# Sprawozdanie
# Sieć neuronowa do rozpoznawania cyfr
# Przedmiot: Projekt indywidualny
# 16 maja 2021

## Spis treści

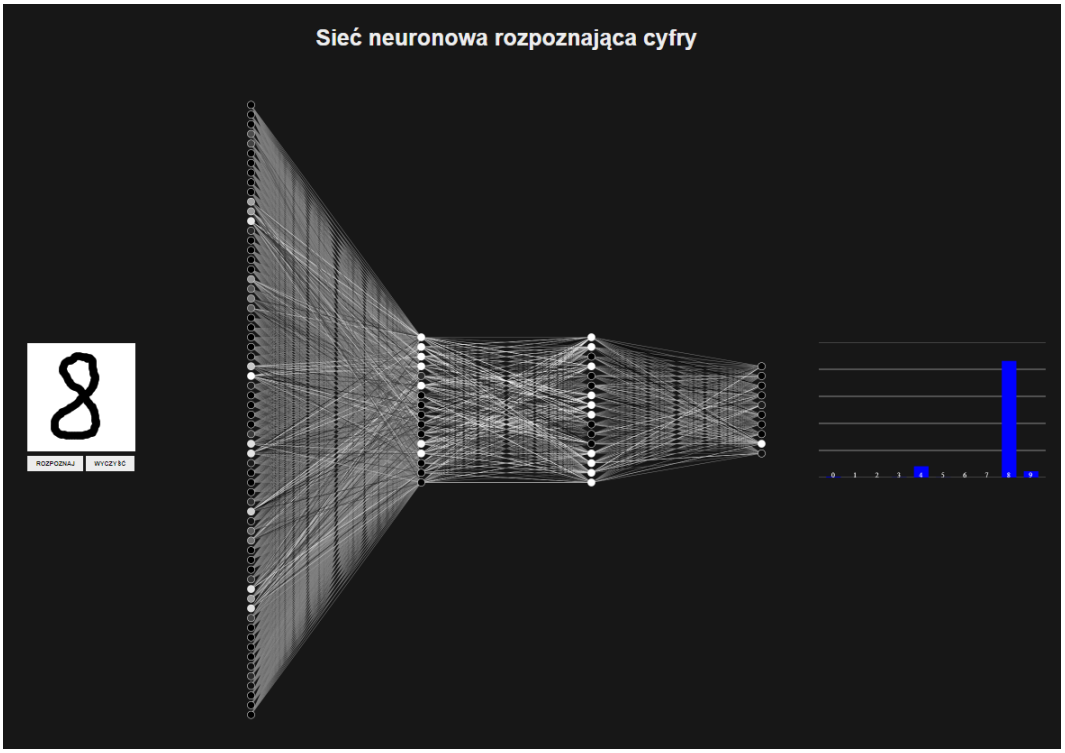


## 1 Cel projektu

Wykonanie strony internetowej wizualizującej działanie prostej sztucznej sieci neuronowej rozpoznającej cyfry.

## 2 Interfejs użytkownika

Do strony użytkownik ma dostęp z przeglądarki internetowej, przez wprowadzenie odpowiedniego adresu.




Wygląd strony

### 2.1 Wprowadzanie danych

Użytkownik wprowadza cyfrę rysując ją myszką na kwadratowym, białym polu. Pole ma wymiary $240 \times 240$ pikseli, szerokość linii to 17 pikseli.

Wciśnięcie przycisku **ROZPOZNAJ** powoduje wczytanie obrazu z pola oraz sklasyfikowanie go przez sieć neuronową.

Wciśnięcie przycisku **WYCZYŚĆ** powoduje wyczyszczenie pola do rysowania.

### 2.2 Dane wyjściowe

Po wciśnięciu przycisku **ROZPOZNAJ** graf sieci neuronowej zmienia kolory, tak aby przedstawić aktywację neuronów i krawędzi.

Wynik klasyfikacji prezentuje diagram słupkowy. Diagram przedstawia prawdopodobieństwo, oszacowane przez sztuczną sieć neuronową, że na rysunku znajduje się

dana cyfra. Oszacowane prawdopodobieństwo to stosunek aktywacji neuronu odpowiadającego danej cyfrze do sumy wszystkich aktywacji.

# 3 Sieć neuronowa

## 3.1 Zbiór danych

Do uczenia sieci neuronowej użyłem zbioru digits z biblioteki scikit-learn. Zbiór ten zawiera 1797 pisanych odręcznie cyfr o wymiarach $8 \times 8$ pikseli. Biblioteka zawiera funkcję **load_digits** pozwalającą na wczytanie dwóch tablic, pierwsza zawiera dane zapisane w wierszu, a druga prawidłowe etykiety.

Aby dostosować dane do uczenia tablicę zawierającą dane normalizuję, a etykiety (cyfry całkowite) zapisuję jako wektor zer i jednej jedynki znajdującej się w komórce o indeksie równym etykiecie.

Podział na zbiór testowy dokonałem za pomocą funkcji **train_test_split**. Zbiór uczący zawiera 70% przykładów, a zbiór testowy pozostałe 30%. Funkcja też zmienia kolejność przykładów.

Nie udało mi się znaleźć podobnego zbioru liter.

## 3.2 Budowa sieci

Sieć neuronowa składa się z:

1. warstwy wejściowej zawierającą 64 neurony,
2. warstwy ukrytej zawierającej 16 neuronów,
   funkcja aktywacji **tangens hiperboliczny**,
3. warstwy ukrytej zawierającej 16 neuronów,
   funkcja aktywacji **tangens hiperboliczny**,
4. warstwy wyjściowej zawierającej 10 neuronów,
   funkcja aktywacji **tangens hiperboliczny**.

Przy użyciu funkcji **ReLU** sieć nie znajdowała optymalnego rozwiązania.

## 3.3 Uczenie sieci neuronowej

Sieć neuronowa wykonuje propagację wsteczną dla każdego przykładu. Najlepsza poprawność w zbiorze testowym jaką udało się osiągnąć to 92%, został osiągnięty dla współczynnika uczenia 0,2 i 700 iteracji. Uczenie sieci neuronowej z tymi parametrami czasami zatrzymuje sią na gorszym rozwiązaniu.

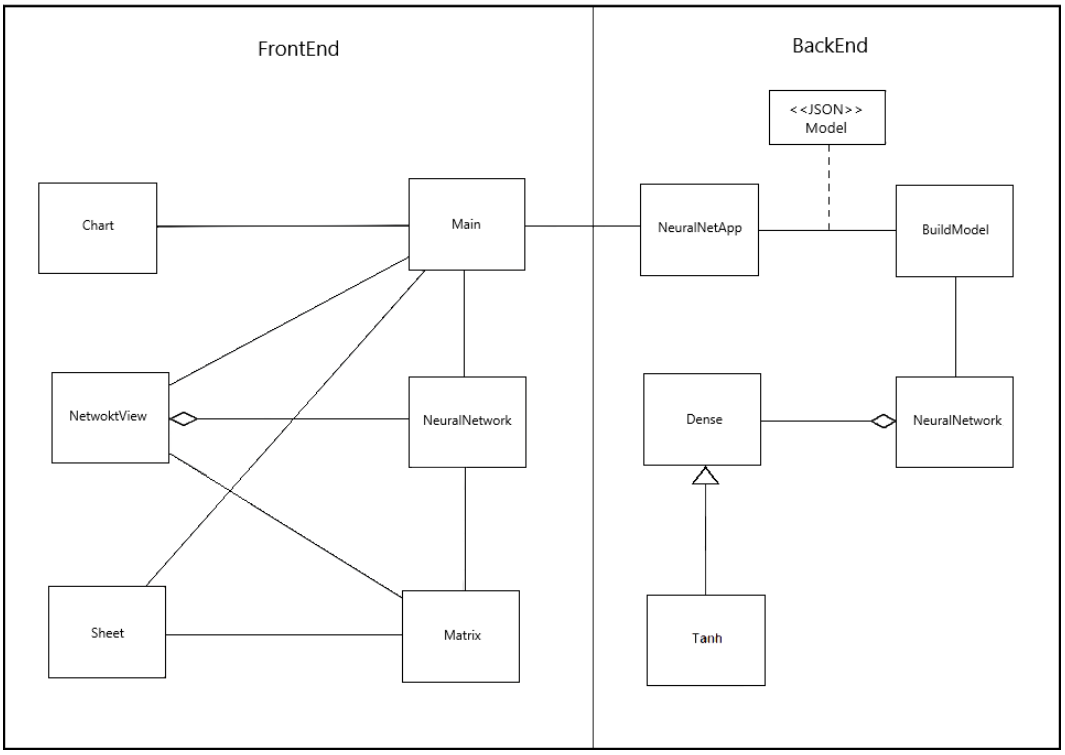
# 4 Implementacja

## 4.1 Użyte technologie

Wykorzystane przy projekcie języki, biblioteki i frameworki:

- Python
  - NumPy
  - scikit-learn
  - Flask
- JavaScript
- HTML
- CSS

## 4.2 Podział na moduły




Krótki opis modółów:

- Chart - odpowiada za wyświetlanie diagramu słupkowego, przedstawiającego procentowy udział cyfr w rezultacie, prawdopodobieństwo szacowanie przez model.

- NetworkView - graficzna część sieci neuronowej, prezentuje działanie modelu.
- Sheet - umożliwia wprowadzanie danych użytkownikowi danych wejściowych, narysowanie cyfry.
- Main - zarządza działaniem strony.
- NeuralNetwork (front-end) - zawiera uproszczony model sieci neuronowej (tylko propagacja w przód).
- Matrix - odpowiada za potrzebne działania na macierzach.
- NeuralNetApp - odpowiada za obsługę zapytań ze strony.
- BuildModel - odpowiada za przygotowanie modelu: przygotowanie danych, stworzenie sieci neuronowej, parametry uczenie sieci i zapisanie modelu do pliku.
- NeuralNetwork (back-end) - udostępnia funkcje potrzebne do stworzenia modelu sieci neuronowej i sprawdzenia jej działania.
- Dense - ogólna klasa warstwy sieci neuronowej (layer).
- Tanh - klasa warstwy sieci neuronowej (layer), ma określoną funkcję aktywacji - tangens hiperboliczny.

# 5 Możliwy dalszy rozwój projektu

Dodanie nowych funkcji aktywacji oraz usprawnienie optymalizacji sieci neuronowej. Udostępnienie użytkownikowi możliwości wyboru modelu sieci neuronowej. Użycie zbioru danych MNIST, o większej rozdzielczości.