WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Katedra Informatyki i Automatyki

Techniki multimedialne

Projekt
Rozpoznawanie cyfr ze zbioru MNIST

Rzeszów, 2019

Andrzej Szymanik

3EF-DI

Spis treści

1. Opis projektu

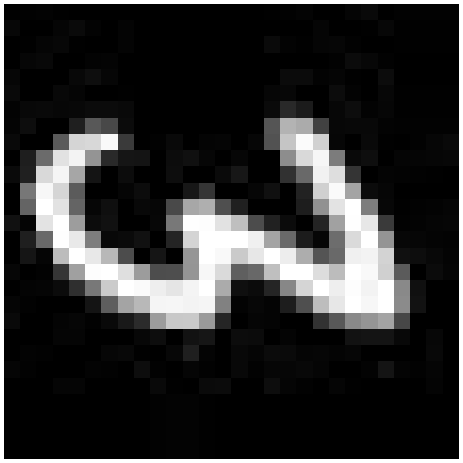
Głównym celem projektu było utworzenie sztucznej sieci neuronowej służącej do klasyfikacji cyfr ze zbioru MNIST w różnej rotacji.

W tym celu zaimplementowana została konwolucyjna sieć neuronowa.

Projekt został wykonany w języku python wykorzystując biblioteki keras i tensorflow.

Dane wykorzystane do uczenia sieci neuronowej zostały podzielone następująco :

- 20000 danych uczących (po 2000 na każdą klasę),
- 2000 danych testowych (po 200 na każdą klasę),
- Dane sprawdzające, które nie brały udziału w procesie nauki sieci.



Rysunek 1: Przykładowy element zbioru uczącego

Kod projektu został podzielony na dwie części. Pierwszą z nich jest część odpowiedzialna za stworzenie konwolucyjnej sieci neuronowej, wczytanie danych uczących oraz testowych oraz naukę sieci neuronowej.

Druga część projektu odpowiada za testowanie skuteczności sieci neuronowej na danych, które nie brały udziału w procesie uczenia. Polega ona na wczytaniu całego folderu zawierającego grafiki oraz przyporządkowanie każdej z nich do jednej z dziesięciu klas.

Pierwszym etapem było pobranie z internetu zbioru MNIST. Z uwagi na to, że było tam zawarte 70000 obrazków musiałem zredukować tą liczbę do 22000 z uwagi na długość nauki sieci neuronowej.

Następnie obróciłem część danych o określony kąt obrotu. Wykorzystałem do tego konsolę Linuxa oraz bibliotekę Imagemagick.

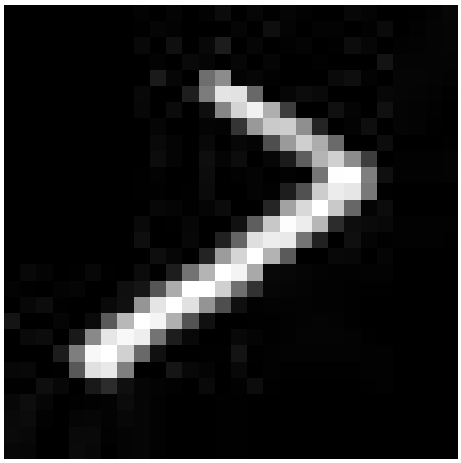
```
#!/bin/bash -e

CUR_DIR=`pwd`
cd "${1}"

        for file in *.jpg; do
            convert "${file}" -distort ScaleRotateTranslate 30 "${file}";
        done
cd CUR_DIR
```

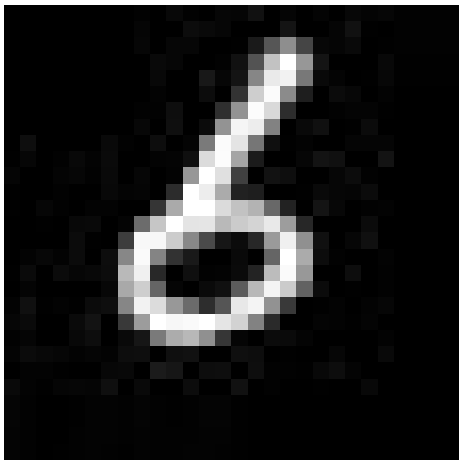
Rysunek 2: Skrypt obracajacy obraz o podany kąt

Zasadniczym problemem występującym podczas klasyfikacji danych jest fakt, że w danych uczących oraz testowych cyfry ‘1’ oraz ‘7’ są w wielu przypadkach niemal identyczne. Przez co sieć neuronowa nie potrafi w każdym przypadku rozróżnić tych dwóch cyfr od siebie.



Rysunek 3: Cyfra klasyfikowana jako 7

Dodatkowym problemem podczas klasyfikacji jest rozróżnianie cyfr ‘6’ oraz ‘9’ ze względu na fakt, że po obróceniu dowolnej z cyfr o 180 stopni, wizualnie wyglądają na inne cyfry. W tym przypadku sieć neuronowa osiąga skuteczność rzędu 58 – 68 %.



Rysunek 4: Cyfra 9 po rotacji o kąt 180 stopni

```
...: import os
...: from keras.preprocessing import image
...: classifier = load_model('/home/andrzej/Applications/MNIST/project.h5')
...: classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
...:
...: import numpy as np
...: success = 0
...: number = 1
...: i = 0
...: images = []
...: for img in os.listdir(os.getcwd()):
...:     img = image.load_img(img)
...:     img = image.img_to_array(img)
...:     img = np.expand_dims(img, axis=0)
...:     images.append(img)
...:
...: images = np.vstack(images)
...: classes = classifier.predict_classes(images,batch_size = 10)
...: for i in range (0,classes.size) :
...:     if classes[i]  == number:
...:         success = success + 1
...:
...: success = success / classes.size * 100
...: print(classes)
...: print(str(success) + ' %')
[1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1]
94.44444444444444 %
```

Rysunek 5 Przykładowe działanie programu dla folderu zawierającego cyfry '1'

Działanie programu polega na wczytaniu nauczonego modelu, a następnie całego folderu zawierającego dane, które nie zostały poddane procesowi uczenia. Program wyświetla do jakich klas przyporządkował poszczególne obrazy a następnie oblicza skuteczność dla całego zestawu danych

2. Biblioteki wykorzystane w aplikacji

2.1 Biblioteki zewnętrzne

Program został napisany w języku Python w wersji 3.7.3. Poniżej zamieszczone zostały wszystkie biblioteki zewnętrzne, które zostały wykorzystane w programie.

- numpy – biblioteka służąca do przetwarzania tablic oraz macierzy,
- keras - otwarto źródlowa biblioteka do sieci neuronowych służąca do szybkich eksperymentów z szeroko pojętą sztuczną inteligencją,
- Tensorflow – biblioteka programistyczna wykorzystywana w uczeniu maszynowym i głębokich sieciach neuronowych,
- os - moduł języka Python służący do wykonywania operacji systemych.

2.2 Metody modelu Sequential

add() - Metoda ta służy do dodania do modelu nowej warstwy. Jej poszczególne argumenty mogą przyjmować postać :

- *def Convolution2D (filters, padding, input_shape, activation_function)* - warstwa konwolucyjna za argumenty przyjmuje filtry konwolucyjne, macierz opisującą parametry wejściowe oraz funkcję aktywacji,
- *def MaxPooling2D (pool_size)* - za argument przyjmuje filtr poolingu, w przypadku filtru (2, 2) roździelczość naszego obrazu zmniejsza się o połowę,
- *def Flatten ()* - nie przyjmuje argumentów, zmienia kształt modelu na model jednoymiarowy,
- *def Dense (output_dim , activation_function)*
- *def compile(optimizer, loss_function, metrics)* - przyjmuje za argumenty optymalizator, funkcję straty oraz metrykę.
- *def fit_generator (training_set,steps_per_epoch,epochs,validation_data,validation_steps)* - przyjmuje za argumenty zbiór danych uczących, liczbę kroków w epoce, liczbę epok, dane testowe oraz ilość kroków walidacji. Odpowiada za proces uczenia się sieci neuronowej.

- *def save (filename)* - za argument przyjmuje nazwę pliku docelowego. Służy do zapisywania modelu w celu późniejszego użycia nauczonego modelu,
- *def predict_classes (images,batch_size)* – przewiduje wynik działania nauczonego modelu sieci neuronowej. Za argumenty przyjmuje zbiór obrazów oraz liczbę klas,
- *def load_model (modelPath)* – wczytuje nauczony model sieci neuronowej. Za argument przyjmuje ścieżkę do pliku.

ImageDataGenerator(rescale, shear_range, zoom_range, horizontal_flip) - konstruktor klasy ImageDataGenerator.

def Flow_from_directory(dir, target_size, batch_size, class_mode) - za argumenty przyjmuje katalog, docelowy oraz rozmiar poszczególnych danych. Class_mode z kolei odpowiada za to w jaki sposób należy przydzielić dane do różnych klas.

2.3 Metody klasy Image

- *def load_img (imagePath) - Metoda ta wczytuje plik graficzny do zmiennej. Przyjmuje za argument ścieżkę do pliku,*
- def img_to_array (image) - metoda ta zamienia wartość zmiennej image na tablicę.

2.4 Pozostałe metody

- *def expand_dims (img, axis)* – dodaje kolejny wymiar do tablicy wielowymiarowej. Za argumenty przyjmuje tablicę obrazu oraz oś, wobec której ma zostać dodany nowy wymiar. Jest to metoda zawarta w bibliotece numpy.
- *def getcwd()* - funkcja zwracająca obecny katalog roboczy.

W przypadku tego projektu nie było konieczne pisanie własnych funkcji. Wykorzystanie istniejących już funkcji występujących w bibliotece keras oraz tensorflow pozwoliło na zaprojektowanie i nauczenie od podstaw sieci neuronowej służącej do klasyfikacji cyfr.