

Bevezetés a mesterséges intelligenciába

Captcha megoldó program

Projekt dokumentáció

Ekart Csaba

2019. május 4.

Tartalomjegyzék

1. Bevezetés	1
2. Működtetés	1
3. A feladványok generálása	1
4. A tanuló algoritmus	2
5. Tanulás	2
6. Végkövetkeztetés	3

1. Bevezetés

A specifikációban leírtak szerint álltam neki a CAPTCHA megoldó program létrehozásához. A korábban már általam hivatkozott Adam Geitgey által írt Mediumos cikk volt a fő ihlet forrásom. A korábban feltöltött "részleges" megoldással sajnos elakadtam, ezért újra kezdtem az egészet. Azt még a TensorFlow CAPTCHA solving tutorial series című YouTube-os videosorozat alapján készítettem. [1] A programot csak Linux alól teszteltem.

A megoldásomhoz a Python 3-as verziójára, az OpenCV és Keras frameworkre, valamint a Google TensorFlow libjére volt szükségem. Ezen dependenciák feltelepítése után lehet futtatni az algoritmust. [2]

2. Működtetés

1. Feltelepítjük a szükséges libeket [2]
2. Terminalban elnavigálunk a projekt mappába.
3. A `python3 train.py` paranccsal elindítjuk
4. A futás ideje körülbelül két perc, az eredménye a 3. ábrán látható

3. A feladványok generálása

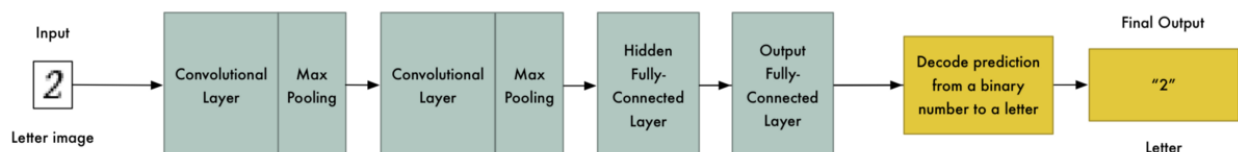
A CAPTCHA feladatok előállítására már önmagában egy érdekes feladat. Mivel rendkívül nehéz és időigényes lenne egyesével kitalálni ezeket más utat kellett találnom. Első nekifutásra találtam egy generátort, mellyel dolgozni is elkezdtem [3], s pár egyszerű módosítás után remek CAPTCHA-kat tudtam generálni. A probléma

a módszerrel az volt, hogy a programmal generált CAPTCHA-k elég bonyolultak, így nagyobb számítási igény lenne szükséges a tanításhoz, mivel ez a kapacitás nem állt rendelkezésemre, így végül más módot kellett találnom. Végül találtam egy kifejezetten CAPTCHA megoldáshoz szánt mélytanulási tanuló anyag forrást. Ezt letöltöttem és használtam a programomhoz. [4]

A Mediumos cikkben és a talált példa egyaránt 10000 CAPTCHA-t bocsájt rendelkezésre a tanúláshoz, melyek már eredendően szegmentálva vannak, így ezzel külön már nem kellett foglalkoznom. Szegmentálást egyébként az OpenCV framework segítségével tudtam volna elvégezni. [5] A 10000 CAPTCHA-ból átlagosan betűnként 2900-3500 egyforma betű eredmény jött ki. Mivel ennek már a fájlkezelőben való kezelése is gondot okozott, úgy döntöttem meg tizedelem és minden betűből csak az első 300-at használom a tanításhoz. Így egész tűrhető sebesség mellett tudtam tesztelni az aktuális kódot, és a memória korlátozás miatt a kernel nem lőtte ki a TensorFlow-t, de a kívánt pontosság megmaradt.

4. A tanuló algoritmus

A tanuló algoritmust a generátorhoz mellékelt leírás [3] és a Mediumos cikk [5] alapján írtam.



1. ábra. A tanuló algoritmus modellje

A tanuló algoritmus cikkben mellékelt ábrája alapján a modell kódja:

```

65 # NEURÁLIS HÁLÓ LÉTREHOZÁSA
66 with train_graph.as_default():
67     # Layer1 - Convolutional
68     conv_layer1 = tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding='SAME', name = 'conv1')
69     relu_layer1 = tf.nn.relu(conv_layer1, name = 'relu1')
70     max_pool_layer1 = tf.nn.max_pool(relu_layer1, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding='SAME', name = 'pool1')
71
72     # Layer2 - Convolutional
73     conv_layer2 = tf.nn.conv2d(max_pool_layer1, W2, strides=[1, 1, 1, 1], padding='SAME', name = 'conv2')
74     relu_layer2 = tf.nn.relu(conv_layer2, name = 'relu2')
75     max_pool_layer2 = tf.nn.max_pool(relu_layer2, ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding='SAME', name = 'pool2')
76
77     # Layer3 - Fully_Connected (Don't forget to flatten the previous layer)
78     flatten_layer3 = tf.contrib.layers.flatten(max_pool_layer2)
79     fc_layer3 = tf.contrib.layers.fully_connected(flatten_layer3, 500, activation_fn=tf.nn.relu, scope = 'fc1')
80
81     # Layer4 - Fully_Connected
82     fc_layer4 = tf.contrib.layers.fully_connected(fc_layer3, n_cls, activation_fn=None, scope = 'fc2')
83     print(fc_layer4)
84

```

2. ábra. A tanuló modell programkódja

A tanuló kódok további részét szintén a generátorhoz mellékelt leírás alapján írtam meg [3]. 10 átfutás a tanuló fájlokon elegendő ahhoz, hogy 99% feletti pontosságot érjünk el. A 10 átfutás során a logban láthatjuk is az aktuális értékeket.

5. Tanulás

A tanulás logjáról készült képek alább láthatóak. Amit biztosan megtudtunk:

- 2 perc alatt tesztekkel együtt lefut a tanulási idővel együtt.
- A mellékelt teszteseteket minden esetben hibátlanul oldotta meg.
- 99% feletti pontosságot értünk el

```

ekaktusz@eab-laptop: ~/Asztal/eab
Instructions for updating:
Use keras.layers.flatten instead.
Tensor("fc2/BiasAddV", shape=(7, 18), dtype=float32)
Tanulasi folyamat inditasa...
2019-05-10 22:09:44.261981: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2019-05-10 22:09:44.282277: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2394385000 Hz
2019-05-10 22:09:44.282887: I tensorflow/compiler/xla/service/service.cc:158] XLA service 0x1a58f8b0 executing computations on platform Host. Devices:
2019-05-10 22:09:44.282934: I tensorflow/compiler/xla/service/service.cc:158] StreamExecutor device (0): <undefined>, <undefined>
0 . epoch kesz:
Aktualis cost: 0: 2.688305
1 . epoch kesz:
Aktualis cost: 1: 1.536953
2 . epoch kesz:
Aktualis cost: 2: 0.445526
3 . epoch kesz:
Aktualis cost: 3: 0.151641
4 . epoch kesz:
Aktualis cost: 4: 0.072992
5 . epoch kesz:
Aktualis cost: 5: 0.044187
6 . epoch kesz:
Aktualis cost: 6: 0.038810
7 . epoch kesz:
Aktualis cost: 7: 0.022763
8 . epoch kesz:
Aktualis cost: 8: 0.018533
9 . epoch kesz:
Aktualis cost: 9: 0.015852
2019-05-10 22:10:24.718842: W tensorflow/core/framework/allocator.cc:124] Allocation of 172800000 exceeds 10% of system memory.
2019-05-10 22:10:25.321201: W tensorflow/core/framework/allocator.cc:124] Allocation of 108800000 exceeds 10% of system memory.
Tanulasi folyamat vege.
Pontossag: 99.888999816742 %
Teszteset inditasa...
Tesztelendo fajok:
['test/HGANAB.png', 'test/XRPGFU.png', 'test/KFLARG.png', 'test/RFAENU.png', 'test/AHCRCP.png']
Eredeti: HGANAB
Tipp : HGANAB
HELYES
Eredeti: XRPGFU
Tipp : XRPGFU
HELYES
Eredeti: KFLARG
Tipp : KFLARG
HELYES
Eredeti: RFAENU
Tipp : RFAENU
HELYES
Eredeti: AHCRCP
Tipp : AHCRCP
HELYES
Helyes tippek szama: 5 , Teszt pontossaga: 100.0 %.
ekaktusz@eab-laptop: ~/Asztal/eab

```

3. ábra. Futtatás eredménye Linuxon.

6. Végkövetkeztetés

Az egyszerűbb CAPTCHA feladatok manapság semmiféle védelmet nem nyújtanak a különféle kártékony programok és botok ellen. A projektben használt CAPTCHA-k feltörése a Mediumos cikk alapján - és a megvalósítás után látva - elegendő ismeretekkel 15 perc alatt megtehető, de egy régi Linuxos lappal és semmiféle korábbi tapasztalattal is pár óra alatt rendkívül magas pontossággal feltörhető. Nem véletlen, hogy az utóbbi években a klasszikus CAPTCHA feladványok elhagyásra kerültek.

Hivatkozások

[1] https://www.youtube.com/watch?v=4D5RN2yKlG4&list=PLbM09c_jUD456277j2fHAUp19xfxIAx0

[2] <https://github.com/hseki/learning-keras/wiki/How-to-install-Keras-to-Ubuntu-18.04>

[3] <https://github.com/lepture/captcha>

[4] <https://github.com/aravindmanoharan/Solving-Captcha-Using-Tensorflow>

[5] https://medium.com/@ageitgey/how-to-break-a-captcha-system-in-15-minutes-with-machine-learning-dbebfbclid=IwAR1rAIlvf4IwPh7zSJvAPdEZU_FzRwFgAjm_WqFTgyqciqiI4dgGAzWdLqE