

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, előkészületek	2
4. A tanuló algoritmus	2
5. Tanulás	3
6. Véggövetkeztetés	5

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 a tanulást
4. A futás ideje pár másodperc, az eredménye a 4. ábrán látható
5. A `python3 test.py` paranccsal elindítjuk a tesztet. Ennek eredményét a 5 képen láthatjuk.

3. A feladványok generálása, előkészületek

A CAPTCHA feladatok előállítása már önmagában egy érdekes feladat. Mivel rendkívül nehéz és időigényes lenne egyesével kitalálni ezeket, ezért 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 CAPTCHÁ-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 mélytanulós tanuló anyag forrást. Ezt letöltöttem és használtam a programomhoz. [4] Az ebben talált CAPTCHÁ-k a lehető legegyszerűbbek: nem tartalmaznak "zavaró tényezőket" tehát nincs bennük zaj, és különféle színes hátterek / vonalak, valamint az egyszerűség kedvéért számokat és bizonyos hasonló betűket nem tartalmaz. Erre azért van szükség mert például a torzított O és D rendkívül hasonló lehet, ami rontja az esélyt, hogy a felhasználó gyorsan megoldja a feladványt.

A Mediumos cikkben és a talált példa egyaránt 10000 CAPTCHA-t bocsájt rendelkezésre a tanuláshoz, 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 használok a tanításhoz, úgys mindössze 18 különböző karaktert kellett csak elemezni. 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.

Egy a cikkből "lopott" algoritmus szükséges a képek átméretezéséhez, mely gyakran van szükség a feladat megoldásában:

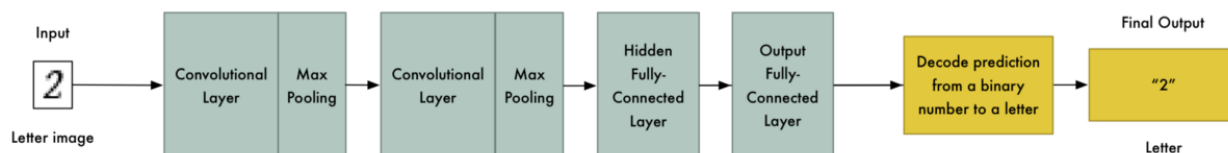
```
4 # Ez egy előre megadott segédfüggvény a képek átméretezéséhez
5 def resize(image, width, height):
6     (h, w) = image.shape[:2]
7     if w > h:
8         image = imutils.resize(image, width=width)
9     else:
10        image = imutils.resize(image, height=height)
11    padW = int((width - image.shape[1]) / 2.0)
12    padH = int((height - image.shape[0]) / 2.0)
13    image = cv2.copyMakeBorder(image, padH, padH, padW, padW,
14                               cv2.BORDER_REPLICATE)
15    image = cv2.resize(image, (width, height))
16    return image
```

1. ábra. Képek újraméretezésére szolgáló függvény

4. A tanuló algoritmus

A tanuló algoritmust a generátorhoz mellékelt leírás [3] és a Mediumos cikkben [5] lévő diagram, illetve kód alapján írtam.

A modellben kétszer szerepel egymás után egy konvolúciós réteg max-pollinggal majd egy rejtett és egy output réteg.



2. ábra. A tanuló algoritmus modellje

A tanuló algoritmus cikkben mellékelt ábrája alapján a modell kódja:

```
44 # NEURÁLIS HÁLÓ
45
46 # Build the neural network!
47 model = Sequential()
48
49 # 1. konvolúciós réteg maxpoolinggal
50 model.add(Conv2D(20, (5, 5), padding="same", input_shape=(20, 20, 1), activation="relu"))
51 model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
52
53 # 2. konvolúciós réteg maxpoolinggal
54 model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
55 model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
56
57 # Rejtett réteg
58 model.add(Flatten())
59 model.add(Dense(500, activation="relu"))
60
61 # Kimeneti réteg
62 model.add(Dense(18, activation="softmax"))
63
64 # Tanulás
65
66 print("Tanulas inditasa...")
67 model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
68 model.fit(X_train, Y_train, validation_data=(X_test, Y_test), batch_size=18, epochs=5, verbose=1)
69 model.save(MODEL_NAME)
70 print("Tanulas kesz! Ha tesztelni akarsz az eredmenyt a test.py scripttel megeleheted!")
```

3. ábra. A tanuló modell programkódja, Kerasban

A tanuló kódok további részét szintén a generátorhoz mellékelt leírás és a Mediumos cikk alapján írtam. [3]. 5 átfutás a tanuló fájlkon elégendő ahhoz, hogy 99% feletti pontosságot érjünk el. A 5 átfutás során a logban láthatjuk is az aktuális értékeket.

5. Tanulás és teszt

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

- Körülbelül pár másodperc alatt futtatható minden a programban
- A mellékelt teszteseteket szinte minden esetben hibátlanul oldotta meg, mindössze 1x tévedett nálam.
- 99% feletti pontosságot értünk el.

```
ekaktusz@eab-laptop: ~/Asztal/eab/Uj$ python3 train.py
Using TensorFlow backend.
Fájlok beolvasása...
Neuralis halo telepítése...
Instructions for updating:
Colocations handled automatically by placer.
Tanulás indítása...
WARNING:tensorflow:From /home/ekaktusz/.local/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 4050 samples, validate on 1350 samples
Epoch 1/5
2019-05-15 00:01:34.633229: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2019-05-15 00:01:34.658197: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2394375000 Hz
2019-05-15 00:01:34.658151: I tensorflow/compiler/xla/service/service.cc:150] XLA Service 0x382a800 executing computations on platform Host. Devices:
2019-05-15 00:01:34.658651: I tensorflow/compiler/xla/service/service.cc:158] StreamExecutor device (0): <undefined>, <undefined>
2019-05-15 00:01:35.074331: W tensorflow/core/framework/allocator.cc:124] Allocation of 19800000 exceeds 10% of system memory.
2019-05-15 00:01:35.074376: W tensorflow/core/framework/allocator.cc:124] Allocation of 19800000 exceeds 10% of system memory.
2019-05-15 00:01:35.081257: W tensorflow/core/framework/allocator.cc:124] Allocation of 17040000 exceeds 10% of system memory.
10/4050 [.....] - ETA: 1:41 - loss: 2.8906 - acc: 0.05562019-05-15 00:01:35.098646: W tensorflow/core/framework/allocator.cc:124] Allocation of 19800000 exceeds 10% of system memory.
2019-05-15 00:01:35.098991: W tensorflow/core/framework/allocator.cc:124] Allocation of 19800000 exceeds 10% of system memory.
4050/4050 [.....] - 6s 1ms/step - loss: 0.6297 - acc: 0.8348 - val_loss: 0.0198 - val_acc: 0.9978
Epoch 2/5
4050/4050 [.....] - 5s 1ms/step - loss: 0.0105 - acc: 0.9985 - val_loss: 0.0221 - val_acc: 0.9978
Epoch 3/5
4050/4050 [.....] - 5s 1ms/step - loss: 0.0134 - acc: 0.9980 - val_loss: 0.0219 - val_acc: 0.9978
Epoch 4/5
4050/4050 [.....] - 5s 1ms/step - loss: 0.0074 - acc: 0.9990 - val_loss: 0.0257 - val_acc: 0.9978
Epoch 5/5
4050/4050 [.....] - 5s 1ms/step - loss: 0.0066 - acc: 0.9990 - val_loss: 0.0227 - val_acc: 0.9978
Tanulás kész! Ha tesztelni akarsz az eredményt a test.py scripttel megnézheted!
ekaktusz@eab-laptop: ~/Asztal/eab/Uj$
```

4. ábra. A train.py futtatási eredménye Linuxon.

```
ekaktusz@eab-laptop: ~/Asztal/eab/Uj$ python3 test.py
Using TensorFlow backend.
WARNING:tensorflow:From /home/ekaktusz/.local/lib/python3.6/site-packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
2019-05-15 00:03:00.383012: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2019-05-15 00:03:00.406182: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2394375000 Hz
2019-05-15 00:03:00.406627: I tensorflow/compiler/xla/service/service.cc:150] XLA service 0x3685190 executing computations on platform Host. Devices:
2019-05-15 00:03:00.406667: I tensorflow/compiler/xla/service/service.cc:158] StreamExecutor device (0): <undefined>, <undefined>
WARNING:tensorflow:From /home/ekaktusz/.local/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Tesztelés indítása...
Tipp    >>> HGANAB
Eredeti >>> HGANAB
OK
Tipp    >>> XRPGFU
Eredeti >>> XRPGFU
OK
Tipp    >>> KFLARG
Eredeti >>> KFLARG
OK
Tipp    >>> RFAENU
Eredeti >>> RFAENU
OK
Tipp    >>> AHCRCP
Eredeti >>> AHCRCP
OK
Helyes tippek száma: 5 . Teszt pontossága: 100.0 %.
ekaktusz@eab-laptop: ~/Asztal/eab/Uj$
```

5. ábra. A test.py futtatási 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 címe 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, és új megoldások vették át a helyét, melyek feltörése egy újabb izgalmas probléma, ám vélhetően pár év múlva szintén percek alatt leküzdhető akadállyá válik.

Hivatkozások

- [1] https://www.youtube.com/watch?v=4D5RN2yKlG4&list=PLbM09c_jUD456277j2fHAUip19xfxIAx0
- [2] <https://github.com/hsekia/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