Contents

[1. Wstęp 3](#_Toc119553111)

[1.1. Cel 3](#_Toc119553112)

[1.2. Opis choroby Altzheimera 3](#_Toc119553113)

[1.3. Opis systemu 3](#_Toc119553114)

[1.4. Przebieg pracy nad systemem 4](#_Toc119553115)

[2. Projekt systemu 5](#_Toc119553116)

[2.1. Opis wykożystanych narzędzi 5](#_Toc119553117)

[2.2. Wykożystane wzorce projektowe 5](#_Toc119553118)

[2.2.1. MVC (Model-View-Controller) 5](#_Toc119553119)

[3. Implementacja systemu 6](#_Toc119553120)

[3.1. Przeprowadzone badania nad modelem sieci neuronowek: 7](#_Toc119553121)

[3.1.1. Badanie augumentacji na model – określenie, czy w wyniku zwiększenia zbioru danych uczących nastąpiła poprawa dokładności klasyfikacji 7](#_Toc119553122)

[3.1.2. Badanie wpływu funkcji aktywacji 7](#_Toc119553123)

[3.1.3. Badanie wpływu funkcji straty 7](#_Toc119553124)

[3.1.4. Badanie ilości warstw gęstych i konwolucyjnych 7](#_Toc119553125)

[3.1.5. Badanie ilości warstw gęstych i konwolucyjnych po augumentacji 7](#_Toc119553126)

[3.1.6. Badanie istotności pikseli – jakie obszary są wiążące – tak jak w książce dla mnista 7](#_Toc119553127)

[3.2. UI 7](#_Toc119553128)

[3.2.1. Panel logowania 7](#_Toc119553129)

[3.2.2. Strona główna 7](#_Toc119553130)

[3.2.3. Lista pacjętów 7](#_Toc119553131)

[3.2.4. Dodawanie pacjęta 7](#_Toc119553132)

[3.2.5. Edycja danych pacjęta 7](#_Toc119553133)

[3.2.6. Usunięcie pacjęta z serwisu 7](#_Toc119553134)

[4. Podsumowanie 7](#_Toc119553135)

[4.1. Podsumowanie wyników przeprowadzonych badań 7](#_Toc119553136)

[4.2. Reklama systemu / uwypulkenie zalet i przeznaczenia aplikacji 7](#_Toc119553137)

[4.3. Dalsze możliwości rozwoju aplikacji 7](#_Toc119553138)

[4.3.1. Sprawdzenie efektywności inych sieci 7](#_Toc119553139)

[4.3.2. Sprawdzenie efektywności głosowania większościowego różnych podsieci 7](#_Toc119553140)

[4.3.3. Wydzielenie w serwisie endpointów do REST API 7](#_Toc119553141)

[4.3.4. Poprawa inferfejsu UI – wykożystanie dedykowanego frameworka 7](#_Toc119553142)

[4.3.5. Wykożystanie bazy danych do przechowywania danych pracowników i pacjętów 7](#_Toc119553143)

[4.3.6. Szyfrowanie danych pracowników i pacjętów 7](#_Toc119553144)

[4.3.7. Wydzielenie serwisu webowego do chmury np. Azure 7](#_Toc119553145)

[1. Argumentacja wyboru oprogramowania: 11](#_Toc119553146)

[2. Kawałek o licencjach 12](#_Toc119553147)

# Wstęp

## Cel

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

## Opis choroby Altzheimera

Choroba Altzheimera dotyka z roku na rok coraz większą liczbę osób, szczególnie narażone są osoby starsze, z ograniczoną aktywnością fizyczną i pamięciową. Chorobę Altzheimera wywołują odkładające się w mózgu białka o patologicznej strókturze, stopniowo psujące neurony, powodując ich obumieranie, w wyniku czego dochodzi do coraz to większych zaników pamięci.

Wyróżnia się trzy stadia rozwoju choroby, z czego każde kolejne stadium charakteryzuje się coraz większym wpływem na organizm chorego, stopniowo uniemożliwiając mu samodzielne funkcjonowanie.

Obecnie nie jesteśmy w stanie w pełni wyleczyć choroby Altzheimera, a jedynie leczyć jej skutki i objawy oraz opóźniać jej rozwój przez odpowiednie leki i ćwiczenia. Dlatego tak ważne jest wykrycie choroby na jej wczesnym stadium rozwoju.

## Opis systemu

W celu usprawnienia procesu diagnozowania choroby Altzheimera wykonany został internetowy system wspomagający pracę lekarzy w ocenie stopnia zaawansowania choroby bądź jej braku.

Po wgraniu zdjęcia rentgenowskiego mózgu pacjęta, 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 rendgenowskich mózgu, zarówno osób chorych jak i zdowych. 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 jak i proces wstępnej obróbki przekazywanego zdjęcia.

Wybrany model, który osiągną największą dokładność w klasyfikacji stopnia rozwoju choroby, wydzielono i wdrożono w system serwisu internetowego. 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 systemie można przeprowadzić takie czynności jak dodawanie, edytowanie i usuwanie danych pacjęta oraz dostęp do historii wcześniej przeprowadzonych badań.

System zapewni również jednoznaczną identyfikację pacjęta dzięki automatycznie inkrementowanemu numerowi id. Głównym zadaniem systemu jest natomiast klasyfigacja i identyfikacja choroby Altzheimera na podstawie przesłanego zdjęcia rendgenowskiego mózgu pacjęta.

Ze względów bezpieczeństwa, system działa w wewnętrznej sieci obiektu, ma to na celu wykłuczenie możliwości połączenia się z nim przez osoby z zewnątrz. Dostęp do aplikacji odbywa się za pomocą przeglądarki internetowej. Dzieki takiemu podejściu, nie wymuszamy na lekarzu posiadania konkretnego systemu operacyjnego, zyskujemy również spójność systemu dla każdego użytkownika oraz bezpieczeństwo przechowywania i skladowania danych.

## Przebieg pracy nad systemem

Projekt systemu zakładał następujące etapy:

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

+ Ewentualnie - Dynamiczne tworzenie systemu za pomocą Jenkinsa i Dockera

# Projekt systemu

## Opis wykożystanych narzędzi

Projekt systemu został rozpoczęty od stworzenia maszyny wirtualnej za pomocą oprogramowania Oracle Virtual Box. W tym celu utworzono nową maszynę wirtulaną z systemem operacyjnym Linux Ubuntu, narzucono maksymalne zużycie pamięci RAM i przydzielono ilość miejsca na dysku. Następnie zwiększono ilość procesorów do dwuch 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 wykożystaniem frameworku Flask. Zdecydowano się na użycie tego framewoku ze względu na brak kosztów licencyjnych oraz dużą dowolność konfigurowania aplikacji. Pozostawia ona programiście możliwość podejmowania decyzji projektowych, przy jednoczesnej możliwości rozszerzania o dodatkowe moduły.

Flask zapewnia również możliwość kożystania z szablonów Jinja2, dzięki czemu jesteśmy w stanie oddzielić logikę aplikacji od warstwy prezentacji, co jest jendym z założeń wzorca projektowego MVC czyli Model-View-Controller (Model-Widok-Kontroler), który staraliśmy się zaimplementować.

Aplikacja zawiera model sieci neuronowej, wyuczony przy wykożystanoiu usługi Google Colab. Zdecydowano się na przeniesienie procesu uczenia i przeprowadzanie badań do usługi Google Colaboratory (w skrócie Colab), ze względu na oferowane przez nią darmowe środowisko wykonawcze, posiadające wydajne procesory graficzne, doskonale sprawdzające się w zastosowaniach uczenia sieci neuronowych. Usługa Google Colab udostępnia nam zasoby za pośrednictwem dynamicznie tworzonch maszyn wirtualnych, które zostają usuniete po upływie określonego czasu pracy lub wyczerpaniu zasobów. W tym czasie możemy przeprowadzić badania, wyuczyć sieć oraz wydzielić i pobrać model który okaże się najlepszy. Pobrany model możemy następnie zaimportować do naszej aplikacji .

Ewentualny opis Jenkinsa + Dockera

## Wykożystane wzorce projektowe

### MVC (Model-View-Controller)

Tak jak sugeruje to nazwa wzorca, zależy nam na rozdzieleniu logiki aplikacji na trzy niezależna, ale powiązane ze sobą bloki. 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 pozostłe bloki zmieniać i dostosowywać logikę oraz kod programu, na przykład poprzez zmianę silnika bazy danych na inny. 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.

Blok widoku reprezentuje interfejs osoby kożystającej ze strony, zawiera warstwę wizualną oraz zajmuje się przedstawieniem danych przekazanych przez kontroler za pośrednictwem modelu. 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. Zyskujemy w ten sposób możliwość zarówno dynamicznych zmian widoku strony, jak również jedno miejsce gdzie należy wprowadzić poprawki w kodzie, aby były widoczne we wszystkich widokach na stronie.

Ostatnim i zarazem najważniejszym blokiem modelu MVC jest kontroler. Blok ten odpowiedzialny jest za przetwarzanie danych modelu i przekazywanie ich do odpowiedniego widoku. Blok ten zajmuje się również przetworzeniem danych otrzymanych od urzytkownika aplikacji za pośrednictwem widoku. W zależności od stopnia rozbudowania serwisu możliwe jest kożystanie z wielu kontrolerów, odpowiadających na żądania urzytkownika wysłane na konkretny adres URL.

## Modele danych

W

# Implementacja systemu

W przypadku naszego serwisu, w skład bloku kontrolera wchodzą kontrolery odpowiedzialne za:

* sprawdzenie danych logowania i blokowanie serwisu przed osobami niezalogowanymi
* dedykowany kontroler dla każdej z operacji dodania, pobrania, edycji i usunięcia pacjęta
* prezentację listy pacjętów
* prezentację listy przeprowadzonych badań na pacjęcie
* badanie stopnia zaawansowania choroby Altzheimera na podstawie przesłanego zdjęcia skanu mózgu
* wylogowanie się urzytkownika z serwisu

W celu wprowadzenia danych i prezencji zwracanych danych urzytkownikowi utworzono następujące widoki:

* widok logowania się urzytkownika
* widok strony głównej serwisu
* widok listy pacjętów wraz z możliwością dodania nowego pacjęta do serwisu oraz usunięcia już istniejącego
* widok edycji danych pacjęta
* widok histori przeprowadzonych badań na pacjęcie
* widok panelu do przeprowadzenia badania

Dodatkowo wprowadzono podwidoki w celu wydzielenia powtarzających się elementów, w tym:

* menu
* blok pacjęta w liście

## Przeprowadzone badania nad modelem sieci neuronowek:

### Badanie augumentacji na model – określenie, czy w wyniku zwiększenia zbioru danych uczących nastąpiła poprawa dokładności klasyfikacji

### Badanie wpływu funkcji aktywacji

### Badanie wpływu funkcji straty

### Badanie ilości warstw gęstych i konwolucyjnych

### Badanie ilości warstw gęstych i konwolucyjnych po augumentacji

### Badanie istotności pikseli – jakie obszary są wiążące – tak jak w książce dla mnista

## UI

### Panel logowania

### Strona główna

### Lista pacjętów

### Dodawanie pacjęta

### Edycja danych pacjęta

### Usunięcie pacjęta z serwisu

# Podsumowanie

## Podsumowanie wyników przeprowadzonych badań

## Reklama systemu / uwypulkenie zalet i przeznaczenia aplikacji

## Dalsze możliwości rozwoju aplikacji

### Sprawdzenie efektywności inych sieci

### Sprawdzenie efektywności głosowania większościowego różnych podsieci

### Wydzielenie w serwisie endpointów do REST API

### Poprawa inferfejsu UI – wykożystanie dedykowanego frameworka

### Wykożystanie bazy danych do przechowywania danych pracowników i pacjętów

### Szyfrowanie danych pracowników i pacjętów

### Wydzielenie serwisu webowego do chmury np. Azure

- linux

- flask

- sieci neuronowe

- trenowanie modelu – colab

- wdrożenie modelu do serwisu flask

- testy funkcjonowania systemu

- wydzielenie stawiania systemu do dockera

- utworzenie pipelinu w jenkinsie – spięcie z repo na githabie – on push – build and test + wdrożenie w kontenerze dockera

----

- jenkins

- docker

Choroba Altzheimera

* + Rosnąca liczba zachorowań
  + Grupa szczególnie narażona
  + Przyczyny
  + Trudna identyfikacja i jednoznaczne stwierdzenie / wykrycie horoby
  + Malejąca liczba specjalistów
  + Potrzeba stworzenia systemu wspomagającego lekarzy który wykrywałby, czy na zdjęciu widoczne są objawy choroby
  + Ostateczna decyzja należy do lekarza (może przeprowadzić wywiad, zna więcej informacji niż samo zdjęcie)
* Sztuczne sieci neuronowe
  + Historia
  + Podobieństwo początkowych neuronów do neuronów w mózgu
  + Perceptron
  + Propagacja wsteczne
  + Wiele warstw perceptronów – głęboka sieć neuronowa
  + Warstwy splotowe
  + Augumentacja
  + Parametry i hiperparametry
    - Funkcja straty
    - Funkcja aktywacji
  + Przetwarzanie danych
    - Wpływ rozmiaru zdjęcia na dokładność
    - Treshold – progowanie - doprowadzenie do zdjęcia 0 / 255
    - Wyostrzenie
* System webowy
  + Potrzeba skalowalności, dużej elastyczności i prostoty wprowadzania zmian
  + Zdecentralizowany model podejmujący decyzję
  + Prostota działania
  + Flask
  + Jenkins
  + Gitlab
  + Colab
    - Platforma do uczenia
    - Duże zasoby
  + Http
* Typy sieci neuronowych
* Badane paramettry sieci neuronowej

Badania przeprowadzić na przyjętej architekturze, po ich zakończeniu skiąć rezultaty w jedno i porównać wyniki!

Poprawić treść strony na polską !!!!!!

# Argumentacja wyboru oprogramowania:

Zdecydowano sie na wykożystanie w pracy systemu operacyjnego Linux Ubuntu 22.04 ze względu na brak kosztów licencyjnych oraz wsparcie społęczności, rozwijającej projekt systemu operacyjnego od 2004 roku.

W projekcie zdecydowano się na wykożystanie systemu operacyjnego Linux Ubuntu, ze względu na:

* brak kosztów licencyjnych
* wsparcie dla pozostałych oprogramowania, modułów i bibliotek wykożystanych w daleszej części pracy
* wbudowany firewall wystarczający do potrzeb projektu
* dużą społeczność udzielającą się na forach dyskusyjnych

W projekcie zdecydowano się na wykożystanie oprogramowania Oracle Virtual Box, ze względu na:

* możliwość tworzenia migawek i kopi bezpieczeństwa całego systemu
* ochrona systemu hosta przed błędami i uszkodzeniami które moą wystąpić w środowisku wykonawczym
* brak ingerencji systemu wirtualnego w ustawienia hosta
* wygodna możliwość wydzielenia zasobów

W projekcie zdecydowano się na wykożystanie frameworka Flask, ze względu na:

* niewielkie wymagania sprzętowe
* brak kosztów licencyjnych
* dobrą dokumentację oraz wsparcie społeczności
* udostępnia możliwość programowania w języku python

W projekcie zdecydowano się na wykożystanie biblioteki Tensor Flow, ze względu na:

* brak kosztów licencyjnych
* możliwość programowania w języku python
* wygodne i szybkie tworzenie modeli sieci neuronowych
* możliwość wydzielenia procesu uczenia modelu do dedykowanych usług chmurowych
* duża społeczność oraz liczne artykuły naukowe pomocne na etapie tworzenia aplikacji

W projekcie zdecydowano się również na skożystanie z usług serwisu Google Colaboratory (w skrócie Colab) ze względu na:

* możliwość uruchamiania wybranych fragmentów kodu napisanego w języku python na wyspecjalizowanej maszynie za pośrednictwem przeglądarki internetowej
* bezpłatny dostęp do procesorów graficznych i jednostek TPU do 12 godzin dzięki czemu jesteśmy w stanie szybciej wytrenować model sieci

W projekcie zdecydowano się na wykożystanie oprogramowa ze względu na:

# Kawałek o licencjach

Oprogramowanie VirtualBox dostępne jest na licencji GNU General Public Licens, dzięki czemu możemu wykożystać je za darmo do użytku osobistego w celu stworzenia projektu do pracy dyplomowej.

System operacyjny Linux Ubuntu dystrybuowany jest na licencji Creative Commons CC-BY-SA wersja 3.0 UK, która zapewnia możliwość bezpłatnego kożystania z wspomnianego systemu operacyjnego.

Framework Flask dostępny jest za darmo pod warunkami zawartymi w licencji Licencja BSD-3-Clause.

Biblioteka Tensor Flow dostępna jest na darmowej licencji Apache License 2.0