Analiza obrazu
Projekt

Jan Bizoń

22 stycznia 2023

1 Opis

1.1 Wstęp

Program służący rozpoznawaniu tekstu oraz cyfr. Projekt zakłada rozpoznawanie jedynie tekstu zawierającego znaki występujące w angielskim alfabecie. Program był testowany na wsl-u.

1.2 Link

- Repozytorium projektu: https://github.com/PenumbraPL/analiza_obrazu
- Aby miec możliwość stworzenia modelu na nowo potrzebna jest instalacja : tego w katalogu /models z nazwą : "*A_Z Handwritten Data.csv*"

1.3 Jak uruchomić?

```
python3 ./main_window.py
```

Aby uruchomić kompilacje modelu należy wejść do katalogu models a następnie:

```
python3 ./train_model.py
```

Czas kompilacji ok. 1 godz.

1.4 Jak przygotować program do działania?

Przykład instalacji za pomocą minicond-y. Pozwala on na szybką instalacje , aktualizacje i deinstalacje pakietów python.

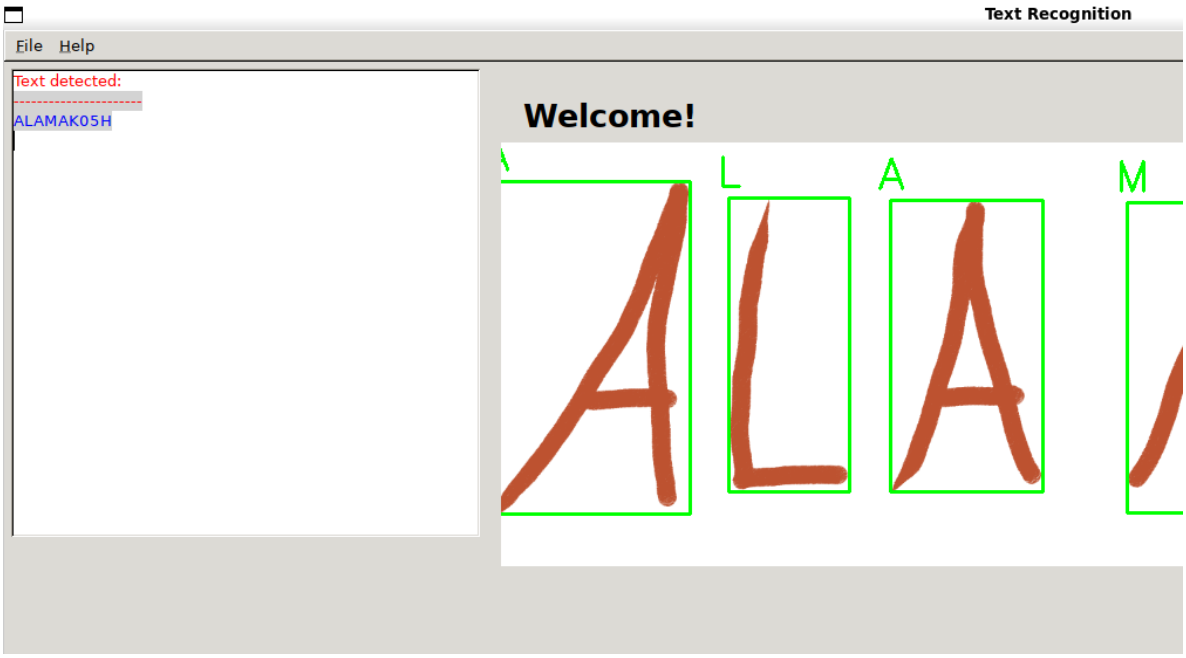
```
conda create -n (name) python=3.7
conda activate (name)
conda install wxpython
conda install matplotlib
conda install -c anaconda scikit-learn
conda install -c conda-forge tensorflow
conda install -c conda-forge opencv
```

1.5 Jak usunąć program?

```
conda deactivate (name)
conda env remove -n (name) # tylko po wyjsciu z srodowiska
```

1.6 Jak działa program?

Po uruchomieniu programu w lewym górnym rogu mamy możliwość wybrania obrazu w formacie .png po czym obraz jest analizowany i wykryty tekst jest zwracany w postaci testu w przeznaczonym do tego miejscu. Dane możemy wczytać za pomocą skrótu klawiszowego CTRL + H. Przykładowe dane wejściowe zostały udostępnione wraz z projektem. Znajdują się one w zakładce /data Na obrazie będą są przedstawione propozycje na to jaka jest to litera.



Rysunek 1: Obraz przedstawiający poprawnie działający program.

1.7 Autorzy

- Jan Bizoń

1.8 Wykorzystane materiały

- https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format
- http://yann.lecun.com/exdb/mnist/

2 Projekt

2.1 Struktura kodu

- main_window.py - aplikacja zawierająca główne okienko oraz jego funkcje
- ocr_lib.py - plik zawierający funkcje przetwarzającą obraz do rozpoznania oraz samo rozpoznanie
- model - skompilowany model sieci neuronowej
- helpers.py - zawiera funkcje pomocnicze wykorzystywane przy tworzeniu modelu - m.in. funkcje wczytujące dane
- resnet.py - zawiera funkcje implementujące residualną sieć neuronową
- train_model.py - zawiera zaimplementowany model sieci neuronowej

- result.png - obraz przedstwiający jakość uczenia
- quality.png - przedstawia przykładowe dane oraz to co zostało do nich przyporządkowane
- A_Z Handwritten Data.csv - należy pobrać z linku - dane wejściowe zawierający litery A-Z
- obrazy - przykładowe obrazy pozwalające przetestować aplikację

```
text_rec
├── main_window.py
├── ocr_lib.py
├── data
│   ├── obrazy
│   │   ....
│   └──
└── models
    ├── model
    ├── helpers.py
    ├── resnet.py
    ├── train_model.py
    ├── result.png
    ├── quality.png
    └── A_Z Handwritten Data.csv
```

2.2 Funkcje

```
# Funkcja    adujca    dane – A–Z z pliku
def load_az_dataset(datasetPath)

# Funkcja    adujca    dane – 0–9 bezpo rednio z Tensorfloww
def load_mnist_dataset()

# Funkcja sklejaj ca obrazy
def build_montages(image_list, image_shape, montage_shape)

# Jednen z blok w neuron w
def convolutional_block(x, filter)

# Jednen z blok w neuron w
def identity_block(x, filter)

# Funkcja implementuje residualn  sie  neuronow
def ResNet34(shape = (32, 32, 3), classes = 10)

# Zaokr gla liczb  number do wielokrotno ci multiple
def round_to_multiple(number, multiple)

# Skaluje obraz albo ze wzg  du na szeroko    albo ze wzgl du na wysoko
# Wysoko   i szeroko    podawane s  w pikselach
def resize(image, width=None, height=None)

# Funkcja odpowiedzialna za obliczenia przetwarzaj ce obraz
# Przystosowuje ona r wnie  dane wej ciowe do modelu
# A nast pnie dane te przekazywane s  do modelu aby okre li
# liter  jak  wykrywany
# Zwracana jest informacja o jako ci dopasowania oraz litery do wy wietlenia
# w kolejno ci wyst powania
def ocr(filepath)
```

Przygotowanie danych Po pobraniu danych z internetu dostosowywane są w taki sposób aby zostały przyjęte do sieci neuronowej w postaci obrazów o rozmiarze 32x32 w postaci czarno-białej. Cyfry dostępne są z bazy danych MNIST, natomiast litery należy pobrać z innego źródła gdzie zostały już przygotowane w taki sposób aby przypominały cyfry z bazy.

Następnie obrazy przepuszczamy dzielimy na testowe i treningowe. Dane treningowe muszą przejść jeszcze proces przekształceń geometrycznych - przesunięć , obrotów itp. aby zwiększyć ilość próbek nie zwiększając obrazów jakie musimy przechowywać w pamięci.

2.3 Tworzenie modelu

Nasz model jest przykładem sieci residualnej. Nasz model zawiera jedynie jedną warstwę "liczącą". Spowodowane to jest długim czasem przetwarzania danych które sięgały nawet 2h dla jednego epoch. Przy jednym bloku czas liczenia zmalał zalednie do 10 min na epoch.

Więcej zysku mamy ze zwiększenia epoch niż ilości bloków liczacych.

Rysunek 2: Jakość modelu wykorzystanego w tym projekcie.

2.4 Analiza obrazu

W procesie analizy obrazu obraz wejściowy zamieniamy na skalę szarości. Dane następnie przechodzą przez filtr wyostrzający z wagami ujemnymi. Następnie dochodzi do wykrycia krawędzi za pomocą algorytmu Cannego - polgającego na wyliczeniu gradienów między pikselami. Wykorzystuje on również pewne bariery które pozwalają na odrzucenie warunków skrajnych. Tak uzyskany obraz posłuży nam do wyznaczenia ram dla każdej litery. Następnie za pomocą progu otsu oddzielimy litery od tła i po przeskalowaniu do wielkości obrazu z procesu uczacego przekarzemy je do naszego modelu. Po dokonaniu predykcji dane sortujemy i zwracamy na ekran.

2.5 Wnioski/Problemy

Program jest w stanie wykryć jedynie litery które nie wymagają segmentacji. Ponadto ma on problemy z niestandardowymi czcionkami. Problemu również mu sprawia wykrywanie nowych linii oraz spacji. Najmniejsza precyzją charakteryzują się litery takie jak O Q oraz 0.

	precision	recall	f1-score	support
0	0.27	0.40	0.33	1381
1	0.99	0.98	0.99	1575
2	0.95	0.95	0.95	1398
3	0.99	0.98	0.98	1428
4	0.91	0.95	0.93	1365
5	0.50	0.95	0.66	1263
6	0.97	0.98	0.97	1375
7	0.98	0.99	0.99	1459
8	0.96	0.98	0.97	1365
9	0.98	0.98	0.98	1392
A	0.99	0.99	0.99	2774
B	0.97	0.99	0.98	1734
C	0.99	0.97	0.98	4682
D	0.87	0.99	0.93	2027
E	0.95	1.00	0.97	2288
F	0.99	0.99	0.99	232
G	0.97	0.95	0.96	1152
H	0.97	0.97	0.97	1444
I	0.98	0.99	0.98	224
J	0.99	0.95	0.97	1699
K	0.96	0.99	0.98	1121
L	0.99	0.98	0.99	2317
M	0.98	0.99	0.99	2467
N	0.99	0.99	0.99	3802
O	0.93	0.85	0.89	11565
P	1.00	0.99	0.99	3868
Q	0.96	0.99	0.97	1162
R	0.99	0.99	0.99	2313
S	0.99	0.88	0.93	9684
T	1.00	0.99	0.99	4499
U	0.99	0.97	0.98	5802
V	0.96	1.00	0.98	836
W	0.98	0.99	0.98	2157
X	0.98	0.99	0.99	1254
Y	0.97	0.97	0.97	2172
Z	0.94	0.95	0.95	1215
accuracy			0.94	88491
macro avg	0.94	0.96	0.94	88491
weighted avg	0.95	0.94	0.95	88491

Rysunek 3: Pewność modelu.