Czytanie tekstu

Projekt zrealizowany w ramach przedmiotu Analiza Obrazów

Wykonali:

Kamil Sudoł

Patryk Śledź

Jakub Strugała

1. Założenia projektu

Założeniem projektu było stworzenie aplikacji umożliwiającej rozpoznanie tekstu z wczytywanego obrazka. Program został napisany przy użyciu oprogramowania MATLAB w wersji R2020b.

2. Opis funkcjonalności

2.1 Słowem wstępu

Do implementacji algorytmu zostały wykorzystane dwa toolbox'y MATLABA, mianowicie *Deep Learning Toolbox* do wytrenowania sieci neuronowej oraz *Image Processing Toolbox* w ramach obróbki obrazu. Program umożliwia wczytanie z pliku obrazu zawierającego interesujący nas tekst, ustalenie, co na podanym obrazie jest napisane oraz wyświetlenie rozpoznanego tekstu. Rozpoznany tekst można zapisać do pliku tekstowego przy użyciu jednego z przycisków.

2.2 Działanie programu

Przygotowanie danych treningowych

Jako dataset, przygotowano skany dużych liter dla czcionki Arial znajdujących się w odpowiadających każdej literze podfolderach w katalogu "*dataset*".

Każdy z folderów zawierający skany liter przekazujemy do funkcji ***imageDatastore()*** z pliku "*wspolczynniki_train.m"* z katalogu "*funkcje"*, której wynik zapisujemy do zmiennej pomocniczej (*ids_a*, *ids_b*, itd.). Następnie te pomocnicze zmienne wykorzystujemy jako argumenty do funkcji ***wczytaj()*** z pliku "*wczytaj.m*". Początkowo w funkcji wywoływana jest metoda ***get_length()***, która zwraca ilość obrazów danej litery znajdujących się w folderze

przekazanym do funkcji jako obiekt ***imageDatastore***. Kolejno, w pętli, dla każdego obrazu z literą funkcja wybiera jeden z kanałów obrazu, dokonuje binaryzacji z wielkością progu równą 0.8, a następnie wykonuje na nim operację otwarcia. Tak przygotowana macierz obrazu trafia do funkcji ***regionprops()***, która zwraca szereg atrybutów wczytanego obrazu.

Następnie, ze zwróconych właściwości każdego obiektu zapisujemy do pomocniczej zmiennej obraz obiektu mieszczącego się w jego Bounding Box'ie, odwołując się do właściwości ***Image***. Kolejno, na wcześniej wspomnianej zmiennej pomocniczej używamy funkcji ***imresize()***, aby zmienić rozmiar obiektu do wielkości 300x300, gdyż sieć potrzebuje do nauki obiektów takiej samej wielkości.

Tak przygotowane obiekty przekazujemy do funkcji poznanych na laboratorium nr 5 do policzenia wartości 8 współczynników dla każdego znalezionego obiektu.

Współczynniki, dla których określiliśmy wartości to:

- Kształt
- Współczynnik Malinowskiej
- CircularityS – kolistość
- CircularityL – kolistość
- Współczynnik Feret'a – określający stosunek średnic z BoundingBox'a - bok poziomy do boku pionowego (promienie Feret'a to inaczej rozpiętość figury w x-ie i y-ku)
- Współczynnik Danielsson'a – średnia odległość piksela od krawędzi(dla koła jest to największa możliwa wartość)
- Współczynnik Haralick'a – średnia odległość pikseli obwodu od środka
- Współczynnik Blair-Bliss'a – średnia odległość od środka dla każdego piksela

Wyżej wymienione współczynniki składają się na wektor ***trainin*** do uczenia sieci neuronowej. Następnie, w zależności od ilości tychże wektorów dla poszczególnych liter, (którą zwróci funkcja ***length_w()*** z pliku "*length_w.m"*) tworzone są odpowiednie wektory wartości logicznych, przy wykorzystaniu funkcji ***repmat()***, które to składają się na wektor ***trainout***.

Sieć neuronowa

Sieć neuronowa została zaimplementowana przy użyciu funkcji ***feedforwardnet()*** z toolbox'a MATLABA *Deep Learning Toolbox* z argumentem '50' oznaczającym rozmiar ukrytych warstw sieci.

Dla odpowiedniego działania sieci ustawiono również inne parametry:

- adaptFcn = 'adaptwb';
- divideFcn = 'dividerand';
- divideMode = 'sample';
- layers{1}.transferFcn = 'logsig';
- layers{2}.transferFcn = 'tansig';
- trainFcn = 'trainlm';
- performFcn = 'mse';

Do nauki sieci wykorzystano funkcję ***train()***, której argumentami były wynik funkcji ***feedforwardnet()*** oraz wektory ***trainin*** i ***trainout*** opisane w poprzednim paragrafie.

Zestaw współczynników obliczonych przy użyciu sieci został zapisany do pliku "*ocr_final_8.mat*", który wykorzystywany jest do późniejszego porównania ze współczynnikami obrazu załadowanego przez użytkownika na przycisk "Rozpoznawanie tekstu".

Rozpoznawanie tekstu

Po załadowaniu pliku przez użytkownika, obraz poddawany jest takim samym operacjom (opisanym w paragrafie *Przygotowanie danych treningowych*) jak obrazy z datasetu wykorzystanego do nauki sieci. Zaleca się, aby wczytywany obraz zawierał na białym tle tekst w kolorze czarnym napisany czcionką Arial.

W skrócie: obliczone współczynniki dla każdego z obiektów znalezionych na obrazie zostają porównane ze współczynnikami wyznaczonymi dla każdej litery, dla której sieć została uprzednio nauczona. Odpowiednia litera zwracana jest przy największym możliwym dopasowaniu.

2.3 Dodatkowe funkcje

Ponadto w ramach aplikacji możliwe jest wyświetlenie efektu binaryzacji obrazu, ponowne wytrenowanie sieci neuronowej, rozpoznanie tekstu przy użyciu nowej sieci (za pomocą przycisku "Sprawdź nową sieć") oraz zapis wyniku rozpoznania tekstu do pliku.

3. Interfejs użytkownika

Projekt został zrealizowany jako program z GUI stworzonym za pomocą aplikacji MATLAB App Designer. Aby uruchomić aplikację należy w programie MATLAB przejść do głównego katalogu projektu, uruchomić plik „TextRecog.mlapp” i nacisnąć przycisk ‘Run’ (zielona strzałka) lub mając wybrana ścieżkę do GUI w MATLABIE - poprzez dwukrotne kliknięcie na „TextRecog.mlapp”.

W przypadku powodzenia powinien pojawić się poniższy ekran.

Rysunek 1: Ekran startowy aplikacji

W ramach GUI dostępne jest kilka przycisków.

Przycisk „Wczytaj obraz” pozwala na wczytanie obrazu z tekstem z pliku. Dla powodzenia rozpoznania tekstu, najlepiej jeśli będzie to plik z czarnym tekstem na białym tle napisany czcionką Arial. Przykładowy plik do załadowania o nazwie „TEST_AO.jpg” znajduje się w głównym katalogu projektu wraz z „TextRecog.mlapp”.

Po wybraniu i wczytaniu obrazu z pliku, powinien pojawić się on w lewej części ekranu, tak jak na poniższym obrazku.

Rysunek 2: Ekran po wczytaniu obrazu z pliku

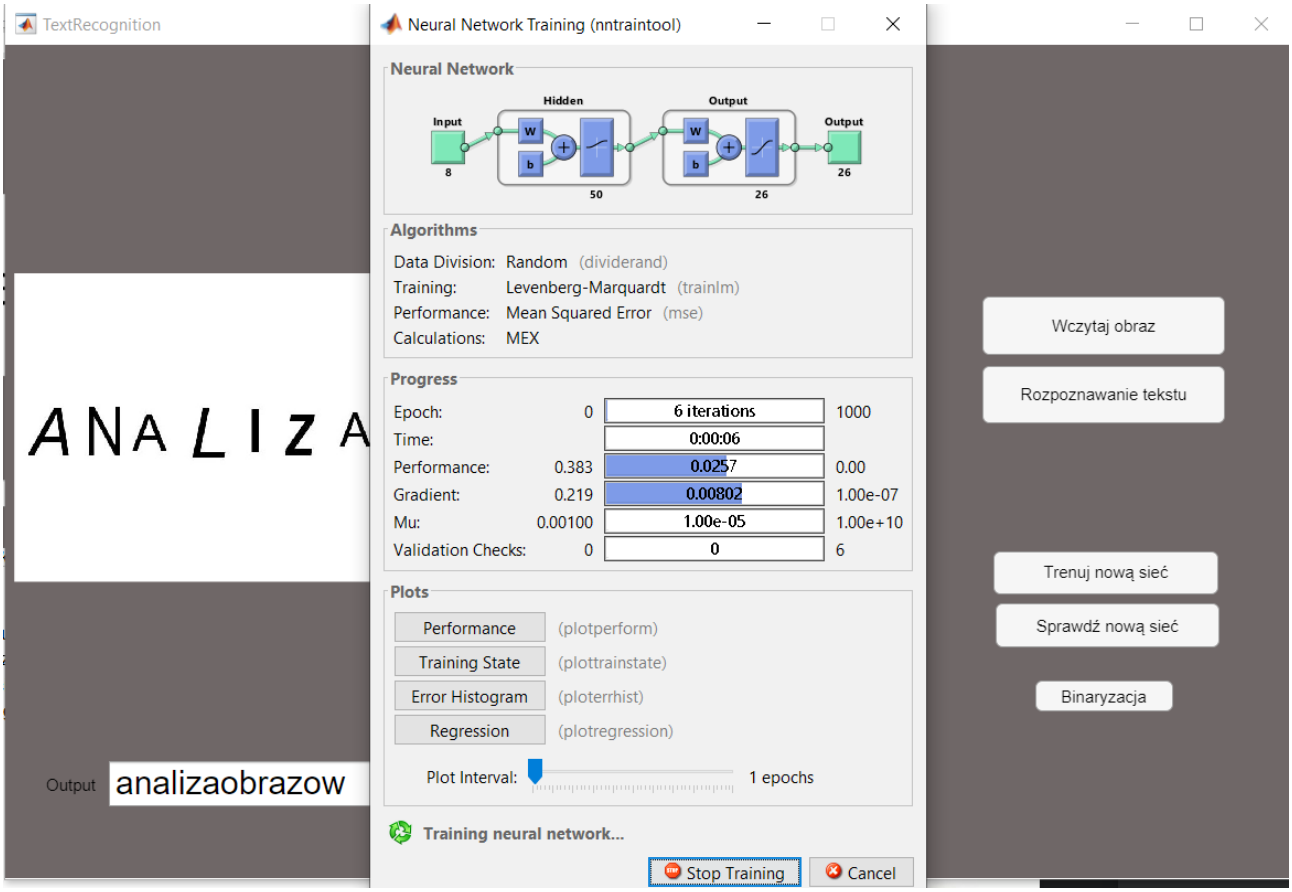
Następny z kolei - przycisk o nazwie „Rozpoznawanie tekstu” przeanalizuje wczytany obraz i zwróci w polu tekstowym „Output” rozpoznany tekst na podstawie zestawu współczynników załadowanych z początkowego uczenia sieci.

Zaraz obok pola tekstowego znajduje się przycisk „Zapisz wynik” umożliwiający zapis rozpoznanego tekstu do pliku tekstowego.

Rysunek 3: Ekran po kliknięciu przycisku "Rozpoznawanie tekstu"

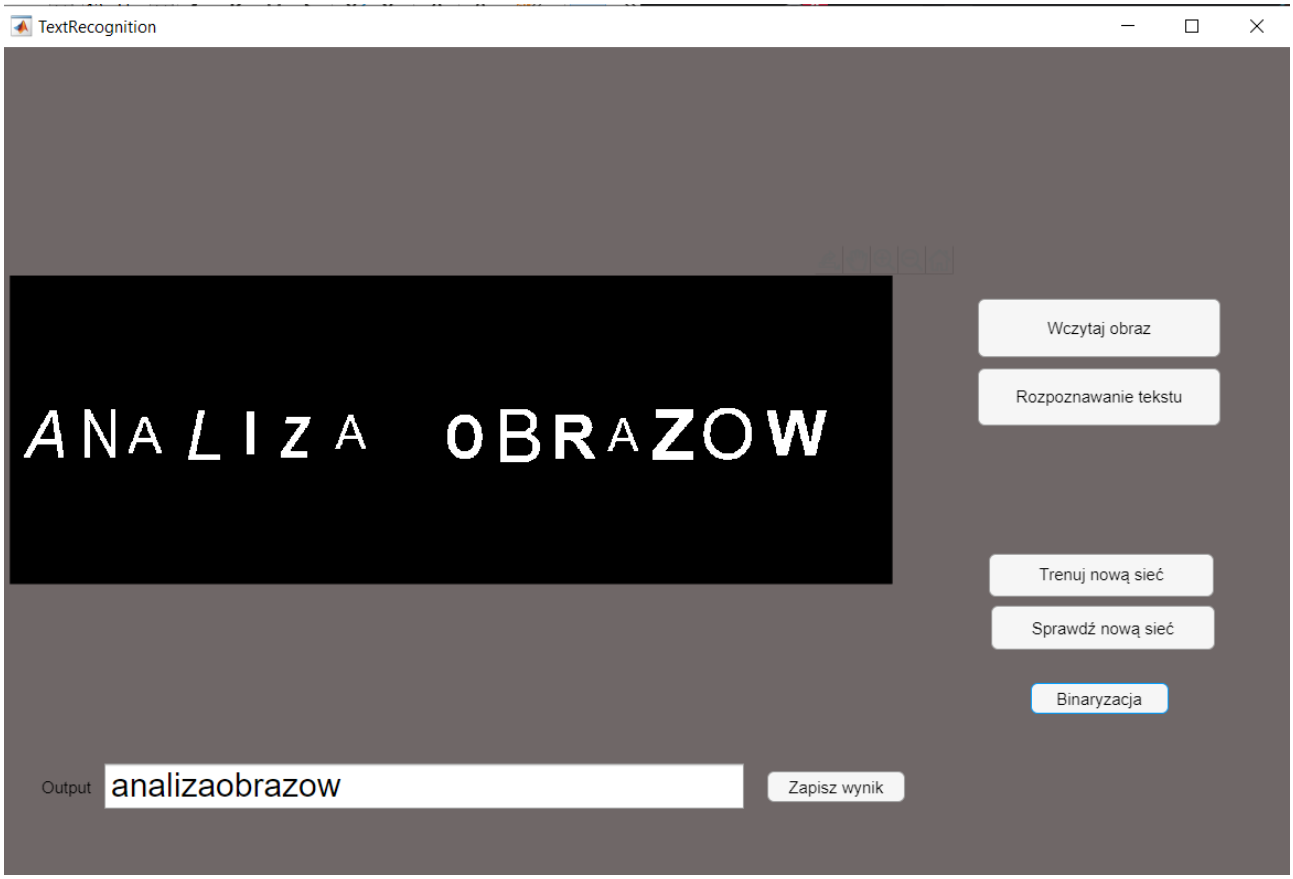
Kolejny przycisk „Trenuj nową sieć” uruchamia proces uczenia nowej sieci. Następujący po nim przycisk „Sprawdź nową sieć” rozpozna tekst przy użyciu nowego zestawu współczynników z nowej sieci, przeciwnie do przycisku „Rozpoznawanie tekstu”, który używał starego, wcześniejszego zestawu obliczonych współczynników.

Analogicznie - rozpoznany tekst zostanie wyświetlony w polu tekstowym „Output”.



Rysunek 4: Ekran po rozpoczęciu trenowania sieci

Ostatni z przycisków „Binaryzacja” pokazuje efekt binaryzacji na obrazie, który jest jednym z etapów obróbki obrazu poprzedzających przekazanie obrazu do procesu uczenia się sieci.



Rysunek 5: Obraz po binaryzacji

4. Wady projektu

- Program nie rozpoznaje małych liter, polskich znaków, liczb, symboli matematycznych i znaków interpunkcyjnych oraz odwróconego tekstu. Przy próbach wprowadzenia zaledwie samych cyfr do obecnego stanu rzeczy, nauczanie sieci kończyło się Validation Check’iem.
- Na ten moment jedyną czcionką, której litery są rozpoznawalne jest Arial, ponieważ podobnie, jak w przypadku rozbudowy rozpoznawania o cyfry, próba powiększenia datasetu o kolejne czcionki kończyła się niepowodzeniem.
- Pomiędzy rozpoznanymi słowami nie występują spacje.
- Warto także dodać, że inputem do naszego projektu powinny być obrazki zawierające tekst napisany czarną czcionką na białym tle, co zostało już wspomniane w paragrafie 2.2 odnośnie rozpoznawania tekstu.

5. Przykładowe usprawnienia

- Z racji, że na ten moment program rozpoznaje jedynie duże litery podstawowym usprawnieniem byłoby wprowadzenie rozpoznawania pozostałych znaków a więc małych liter, cyfr, symboli matematycznych i znaków interpunkcyjnych.
- Wprowadzenie rozpoznawania znaków pochodzących z innych czcionek.
- Uniezależnienie poprawności zwracanego wyniku od koloru czcionki i tła wprowadzanego skanu.
- Oddzielanie rozpoznanych słów spacją.

6. Podział obowiązków

Implementacja sieci neuronowej, przygotowanie zestawu liter, trenowanie sieci - Kamil Sudoł

GUI, trenowanie sieci - Patryk Śledź

Dokumentacja, trenowanie sieci - Jakub Strugała