GÉPI LÁTÁS

TKNB_INTM038

KÖZLEKEDÉSI TÁBLA FELISMERŐ ALKALMAZÁS

CSIZI SZEBASZTIÁN MÁRK

VUADF2

2019/20/2

Tartalom

1 Bevezetés

1.1 Feladatleírás

A féléves feladatom egy közlekedési tábla felismerő alkalmazás elkészítése volt. A program képes detektálni és felismerni 43 különböző KRESZ táblát. A megoldásomban több kép feldolgozási eljárást használok, mint a küszöbölés, kontúrok detektálása, CNN. A dokumentumban részletezni fogom megoldásom elméleti hátterét, programom felépítését, határait és használatát.

1.2 Közlekedési táblák

A KRESZ táblák olyan objektumok, amelyek jól megfogalmazható szabályokkal rendelkeznek. A táblák esetében ilyenek a színek és a formák. Ezek a tulajdonságaik az esetek többségében jól kiemelik őket a környezetükből. Az alakzatok a: kör, oktagon, rombusz, háromszög, négyzet. A színek pedig a: vörös, kék és sárga. Ezek figyelembevételével a detektálás megadhatja a számunkra fontos részeit a képnek (ROI: Region Of Interest).

1.3 Potenciális kihívások

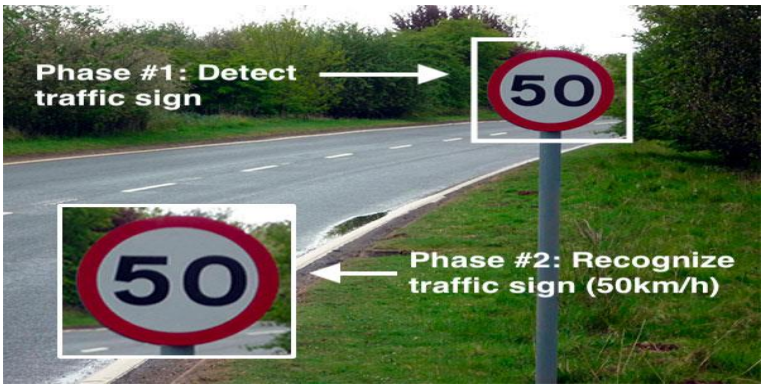
A táblák felismerését, rengeteg tényező befolyásolhatja:

- Táblák megrongálódása.
- Táblákhoz hasonló színek, vagy formák. (épületek, plakátok)
- Környezeti hatások.
- Különböző fényviszonyok.

1.ábra: kihívások

1.4 Objektumok felismerése

Objektumok felismerésének három fő lépése van. A kép előfeldolgozása, detektálás és a felismerés. Sok esetben az első kettőt egy lépésként értelmezik.



2.ábra: detektálás és felismerés

1.5 Előfeldolgozás

A képek eltérő minősége miatt érdemes előfeldolgozni őket. Ilyenek a zajszűrés vagy a méretezés. A méretezés során egyes pixelek értéke nem egyértelmű, ezért érdemes egy olyan interpolációs függvény használata, ami a kép minőségét kevésbé rontja. Ez a függvény egy feltételezés arra vonatkozóan, hogy bizonytalan pixeleknek milyen színe lehetett. A zajszűrésnek sok fajtája ismert, ahol a kis kiterjedésű impulzus szerű részeit a képnek próbáljuk csökkenteni.

1.6 Detektálás

Detektálás célja, hogy a potenciális objektumokat szétválasszuk azok környezetétől. Esetünkbe ez két fő lépésből áll.

1.6.1 Színek intervallumokra osztása

1.6.1.1 Küszöbölés

Küszöbölés alatt azt a folyamatot értjük amikor intenzitástartományt intervallumokra osztjuk, és az egyes intervallumokhoz 1-es vagy egy 0 értéket rendelünk, végeredményül pedig egy bináris képet kapunk.

1.6.1.2 HSV

Sok TSR rendszer HSV (H(Hue)-színárnyalat, S(Saturation)-színtelítettség, V(Value)-világosság) színtérrel végzi el a küszöbölést. Ezt a színteret kifejezetten a képfeldolgozáshoz készítették el. Az RGB és más színterekhez képest sokkal gyorsabb a detektálási sebessége, ami a kisseb számítási komplexitásnak köszönhető. Továbbá a fényviszonyok kevésbé vannak hatással rá. [2][3]

1.6.2 2. Kép formákra osztása

A színekkel való detektálásra a fényviszonyok hatással vannak. Ebben az esetben érdemes a képet formákra osztani és külön-külön elemezni őket, hogy megfelelnek-e a keresendő

objektum paramétereinek. A feladatom megoldásában a már színekkel detektált formák transzformálása, elemzése zajlik.

1.6.2.1 Kontúrok detektálása

A körvonalakat detektálása a bináris képeken, lehetővé teszik a formák és méretük megismerését.

1.6.2.2 Morfológia

Kép alakjának és struktúrájának olyan elemzése, ahol képet összehasonlítjuk egy strukturáló elemmel. A strukturáló elem ütköztetésével az objektumokat a kívánt alakra hozzuk. Két nagyobb fajtája van erózió és dilatáció.

1.6.2.3 Gépi tanulás

Rengeteg gépi tanulási módszer létezik: mint a Tartóvektor-gép (SVM: Support Vector Machine) és a konvolúciós neurális hálózat (CNN: Convolution Neural Network). Általában ezeket használják táblák detektáláshoz és vagy a felismeréshez. [3]

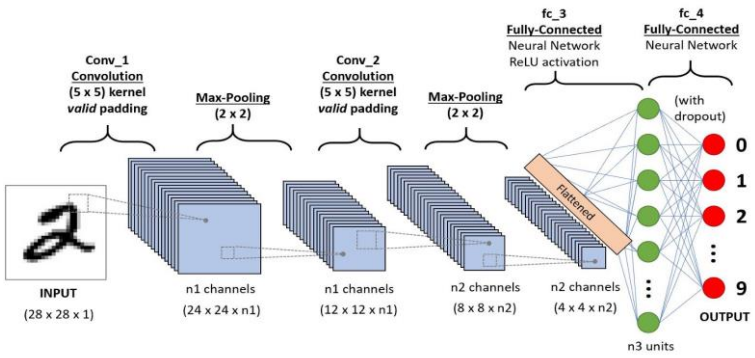
1.6.2.4 CNN

A CNN egy matematikai modell, amivel szimulálni akarjuk a biológiai neurális hálózattokat. A CNN tervezésekor az agy vizuális kérgének utánozása volt a cél. Ennek a kéregnek a sejtjei azok, amelyek felelősek a fény érzékeléséért. Egy neurális hálózatok bementére érkező adatokat manipulálja „súlyok" beállításával, amelyek feltevések arra, hogy bemeneti jellemzők hogyan kapcsolódnak egymáshoz és az objektumok osztályaihoz. [1]

A hálózat kiképzésekor a súlyok értékeit beállítják. A súlyokat a konvolúció nevezetű matematikai művelet hozza létre, ezek a súlyok lénygében a kép egyes részeinek ábrázolása. A kernel vagy szűrő ezen súlyok összesége. A létrehozott szűrő kisebb, mint a teljes bemeneti kép, csak a kép egy részét fedi le. A szűrőben szereplő értékeket megszorozzuk a képen szereplő értékekkel. A szűrőt ezután áthelyezik a kép új részének ábrázolására, és a folyamatot addig ismétlik, amíg a teljes képet le nem fedi. Az egész bemeneti kép körül mozgatott szűrők kimenete egy kétdimenziós tömb, amely a teljes képet ábrázolja. Ezt a tömböt „feature map" - nek hívják.[1]

A CNN folyamatosan fejlődik az adatok növekedésével, mivel képesek alkalmazkodni a megtanult tulajdonságokhoz és saját szűrőket fejleszteni. A "hideg indításkor" (cold start) ezeket a szűrőket kézzel kel beállítani.[6]

Tehát ez az algoritmus képes egy bemeneti kép különböző tulajdonságainak megtanulására, de sok más területen is alkalmazzák. Ez lehetővé teszi a fontosabb objektumok megjegyzését és azok felismerését. Egyik előnye a többi gépi tanulás hoz képest az, hogy az adatokat önmagukban előfeldolgozza. Így csökkentve az erőforrás igényeket. [6]



3.ábra: tipikus CNN felépítés

2 Megoldásom

2.1 Program felépítése

A közlekedési táblák felismeréséhez két kódot használtam fel. Az első egy olyan python kód (train.py) amely megtanítja egy kész adatbázis elemeinek felismerésére (GTSRB). A kódot elemeztem, de mivel tesztelni és lefuttatni nem tudtam ezért a végeredményül kapott h5 súly fájlt használtam a felismeréshez. A második python fájl (TSR.py) a kapott képeken detektálja, majd a korábban edzett CNN-el felismerni azokat.

CNN modell elkészítése (train.py)

- Adatbázis betöltése és előfeldolgozása
- CNN modell felépítése
- Képzés (train)
- Modell tesztelése a teszt adatkészlettel

Detektálás és felismerés (TSR.py)

Detektálás:

- Kép betöltése.
- A képet RGB színtérről HSV színtérre alakítása.
- Küszöbölés.
- Morfológiai nyitás.
- Kontúrok detektálása.
- Méret és formák értelmezése. Ez alapján a potenciális részek kiválasztása.

Felismerés:

- Osztályozás a korábban készített CNN-nel.

Eredmények rajzolása a képre és megjelenítése.

2.2 Adatbázis

A German Traffic Sign Recognition Benchmark egy olyan adatbázis, ami 43 különböző német közlekedési tábla képét tartalmazza különböző db számban. Tartalmazza továbbá a címkéket és a teszteléshez használt képeket.

2.3 Detektálás

2.3.1 A képet RGB/BGR színtérről HSV színtérre alakítása

Ahogyan korábban szó volt róla, érdemes a színteret HSV-re alakítani.

4.ábra: RGB és HSV

2.3.2 Küszöbölés

A küszöböléshez a H és S értékeket használják, mivel a V nem hordoz információt a színről. Ahhoz, hogy a meghatározott színek legyenek láthatóak a képen szükség van a megfelelő küszöbérték meghatározására. Ezeket a táblázatban láthatjuk **1.táblázatban**. Azok az értéket, amik beleesnek ezekbe a tartományokba egyesek lesznek, amik nem azok pedig nullák. Érdemes a küszöbölés elött zajszűrést végezni, amit átlagoló szűréssel old meg a program.

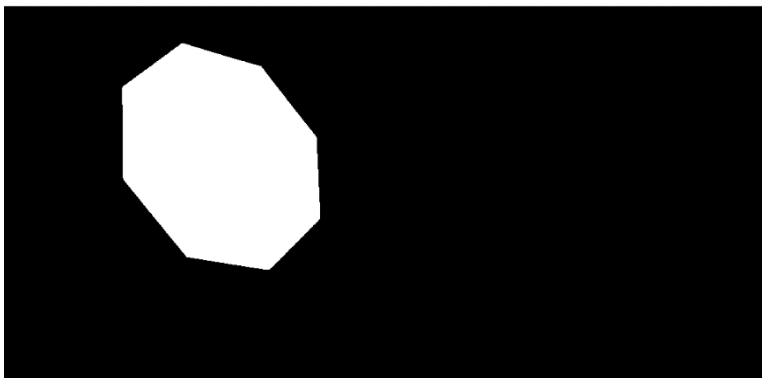
1.**táblázat**: H és S küszöbértékek

	H	**S**
Vörös	0-12 és 160-210	132-255
Kék	100-140	100-255
Sárga	15-25	110-255

5.ábra**:** küszöbölés

2.3.3 Morfológia

A képen sok zaj is megjelenhet a küszöbölés során, amik megfelelnek a kívánt szín értékeknek, de program szempontjából feleslegesek. Ebben az esetben a zaj szűrése morfológiai nyitással történik.



6.ábra: zajszűrés

2.3.4 Kontúrok detektálása

A kontúrok detektálásából következtethetünk arra, hogy a fehér foltok mekkorák és milyen a formájuk. A zajszűrésből megmaradd fehér formák területét feltöltjük. A megmaradt fehér foltok kontúrjait még zajosak lehetnek, ezért érdemes egy kisebb faktorral közelíteni hozzájuk, hogy szabályos formákat kapjunk.

2.3.5 Méret és formák értelmezése.

Bizonyos formák csak bizonyos színekkel értelmezhetőek a feladat szempontjából. Például az oktagon csak a vörös szín küszöböléséből származhat, ha kékbe találnánk ilyet az nem lenne közlekedési tábla. Ezeket a program azzal szűri, hogy a küszöbölést külön végzi el az egyes színekre és azok eredményei csak a megfelelő formákat engedi át.

2.3.6 Felismerés

Az eddigi szűréseken átment területei a képnek, levágásra kerülnek. Ezeket 30*30 pixelre méretezi és osztályozza a korábban készített CNN-el felismeri.

3 Tesztellés

3.1 Pozitív teszt eredmények

7.ábra: pozitív teszt eredmények

3.2 Negatív teszt eredmények

8.ábra: Jó detektálás, de rossz felismerés.
Nincs detektálás
Nem létező detektálás
Kiváló detektálás, de rossz felismerés

3.3 A rendszer határainak megállapítása:

- Jó detektálások tiszta eltérő hattérrel rendelkező tábláknál
- Kör detektálása nem pontos
- Felismerés gyakran téved. Lehet, hogy a számítási kapacitás az oka?

4 Használati útmutató

1. A train.py cmd-ből való futtatása előtt szükséges, az adatbázis kicsomagolása és annak egy mappában történő elhelyezése.
2. A TSR.py és a cnn-ből elkészített traffic_classifier.h5 fájl egy mappában valós elhelyezése.
3. A cmd-ből futtassuk a TSR-t a következő paranccsal: TSR.py --image fájl útvonala. Egy lehetséges példa TSR.py --image /test/1.png.

Szükséges könyvtárak:

- Pillow 7.0.0
- tensorflow 2.0.0rc1
- sklearn: 0.0
- pandas: 1.0.3
- opencv-python: 4.2.0.34
- numpy: 1.18.2
- matplotlib: 3.2.1
- Keras: 2.3.1

5 Felhasznált irodalom

A felhasznált irodalom a dokumentumban számokkal megadott irodalom nevű mappában találhatók. Továbbá felhasználatam:

[4]https://www.pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-and-deep-learning/

[5]https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-with-keras-deep-learning-and-python/

[6]https://data-flair.training/blogs/python-project-traffic-signs-recognition/

[7]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[8]https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0052_13_digitalis_kepelemzes_alapveto_algoritmusai/index.html

Órai jegyzeteket