**Logófelismerés dokumentáció**

*Készítette:*

**Adonics Nikolett**

**Kun Réka Bianka**

**Kovács Marcell**

**2023**

**Tartalomjegyzék**

[**Feladat leírása** 3](#_Toc152532405)

[**Projekt futtatása** 4](#_Toc152532406)

[**Feladat elkészítése** 5](#_Toc152532407)

[**1.** **Algoritmus megtervezése** 5](#_Toc152532408)

[**2.** **Képek gyűjtése a tanításhoz és teszteléshez** 5](#_Toc152532409)

[**3.** **Logók felcímkézése a képeken** 5](#_Toc152532410)

[**4.** **Képek méretének normalizálása** 5](#_Toc152532411)

[**5.** **Modellek** 5](#_Toc152532412)

[**a.** **CNN** 5](#_Toc152532413)

[**b.** **YOLOv5** 5](#_Toc152532414)

[**6.** **Training – YOLOv5** 6](#_Toc152532415)

[**7.** **Grafikus felhasználói felület** 6](#_Toc152532416)

[**8.** **Dokumentáció és prezentáció készítése** 6](#_Toc152532417)

[**Eredmények kiértékelése** 7](#_Toc152532418)

[**Források** 8](#_Toc152532419)

# **Feladat leírása**

Feladatunk négy cég logójának felismerése, detektálása színes képeken. Ezek a logók a McDonald's, a Nike, az Apple és az Audi logója.

Az alkalmazás először a YOLOv5 és CNN modell-t készíti elő betanításra.

A YOLOv5 txt fájlokba kiírja a képek logóinak adatait [0, 1] tartományban, majd ezeket a képeket és logókat három fájlba osztja: train, test, val. Ezután PyTorch segítségével kerül sor a betanításra. Végül az alkalmazás GUI-ja a kapott képen futtatja a betanított algoritmust, majd megjeleníti az eredményt.

A CNN 100x100-as képek létrehozása után a képeket egy konvolúciós neurális hálózat (CNN) különböző rétegein küldi keresztül. Ez nem került betanításra.

# **Projekt futtatása**

Python 3.9 verzióval készült a program.

* pip install -r requirements.txt parancs
* yolov5\_yaml.py
* model\_yolo.py
* train\_yolo.py
* logo\_detect\_yolo.py / logos\_detect\_yolo.py
* gui.py

# **Feladat elkészítése**

## **Algoritmus megtervezése**

Az algoritmus megtervezése céljából különböző cikkeket olvasott ***Nikolett*** és ***Réka***. Ezek alapján Nikolett összeállított algoritmus terveket a CNN, YOLOv5 és a PyTorch alkalmazásával.

## **Képek gyűjtése a tanításhoz és teszteléshez**

Mind a hat logóból 30-30 darab különböző szemszögű képet gyűjtöttünk össze.

* A Nike és az Audi logók gyűjtése ***Nikolett*** feladata volt.
* Az Apple és a McDonald’s logók gyűjtése ***Réka*** feladata volt.

***Niki és Réka*** az általa gyűjtött logókhoz készített logónként egy-egy txt fájlt, ami tartalmazza az adott logó minden egyes képéhez a kép nevét, a logót határoló doboz koordinátáit, a logó típusát és a logó típusának számszerű azonosítóját, az alábbi formában néz ki egy sora:

fájl\_neve, bal\_felső\_x, bal\_felső\_y, jobb\_alsó\_x, jobb\_alsó\_y, típus, típusid

Ezek a logók kerültek összesítésre az logos/*images* mappába és az adataik az logos/*images/images.txt*-be.

## **Logók felcímkézése a képeken**

Ez ***Réka*** feladata volt. A *read\_txt.py* által kerülnek beolvasásra a képek és a txt-k, majd ezeket a képeket a txt-k tartalma alapján felcímkézi logó megnevezéssel. Végül kiírja őket az *original\_logos/output* mappába.

## **Képek méretének normalizálása**

Ez ***Marcell*** feladata volt. A *normalize\_images.py* a beállított méretre vágja a képeket, ezzel normalizálva, hogy minden kép mérete megegyezzen.

## **Modellek**

Ez ***Nikolett*** feladata volt. A CNN és a YOLOv5 algoritmus modellt készítette el a betanításhoz.

### **CNN**

A *model\_cnn.py*-banRéka és Marcell fenti megoldásai kisebb-nagyobb változtatásokkal és átalakításokkal kerültek összehangolásra. Minden képen a logó körül közepétől kiindulva 100x100-as méretben kerül kivágásra a logó és ezek a *logos/output*-ba kerülnek kiírásra.

Ezután egy konvolúciós neurális hálózat (CNN) különböző rétegein küldi keresztül a képeket.

### **YOLOv5**

A *model\_yolo.py* képenként egy szöveges fájlt (txt) hoz létre a labels mappába, ahol a szövegfájl a következőket rögzíti egy sorban:

class\_id, center\_x, center\_y, határoló doboz szélessége, magassága

Ezek mind [0, 1] tartományba eső számokká vannak alakítva három tizedesjegy pontossággal.

Végül a képeket és label-öket három részre osztja:

1. *logos/train*: a tanító adatbázis
2. *logos/test*: a teszt adatbázis
3. *logos/val*: a validációs adatbázis

A *yolov5\_yaml.py tartalmazza* a YOLOv5 GIT Repo-jának klónozását és a data mappájába konfiguráció létrehozását a *logo\_detection.yaml* fájlba, valamint a YOLOv5 requirements.txt futtatását.

## **Training – YOLOv5**

Ez ***Réka*** feladata volt. A YOLOv5 modell kiképzése az adathalmazon PyTorch segítségével történt.

* A *train\_yolo.py* tartalmazza a betanításhoz szükséges parancsokat.
* A *logo\_detect\_yolo.py* tartalmazza egy kiválasztott képen lévő logó detektálását.
* A *logos\_detect\_yolo.py* tartalmazza több kép egyszerre történő detektálását.

## **Grafikus felhasználói felület**

Ez ***Marcell*** feladata volt. Ezt a *hook-tkinterdnd2.py* és a *gui.py* tartalmazza.

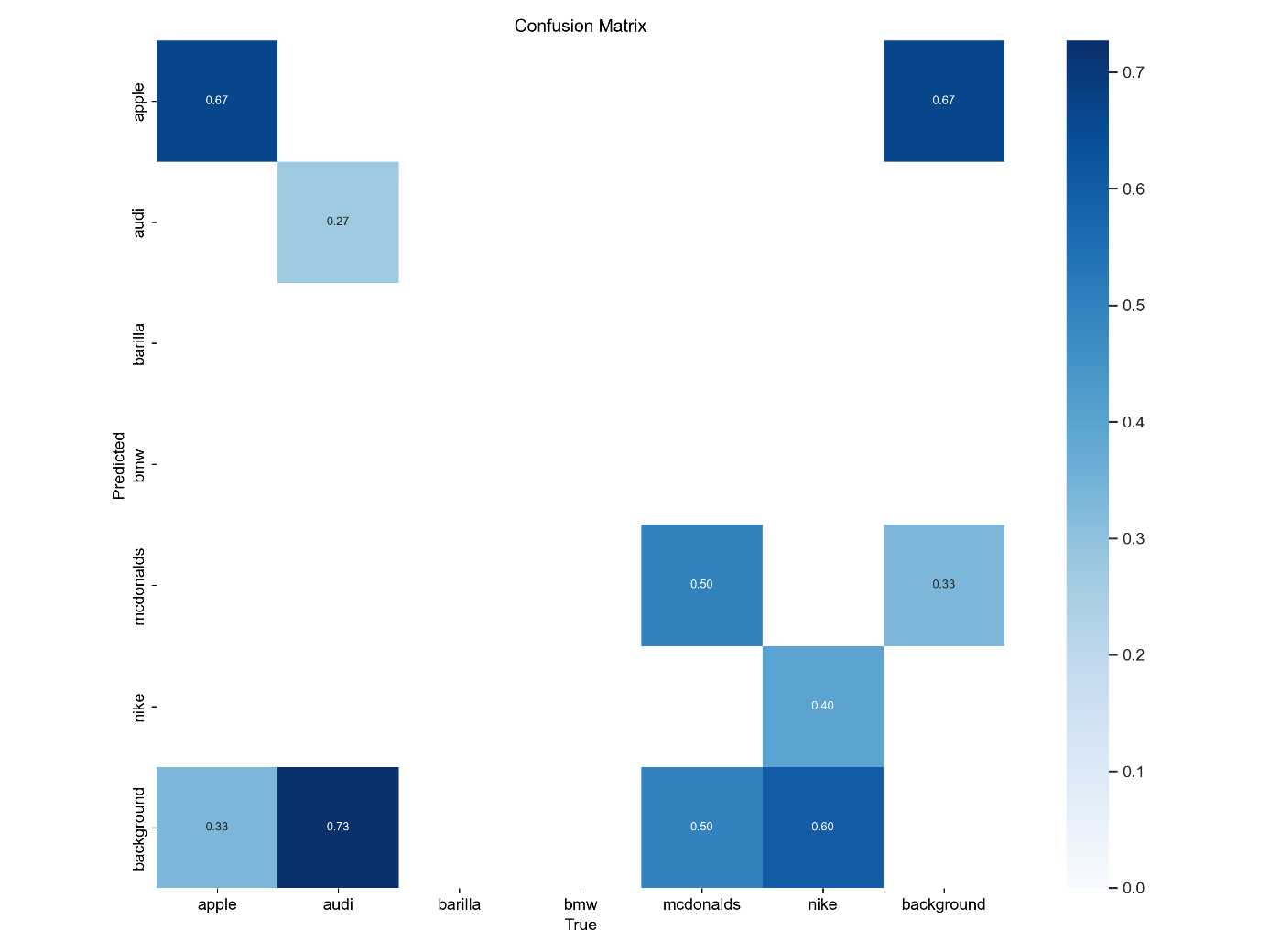
A GUI-ban található egy drag and drop rendszer, mely segítségével a meglévő képünket elég behúzni az alkalmazás ablakába, ezáltal meghatározásra kerül a kép útvonala. A kép elemzése nevű gombra kattintva, a beolvasott képet a modell elemzi, és egy értékelést ad. A bemeneti képre – opencv segítségével – rárajzolódnak a bounding boxok, illetve megjelennek a kitalált logók, illetve konfidencia értékek, majd ezt a képet megjeleníti a felhasználónak.

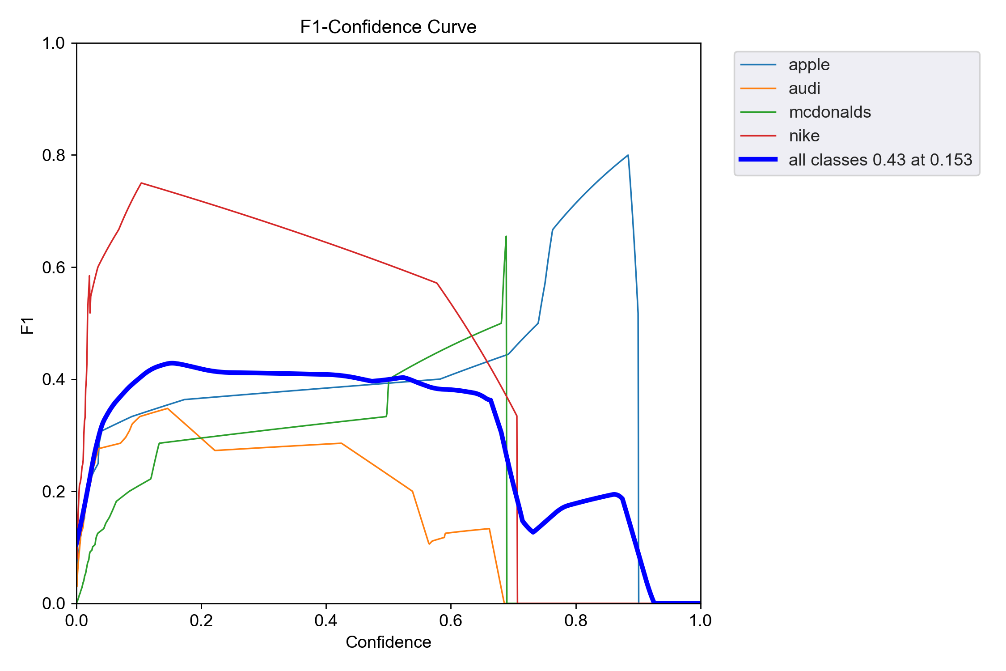
## **Dokumentáció és prezentáció készítése**

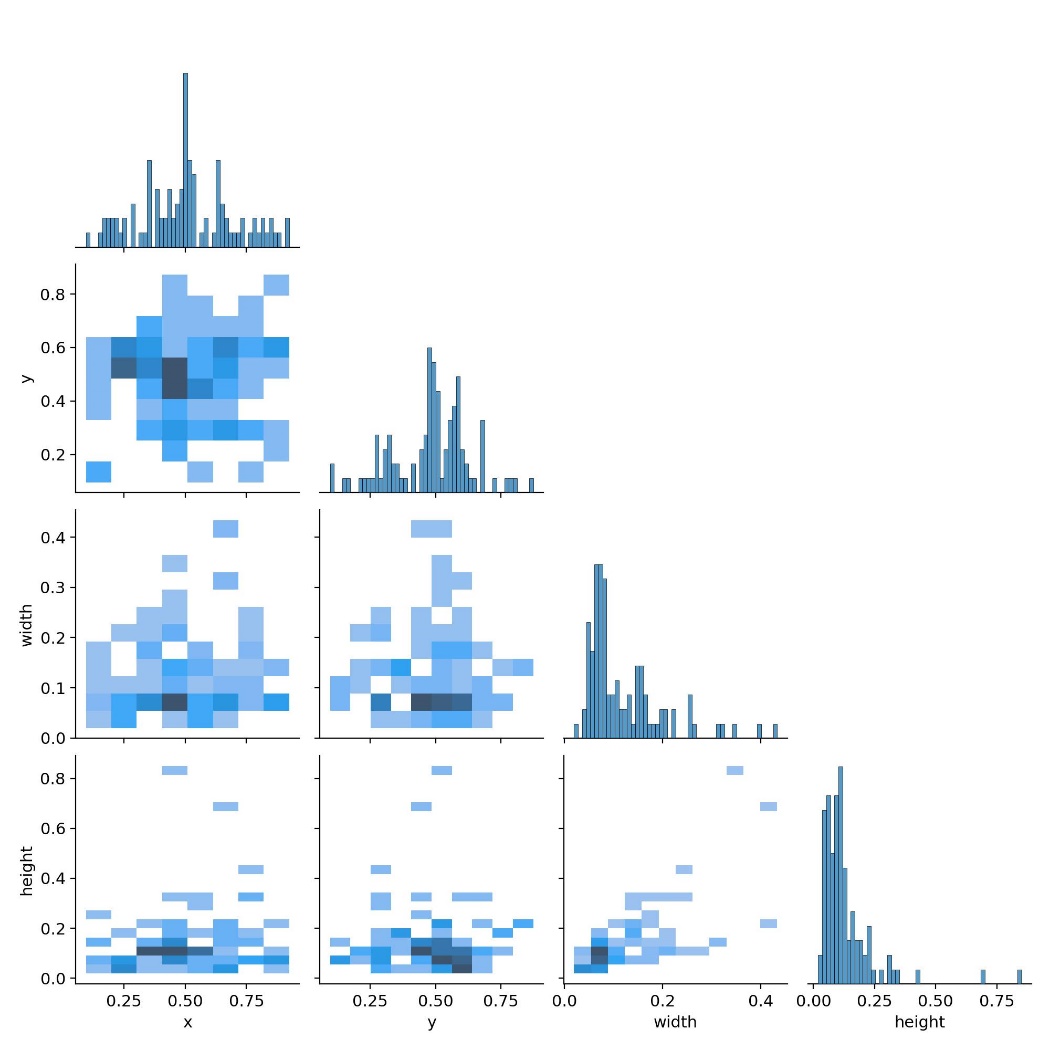
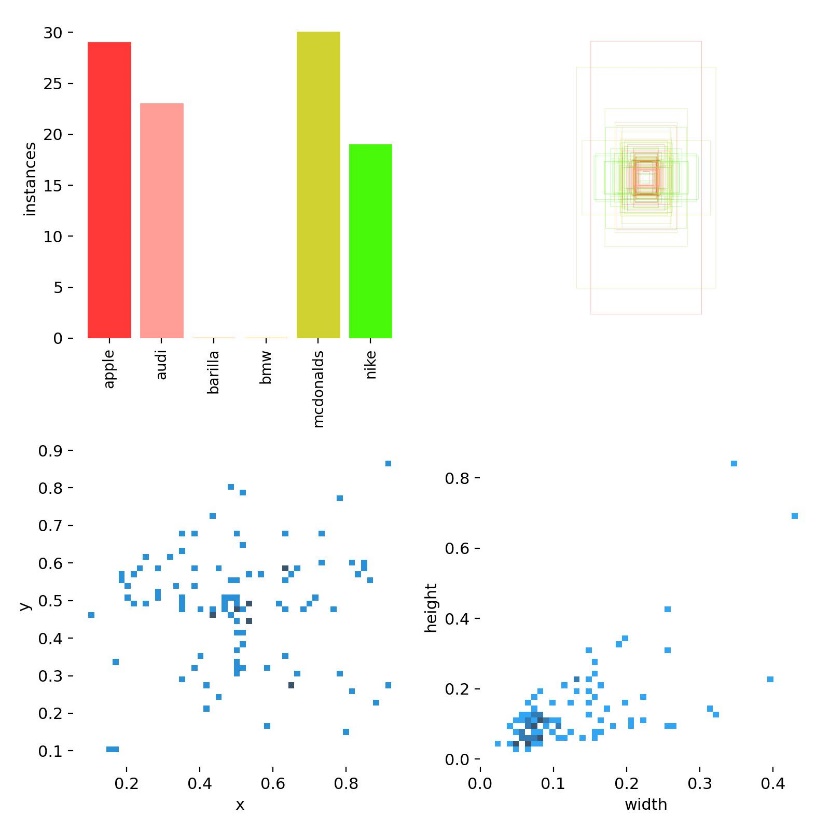
A dokumentációt és a prezentációt ***Nikolett*** készítette el.

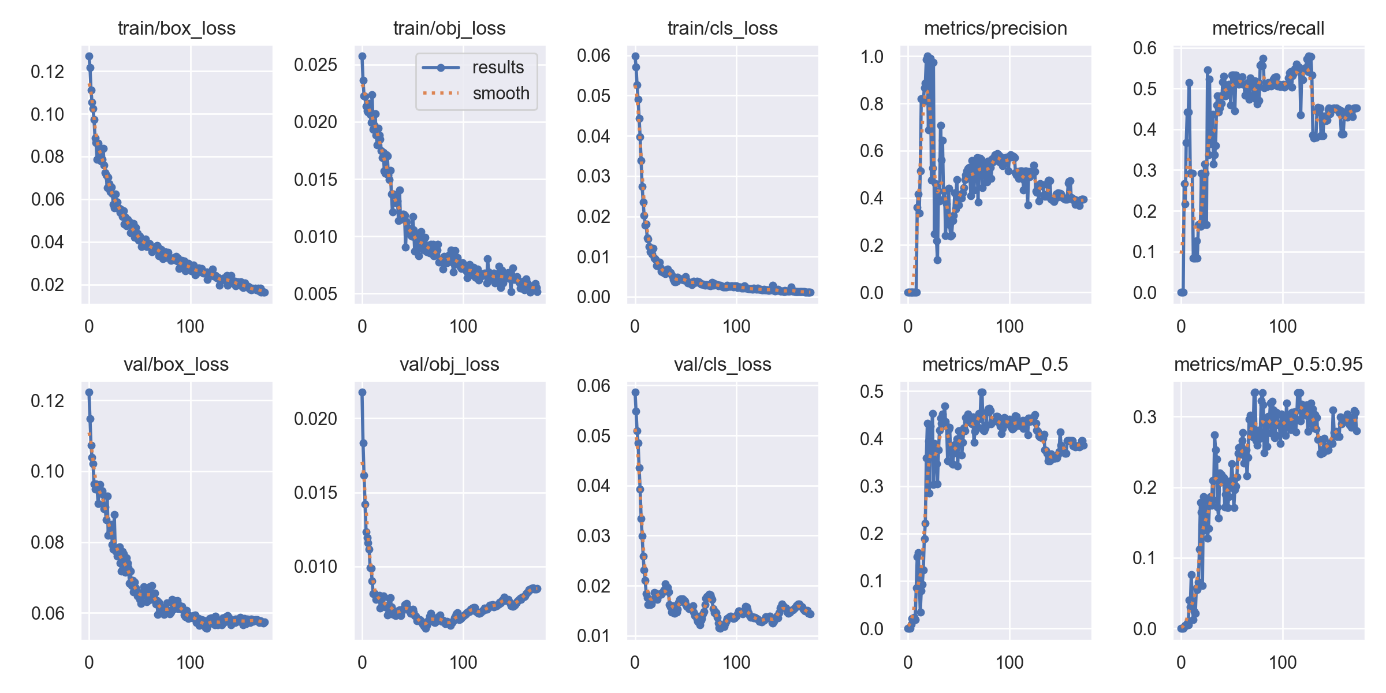
# **Eredmények kiértékelése**

Az eredmények megtalálhatók a yolo\_train/train/yolo\_logo\_detect/ mappában









# **Források**

Ezek sok más mellett a fő források, amiket felhasználtunk a projekt elkészítéséhez.

<https://thebojda.medium.com/tensorflow-alapoz%C3%B3-d2d1ee97c9db>

<https://medium.com/@f.a.reid/custom-training-yolov5-for-logo-detection-d29320f10b39>