Akademia Górniczo-Hutnicza

Wydział Informatyki, Elektroniki i Telekomunikacji

Kierunek Elektronika i Telekomunikacja - Systemy Wbudowane

Wykorzystanie OCR do Automatycznego Rozpoznawania Znaków na Tablicach Rejestracyjnych.

Radosław Borczuch

Kamil Godek

Szymon Frączek

Kraków, 10 czerwiec 2024

1 Cel projektu:

Celem naszego projektu była implementacja algorytmów uczenia maszynowego do odczytywania tablic rejestracyjnych pojazdów na podstawie zdjęć, z wykorzystaniem technologii OCR. Do realizacji tego zadania użyliśmy sieci neuronowych.

OCR (Optical Character Recognition): Technologia pozwalająca na automatyczne rozpoznawanie i przetwarzanie tekstu z obrazów do postaci cyfrowej.

TensorFlow: Otwartoźródłowa biblioteka do uczenia maszynowego, opracowana przez Google, umożliwiająca tworzenie, trenowanie i wdrażanie modeli sztucznej inteligencji.

Keras: Wysokopoziomowe API dla sieci neuronowych, działające na podstawie TensorFlow, ułatwiające szybkie tworzenie i trenowanie modeli głębokiego uczenia.

OpenCV: Biblioteka do przetwarzania obrazów i widzenia komputerowego, oferująca narzędzia do analizy obrazów, rozpoznawania wzorców oraz różnych zadań związanych z wideo i obrazami.

Projekt został zrealizowany w środowisku Google Colab i podzielony na trzy pliki. Pierwszy plik odpowiadał za obróbkę i przygotowanie zbioru danych do uczenia. Drugi plik zawierał implementację sieci neuronowej i służył do jej trenowania. Trzeci plik wykorzystywał wytrenowany model z drugiego pliku.

Do trenowania modelu wykorzystano zbiór obrazów zawierających zdjęcia tablic rejestracyjnych oraz informacje o segmentacji z witryny Roboflow, dostępny pod następującym linkiem: https://universe.roboflow.com/metehan-yasar/heyyy/dataset/1

Pliki projektu dostępne w repozytorium: https://github.com/RBorczuch/PlatesOCR

2 Implementacja.

2.1 Przygotowanie danych do uczenia.

W pliku "license_plates_extraction.ipynb" zaimplementowano program odpowiadający za dostosowanie obrazów do wymagań sieci neuronowej. Program przetwarza dane w następujących krokach:

1. **Pobranie danych z strony Roboflow.**

2. **Wycięcie obiektów na podstawie segmentacji i przeskalowanie do jednolitych wymiarów.**

3. **Wyrównanie histogramu, zamiana obrazu na odcienie szarości i progowanie barw.**

4. **Zastosowanie gotowego modelu EasyOCR do nadania nazw obrazom.** Niestety EasyOCR nie poradził sobie z odczytem większości z tablic i pliki musieliśmy nazywać ręcznie.
5. **Skompresowanie obrazów do pliku zip i pobranie.**

2.2 Przygotowanie sieci neuronowej i uczenie modelu.

W pliku „PlatesOCR_learning.ipynb” zaimplementowano program zawierający sieć neuronową i odpowiadający za jej uczenie. Program przetwarza dane w następujących krokach:

1. **Rozpakowanie danych z pliku zip.**
2. **Przygotowanie obrazów do obróbki i pobranie etykiet z ich nazw.**
3. **Podział danych na zestawy treningowy 80% i walidacyjny 20%.**
4. **Liczenie unikalnych obrazów, etykiet oraz znaków.**
   Nasz zestaw danych zawierał 250 unikalnych obrazów i etykiet obejmujących 34 różne znaki. Baza danych, na której trenujemy model, jest stosunkowo mała. Nie wszystkie z wymienionych obrazów są unikalne, ponieważ zostały poddane augmentacji danych. Augmentacja danych to technika stosowana w uczeniu maszynowym, polegająca na sztucznym zwiększeniu liczby przykładów w zbiorze treningowym poprzez wprowadzenie różnych modyfikacji do istniejących danych.
5. **Wizualizacja części danych treningowych.**

6. **Implementacja sieci neuronowej o następującej budowie:**

Layer (type)	Output Shape	Param #	Connected to
image (InputLayer)	[(None, 200, 50, 1)]	0	[]
Conv1 (Conv2D)	(None, 200, 50, 32)	320	['image[0][0]']
pool1 (MaxPooling2D)	(None, 100, 25, 32)	0	['Conv1[0][0]']
Conv2 (Conv2D)	(None, 100, 25, 64)	18496	['pool1[0][0]']
pool2 (MaxPooling2D)	(None, 50, 12, 64)	0	['Conv2[0][0]']
reshape (Reshape)	(None, 50, 768)	0	['pool2[0][0]']
dense1 (Dense)	(None, 50, 64)	49216	['reshape[0][0]']
dropout (Dropout)	(None, 50, 64)	0	['dense1[0][0]']
bidirectional (Bidirectional)	(None, 50, 256)	197632	['dropout[0][0]']
bidirectional_1 (Bidirectional)	(None, 50, 128)	164352	['bidirectional[0][0]']
label (InputLayer)	[(None, None)]	0	[]
dense2 (Dense)	(None, 50, 36)	4644	['bidirectional_1[0][0]']
ctc_loss (CTCLayer)	(None, 50, 36)	0	['label[0][0]', 'dense2[0][0]']

Total params: 434660 (1.66 MB)
Trainable params: 434660 (1.66 MB)
Non-trainable params: 0 (0.00 Byte)

- **Wejścia modelu:**

  Obraz wejściowy (input_img) o kształcie (img_width, img_height, 1), gdzie 1 oznacza kanał skali szarości.

  Etykiety (labels) o zmiennej długości, które są prawdziwymi wartościami do porównania z przewidywaniami modelu.
- **Bloki konwolucyjne:**

  Pierwszy blok konwolucyjny: Składa się z warstwy konwolucyjnej z 32 filtrami, aktywacji ReLU, inicjalizacji he_normal i paddingu 'same', co zachowuje rozmiar obrazu. Następnie zastosowana jest warstwa MaxPooling2D, która zmniejsza rozmiar mapy cech.

  Drugi blok konwolucyjny: Składa się z warstwy konwolucyjnej z 64 filtrami, aktywacji ReLU, inicjalizacji he_normal i paddingu 'same'. Następnie zastosowana jest kolejna warstwa MaxPooling2D.
- **Przekształcenie danych:**

  Po dwóch warstwach MaxPooling2D, rozmiar map cech jest zmniejszony 4-krotnie. Dane są przekształcane do nowego kształtu za pomocą warstwy Reshape. Następnie przechodzą przez warstwę Dense z 64 jednostkami i aktywacją ReLU oraz warstwę Dropout, co pomaga w regularizacji modelu.
- **Warstwy RNN:**

  Model zawiera dwie dwukierunkowe warstwy LSTM. Pierwsza warstwa ma 128 jednostek, a druga 64 jednostki, obie z return_sequences=True i dropoutem 0.25, co pomaga w radzeniu sobie z długoterminowymi zależnościami w sekwencjach.
- **Warstwa wyjściowa:**

  Warstwa Dense z aktywacją softmax, która generuje prawdopodobieństwa dla każdego z unikalnych znaków plus jeden dodatkowy dla oznaczenia końca sekwencji.
- **Warstwa CTC:**

  Warstwa CTCLayer jest dodana na końcu modelu, aby obliczać stratę CTC na każdym kroku treningowym. To pozwala modelowi na prawidłowe trenowanie na danych sekwencyjnych. CTC eliminuje konieczność ręcznego wyrównywania danych wejściowych i wyjściowych.

7. **Trenowanie modelu i jego zapisywanie.**

   Trenujemy model przez 300 epok.

8. Predykcja i wizualizacja danych wyjściowych:

Predykcja jest dokładna tylko niektóre z tablic zostały błędnie zidentyfikowane.

Prediction: 01DJ6 [UNK][UNK]

Prediction: 01ABH447

Prediction: 01EL122

Prediction: 01AHM001

Prediction: 01DMJ09

Prediction: 01DFZ96

Prediction: 01BVF33

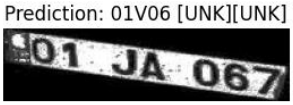
Prediction: 01UM176

Prediction: 01SU001

Prediction: 01AGM485

Prediction: 01DRT64

Prediction: 01V06 [UNK][UNK]



Prediction: 01EV151

Prediction: 01B7307

Prediction: 01AHV11[UNK]

Prediction: 01TA002

Prediction: 01D2881

Prediction: 01ETN75

Prediction: 01DKA80

Prediction: 01ALL229

Prediction: 01AAK545

Prediction: 01SU001

Prediction: 01AGU35

Prediction: 01AHK734

Prediction: 01HL994

Prediction: 01KH964

Prediction: 01AG849

Prediction: 01DHC63

Prediction: 01APL40

Prediction: 01AGM485

Prediction: 01ETN75

Prediction: 01AZS69

2.3 Testowanie modelu.

Ostatni plik „PlatesOCR_model_test.ipynb” odpowiada za testowanie wyuczonego modelu. Program przetwarza dane w następujących krokach:

1. **Przekształcenie obrazu wejściowego:**

2. **Predykcja obrazu na podstawie modelu:**

3 Wnioski:

Nasz wytrenowany model przewyższa EasyOCR w odczytywaniu znaków na tych konkretnych obrazach. Jest on zoptymalizowany wyłącznie do tureckich tablic rejestracyjnych, ponieważ zestaw danych składał się wyłącznie z tego typu tablic. Model ten mógłby osiągać lepsze wyniki, gdyby był wytrenowany na większym i bardziej zróżnicowanym zestawie danych.

Możliwe jest także zastosowanie naszego modelu w połączeniu z modelem opartym na YOLO. Model YOLO (You Only Look Once) wykrywałby tablice rejestracyjne na obrazach, a nasz model służyłby do ich dekodowania ich znaków.