Bevezetés a mesterséges intelligenciába

Captcha megoldó program

Projekt dokumentáció

Ekart Csaba

2019. május 4.

Tartalomjegyzék

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 mélytanulásos 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 használom a tanításhoz, úgyis 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 TensorFlowt, de a kívánt pontosság megmaradt.

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

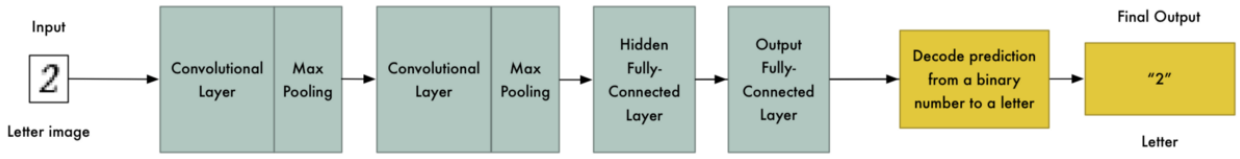
```
4   # Ez egy elore megadott segedfuggveny a kepek atmeretezesehez
5   def resize(image, width, height):
6       (h, w) = image.shape[:2]
7       if w > h:
8           image = imutils.resize(image, width=width)
9       else:
10          image = imutils.resize(image, height=height)
11      padW = int((width - 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:

```
# N E U R Á L I S  H Á L Ó

# Build the neural network!
model = Sequential()

# 1. konvulóciós réteg maxpoolinggal
model.add(Conv2D(20, (5, 5), padding="same", input_shape=(20, 20, 1), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# 2. konvulóciós réteg maxpoolinggal
model.add(Conv2D(50, (5, 5), padding="same", activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Rejtett réteg
model.add(Flatten())
model.add(Dense(500, activation="relu"))

# Kimeneti réteg
model.add(Dense(18, activation="softmax"))

# Tanulás

print("Tanulas inditasa...")
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), batch_size=18, epochs=5, verbose=1)
model.save(MODEL_NAME)
print("Tanulas kesz! Ha tesztelni akarod az eredmenyt a test.py scripttel megteheted!")
```

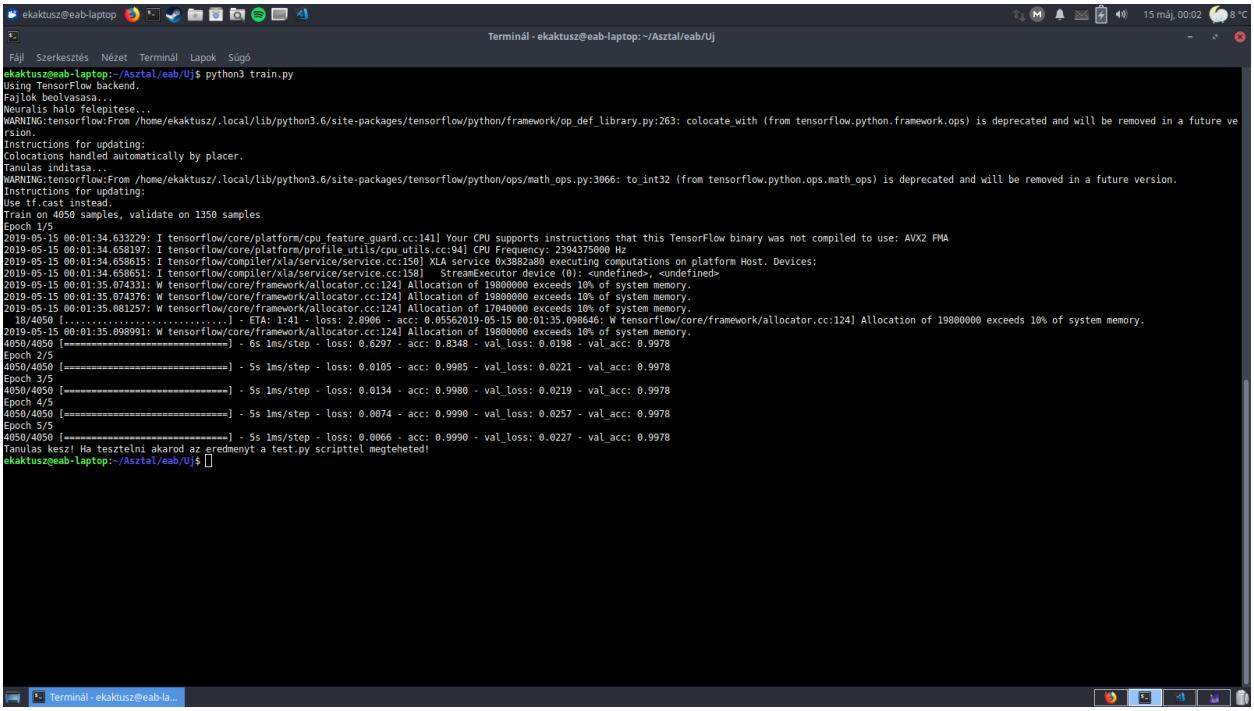
3. ábra. A tanuló modell programkódja, Kerasban

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

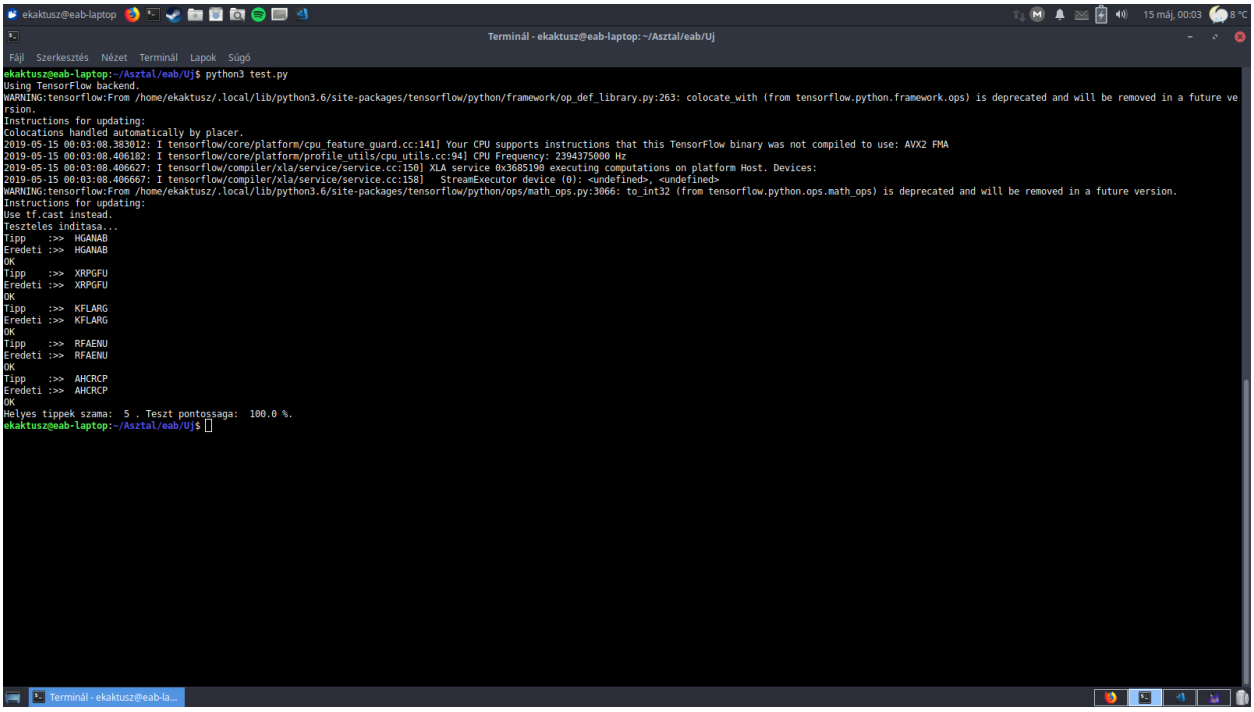
5. Tanulás és teszt

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.



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



5. ábra. A test.py futtatási eredménye Linuxon.

x

6. Végkövetkeztetés

Az egyszerűbb CATPCHA feladatok manapság semmiféle védelmet nem nyújtanak a különféle kártékony programok és botok ellen. A projektemben használt CAPTCHA-k feltörése a Mediumos cikk címe alapján - és a megvalósítás után látva - elegendő ismeretekkel 15 perc alatt megtehető, de egy régi Linuxos laptoppal 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=PLbMO9c_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-dbeb
fbclid=IwAR1rAIlvf4IwPh7zSJvAPdEZU_FzRwFgAjm_WqFTgyqciqiI4dgGAzWdLqE