Systemy sztucznej inteligencji

dokumentacja projektu DigitRecognizer

Jambor Daniel
Grupa 2D

Kozieł Wojtek
Grupa 2D

Matula Kamil
Grupa 2D

19 maja 2020

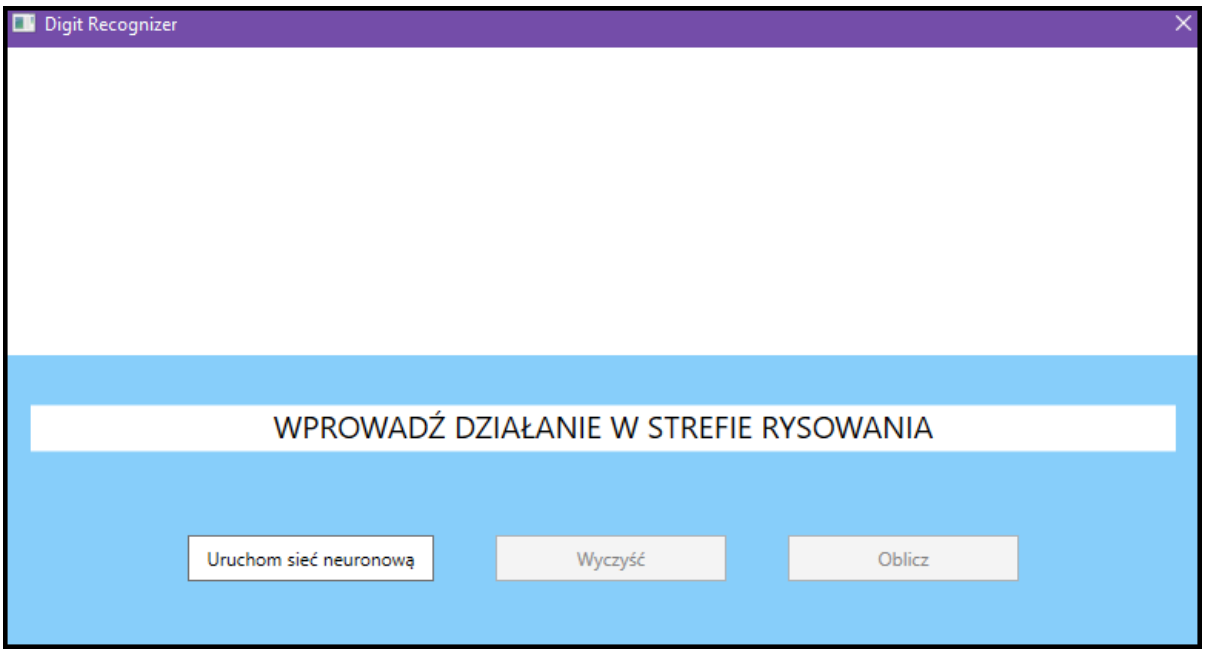
Część I

Opis programu

Program ***DigitRecognizer*** służy do rozpoznawania ręcznie napisanych działań i wyświetlania ich wyniku. Użytkownik pisze na specjalnym polu liczby całkowite oraz dowolne z czterech zaimplementowanych znaków arytmetycznych (odpowiadających działaniom dodawania, odejmowania, dzielenia i mnożenia), a program wyświetla końcowy wynik podanego wyrażenia. Program korzysta z bazy danych „MNIST", która składa się łącznie z 70 000 ręcznie napisanych cyfr oraz z autorskiej bazy cyfr i oznaczeń matematycznych.

Instrukcja obsługi

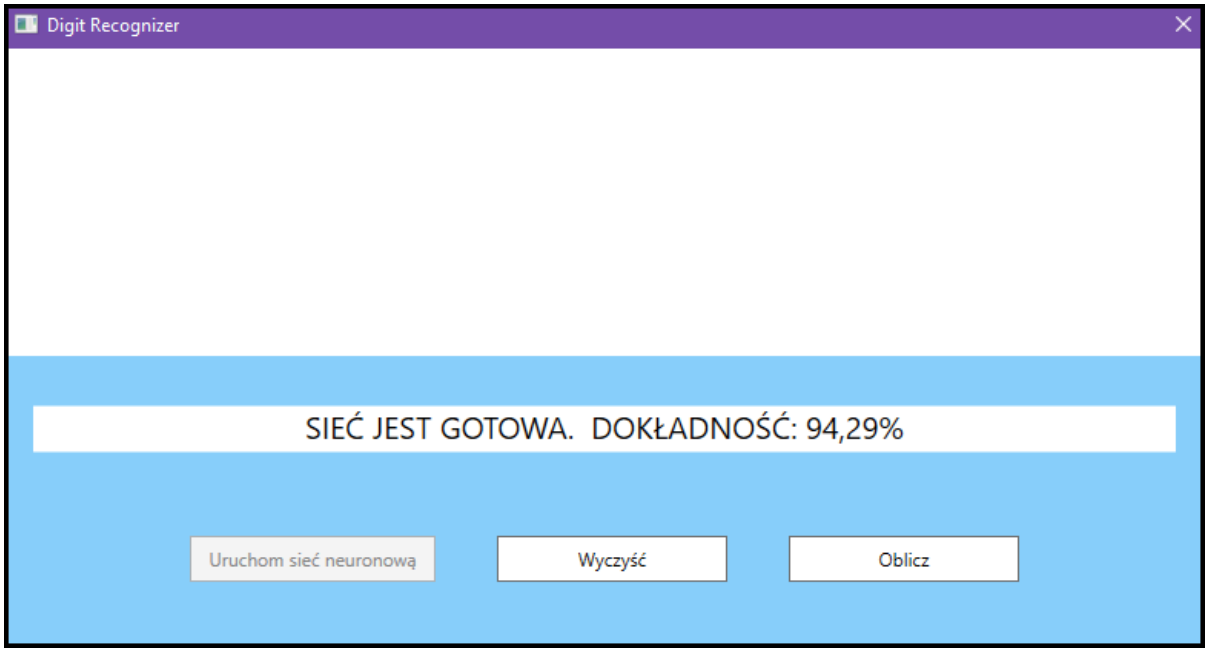
W celu włączenia programu należy uruchomić plik wykonawczy *DigitRecognizer.exe*. Do prawidłowego działania aplikacji wymagany jest plik *weights.txt*, który powinien znajdować się w tym samym folderze, co plik *DigitRecognizer.exe*. Choć sieć nie będzie nauczana przy każdym uruchomieniu aplikacji, w folderze projektowym powinien znajdować się także folder *Datasets* z wykorzystywanymi do uczenia bazami danych - pozwoli to na wczytanie baz, zbudowanie sieci, przypisanie wag synapsom i wyliczenie dokładności sieci. W przypadku nieodnalezienia pliku z wagami pojawi się komunikat *„PLIK Z WAGAMI SYNAPS NIE ISTNIEJE!"*. Z kolei w przypadku braku folderu *Datasets* lub bazy MNIST użytkownik zobaczy informację *„NIE ODNALEZIONO FOLDERU DATASETS!"*. Ponadto, jeśli plik z wagami jest nieprawidłowy (np. zawiera nieodpowiednią ilość linii z powodu uczenia sieci neuronowej o innej budowie), również pojawi się stosowny komunikat. Po uruchomieniu programu użytkownik zobaczy poniższy interfejs graficzny:



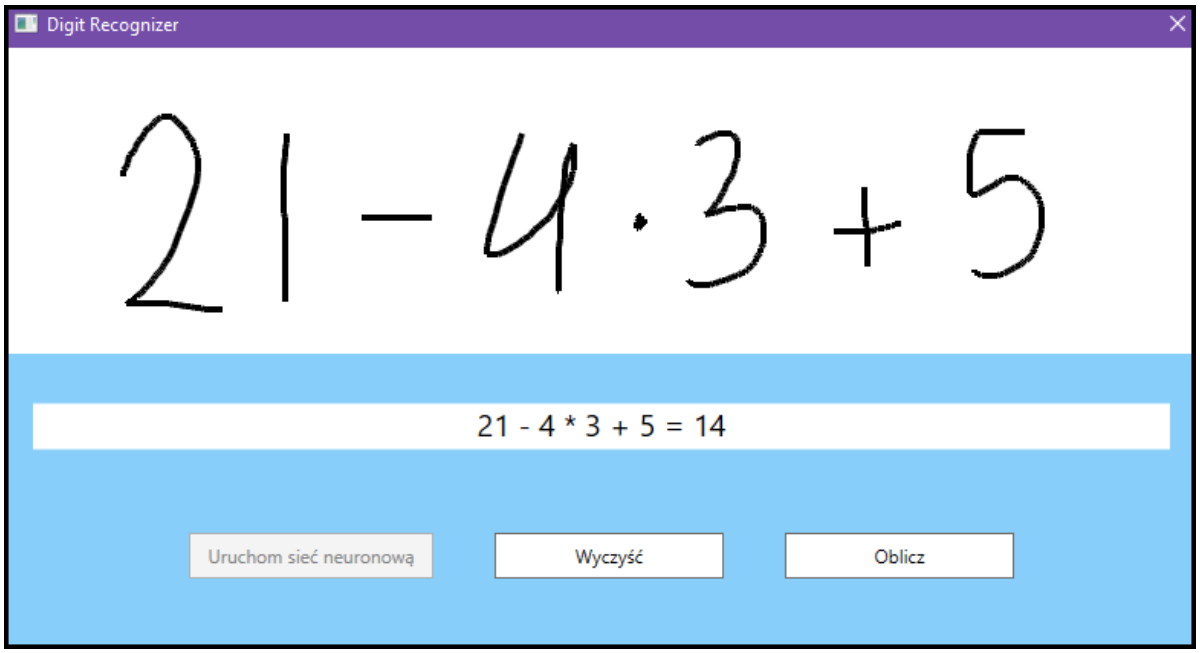
Jak widać na zamieszczonym na poprzedniej stronie zrzucie ekranu aplikacja składa się z:

- białego pola, zwanego „strefą rysowania”, na którym użytkownik może zapisywać wyrażenia arytmetyczne,
- szerokiego pola tekstowego, gdzie wyświetlane są wszystkie komunikaty związane z działaniem programu włącznie z wynikiem zapisanego wyrażenia,
- przycisku „Uruchom sieć neuronową”, który jako jedyny jest dostępny po uruchomieniu programu - pozwala on zbudować sieć neuronową i wczytać wcześniej wyuczone wagi,
- przycisku „Wyczyść”, który czyści zawartość strefy rysowania,
- przycisk „Oblicz”, który zleca sieci neuronowej pobranie narysowanych przez użytkownika cyfr i znaków, a także zwraca obliczony wynik.

Po wciśnięciu przycisku „Uruchom sieć neuronową” w polu tekstowym pojawi się informacja o tym, że sieć jest już gotowa oraz ile wynosi jej wyuczona dokładność. Ponadto przycisk ten stanie się nieaktywnym, a pozostałe dwa uaktywnią się, co widać na poniższym obrazku:



W celu obliczenia wartości pewnego wyrażenia wystarczy je napisać w strefie rysowania, a następnie wcisnąć przycisk „Oblicz” np.:



Dodatkowe informacje - DO UZUPEŁNIENIA!

Wymagania itd. - TUTAJ MOŻEMY IMO NAPISAĆ O NIEDOKŁADNOŚCIACH SIECI (I NA PRZYKŁAD O CANVAS), A TAKŻE O TECHNOLOGIACH JAKIE WYKORZYSTALIŚMY

Część II

Opis działania - DO UZUPEŁNIENIA!

Tutaj uwzględniamy część matematyczną. Opisujemy całą teorię np.: dla zadania związanego z sieciami neuronowymi - opisujemy całą budowę, algorytm uczenia i wszystkie wzory. Dla zadania związanego z kombinatoryką opisujemy całą teorię kombinatoryczną potrzebną do zrozumienia zadania (mile widziany przykład obliczeniowy).

Algorytm - DO UZUPEŁNIENIA!

Tutaj opisujemy rozwiązanie zadania. Dla przedmiotu programowanie będzie to wykorzystanie matematyki z poprzedniego zadania itd. Dla SSI będzie to ogólne działanie przetwarzania danych w oparciu o modele matematyczne z poprzedniego zadania.

Pseudokod tworzymy w LaTeX. Przykład:

Data: Dane wejściowe liczba k
Result: Brak
$i := 0$;
while $i < k$ **do**
  Drukuj na ekran liczbę i;
  if $i\%2 == 0$ **then**
    Wydrukj informację, że liczba i jest liczbą parzystą;
  else
    Wydrukj informację, że liczba i nie jest liczbą parzystą;
  end
end

Algorithm 1: Algorytm drukowania informacji o liczbie parzystej/nieprarzystej.

Bazy danych - DO ROZWINIĘCIA!

Nauka sieci neuronowej wykorzystywała dwie bazy danych:

MNIST

THE MNIST DATABASE of handwritten digits - baza danych składająca się z 60 000 próbek treningowych oraz 10 000 próbek walidacyjnych. Dane są przechowywane w formacie *.idx3-ubyte* (w przypadku samych obrazków) oraz *.idx1-ubyte* (w przypadku etykiet). Pliki składają się z wartości typu ubyte i 32-bit Integer. W przypadku plików przechowujących grafiki, pierwsze cztery wartości przechowują:

- 'Magic number' - liczba kontrolna
- Liczba zdjęć
- Szerokość jednego zdjęcia
- Wysokość jednego zdjęcia

- Kolejne pixele zdjęć. Pixele te przyjmują wartości od 0 (które reprezentuje kolor biały) do 255 (reprezentacja koloru czarnego)

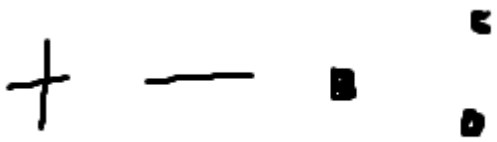
Pliki przechowujące etykiety zbudowane są na podobnej zasadzie:

- 'Magic number' - liczba kontrolna
- Liczba etykiet
- Kolejne etykiety

Wczytywanie bazy danych polega na wczytaniu pierwszych czterech wartości z pliku zawierającego zdjęcia oraz pierwszych dwóch z pliku zawierającego etykiety. Następnie do tablicy dwuwymiarowej wczytywane jest 784 pixeli (28 x 28), a do drugiego wymiaru odpowiadająca zdjęciu etykieta (dla jedynki będzie to 1 itp.). Takie dane są już gotowe do przesłania ich do sieci, jednak etykiety zostały przebudowane na potrzeby projektu.

Baza oznaczeń matematycznych

Baza oznaczeń matematycznych została zrobiona na potrzeby projektu. Baza opiera się na plikach .png, które zawierają zestawy każdego znaku:



Jeden plik zawiera 50 takich zestawów umieszczonych w jednym wierszu. Łącznie baza danych składa się z 600 znaków, z czego 10% jest traktowana jako część walidacyjna. Za pomocą napisanego algorytmu, plik jest dzielony na 200 osobnych obrazków, których pixele są zapisywane w tablicy tak samo jak w przypadku bazy danych MNIST. Z racji tego, że możliwe są tylko cztery etykiety, dodawane one są naprzemiennie, gdyż kolejność oznaczeń jest taka sama(plus, minus, mnożenie, dzielenie).

Implementacja - DO UZUPEŁNIENIA!

Opis, zasada i działanie programu ze względu na podział na pliki, nastepnie funkcje programu wraz ze szczegółowym opisem działania (np.: formie pseudokodu, czy odniesienia do równania)

```
1  Tutaj wklejamy fragment kodu, ktory chcemy opisac
2  (bez polskich znakow).
```

Testy - DO UZUPEŁNIENIA!

Tutaj powinna pojawić się analiza uzyskanych wyników oraz wykresy/pomiary.

Pełen kod aplikacji - DO UZUPEŁNIENIA NA KONIEC!

```
1 Tutaj wklejamy pelen kod.
```
