A vizuális adatok automatikus feldolgozását képfeldolgozásnak (Imafe Processing) vagy gépi látásnak (Computer Vision) nevezzük. A képek és videók feldolgozására napjainkra számtalan alkalmazás került kifejlesztésre. A tantárgy keretében elkészítendő projekt célja egy jelzőtábla-felismerő program megvalósítása.

A feladathoz Macbook Air M1-es laptopot fogok használni. A készülék specifikációja:

* Processzor: Apple M1 chip (8 magos CPU, 7 magos GPU, 16 magos Neural Engine), órajel: 3,2 GHz
* Memória: 16 GB
* Tárhely: 256 GB SSD
* Kijelző: 13,3 hüvelykes képátló, LED-es háttér-világítású kijelző IPS technológiával, 2560 X 1600 képpontos natív felbontás, 400 nites fényerő
* Videokártya: Integrated M1 Graphics
* Operációs rendszer: macOS

A feladatot az ilyen jellegű (neurális hálózatok, deep learning) munkáknál legelterjedtebb, és a tantárgy keretében is leginkább ajánlott Python programozási nyelven készítem el. A képfeldolgozáshoz számos szoftveres eszköz áll rendelkezésre. A mesterséges intelligencia tantárgy keretében több lehetséges eszköz került bemutatásra, úgymint:

* Caffe/Caffe2 – mélytanulási keretrendszer
* PyTorch – Python nyelvre épül, NumPy alternatíva
* TensorFlow
* Cognitive Toolkit – Microsoft
* Keras
* DeepLearning4j – java nyelvhez
* Jupyter Notebook – web alapú alkalmazás
* NLP Architect – nyelvfeldolgozás fókuszú
* Weka
* OpenCV – gépi látás és gépi tanulási könyvtár, támogatja a macOS-t és a Pythont
* R – főként statisztikai számításokhoz

A használni tervezett szoftverek kiválasztásának fő szempontja az operációs rendszerrel és a feladattal való kompatibilitás, illetve a processzorral történő számítás volt (ami ugyan lassabban számol, mint a videokártya, ugyanakkor egyszerűbb a használata). Ezek alapján elsősorban a Keras használatát tervezem, mivel ez Python alapú, nyíltforrású szoftver, amely képes együttműködni más népszerű könyvtárakkal is, mint pl. a TensorFlow. Ugyanakkor a szükséges számítások elvégzéséhez, vizualizációhoz (amennyiben szükséges) kiegészítőként az OpenCV könyvtár is szükséges lehet. Fejlesztőkörnyezetként a Poetry-t tervezem használni, mivel a PyTorch nem elérhető az M1-es macbook-ra. Amennyiben ezen eszközök telepítése, működtetése problémába ütközne, akkor a Parallels szoftver segítségével feltelepített virtuális Windows felületen fogom elkészíteni a feladatot, és ehhez illeszkedő Python alapú szoftvereket fogok használni. Ebben az esetben az alkalmazott szoftverek listáját frissíteni fogom.

A modell elkészítéséhez rendszerint két programkód szükséges. Az egyik betanítja, míg a másik teszteli a modellt. A tanításhoz a <https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign> oldalon elérhető German Traffic Sign Recognition Benchmark (GTSRB) adatbázist tervezem felhasználni.

*„A feladat kétdimenziós alakelemzésen alapuló megkülönböztetést és osztályozást tesz szükségessé. Ennek két fő módszere létezik:*

* *terület alapú*
* *kontúr alapú*

*A terület alapú alakelemzési módszerek jellemzője:*

* *az alakzat teljes területén operálnak,*
* *a pontokat 2D-ben, a képsíkon rendezik,*
* *támogatják a 2D-s lokális műveleteket,*
* *számításigényük az alakzat területétől függ.*

*A kontúr alapú alakelemzési módszerekre ezzel szemben az a jellemző, hogy:*

* *az alakzat kontúrján operálnak,*
* *a pontokat a kontúr mentén rendezik,*
* *nem támogatják a 2D-s lokális műveleteket,*
* *számításigényük az alakzat kerületétől függ.*

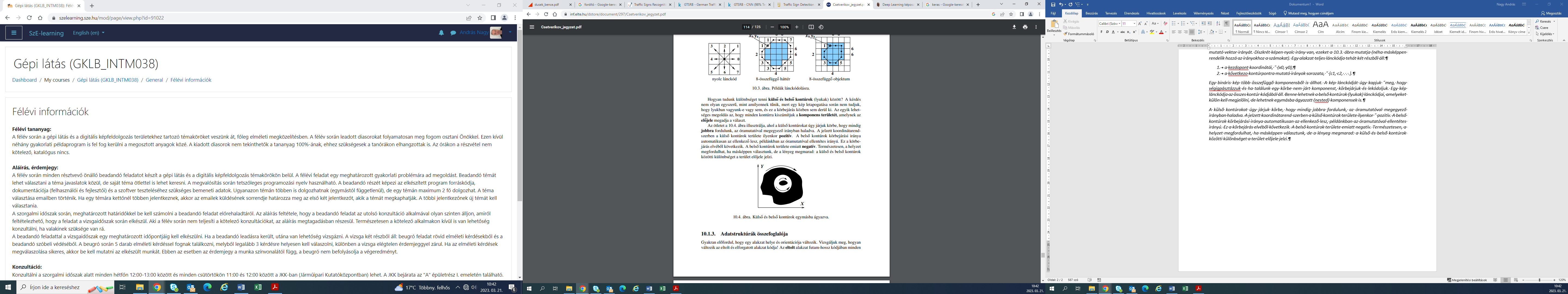
*Esetünkben a kontúr alapú alakelemzési módszer tűnik megfelelőnek. Az ehhez illeszkedő adatstruktúra a lánc-kód (chain code).*

*Amikor csak egy alakzat (összefüggő komponens) van, lánckódolása a következő lépésekből áll. Kiválasztjuk a kezdő kontúrpixelt és a körbejárási irányt. Végigkövetjük az alakzat kontúrját, amíg vissza nem térünk a kezdőpontba. Közben eltároljuk a lánckódokat, azaz a következő kontúrpontra mutató vektor irányát. Diszkrét képen nyolc irány van, ezeket a 10.3. ábra mutatja (néha másképpen rendelik hozzá az irányokhoz a számokat). Egy alakzat teljes lánckódja tehát két részből áll:*

1. *a kezdopont koordinátái, ˝ (x0, y0);*
2. *a következo kontúrpontra mutató irányok sorozata, ˝ {c1, c2, . . .}.*

*Egy bináris kép több összefüggő komponensből is állhat. A kép lánckódját úgy kapjuk ˝meg, hogy végigpásztázzuk és ha találunk egy körbe nem járt komponenst, körbejárjuk és lekódoljuk. Egy kép lánckódja az összes kontúr kódjából áll. Benne lehetnek a belső kontúrok (lyukak) lánckódjai, amelyeket külön kell megjelölni, de lehetnek egymásba ágyazott (nested) komponensek is.*

*A külső kontúrokat úgy járjuk körbe, hogy mindig jobbra fordulunk, az óramutatóval megegyező irányban haladva. A jelzett koordinátarend-szerben a külső kontúrok területe ilyenkor ˝ pozitív. A belső kontúrok körbejárási iránya automatikusan az ellenkező lesz, példánkban az óramutatóval ellentétes irányú. Ez a körbejárás elvéből következik. A belső kontúrok területe emiatt negatív. Természetesen, a helyzet megfordulhat, ha másképpen választunk, de a lényeg megmarad: a külső és belső kontúrok közötti különbséget a terület előjele jelzi.* [1]



A kontúr alapú alakelemzésnél fontos szerepet játszik az alaktényező (shape factor). Definíció szerint.

ahol S a terület, L a kerület és F a körszerűség (circularity) értéke 0 =< F <= 1. Közlekedési táblákra vonatkoztatva az értékek:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Alakzat leírása | Aspect ratio | Circularity | Covexity | Solidity |
| kör | 1 | 1 | 1 | 1 |
| háromszög | 1 | 0,78 | 1 | 1 |
| négyszög | 1 | 0,8 | 1 | 1 |
| nyolcszög | 1 | 0,9 | 1 | 1 |

Forrás: <https://particletechlabs.com/ptl-press/particle-shape-factors-and-their-use-in-image-analysis-part-1-theory/>

Szükség lesz továbbá a képek valamilyen fokú előfeldolgozására is. A leggyakoribb előfeldolgozási lépések a képek egységes méretűvé alakítása nyújtással, a színcsatornák egységesítése és a felesleges képrészletek levágása.

Összefoglalva tehát a következő lépéseket tervezem követni a feladat elkészítése során:

1. **Adatok letöltése és előkészítése:**

Letöltöm a GTSRB adatbázist, majd felhasználom az OpenCV csomagot, hogy előkészítsem a képeket a gépi tanuláshoz. A képeket átméretezem, kiszűröm a zajt, normalizálom a színeket és kiszámítom a képek HOG (Histogram of Oriented Gradients) jellemzőit, amelyek az algoritmusunk számára hasznosak lesznek a táblák felismeréséhez.

1. **Adatok felosztása tanító- és tesztelőhalmazra**:

Ebben a lépésben az adatokat két részre osztom: tanítóhalmazra és teszthalmazra. A tanítóhalmazt az algoritmus tanítására, míg a teszthalmazt arra használom, hogy teszteljem, mennyire hatékony a tanított algoritmus.

1. **Algoritmus kiválasztása és tanítása:**

Ezután egy olyan algoritmust keresek, amely képes lesz a táblák felismerésére. Az algoritmusok közül például a Support Vector Machines (SVM) algoritmus ilyen. A tanításhoz felhasználom a tanítóhalmazt, majd elvégzem az algoritmust finomhangolását a teszthalmazon.

1. **Algoritmus értékelése**:

Utolsó lépésben kiértékelem az algoritmust a teszthalmazon. Az értékeléshez a Confusion Matrixot tervezem használni, amely segít meghatározni a helyes és helytelen osztályozások számát.

A fenti lépések végrehajtásához minimálisan a következő Python csomagokra lesz(lehet) szükség:

* OpenCV
* Scikit-learn
* Matplotlib
* Pandas

Irodalomjegyzék

[1] Csetverikov Dmitrij, Digitális képelemzés alapvető algoritmusai, ELTE Informatikai Kar, 2015., letöltve: <https://dtk.tankonyvtar.hu/xmlui/bitstream/handle/123456789/3786/2011-0052_13_digitalis_kepelemzes_alapveto_algoritmusai.pdf?sequence=2> , 2023.03.21.