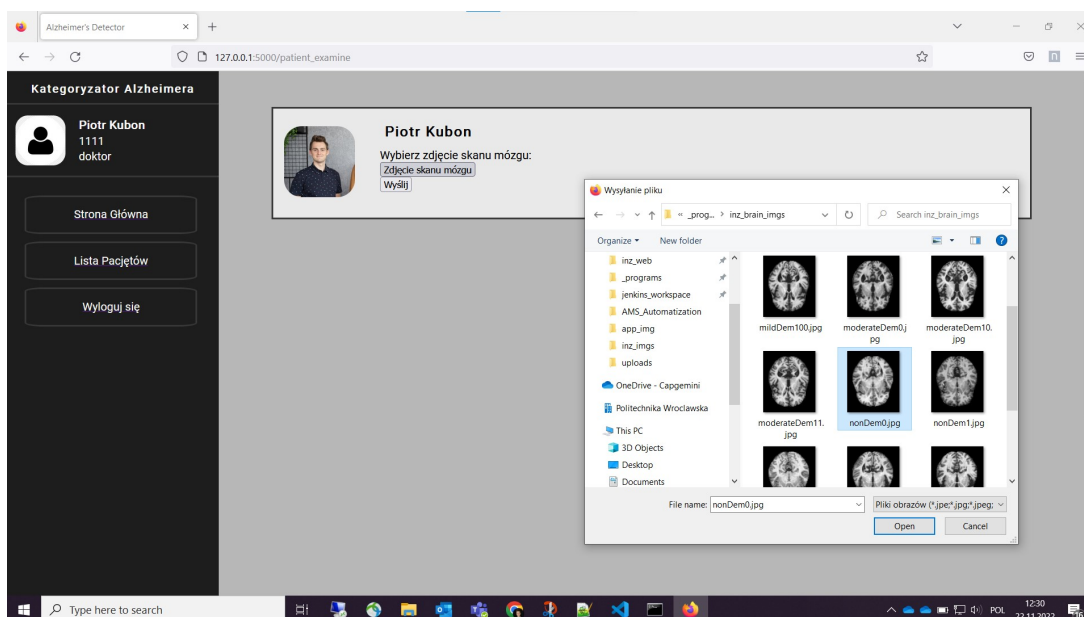
### 5.1.6. Panel badania pacjenta

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.



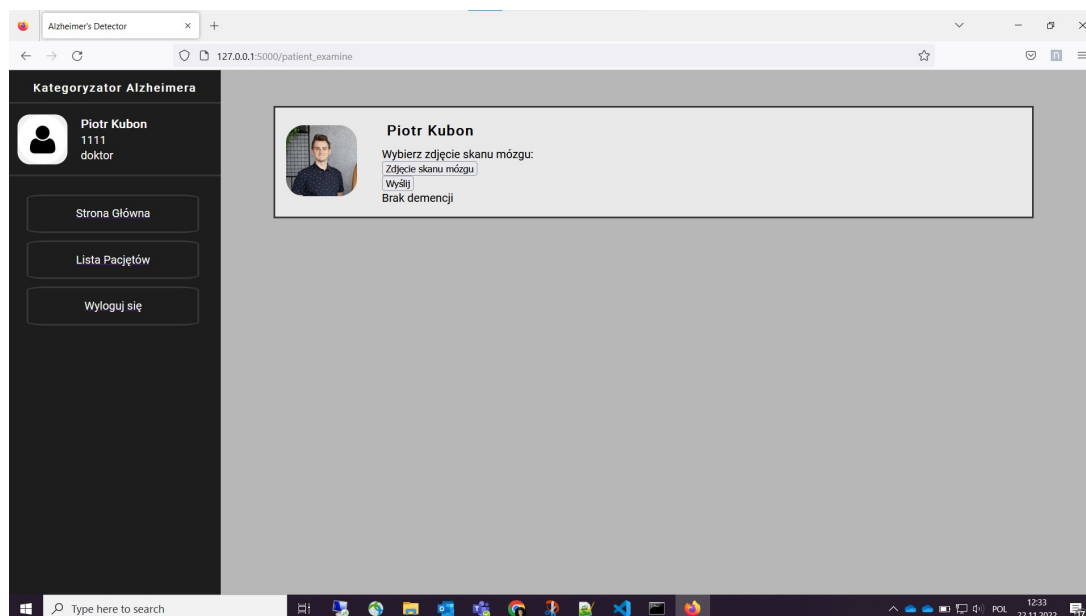
Rys. 5.6: Panel badania pacjenta

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

W wyniku przeprowadzonych badań udało się stworzyć model cechujący się wysoką skutecznością klasyfikacji zdjęć rentgenowskich mózgu. Udało się także stworzyć serwis internetowy wykorzystujący wcześniej wspomniany model do wykrywania choroby Alzheimerera.

Wyciągnięto również następujące wnioski:

- Normalizacja danych w znacznym stopniu poprawiła wyniki klasyfikacji modelu.
- Najlepsze wyniki klasyfikacji osiągnął model wykorzystujący 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, dzięki czemu zmniejszamy ryzyko przetrenowania modelu. Wprowadzając zbyt duże zmiany oryginalnych zdjęć, możliwe jest pogorszenie zdolności klasyfikacyjnych modelu, ponieważ model może nauczyć się wykrywać błędne przekształcenia, zamiast cechy obiektu na zdjęciach.
- Model wdrożony do serwisu internetowego cechuje się bardzo dobrymi zdolnościami klasyfikacyjnymi, co zostało przedstawione w tabeli 4.14. Model ten jest w stanie rozróżnić zdjęcia z etykietą “Brak demencji” od zdjęć z etykietą “Bardzo łagodna demencja” co potwierdza macierz pomyłek zawarta na zdjęciu 4.53. Rozróżnienie zdjęć z wspomnianymi etykietami stanowiło poważny problem dla wcześniej badanych modeli.

# 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] 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].
- [6] Framework. <https://pl.wikipedia.org/wiki/Framework> [Dostęp dnia 14 grudnia 2022].
- [7] Jinja2 introduction. <https://jinja.palletsprojects.com/en/3.1.x/intro/> [Dostęp dnia 14 grudnia 2022].
- [8] 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].
- [9] Model-view-controller. <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller> [Dostęp dnia 14 grudnia 2022].
- [10] Mvc - model view controller. <https://docs.phalcon.io/4.0/pl-pl/mvc> [Dostęp dnia 14 grudnia 2022].
- [11] Numbers of people with dementia worldwide. <https://www.alzint.org/resource/numbers-of-people-with-dementia-worldwide/> [Dostęp dnia 14 grudnia 2022].
- [12] Perceptron. <https://en.wikipedia.org/wiki/Perceptron> [Dostęp dnia 14 grudnia 2022].
- [13] Perceptron wielowarstwowy. [https://pl.wikipedia.org/wiki/Perceptron\\_wielowarstwowy](https://pl.wikipedia.org/wiki/Perceptron_wielowarstwowy) [Dostęp dnia 14 grudnia 2022].
- [14] phalcon controllers. <https://docs.phalcon.io/4.0/pl-pl/controllers> [Dostęp dnia 14 grudnia 2022].

- [15] Top 10 machine learning algorithms in 2022. <https://www.spiceworks.com/tech/artificial-intelligence/articles/top-ml-algorithms/> [Dostęp dnia 14 grudnia 2022].
- [16] Top 7 python neural network libraries for programmers. <https://analyticsindiamag.com/top-7-python-neural-network-libraries-for-developers/> [Dostęp dnia 14 grudnia 2022].
- [17] A. Geron. *Uczenie maszynowe z użyciem scikit-learn i tensorflow*, wydanie ii. Helion, 2019.
- [18] L. LeCun, Bottou. *Gradient-based learning applied to document recognition*. 1998.
- [19] W. McCulloch. *A logical calculus of the ideas immanent in nervous activity*. 1943.
- [20] M. Minsky, Papert. *Perceptrons*. 1969.
- [21] W. Rumelhart, Hinton. *Learning representations by back-propagating errors*. 1986.