ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS INFORMATIKAI KAR**

# SZAKDOLGOZAT


**OE-NIK**
**2023**

Hallgató neve: **Szücs Klaudia**
Hallgató törzskönyvi száma: **T/006399/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

# SZAKDOLGOZAT FELADATLAP

| | |
|---|---|
| Hallgató neve: | **Szücs Klaudia** |
| Törzskönyvi száma: | T/006399/FI12904/N |
| Neptun kódja: | GEL5FF |

A dolgozat címe:

**Pajzsmirigy csomók detektálása ultrahangos képek alapján mély konvolúciós hálózatokkal**
**Thyroid nodule detection based on ultrasound images with deep convolutional networks**

| | |
|---|---|
| Intézményi konzulens: | Dr. Kertész Gábor |
| Külső konzulens: | |
| Beadási határidő: | 2023. május 15. |
| A záróvizsga tárgyai: | Számítógép architektúrák<br>Szoftvertervezés és –fejlesztés specializáció |

**A feladat**

A lakosság számottevő többsége szenved valamilyen pajzsmirigy rendellenesség által okozott betegségtől. Ezen a szervünkön lévő apró elváltozások mielőbbi észlelése is akár kulcsfontosságú lehet a megfelelő és precíz diagnosztika szempontjából.
A dolgozat célja egy olyan orvosi döntést támogató megoldás készítése, amely ultrahangos felvétel alapján képes az elváltozást detektálni. A megoldás alapja egy mély konvolúciós neurális hálózat, amely a bemenetként érkező ultrahangos kép alapján megadja, hogy a beteg milyen malignitású kategóriába tartozik a TI-RADS osztályozás szerint. A dolgozat részeként készített webalkalmazás segítségével a kezelőorvos páciensprofilokat tud létrehozni, illetve különböző statisztikai kimutatások is elérhetőek a betegségek gyakoriságáról korosztályok, vagy nemek szerinti bontásban.

**A dolgozatnak tartalmaznia kell**:

- a kapcsolódó szakirodalom részletes áttekintését, a kapcsolódó módszerek (úgy mint R-CNN, YOLO) részletes bemutatását,
- hasonló problémák, hasonló módszerek bemutatását,
- az elérhető adatok bemutatását, analízisét,
- a megoldás részletes tervét, a módszertan leírását,
- a megoldás implementációjának bemutatását, az eredmények részletes kiértékelését,
- a további kutatási-fejlesztési lehetőségek bemutatását.

ÓBUDAI EGYETEM Neumann János Informatikai Kar 1. Budapest

Ph.

Vámossy Z.

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2025. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

# HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2023. 05. 05.

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar
9. sz. melléklet

# KONZULTÁCIÓS NAPLÓ

Hallgató neve: SZÜCS KLAUDIA

Neptun kód: GEL5FF

Tagozat: NAPPALI

Telefon: +36301584772

Levelezési cím (pl: lakcím): 8681 LÁTRÁNY EÖTVÖS LÓRÁNT UTCA 5

Szakdolgozat / Diplomamunka[1] címe magyarul:

Pajzsmirigy csomók detektálása ultrahangos képek alapján mély konvolúciós hálózatokkal

Szakdolgozat / Diplomamunka[2] címe angolul:

Thyroid nodule detection based on ultrasound images with deep convolutional networks

Intézményi konzulens: DR. KERTÉSZ GÁBOR

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2022.09.29. | Elkészült dolgozat átbeszélése, féléves ütemterv egyeztetése | |
| 2. | 2022.10.18. | Feladatlap elkészítése | |
| 3. | 2022.11.21. | Dolgozat állapotának ellenőrzése | |
| 4. | 2022.11.29. | Dolgozat véglegesítése, prezentáció áttekintése | |

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022. 11. 29.

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

9. sz. melléklet

# KONZULTÁCIÓS NAPLÓ

Hallgató neve: SZÜCS KLAUDIA

Neptun kód: GEL5FF

Tagozat: NAPPALI

Telefon: +36301584772

Levelezési cím (pl: lakcím): 8681 LÁTRÁNY EÖTVÖS LÓRÁNT UTCA 28

Szakdolgozat / Diplomamunka[1] címe magyarul:

Pajzsmirigy csomók detektálása ultrahangos képek alapján mély konvolúciós hálózatokkal

Szakdolgozat / Diplomamunka[2] címe angolul:

Thyroid nodule detection based on ultrasound images with deep convolutional networks

Intézményi konzulens: DR. KERTÉSZ GÁBOR

Külső konzulens:

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2023.03.08. | Féléves ütemterv egyeztetése | |
| 2. | 2023.04.12. | Neurális háló működésének áttekintése | |
| 3. | 2023.04.25. | Dolgozat állapotának ellenőrzése | |
| 4. | 2023.05.02. | Módosítási javaslatok áttekintése, dolgozat véglegesítése | |

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2023.05.05

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

.

# Tartalomjegyzék


**1 Absztrakt** **3**

**2 Abstract** **4**

**3 A megoldandó probléma bemutatása** **5**

3.1 Pajzsmirigy és betegségei . . . . . 5

3.2 TI-RADS rendszer . . . . . 7

3.3 Kutatás célja . . . . . 8

**4 Irodalomkutatás** **9**

4.1 Módszertan kiválasztása . . . . . 9

4.1.1 Neurális hálózat . . . . . 9

4.1.2 Konvolúciós neurális hálózatok . . . . . 13

4.2 Architektúrák áttekintése . . . . . 15

4.2.1 R-CNN . . . . . 15

4.2.2 Fast R-CNN . . . . . 17

4.2.3 Faster R-CNN . . . . . 18

4.2.4 ResNet . . . . . 18

4.2.5 YOLOv3 . . . . . 19

4.3 Terv megvalósításához felhasználható technológiák . . . . . 21

4.3.1 Python . . . . . 21

4.3.2 Tensorflow . . . . . 21

4.3.3 Pytorch . . . . . 22

4.3.4 Google Colaboratory Pro (Colab) . . . . . 22

4.3.5 ASP.NET Core . . . . . 22

4.3.6 Spring Boot . . . . . 23

4.3.7 Frontend keretrendszerek . . . . . 24

4.4 Irodalomkutatás összegzése . . . . . 24

4.4.1 Architektúrák a probléma szempontjából . . . . . 24

4.4.2 Felhasználandó technológiák . . . . . 25

**5 Rendszerterv** **27**

5.1 Követelmény specifikáció . . . . . 27

5.2 Architektúra áttekintése . . . . . 27

**6 Adathalmaz bemutatása** **29**

6.1 Adatok előfeldolgozása . . . . . 31

**7 Fejlesztés** **33**
7.1 Szoftver bemutatása . . . . . 33
7.1.1 Neurális hálózat fejlesztése . . . . . 33
7.1.2 Webalkalmazás . . . . . 38
7.1.3 Objektum detektáló alkalmazás fejlesztése . . . . . 40
7.2 Tesztelés . . . . . 41

**8 Eredmények** **45**
8.1 Eredmények összevetése a szakirodalommal . . . . . 45
8.2 Eredmények értékelése . . . . . 46

**9 Továbbfejlesztési lehetőségek** **48**

**10 Összefoglaló** **49**

**11 Summary** **51**

**Ábrajegyzék** **53**

**Táblázatjegyzék** **54**

**Irodalomjegyzék** **54**

# 1 Absztrakt

Dolgozatom témája, a pajzsmirigy rendellenességek által okozott betegségek növekvő esetszámából kiindulóan, egy orvosi döntést támogató webalkalmazás létrehozása.

Az elváltozás detektálása egy előre tanított konvolúciós neurális hálóval történik, ami képes a bemenetként érkező ultrahangos kép alapján meghatározni a pajzsmirigy csomók helyzetét, valamit besorolni, hogy a beteg milyen malignitású kategóriába tartozik az egységesített TI-RADS pontozási rendszer szerint. A felhasználóval való kommunikációs interfész pedig egy webes alkalmazás, ahol az orvosok képesek a detektálás futtatására, illetve statisztikai kimutatások elvégzésére a betegségek gyakoriságáról, akár különböző korosztályokban, vagy nemi alapú megkülönböztetés szerint.

Az orvosi objektumdetektálás ezen területén folytatott kutatások eredményei alapján a Faster R-CNN neurális hálózati modellt választottam, mivel ezen modell segítségével nagyobb csomópont lokalizációs eredményt értek el, hozzám hasonló összetételű, kis mennyiségű adathalmazon, mint más korszerű neurális modell architektúrákkal.

Az adathalmaz számos átalakítását követően, sikerült 73%-os pontosságot elérnem a pajzsmirigy csomók bináris klasszifikálása során.

# 2 Abstract

Based on the growing number of diseases caused by thyroid disorders, the topic of my thesis is to create a web application to support medical decision making.

The detection of the lesion is performed by a pre-trained convolutional neural network, which is able to determine the position of the thyroid nodules based on an input ultrasound image, and to classify the malignancy category according to the unified TI-RADS scoring system. The user interface is a web application where doctors beside running the object detection itself, they can perform statistical analysis of the prevalence of the disease, either in different age groups or gender distribution.

Based on the results of research in medical object detection, I chose the Faster R-CNN neural network model. This model has been used to achieve higher node localization results on a small dataset of similar composition to mine, than other state-of-the-art neural model architectures.

After several transformations of the dataset, I was able to achieve 73% accuracy in the binary classification of thyroid nodes.

# 3 A megoldandó probléma bemutatása

A lakosság számottevő többsége szenved valamilyen endokrin betegségtől, amelynek legtöbb esetben okozója valamilyen pajzsmirigy rendelleneség (lásd 3.1 fejezet). Azonban ezeknek a betegségeknek a diagnosztizálása igen nagy kihívást jelentő feladat. Általában az orvos szakképzettségén és tapasztalatán múlik. Azonban mihamarabbi pontos diagnosztika felállítása meghatározó lehet a páciens életének szempontjából.

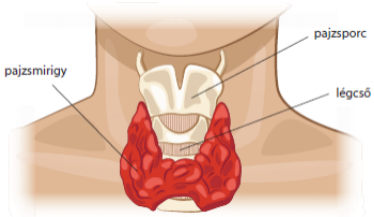
A pajzsmirigyrák kutatása az orvosi közösség és a társadalom széleskörű aggodalmát kiváltó témává vált napjainkban. A pajzsmirigyrákok többsége pajzsmirigydaganat formájában jelenik meg, amelyet gyakran véletlenszerűen fedeznek fel a nyak képalkotó diagnosztika vizsgálata során. Ezek többsége nagy valószínűséggel ártalmatlan elváltozás, azonban a nem várt észlelések ellenére fontos, hogy minél több rosszindulatú nodolust időben azonosítsanak. Továbbá kizárják azokat, amelyek nagy valószínűséggel jó indulatúak.

A vizsgálat által készült ultrahangos képek információt tartalmaznak a pajzsmirigy csomók jellemzőiről, mint például összetétel, echogenitás, peremek, vagy akár a méret. Ezen tulajdonságok alapján a radiológusok egy szabványosított pontozási rendszer (lásd 3.2 fejezet) segítségével megállapíthatják a pajzsmirigycsomók rosszindulatúságának kockázatát. Azaz a kapott pontok alapján kategorizálható, hogy az adott csomó rosszindulatú vagy jóindulatú elváltozás.

Ahogy már említettem szabadszemmel csak kellően nagy tapasztalattal rendelkező szakorvos tudja megállapítani a betegség súlyosságát. Azonban a rohamosan fejlődő informatikai területen megjelenő mélytanulási technológiák számos új lehetőséget nyitottak az orvostudomány területén is. Számos objektumdetektálásra alkalmazható architektúra létezik, amiket megfelelően finoman hangolva nagyban segíthetik a radiológusok munkásságát is.

## 3.1 Pajzsmirigy és betegségei

Legfontosabb belső elválasztású mirigyünk. A pajzsmirigy egy kis, pillangó alakú mirigy, amely a nyak alsó részén, a bőr és az izomrétegek alatt található.

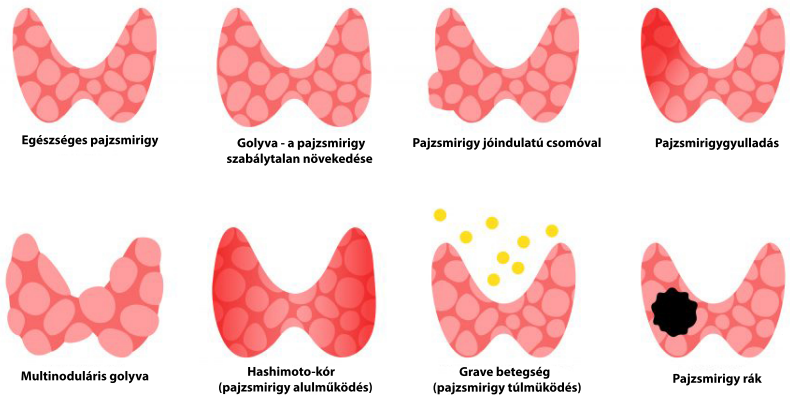



Ábra 3.1: Pajzsmirigy elhelyezkedése

A pajzsmirigy [1] két aktív pajzsmirigyhormont termel, a levotiroxint (T4)

és a trijódtironin (T3), amiknek fontos szerepe van egészséges metabolizmus fenntartásában. Ezen hormonok hatással vannak a fehérje termelődésre, egészséges testhőmérséklet fenntartására, energiatermelésre, illetve szívverés növelésével, légzés növelésével fokozzák a mitokondriumok aktivitását. Ennek eredményeképpen a pajzsmirigy működése a szervezet minden lényeges szervére hatással van.

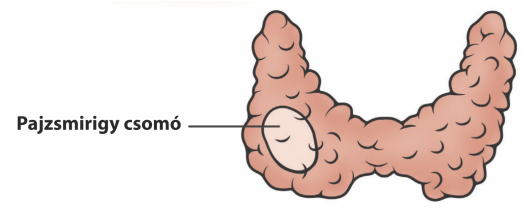
A pajzsmirigy számos különböző problémára hajlamos, amelyek közül néhány rendkívül gyakori betegségeknek mondhatóak az emberiség körében. A túl kevés pajzsmirigyhormon termelése pajzsmirigy-alulműködést, a túl sok pajzsmirigyhormon termelése pedig pajzsmirigy túlműködést okoz. Az előbbi esetben olyan állapotról van szó, amikor a pajzsmirigy alul működik, és nem képes elegendő pajzsmirigyhormont termelni. Ezzel szemben az utóbbi esetben a pajzsmirigy túlműködik, és túl sok pajzsmirigyhormont termel. Túlműködés esetén kialakulhat úgynevezett Basedow Graves-kór, ami egy autoimmun betegség, illetve alulműködés esetén jellemző betegség a Hashimoto-kór.




Ábra 3.2: Pajzsmirigy betegségek

Betegségek nehezen észrevehetőek, az orvosok megannyi tényezőt, többek között klinikai értékelést, vérvizsgálatokat, képalkotó vizsgálatokat, biopsziát és egyéb vizsgálatokat vonhatnak be a pajzsmirigybetegség diagnosztizálásába.

Én dolgozatom szempontjából a képalkotó vizsgálatokkal észlelhető betegségek lesznek a mérvadók, mint például az ultrahangos képek alapján diagnosztizálható pajzsmirigy csomók. A pajzsmirigy csomók a pajzsmirigy belsejében kialakuló szilárd vagy folyadékkal teli daganatos csomók, amelyek lehetnek rosszindulatúak vagy jóindulatúak.

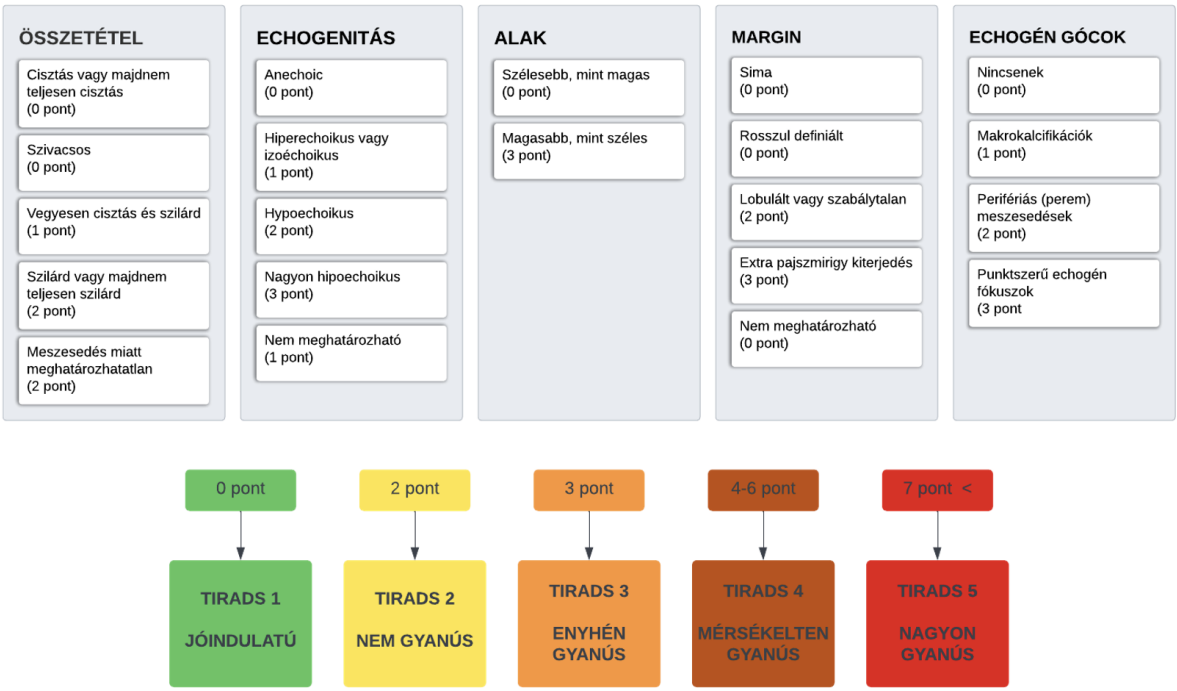



Ábra 3.3: Pajzsmirigy csomó elhelyezkedése

## 3.2 TI-RADS rendszer

Egy Amerikai Radiológiai Társaság egységes kifejezést javasolt a pajzsmirigy csomók malignitásának meghatározására, miszerint az adott nodulust pontszámmal értékelik a heterogenitása, echognitása, alakja, széle és echogén gócok alapján, hogy megítélhessék indokolt-e biopsziás vizsgálat, vagy elegendő az ultrahangos követés a kezelés szempontjából.

Az egységes kifejezési rendszernek köszönhetően a göböt az 3.4 ábrán szemléltetett táblázatban meghatározott értékek alapján öt különböző malignitási csoportba lehet sorolni.




Ábra 3.4: TIRADS pont kalkulátor [2]

Az így kapott osztályok nagyon hatékonyan alkalmazhatóak lehetnek egy neurális hálózati algoritmus szempontjából, amely képes a képről kinyert jellemzőket felis-

merni és azok alapján meghatározni a csomók malignitási szintjét. Az orvosok is a kategorizálást az így kinyerhető tulajdonságok alapján végzik.

## 3.3 Kutatás célja

A fentebb említett betegség növekvő esetszáma miatt, dolgozatom fő célja egy olyan alkalmazás implementálása, ami képes felgyorsítani a klinikai diagnosztika létrehozást mélytanulási algoritmusok segítségével.

Emellett felhasználó számára különböző statisztikai adatok megjelenítésére, illetve szűrésére is szolgálna, a már felvett esetekből. Ezen vizualizációs funkció is hozzájárulna a precízebb lelet készítéséhez.

# 4 Irodalomkutatás

## 4.1 Módszertan kiválasztása

### 4.1.1 Neurális hálózat

A neurális hálózatok[3] a gépi tanulás egy algoritmustípusa, amelyek egymással összekapcsolt csomópontokból állnak. A csomópontok egymással együttműködve dolgoznak, hogy valamilyen összetett, bonyolult adathalmazokban ismétlődések fedezzenek fel.

Ezek a csomópontok, másnéven neuronok az alapvető építő kövei a neurális hálózatoknak. Minden egyes neuron egy vagy akár több bemenetet is kaphat, amin majd számítást fog végezni, és valamilyen kimenetet fog előállítani.

A neuronok rétegeket alkotva kapcsolódnak egymáshoz, és minden egyes kapcsolathoz súlyt rendelnek. Ezek a hozzárendelt súlyok meghatározzák, hogy az egyik neuron kimenete milyen erősen befolyásolja egy másik neuron bemenetét. A képzés során a súlyokat úgy állítjuk be, hogy fontosabb információk nagyobb súllyal fognak rendelkezni, majd minimalizáljuk a hálózat előre jelzett kimenete és a tényleges kimenet közötti különbséget. Azaz a hálózat folyamatosan tanulni fog, hogy számára melyek a hasznos információk, és ezek alapján finom hangolni fogja a súlyokat.

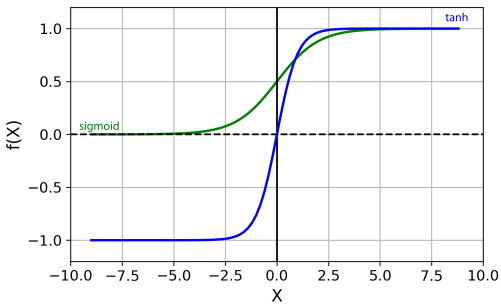
A súlyok mellett minden neuron rendelkezik egy eltolási értékkel (bias), amely egy állandó érték. Ez a torzító kifejezés minden neuron kimenetéhez hozzáadódik mielőtt a következő rétegbe kerülne. Az eltolási érték szükség szerint lehetővé teszi a hálózat számára, hogy az előrejelzéseit felfelé vagy lefelé tolja el.

A neuronok kimenete jellemzően egy aktivációs függvényen halad át, amely a neuron végső kimenetét fogja meghatározni. Az aktiválási függvény dönti el, hogy az adott neuron tartalmazz-e hasznos információt az előrejelzés szempontjából vagy nem. Általános aktiváló függvények közé tartozik a szigmoid függvény, a hiperbolikus érintő függvény és a kiegyenesített lineáris egység (ReLU) függvény.

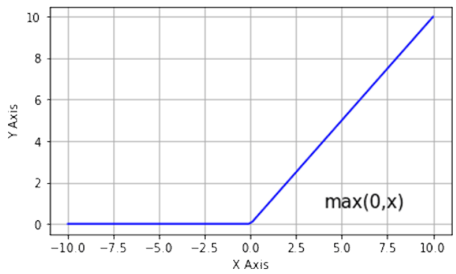
Mindegyik függvénynek sajátos felhasználása van. A szigmoid függvényt bináris klasszifikáció esetén szokták használni, míg a hiperbolikus tangens (tanH) hasonlít a szigmoid függvényre, azonban más érték tartománnyal rendelkezik.

A softmax függvényt gyakran használják a neurális hálózat utolsó aktiválási függvényeként, hogy normalizálja a hálózat kimenetét a megjósolt kimeneti osztályok valószínűségi eloszlásának tekintetében. A ReLU függvény használatának előnye, hogy nem aktiválja az összes neuront egyidőben.

A kimenet tanítás során, egy veszteségfüggvény segítségével a hálózat összehasonlítja a tényleges és az előjelzett kimenet közötti különbséget. A tanítás célja a hálózat súlyainak, illetve az eltolási érték beállítása oly módon, amellyel a veszteségfüggvény minimalizálható. Ezen értékek beállításához különböző optimalizációs algoritmusokat használnak.




Ábra 4.1: Szigmoid és tanH függvény összehasonlítása




Ábra 4.2: ReLU függvény ábrázolása

Azonban a neurális hálózatok felépítése típusuktól és céljuktól függően változhat. Fentebb néhány alapvető összetevőt ismertettem, ami a legtöbb neurális hálózattípus alapjául szolgált.

Ilyen másfajta neurális hálózat például a Konvolúciós Neurális Hálózatok (Convolutional Neural Network, CNN). Képfelismerési feladatokra használják őket, és konvolúciós, illetve pooling rétegeket használnak. A Visszacsatolt Neurális Hálózatok (Recurrent Neural Netwroks, RNN) általában természetes nyelvfeldolgozási feladatokra használják, és speciális szerkezetük a neuronok közötti visszacsatolt kapcsolatokat tartalmazza.

A fentebb említett hálózatokat gyakran használják felügyelt tanítási (supervised learning) feladatokban. Ilyenkor címkézett adatok felhasználásával képeznek ki egy hálózatot, hogy megtanulja megjósolni a helyes kimenetet egy adott bemenetre. Természetesen oly módon, hogy tanítás közben minimalizálja a veszteség függvényt. Azonban a felügyelt tanulással működő neurális hálózat által készített előrejelzések minősége a tanításhoz használt címkézett adatok minőségétől és mennyiségétől függ.

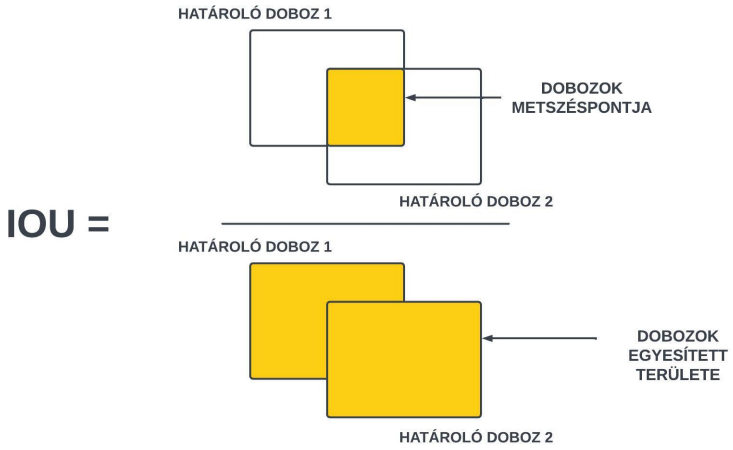
Az adathalmaz eloszlása döntő szerepet játszik a betanított modell teljesítményének meghatározásában. A modell nem biztos, hogy jól teljesít az új, nem látott adatokon, ha a betanított adathalmaz eloszlása jelentősen eltér a valós adatok eloszlásától. A tanítási adathalmaz gondos kiválasztása és előfeldolgozása, valamint az olyan technikák, mint az adatbővítés (data augmentation) és a

túlmintavételezés (oversampling), segíthetnek javítani a modell új adatokra való általánosítási képességét, azaz pontosságát.

A neurális hálózatok pontossága különböző mérőszámokkal mérhető, a konkrét feladattól és a használt hálózat típusától függően.

Szakdolgozatomban a konvolúciós neurális hálózatokat (melynek ismertetése 4.1.2 fejezet) használtam fel, így ezekre vonatkozó fő metrikákat hangsúlyoznám.

Az egyik ilyen használatos mérő metrika az Intersection over Union (IoU) [4]. Ez a mérőszám a képen lévő objektum előre jelzett határoló doboza és az alapigazság határoló doboza közötti átfedést méri. Az IoU kiszámítása igen egyszerű, a megjósolt határoló doboz és az alapigazság határoló doboz metszéspontjának területét kell elosztani az egyesítésük területével. Az alábbi ábrán szemléltetve.



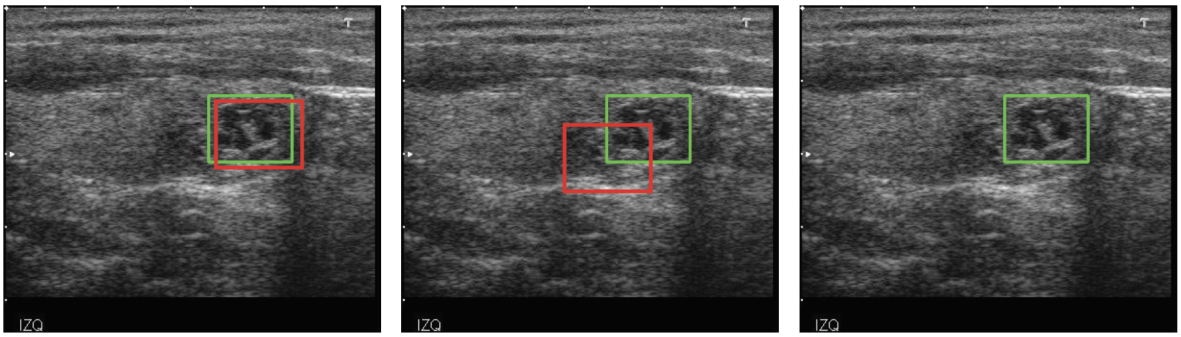

Ábra 4.3: A IoU kiszámításának ábrázolása

Más szóval egy arányszámot fog megadni, a két határoló doboz átfedő területének és az általuk lefedett teljes területnek a mérésével. Ebből kiindulva az IoU metrika értéke 0 és 1 között mozog, ahol a 0 azt jelenti, hogy a két határoló doboz között nincs átfedés, míg az 1 pedig a tökéletes átfedést jelzi. A magasabb IoU pontszám azt jelzi, hogy a megjósolt határoló doboz pontos, és nagymértékben megegyezik az alapigazság határoló dobozával. Az IoU-t gyakran használják küszöbértékként is, annak meghatározására, hogy egy előre jelzett határoló doboz valódi pozitív (True Positive, TP), hamis pozitív (False Positive, FP), hamis negatív (False Negatív, FN), vagy igaz negatívok (True Negative, TN) detektálásnak tekinthető.

A megjósolt határoló doboz helyességét abban az esetben tekintjük igaznak, ha meghaladja az előre meghatározott küszöbértéket. Ennek kiszámítási módja az alábbi módon tér el attól, amelyet korábban a diagrammon bemutattam.

$$IoU = \frac{area(B \cap B_g)}{area(B \cup B_g)} \geq a_0$$

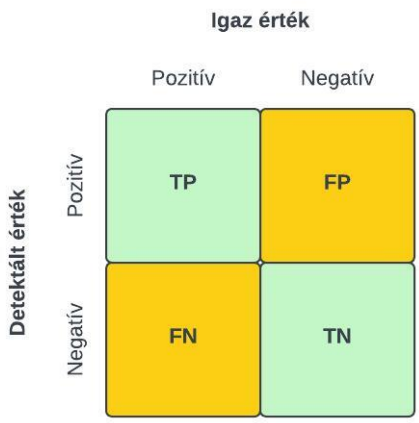
Az 4.4 ábrán egy példa látható, hogy hogyan lehet értékelni egy detektálás pontosságát egy adott küszöbérték alapján. Az alap igazság értékét zöld kerettel jelölt téglalap reprezentálja, míg a "detektált" értéket piros keret veszi körül.




**True Positive** **False Positive** **False Negative**

Ábra 4.4: Detektálás pontosságának reprezentálása

A pozitív-negatív fogalmak, amelyet fentebb említettem, a konfúziós mátrix [4] készítésének is alapjául szolgálnak, amely segít megérteni, hogy a modellünk hogyan teljesít különböző kategóriákban vagy osztályokban.




Ábra 4.5: Konfúziós mátrix felépítése

A konfúziós mátrix további segítéséget nyújt a modell által előrejelzett és a valós értékek közötti eltérések megállapítására, valamint meghatározhatók a különböző pontossági mutatók, például a pontosság (accuracy), a pontatlanság (error), a precizitás (precision) és a felismerés (recall).

A precizitás (Precision) [4] megadja a valódi pozitív észlelések arányát a modell által. Azaz egy adott IoU küszöbérték mellett észlelt valódi pozitív esetek száma a

végzett észlelések teljes számához képest.

$$\mathbf{P = TP/(TP + FP) = TP\ /\ Total\ Predictions}$$

Az érték 0 és 1 között mozog.

A felismerési (Recall) [4] pedig megadja a valódi pozitív észlelések arányát, a képen lévő alapigazság objektumok teljes számához képest.

$$\mathbf{R = TP\ /\ (TP + FN) = TP\ /\ Total\ Ground\ Truths}$$

Hasonlóan a precizitáshoz, az érték 0 és 1 között mozog.

Egy másik precizitás mérésére használatos metrika, az átlagos pontosság (Average Precision, AP). Kiszámítása úgy történik, hogy kiszámítjuk a pontosságot (Precision) különböző felismerési (Recall) szinteken, és ezen értékek átlagát vesszük. Ez azt jelenti, hogy az adathalmaz minden egyes osztályára külön-külön megállapítjuk a pontosság és a felismerési értékét, majd a pontosság-felismerési görbe (Precision-Recall Curve) alapján kiszámítjuk az AP pontszámot az adott osztályra.

Ezután az egyes osztályok AP-pontszámait átlagolva megkaphatjuk a teljes adathalmazra vonatkozó átlagos pontosság (mean Average Precision, mAP) pontszámát [4].

$$\mathbf{mAP = 1/n\ *\ sum(AP)}$$

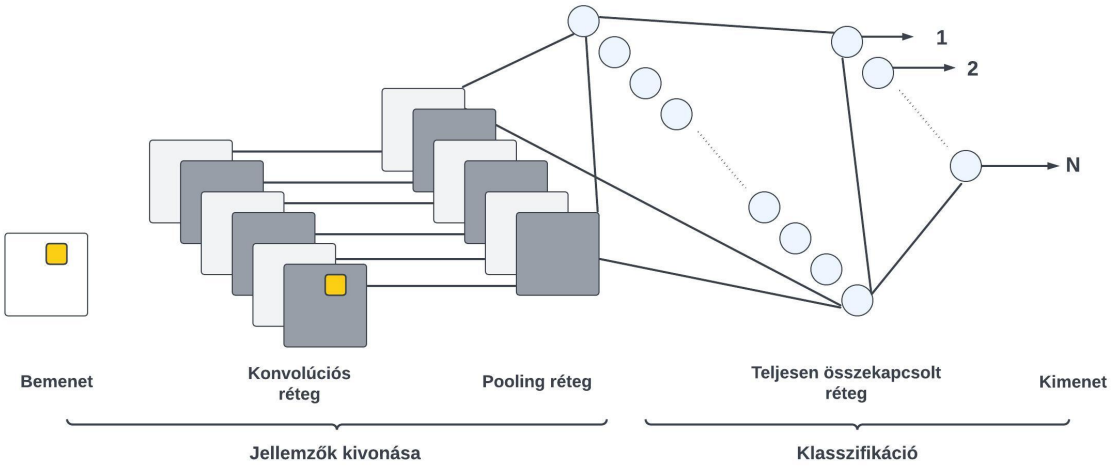
ahol n = osztályok száma. A mAP pontszám átfogó érékelést ad a modell teljesítményéről az osztályban. Az mAP pontszám szintén 0 és 1 érték között mozoghat, és természetesen a magasabb pontszám jobb teljesítményt jelez.

### 4.1.2 Konvolúciós neurális hálózatok

A konvolúciós neurális hálózat (Convolutional Neural Network, CNN) [5] egy olyan speciális neurális hálózat, amelyet általában kép- és videóelemzési feladatokra használnak, például képosztályozásra, objektumdetektálásra és képszegmentálásra.

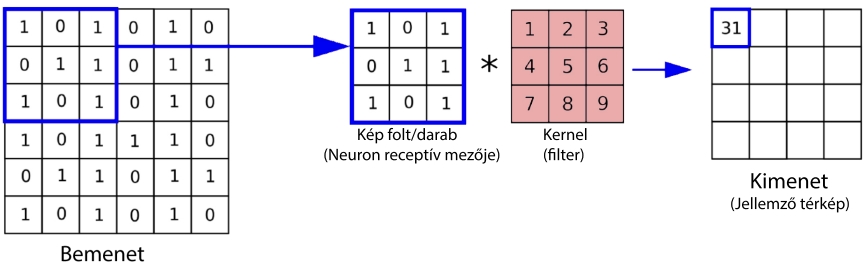
A konvolúciós neurális hálók az adatokat térbeli adatokként kezelik. Ahelyett, hogy a neuronok az előző rétegekben lévő neuronokhoz kapcsolódnának, mint a hagyományos neurális hálókban, a konvolúciós neurális hálón belüli neuronok csak a hozzájuk közeli neuronokhoz kapcsolódnak. A hálózat architektúráját a konvolúciós rétegek, pooling rétegek és a teljesen összekapcsolt rétegek különböztetik meg a többi neurális hálózat felépítésétől.

A konvolúciós réteg az első réteg, amely a különböző jellemzők kinyerésére szolgál a bemeneti képekből. Ebben a rétegben egy szűrőt vagy Kernel-módszert használunk a bemeneti kép jellemzőinek kinyerésére. Ez egy olyan folyamat, amiben vesszük a kernelnek vagy szűrőnek nevezett számmátrixunkat, majd átvezetjük a képünkön és




Ábra 4.6: Konvolúciós neurális hálózat felépítése

a szűrő értékeinek megfelelően átalakítjuk azt. Ez a réteg megőrzi a pixelek közötti korrelációt a képjellemzők megtanulásával, a bemeneti adatok kis négyzeteinek felhasználásával. Lényegében egy olyan matematikai művelet, amelynek konklúziója a képmátrix és a filter segítségével egy jellemzőtérkép.

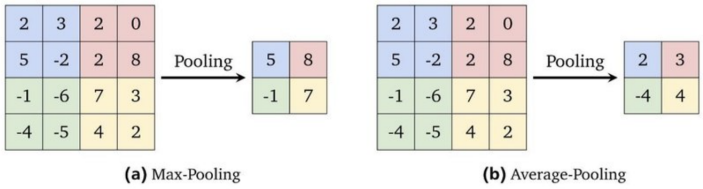



Ábra 4.7: Jellemzők kinyerése

A fentebb említett másik fontos alkotó elem a pooling réteg. Ezen rétegnek az elsődleges célja a számítási költségek csökkentése. Ennek érdekében képes csökkenteni a jellemzőtérkép mintájának méretet a rétegek közti kapcsolatok redukálásával, így gyorsabbá téve ezeknek a feldolgozását. Azaz jellemzők általánosítására szolgáló technika, amely segít a hálózatnak a képen való elhelyezkedésüktől független jellemzők felismerésében. A pooling réteg kimenete egy összevont jellemzőtérkép lesz.

Az alkalmazott módszertől függően többféle pooling művelet létezik a maximum kiválasztás (max pooling), illetve átlagoló szűrés (average pooling). A pooling műveletek legnépszerűbb formája a max pooling, amely a bemeneti

jellemzőtérképből foltokat von ki úgy, hogy minden foltban megtalálja a legnagyobb értéket, majd csak azt a legnagyobb értéket tartja meg, és az összes többit elveti. Ellentétben a másik megoldással, az átlagoló szűrés (average pooling) során a szűrő végigfutása közben kiszámítja a bemeneten található értékek átlagát, majd ezeket egy kimeneti tömbben összegezve tárolja.




Ábra 4.8: Pooling módszerek

Ezen módszerek csökkentik a komplexitást, javítják a hatékonyságot, illetve korlátozzák a túlillesztés kockázatát. A konvolúciós neurális architektúra utolsó néhány rétegét a teljesen összekapcsolt rétegek alkotják. Közvetlenül a kimenet előtt helyezkednek el. Létrehozzák a hálózat végső kimenetét az előző rétegek és azok különböző szűrői által kinyert jellemzők alapján.

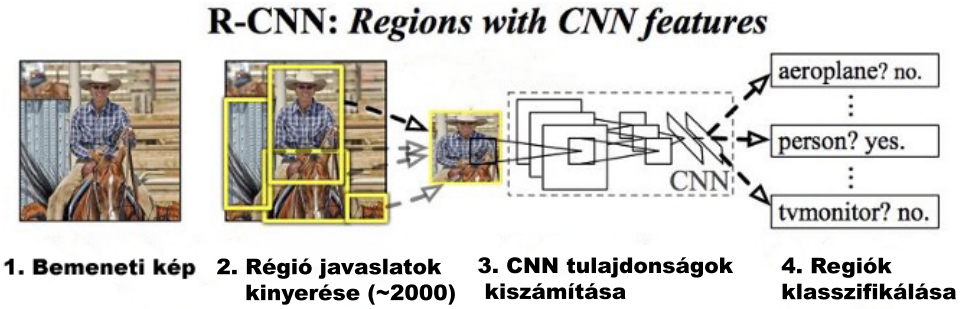
A konvolúciós és pooling rétegek általában ReLU aktivációs függvényeket használnak, míg a teljesen összekapcsolt rétegek softmax függvényt alkalmaznak az osztályozáshoz. A softmax függvény normalizálja az összes bemenetet, így a kimeneti értékek 0 és 1 közötti valószínűségeket jelentenek, amelyek megmutatják, hogy az adott bemenet milyen valószínűséggel tartozik az egyes osztályokhoz. Ez lehetővé teszi a hálózat számára, hogy különböző valószínűségeket állítson elő az adott osztályokra, amelyek alapján az osztályozást végezheti.

## 4.2 Architektúrák áttekintése

### 4.2.1 R-CNN

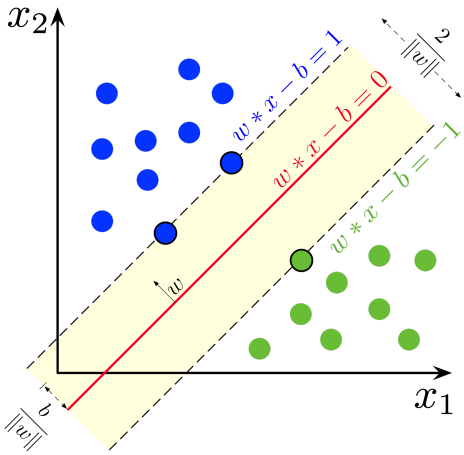
A gépi látás terjedésével, egyre nagyobb hangsúly helyeződött az objektumdetektálás témakörére. Az első igazi áttörés a területen 2014-ben következett be, az RCNN [6] publikálásával, melynek sokkal magasabb performancia szintje volt, az akkori más elterjedt objektumdetektáló metodikákkal szemben. Az addigi próbálkozások nagy komplexitású- és számításigényűek voltak, ami a detektáláshoz szükséges régiók kiszámításának költséges műveletének volt köszönhető. Bemeneti képeken külön elhelyezkedéssel és mérettel találhatók objektumok, ebből következően a régiók meghatározásának folyamata különböző méretű, és nagyon nagy mennyiségű régió detektálását és kiértékelését igényli.

Erre a problémára ad megoldás a RCNN (Regions with CNN).Ezen architektúra három lépést tartalmazz ami a lentebb mellékelt ábrán látható.




Ábra 4.9: RCNN működése
[6]

Az RCNN pipeline első lépése a ′régiójavaslatok′ létrehozása, azaz olyan régiók létréhozása, amelyek egy objektumhoz tartoznak. Egy Szelektív Kereső algoritmus a fő működési motorja, ami úgy működik, hogy a kép olyan alszegmentációt hozza létre, amelyek egy objektumhoz tartoznak – szín, textúra, méret és alak alapján –, és a hasonló régiókat iteratív módon kombinálja objektumok kialakításához. Ezen kereső algoritmus segítségével ∼2000 darab független régiójavaslatot generál, amiken később egy konvolúciós neurális hálózat segítségével minden egyes régiójavaslathoz 4096 dimenziós jellemzővektort kapunk. Jellemzők kinyerése után, azoknak feldolgozása fog történni egy egyedi lineáris SVM (Support Vector Machine) nevű klasszifikációs módszerrel. Ez e a módszer a régiók közötti legjobb határvonal megtalálására ad megoldást, tehát probléma szempontjából besorolja egy adott osztályba, vagy jelzi annak a hiányát.




Ábra 4.10: SVM működése

A lokalizációs teljesítmény javítása érdekében, klasszifikációs fázis után egy bounding-box regressziós nevű lépés kerül végrehajtásra, ami a megtalálandó objektumok köré úgynevezett határoló dobozokat helyez.

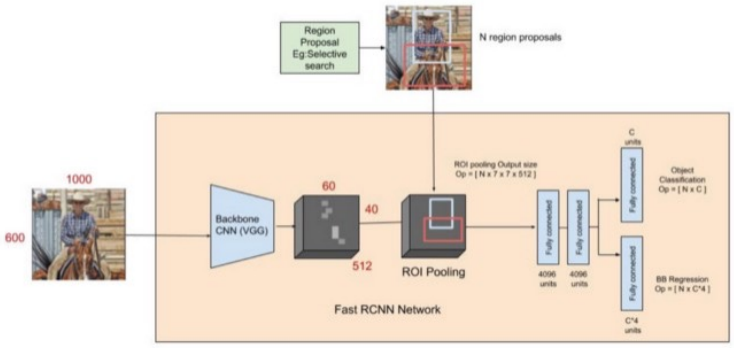
Azonban ennek a modelnek számos hátránya van.

- Ez egy többlépcsős modell, ahol minden egyes lépés egy független komponens.
- Az előre tanított CNN-ből kinyert jellemzőket a lemezen tárolja, hogy később betaníthassa az SVM-eket. Ehhez azonban nagy mennyiségű szabad tárhelyre van szükség.
- A Szelektív Kereső algoritmustól függ a régiójavaslatok létrehozása.
- Minden egyes régió javasalatot függetlenül adunk át a neurális hálózatnak a jellemzők kinyeréséhez, ezért az R-CNN valós idejű futtatása lehetetlen.

### 4.2.2 Fast R-CNN

A fejlődő technológia és növekvő számítási teljesítmény lehetővé tette az R-CNN korlátjainak felismerését. A Fast R-CNN ezen problémák megoldására törekszik, melynek célja egy pontosabb, gyorsabb és kevésbé erőforrásigényes megoldás nyújtása.

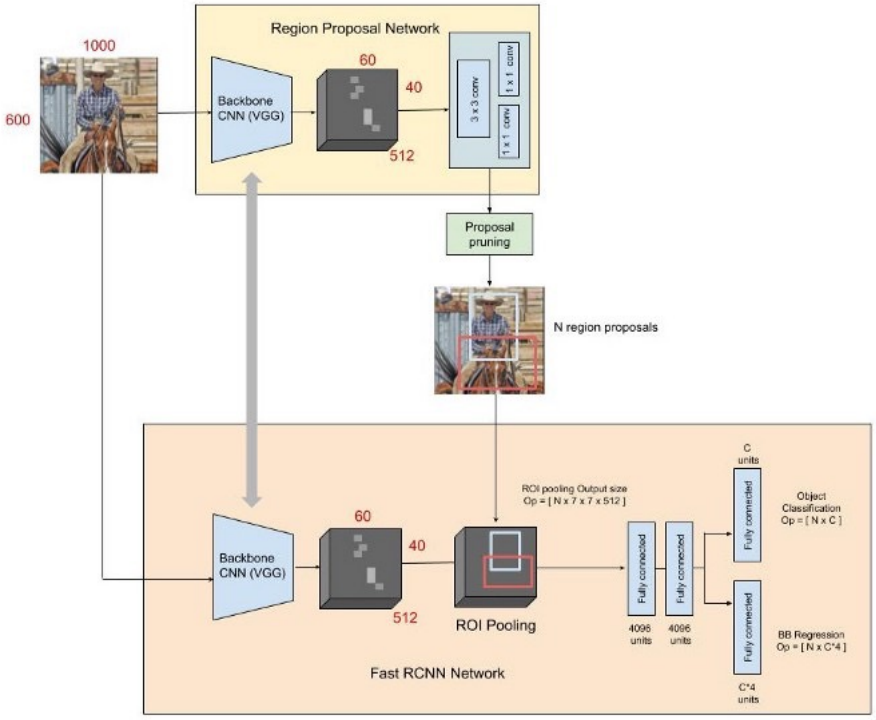
Míg az R-CNN alapötletében minden egyes képhez körülbelül 2000 régió javaslatot generál, és minden egyes régió javaslatot betáplál a mögöttes hálózati architektúrába, a Fast R-CNN esetén a képeket csak egyszer adjuk hozzá a hálózathoz, majd a Szelektív Keresés által generált régiójavaslatokat a CNN segítségével készített jellemző térképekre (feature map) vetítjük. Ezt a folyamatot Region Of Interest (ROI) Projectionnek nevezzük. Ezután a fontos területek feltérkepézésére az Support Vector Machine (SVM) klasszifikálást lecserélve egy Softmax nevezetű függvényt vezettek be. Ebben az aktivációs rétegben történik az objektumok osztályainak meghatározása. A Softmax függvény célja, hogy a CNN kimenetét normalizálja, azaz nulla és egy közé sorolja be. Végül ezek a normalizált értékek megjelennek a határoló dobozokkal együtt.




Ábra 4.11: Fast RCNN működése [6]

### 4.2.3 Faster R-CNN

A Faster R-CNN [6] 2015-ben került publikálása, az előbbiekben ismertetett Fast R-CNN után. A Fast R-CNN már jelentős előre lépést mutatott a R-CNN architektúrák családjában, azonban a detektálás során használt szelektív keresési régiójavaslat generáló algoritmus jelentős teljesítményt igényelt. Ez az algoritmus ezért nem volt alkalmas közel valós idejű működés eléréséhez. Képenkénti $\sim$ 2000 régiójavaslatot, egy új réteg bevezetésével váltották ki, ezt Region Proposal Network-nak nevezzük, amely már minimális veszteséggel dolgozott. Mivel a régiójavaslatokat egy előre tanított hálózattal generálja, most már az adott probléma paramétereire lehet szabni. Így jobb régiójavaslatokat állít elő, az olyan általános módszerekhez képest, mint a Szelektív Keresés.

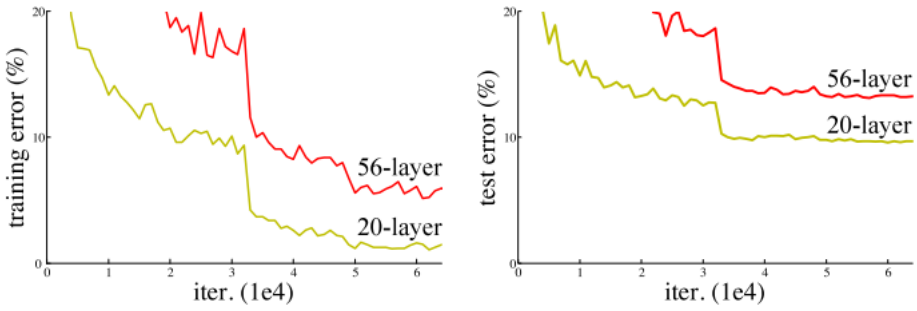



Ábra 4.12: Faster RCNN működése
[6]

### 4.2.4 ResNet

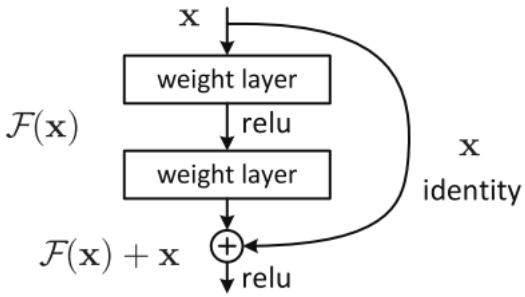
A mélyebb neurális hálózatók betanítása kezdetekben nehéz volt az úgy nevezett eltűnő gradiens (vanishing gradient) probléma miatt. Ez annyit jelentett, ha egyre több réteg van a hálózatban, azaz egyre mélyebb a hálózat, akkor a derivált szorzatok értéke folyamatosan csökken. Az ismétlődő szorzás következményeképpen, a gradiens megközelíti a nullához közelei értéket, majd eltűnik. A hálózat egyre mélyebbé

válásával a teljesítménye telítődik, és a tanítás folyamata leáll. Azért érdemes mélyebb, több rétegből álló hálózatok létrehozására törekedni, mert már önmagában ezzel jelentős javulásokat érhetünk el a tanítás során. A mélyebb hálózatok következménye a hosszabb tanítás szükségessége, amit ha jól konfigurálunk magával vonja a magasabb precizitást.




Ábra 4.13: A halózat mélysége és a performancia kapcsolata [7]

Ennek a problémának orvosolására úgynevezett reziduális blokkokat vezettek be. Ezekben a blokkokban a kihagyásos kapcsolatok technikának az alkalmazásával, a rétegek aktivációit úgy kapcsolja össze a további rétegekkel, hogy közben egy vagy több réteget kihagy.




Ábra 4.14: A reziduális blokk felépítése [7]

Előnye, hogy ha bármelyik réteg károsítaná az architektúra teljesítményét, a kötegnormalizálásnak (batch normalization) köszönhetően a kihagyja ezt.

E tulajdonsága miatt a ResNet a fentebb említett probléma nélküli mélytanulási hálózat megvalósítására képes.

### 4.2.5 YOLOv3

Az összes fentebb említett objektumdetektáló algoritmus régiók segítségével lokalizálja az objektumot a képen belül. Tehát a hálózat nem a teljes képet vizsgálja. Ehelyett a kép azon részeit vizsgálja, amelyek nagy valószínűséggel tartalmazni fogják a keresett tárgyat.

A YOLO vagy You Only Look Once egy olyan objektumfelismerő algoritmus, amely nagyban különbözik a fentebb látott régióalapú algoritmusoktól. Félépítésükben több konvolúciós műveletet és pooling rétegeket tartalmaznak. Korábbi módszerek, mint például az R-CNN modellek is, több ezer hálózati kiértékelést igényelnek ahhoz, hogy egy képre való előrejelzést készítsenek, aminek optimalizálása is jóval bonyolultabb. A YOLO egyesíti a különböző folyamatokat egyetlen neurális hálózatban, így hatékonyabbá válik a futási idő. Azonban a módszer nem képes nem képes felismerni a kis tárgyak csoportját vagy a szabálytalan alakú tárgyakat egy képen belül.

Egy hatékonyabb implementációja a YOLOv3 [8], amely már javított pontosságot is mutat. A modell a DarkNet-et veszi alapul, ami eredetileg 53 réteggel rendelkező neurális hálózatok képzésére szolgáló keretrendszer. A detektálás elvégzéséhez újabb 53 réteget helyeztek rá, így összesen 106 rétegű teljesen konvolúciós architektúra jött létre.

| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| 1× | Convolutional | 32 | 1 × 1 | |
| | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| 2× | Convolutional | 64 | 1 × 1 | |
| | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| 8× | Convolutional | 128 | 1 × 1 | |
| | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| 8× | Convolutional | 256 | 1 × 1 | |
| | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| 4× | Convolutional | 512 | 1 × 1 | |
| | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Ábra 4.15: Darknet-53 [9]

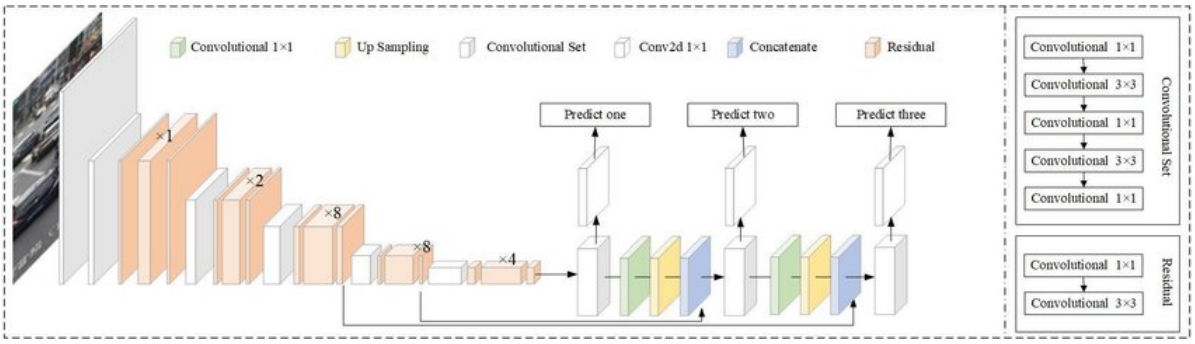
Előző verzióhoz hasonlítva azonban, a sebesége lecsökkent, de sokkal magasabb precízitással képes apróbb objektumok detektálására. Maga a detektálás egy 1x1 -es detektáló kernel segítségével történik, három különböző méretű jellemző térképen, a hálózat három különböző helyen. A detektáló kernel alapja:

$$\mathbf{1 \text{ x } 1 \text{ x } (B + (5 + C))},$$

ahol B, az előre megbecsülhető határoló dobozok száma. Az 5-ös szám a négy határoló doboz attribútumaira, illetve az egy objektum bizalomra, vagy másnéven

bizalmi pontszámra utal (mennyire valószínű, hogy a tartalmazz objektumot, vagy nem). C pedig az osztályok száma.

Az algoritmus három skálán készít előrejelzéseket, amelyeket az adott input kép dimenzióinak 32,16, és 8 mintavételezése ad meg. A YOLO v3 összesen 9 horgonyzó dobozt (anchor box) használ, minden skálázáshoz hármat. K-Means algoritmus segítségével megfelelően klaszterezéssel létrehozható a 9 horgonykocka. Az első skálához tartozik a három legnagyobb, a másodikhoz a három közepes, és az utolsóhoz a három legkisebb keret. Azaz dimenziók szerint csökkenő sorrendbe vannak rendezve. Így minden réteg a nagy, közepes vagy kis objektumok felismerésében jeleskedik.




Ábra 4.16: YOLOv3 architektúra részletei [9]

## 4.3 Terv megvalósításához felhasználható technológiák

### 4.3.1 Python

A Python [10] egy általános célú, magas szintű programozási nyelv, amely sokféle területen alkalmazható. A Python kódja egyszerűen olvasható és írható, és sok nyelvi funkciója segíti a hatékony és gyors fejlesztést. Továbbá széleskörű könyvtár készletének és keretrendszer támogatottságának köszönhetően, ez a nyelv bizonyult eddig a legmegfelelőbbnek gépi tanulás terén. Következtetésképpen az objektum detektáló részt ezzel fogom elkészíteni.

### 4.3.2 Tensorflow

A Tensorflow [11] a Google által kifejlesztett nyílt forráskódú gépi tanulási keretrendszer, ami 2015-ben került publikálásra. Eredetileg a Google Brain mérnöki és kutatói csapata fejlesztette ki a gépi tanulás és mély tanulási hálózatok kutatására. Jelenleg a leginkább alkalmazott keretrendszer ezen a területen. Széleskörű, flexibilis eszközeinek és könyvtárainak köszönhető a népszerűsége, amelyek nagyban megkönnyítik a gépi tanulási alkalmazások építését, illetve telepítését. Továbbá a

jó dokumentáció mellet előre tanított modellkészlettel is rendelkezik, amely jó kiindulópontként szolgálhat a témával idegeneknek a különböző modellek fejlesztési folyamatának megértéséhez.

Ezek mellett a Tensorflow-ba integrálva elérhető a Keras API, ami egy magas szintű, mély tanulási API, amelyet a Google fejlesztett ki a neruális halózatok hatékonyabb, gyorsabb megvalósítására.

### 4.3.3 Pytorch

A Pytorch [12] egy viszonylag új, Torch-on alapuló mélytanulási keretrendszer. A Facebook AI-kutatócsoportja fejlesztette ki, és publikálta 2017-ben. Ezen keretrendszer főként az egyszerűségéről, könnyű használhatóságáról, rugalmasságáról, hatékony memória használatáról és dinamikus számítási gráfjairól híres. Emellett natív érzetet kelt, ami a kódolást könnyebben kezelhetővé teszi és növeli a feldolgozási sebességet.

### 4.3.4 Google Colaboratory Pro (Colab)

A Google Research terméke, ami lehetővé teszi, hogy bárki tetszőleges python kódot írjon és futtasson a böngészőn keresztül, és különösen alkalmas gépitanuláshoz, illetve adatelemzéshez. Technikailag a Colab egy olyan hosztolt Jupyter notebook szolgáltatás, amelynek használatához nincs szükség beállításra, ugyanakkor ingyenes hozzáférést biztosít a számítási erőforrásokhoz, beleértve a CPU-kat, GPU-kat és TPU-kat is. [13]

### 4.3.5 ASP.NET Core

Az ASP.NET Core [14] a .NET platform egy újabb verziója, ami egy nyílt forráskódú és felhőoptimalizált webes alkalmazások fejlesztésére szolgáló cross-platform keretrendszer, amely támogatja az MVC (Model-View-Controller) tervezési mintát.

Azaz, olyan projekt sablonokat tartalmaz, amik segítségével könnyen létre hozhatunk egy webes alkalmazást a kezdetektől, hiszen az alap projekt struktúrát le generálja a fejlesztőnek, így lerövidítve fejlesztési környezet kialakítását.

Különböző előre definiált osztályokat is tartalmaz, továbbá attribútumokat is, amik segítségével könnyen kialakítható a Controller és a View réteg közötti kommunikáció. View réteg megvalósítására számos lehetőségünk van ASP.NET-ben. Egyik legelterjedtebb a Razor Layout, amivel dinamikus weblap készíthető olyan módon, hogy HTML szintakszisba szerver oldali kódot is beágyazhatunk. Továbbá az adatkötés is könnyen megvalósítható, a View rétegből értekező adatok automatikus hozzárendelődnek a Controller rétegben található egyes metódusok paramétereihez.

Az ASP.NET Core könnyen integrálódik a később említett közkedvelt ügyféloldali keretrendszerekkel és könyvtárakkal is, mint például az Angular-ral,

React-tal és a Vue.JS-el. Ezek mellé felsorolható a Blazor keretrendszer is, ami Javascript helyett C#-t használ a felhasználói felületek készítéséhez. A Blazor alkalmazások lényegében olyan újra felhasználható komponensekből állnak, amelyeket C#, HTML és CSS segítségével implementáltak.

Az adatelérés is zökkenőmentesen implementálható az Entity Framework ORM (Object Relational Mapping) keretrendszerrel. Az így keletkezett adatbázis kontextusban hozzáférünk az egyes mezőkhöz, amiken már LINQ-t (Language Integrated Query) is használhatjuk. A LINQ lényegében a C# programozási nyelvbe integrált API-készlet, amely segítségével gördülékenyen végrehajthatunk különböző adatforrásokból történő lekérdezéseket, mint pl. Where(), OrderBy().

A token alapú JWT autentikáció és autorizáció is egyszerűen implementálható egy NuGet csomag telepítésével.

### 4.3.6 Spring Boot

A Spring keretrendszer [15] maga egy szintén nyílt forráskódú, vállalati szintű keretrendszer, ami Java virtuális gépen (JVM) futó önálló alkalmazások létrehozására szolgál. A Spring Boot olyan eszközöket biztosít, amely a Spring keretrendszer segítségével lehetővé teszi, hogy minimáliás konfigurációval és beállíttassál piackész webes alkalmazásokat lehessen létrehozni.

A Spring Boot keretrendszer leginkább a Dependency Injection megvalósításának módjától lett közkedvelt a mikro szolgáltatások fejlesztése körében. A különböző objektumoknak lehetséges függőségeket definiálni, amiket később a Spring konténer injektál hozzájuk, így könnyedén létrehozhatunk lazán összekapcsolt komponensekből álló moduláris alkalmazásokat. Ezeket a függőségeket különböző annotációkkal tudjuk megjelölni, mint például @Service, @Repository, @Controller. Így ezekkel az annotációkkal ugyanúgy kialakítható a MVC tervezési mintának megfelelő struktúra, mint az előbb említett ASP.NET Core-ban.

Maga a Controller rétegben elhelyezett osztályok felépítése igen hasonlóan történik, csak másképpen nevezett annotációkkal valósítsuk meg őket.

Spring Boot-ban a Hibernate nevezetű ORM (Object Relational Mapping) keretrendszert használhatjuk. Itt nem adatbázis kontextust hozunk létre, hanem különböző tároló interfészeket, amit a Spring Data JPA-ba beépített osztályok egyikével bővítünk ki. (Ilyen lehet például a JpaRespository) Majd az így keletkezett objektumokhoz alapvető lekérdezéseket definiál nekünk a JPA, mint például findAll(). Azonban lehetőségünk is van egyéni lekérdezésekkel is bővíteni ezeket.

A Spring Boot-ban a biztonsági konfiguráció sokkal részletesebb, mint a fentebb említett ASP.Net Core keretrendszerben. Ez az úgynevezett Spring Security dependencia egy igen jól személyre szabható megoldást kínál.

A felhasználói felület megvalósítására itt is könnyedén integrálható a fentebb említett néhány keretrendszer, mint például a React, Angular vagy a Vue.JS.

### 4.3.7 Frontend keretrendszerek

Az alábbi táblázatban a jelenleg piacvezető keretrendszerek összehasonlítása látható fontosabb tulajdonságaik alapján.

| | **Angular** | **React** | **Vue** |
|---|---|---|---|
| **Technológia típusa** | Typescript<br>MVC keretrendszer | Javascript<br>Javascript könyvtár | Javascript<br>MVVM keretrendszer |
| **Elterjedtség** | Népszerű maradt a fejlesztők között hosszabb ideje | Legnépszerűbb | Növekvő népszerűség,<br>legnagyobb csillag szám GitHub-on |
| **Tanulási görbe** | Legnehezebb | Könnyebb, mint az Angular,<br>de ugyanúgy meredekebb tanulási görbe | Legkisebb tanulási görbe,<br>alap Javascript tudás |
| **Legalkalmasabb** | Egyoldalas alkalmazások | Minden méretű alkalmazások | Minden méretű alkalmazások |
| **Fenntarthatóság** | Nehezebb,<br>sok új verzió,<br>nagy különbségek | Könnyű,<br>hibamentes | Könnyű,<br>hibamentes |

Táblázat 4.1: Frontend keretrendszerek összehasonlítása [16][17][18]

## 4.4 Irodalomkutatás összegzése

Ebben a fejezetben összefoglalásra kerül az előző részekben tárgyalt irodalomkutatás eredményei, amelyek alapján kiválasztottam a legmegfelelőbb keretrendszereket és architektúrákat a rendszer megvalósítására.

### 4.4.1 Architektúrák a probléma szempontjából

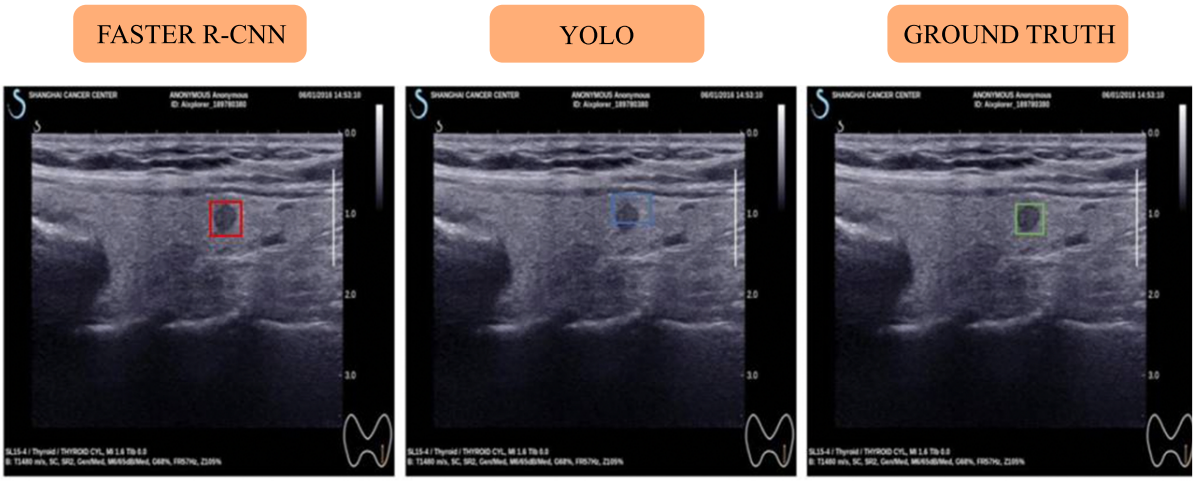
Egy cikk alapján, továbbfejlesztett CNN modellel [19] képesek voltak a nodulosok automatikus detektálására. Ezen tanulmány az általam választott nyíltan hozzáférhető adatbázison (Digital Database of Thyroid Ultrasound Images, DDTI [20]) lett validálva. Az ígért módszerrel, módosított Resnet-50 alapú hálózattal 87.131%-os pontosságot sikerült elérniük. Azonban az adathalmaz számosságából kiindulóan, csak a rosszindulatú csomókat klasszifikálta sikeresen.

További kutatásokban, nagyméretű adathalmazokkal (8000 + 1200 kép), a YOLOv3 [21] és a Faster R-CNN [22] modellekkel, 94%-os pontosságot sikerült elérni.

Azonban számomra nem állt rendelkezésre ekkora méretű adatbázis, így nem tudtam megalapozni a modellválasztásomat az említett kutatások eredményeire.

Sikerült találni egy tanulmányt, amely összehasonlítja ezen két modern objektumdetektálási módszert, melynek alapja a DDTI adathalmaz és egy 200 képből álló belső adatbázis. Így nem csak a modell választás terén volt ezen kutatásnak döntő szerepe, de a későbbiekben lehetőségem volt a tanulmányban elért sikerekkel releváns összehasonlítási metrikák kimutatására, valamint az alkalmazott módszerem finomhangolására.

Az összehasonlító képen ( 4.17 ábra) jól látható, hogy a Faster R-CNN nagyobb lokalizációs pontossággal rendelkezik mint a YOLO. Teljesebb csomópont lokalizációs eredményt kaphatunk vele, amely jobban hasonlít az alapigazsághoz.




Ábra 4.17: Két módszer összehasonlításának eredménye [23]

Ezen állaspontot tovább igazolják, a 4.2 táblázatban összegzett értékek, melyek szerint a határértékek emelésével is nagyobb precizitást mutatt a Faster R-CNN model.

| **Method** | **Faster R-CNN** | **YOLO** |
|---|---|---|
| IoU threshold | mAP | mAP |
| 0.5 | 0.68 ±10 | 0.63 ±0.07 |
| 0.75 | 0.53 ±15 | 0.30 ±0.11 |
| 0.90 | 0.36 ±19 | 0.11 ±10 |

Táblázat 4.2: A mAP értékek összehasonlítása [23]

### 4.4.2 Felhasználandó technológiák

A neurális hálózat tanítását, ahhoz szükséges környezet kialakítását, illetve az adatok előfeldolgozását a PyTorch keretrendszerrel fogom elvégezni. Ezen keretrendszer sokkal inkább a fejlesztők számára optimalizált és felhasználóbarátabb megközelítést kínál, illetve könnyebb hiba keresési lehetőségeket tartalmaz, ami az én megközelítésemből sokkal hasznosabb, mint a vizualizációs eszközök. Az adathalmaz, amelyet későbbi fejezetben bemutatok, számos módosítást igényel. A nyers adatokkal való munka egyszerűbb lehet, mint az adatok beágyazása a TFRecordok struktúrájába. Ennek következtében is a PyTorch sokkal jobban illik a céljaimhoz.

A .NET keretrendszert választottam a webalkalmazás fejlesztéséhez, mivel a modern C# programozási nyelv előnyösebb, mint más alternatívák. Ráadásul a beépített LINQ (Language Integrated Query) támogatás lehetővé teszi, hogy

egyszerűen és hatékonyan végezhessek adatbázis-lekérdezéseket a fejlesztés során. Továbbá, az Azure felhő alapú szolgáltatás integrálása számomra kiemelten fontos volt, és a .NET keretrendszer remekül támogatja ezt.

A felhasználói felület megvalósítására pedig egy olyan keretrendszer szerettem volna választani, amivel könnyen lehet hibamentes, de jól bővíthető alkalmazásokat készíteni és természetesen a performanciát is szeretném figyelembe venni a felhasználói élmény miatt. Így a Vue.JS keretrendszerre esett a választásom.

# 5 Rendszerterv

## 5.1 Követelmény specifikáció

A rendszer fő célja, hogy hatékonyabbá tegye a pajzsmirigybetegségek diagnosztizálását és kezelését. Ennek érdekében a szoftvernek képesnek kell lennie felismerni és lokalizálni a pajzsmirigy csomóit, valamint a csomók rossz- vagy jóindulatúságának meghatározására is. Továbbá az alkalmazás lehetővé teszi a korábban diagnosztizált betegek statisztikai adatainak vizualizálását, így az orvosok és betegek hatékonyabban tudnak döntéseket hozni a kezeléssel kapcsolatban.

Az alkalmazás használatához először is egy intuitív grafikus felületet kell biztosítani, amelyen keresztül a felhasználó könnyedén feltöltheti az ultrahang képet. A bemeneti kép minden esetben egy nyaki ultrahang felvétel, amelyen jól látható a pajzsmirigy. Az alkalmazásban elérhetőnek kell lennie egy lekérdező felületnek is, amelyen keresztül a felhasználó szűrheti a korábban diagnosztizált betegek adatait, és így pontosabb információkhoz juthat az adott betegséggel kapcsolatban.

Majd az alkalmazás detektáló algoritmusa kiértékeli a bemeneti képet, majd megjeleníti a csomó helyét és rossz- vagy jóindulatúságát. Az eredményeket egyértelműen és érthetően jeleníti meg a felhasználó számára, így a betegség diagnosztizálása és kezelése hatékonyabbá téve.

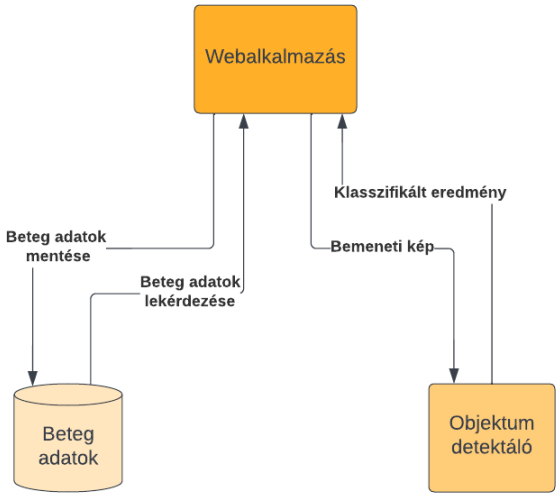
## 5.2 Architektúra áttekintése

Az architektúra felépítése több rétegből áll. Az első réteget a felhasználói interakciókra alkalmas webalkalmazás képezi, amely az egész rendszer központi eleme. A felhasználók a webalkalmazás segítségével tudják feltölteni a bemeneti képeket az objektum felismerő alkalmazás számára, amely felismeri és lokalizálja a pajzsmirigy csomókat, valamint klasszifikálja azokat. A detektált értékeket a REST API-n keresztül küldi vissza a webalkalmazásnak, ahol a felhasználók számára megjelenítésre kerülnek.

Az adatbázisréteg lehetővé teszi a beteg adatok tárolását, amely alapján számos statisztikai lekérdezés jeleníthető meg a webalkalmazásban. A beteg adatok lekérdezése az alkalmazás REST API végpontjain keresztül történik.

Az architektúra felépítése tehát a fenti elemek kombinációjából áll, amelyek között a webalkalmazás és az objektum detektáló python alkalmazás közötti kommunikáció a REST API interfész segítségével történik.

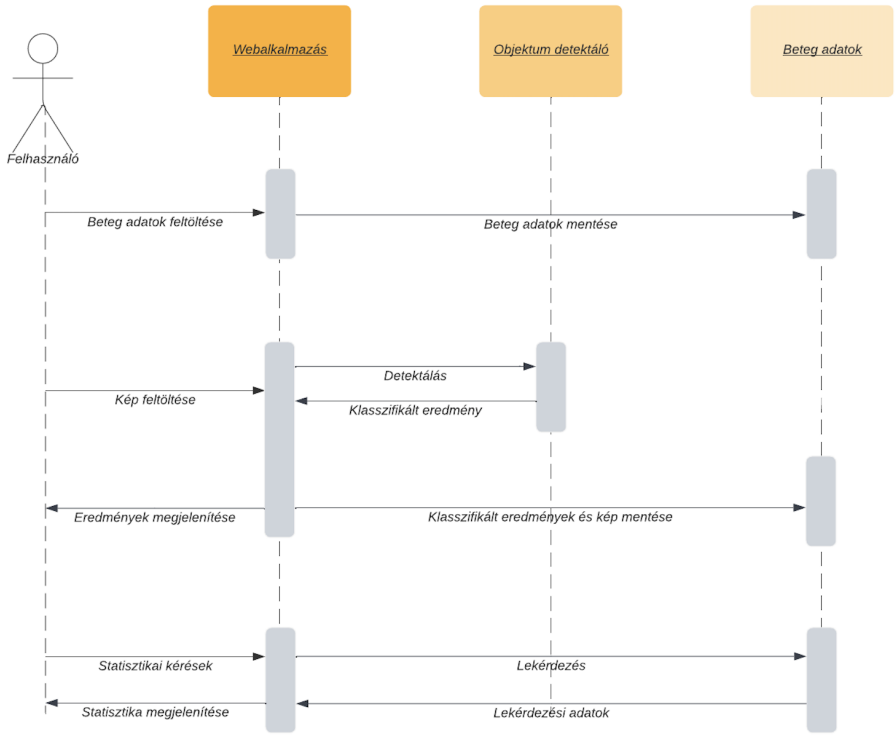
Az architektúra pontosabb és vizuálisabb ábrázolása a 5.1 ábrán látható.




Ábra 5.1: Arhitektúra felépítése

A 5.2 ábrán részletesen bemutatom a különböző folyamatokban szereplő komponensek közötti megvalósítást.

Ezen ábra által nyújtott információk segítségével könnyedén átláthatóvá válnak a komponensek közötti folyamatok, azok sorrendje, így könnyebben megérthetőbb a rendszernek működése.

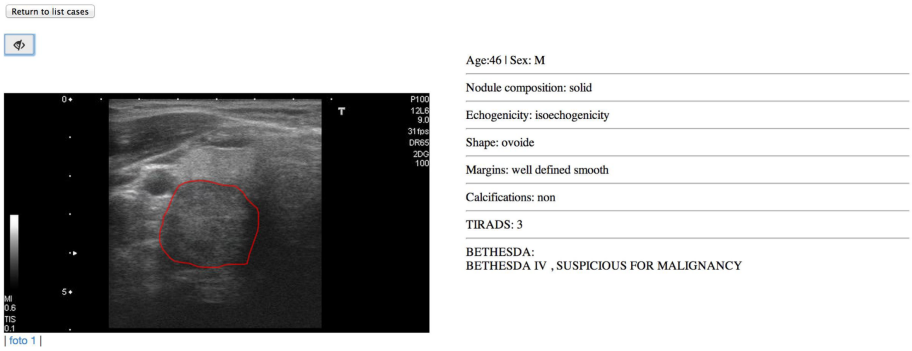



Ábra 5.2: Folyamatábra

# 6 Adathalmaz bemutatása

Egy nyíltan hozzáférhető adatbázis ami a pajzsmirigy ultrahangos képeit (Digital Database of Thyroid Ultrasound Images, DDTI [20]) tartalmazza. Az adathalmaz Kolumbia Nemzeti Egyetem által került publikálásra az alábbi linken: `http://cimalab.unal.edu.co/applications/thyroid/`

Ezen adatbázis JPEG kiterjesztésű B-modú ultrahangos képeket tartalmaz a pajzsmirigy csomókról, amely az intézet által kifejlesztett speciális annotációs eszközzel került felcímkézésre. Ez egy online Javascript rajzoló interfész, amely számos platformon működve lehetővé teszi a csomók klasszifikálását az adatbázisba feltöltött képeken.




Ábra 6.1: Annotációs eszköz bemutatása
[20]

Ezen új annotációs eszköz célja, hogy lehetővé tegye a szegmentált nodulusok és a megjegyzésekkel ellátott címkék azonnali megosztását az interneten keresztül (fentebb említett link), így folyamatosan bővíthető az adathalmaz.

A legutóbbi feljegyzés szerint az adathalmaz 428 olyan esetet tartalmaz, amelyek a gyanús pajzsmirigy elváltozásának teljes annotációját és diagnosztikai leírását tartalmazzák. A különböző esetek szakértő radiológusok közreműködésével a TI-RADS (lásd 1.2 fejezet) osztályozásnak megfelelően voltak végrehajtva. A lentebb látható ábra reprezentálja az egyes esetek számosságát.

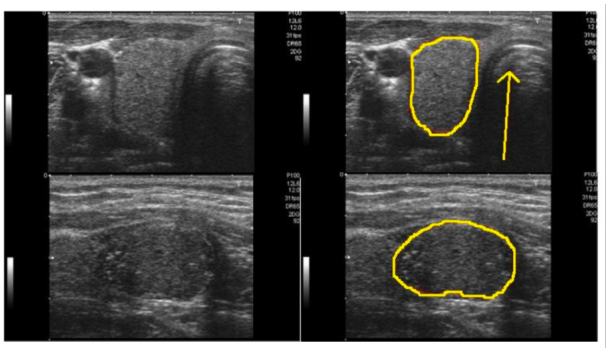
A radiológusok által felállított diagnózist ultrahang képek alapján patológusok is felülvizsgáltak és kategorizáltak őket a Bethesda rendszer [24] szerint.

A patológusok által készített leírások szintén szerepelnek XML formátumú fájlokban. Ezekben az XML-fájlokban esetenként megtalálható, hogy az adott elváltozás rosszindulatú vagy jóindulatú, emellett a betegről is számos információt szolgáltat, mint például: életkor és nem. A csomók egyéb rosszindulatúságra utaló jellemzőit is azonosították: összetétel, echogenitás, margójellemzők, mikro kalcifikációkat, kompozícióit, TI-RADS pontszám.

```
<case>
    <number>129</number>
    <age>49</age>
    <sex>F</sex>
    <composition>predominantly solid</composition>
    <echogenicity>isoechogenicity</echogenicity>
    <margins>well defined</margins>
    <calcifications>microcalcifications</calcifications>
    <tirads>4b</tirads>
    <reportbacaf></reportbacaf>
    <reporteco>Nodule 2</reporteco>
    <mark>
        <image>2</image>
        <svg>[{"points": [{"x": 214, "y": 92}, {"x": 212, "y": 82}, {"x": 208, "y": 77}, {"x": 202, "y": 73}, {"x": 186, "y": 69}, {"x": 176, "y": 65}, {"x":
            166, "y": 63}, {"x": 142, "y": 63}, {"x": 130, "y": 63}, {"x": 122, "y": 63}, {"x": 110, "y": 69}, {"x": 98, "y": 83}, {"x": 95, "y": 87}, {"x":
            91, "y": 94}, {"x": 91, "y": 101}, {"x": 109, "y": 141}, {"x": 111, "y": 143}, {"x": 117, "y": 154}, {"x": 123, "y": 156}, {"x": 147, "y": 157},
            {"x": 195, "y": 146}, {"x": 222, "y": 115}, {"x": 222, "y": 104}, {"x": 221, "y": 96}, {"x": 212, "y": 90}], "annotation": {}, "regionType":
            "freehand"}, {"points": [{"x": 407, "y": 88}, {"x": 406, "y": 82}, {"x": 398, "y": 76}, {"x": 395, "y": 69}, {"x": 390, "y": 67}, {"x": 384, "y":
            67}, {"x": 373, "y": 67}, {"x": 363, "y": 66}, {"x": 354, "y": 66}, {"x": 324, "y": 81}, {"x": 316, "y": 85}, {"x": 309, "y": 89}, {"x": 301, "y":
            97}, {"x": 298, "y": 102}, {"x": 310, "y": 129}, {"x": 318, "y": 146}, {"x": 331, "y": 149}, {"x": 336, "y": 149}, {"x": 356, "y": 157}, {"x": 361
            , "y": 159}, {"x": 387, "y": 161}, {"x": 389, "y": 161}, {"x": 396, "y": 159}, {"x": 405, "y": 150}, {"x": 413, "y": 138}, {"x": 415, "y": 131},
            {"x": 414, "y": 123}, {"x": 414, "y": 114}, {"x": 415, "y": 104}, {"x": 415, "y": 95}, {"x": 415, "y": 90}, {"x": 409, "y": 87}, {"x": 407, "y":
            86}], "annotation": {}, "regionType": "freehand"}]</svg>
    </mark>
</case>
```


Ábra 6.2: Adathalmaz felépítése

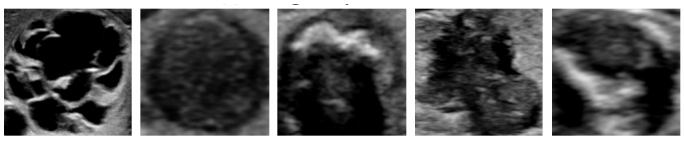
Illetve a ′points′ tömb a szabadkézzel annotált csomók koordinátáit tartalmazza. A ábra illusztrálja a szabadkezű annotálást a fentebb említett Javascript rajzoló interfész segítségével.



Ábra 6.3: Eredeti és annotált kép bemutatása
[20]

A következőekben néhány példát szeretnék bemutatni a pajzsmirigy jó- és rosszindulatú csomóira az adatbázisban található ultrahang képek alapján. Az első sorban lévő képek (6.4) jóindulatúak, a többiek pedig rosszindulatúak (6.5).

Ábra 6.4: Jóindulatú pajzsmirigy csomó
[25]

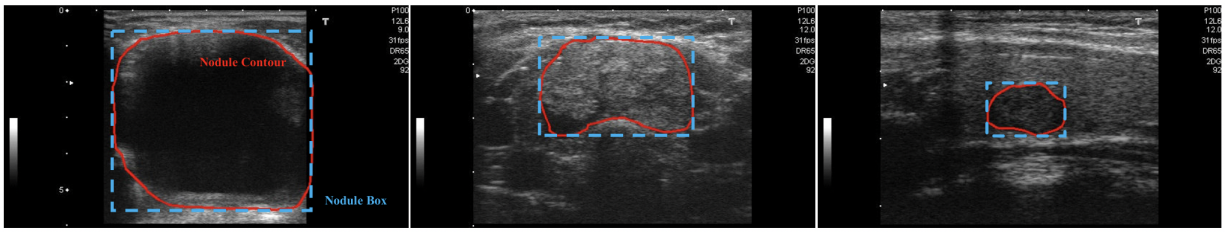


Ábra 6.5: Rosszindulatú pajzsmirigy csomó
[25]

## 6.1 Adatok előfeldolgozása

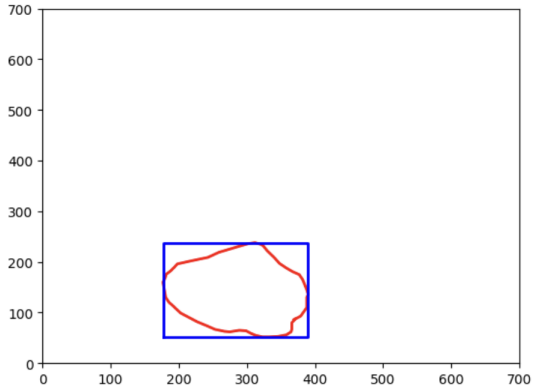
A fentebb ismertetett adathalmazt a tanítás megkezdése előtt szükséges átalakítani. Számomra releváns adatot főként a points tömb, illetve a tirads pontszám. Az utóbbi a csomó elhelyezkedéséről szolgáltat információt, míg a másik a malignitás megállapításában lesz fontos.

Azonban az előbb említett tömb, olyan pontok halmaza, ami megadja a szabadkézzel felrajzolható annotációját a csomónak. Én esetemben határoló dobozokra (bounding box), tehát egy négyszög koordinátáira lesz szükség. Ezen tömb nagyon könnyen átalakítható. A megadott koordinátákból mindig az adott tengelyen megfelelő maximum és minimum értéket megkeresve előállítható a szükséges négy pont.




Ábra 6.6: Átalakítás illusztrációja tanulmány szerint
[25]

```
Freehand points: [{'x': 371, 'y': 87} {'x': 366, 'y': 80} {'x': 366, 'y': 67}
 {'x': 365, 'y': 62} {'x': 358, 'y': 56} {'x': 346, 'y': 53}
 {'x': 333, 'y': 52} {'x': 323, 'y': 52} {'x': 313, 'y': 55}
 {'x': 306, 'y': 59} {'x': 299, 'y': 64} {'x': 289, 'y': 65}
 {'x': 275, 'y': 62} {'x': 267, 'y': 63} {'x': 253, 'y': 67}
 {'x': 243, 'y': 73} {'x': 227, 'y': 82} {'x': 203, 'y': 99}
 {'x': 195, 'y': 109} {'x': 192, 'y': 113} {'x': 186, 'y': 120}
 {'x': 181, 'y': 130} {'x': 177, 'y': 160} {'x': 180, 'y': 168}
 {'x': 182, 'y': 176} {'x': 188, 'y': 182} {'x': 194, 'y': 190}
 {'x': 198, 'y': 196} {'x': 218, 'y': 202} {'x': 243, 'y': 209}
 {'x': 259, 'y': 219} {'x': 289, 'y': 231} {'x': 302, 'y': 236}
 {'x': 312, 'y': 238} {'x': 320, 'y': 235} {'x': 324, 'y': 232}
 {'x': 330, 'y': 222} {'x': 340, 'y': 209} {'x': 348, 'y': 197}
 {'x': 358, 'y': 188} {'x': 367, 'y': 181} {'x': 377, 'y': 175}
 {'x': 382, 'y': 165} {'x': 388, 'y': 145} {'x': 390, 'y': 136}
 {'x': 388, 'y': 130} {'x': 388, 'y': 113} {'x': 388, 'y': 110}
 {'x': 379, 'y': 93} {'x': 369, 'y': 86}]
Bounding box: BoundingBox(177, 390, 52, 238)
```




Ábra 6.7: Átalakítás illusztrációja általam

Az így keletkezett értékeket elmentettem és a további könnyebb feldolgozás érdekében az XML állományt átalakítottam JSON fájlokká.


```
{
    "case_id": "162_1",
    "age": "40",
    "sex": "F",
    "composition": "solid",
    "echogenicity": "hypoechogenicity",
    "margins": "ill defined",
    "calcifications": "non",
    "tirads": "4c",
    "reportbacaf": "",
    "reporteco": "",
    "bboxes": [
        {
            "x": 367,
            "y": 78,
            "w": 84,
            "h": 95
        },
        {
            "x": 101,
            "y": 86,
            "w": 126,
            "h": 89
        }
    ]
},
```


Ábra 6.8: JSON fájl felépítése

# 7 Fejlesztés

## 7.1 Szoftver bemutatása

### 7.1.1 Neurális hálózat fejlesztése

A területen történő számos kutatás átolvasása után tehát a Faster R-CNN modell mellett döntöttem. A dolgozatomban azt vizsgáltam meg, hogy hogyan lehet felhasználni a Pytorch előtanított Faster R-CNN modelljét a pajzsmirigy csomók felismerésére. A modellt finomhangoltam a fentebb ismertetett ultrahang képek és hozzájuk tartozó annotációk felhasználásával. Maga az architektúra egy ResNet gerinchálózatból, egy RPN-ből a javasolt területek kinyeréséhez, és egy Fast R-CNN hálózatból áll az objektumok osztályozásához és a határoló dobozok kijelöléséhez. A modell előzetesen tanított súlyai a COCO adatkészletből származtak, és az SGD (stochasztikus gradiens csökkentés) algoritmus segítségével finomítottam őket. A beállított modellt az egyes kísérletek után egy külön validációs adatkészleten értékeltem az átlagos pontosság (mAP) és az átfedési együttható (IoU) metrikák segítségével.

A tanítási környezet előkészítése és a különböző algoritmusok implementálása, csak úgy, mint az adathalmaz előfeldolgozása, a szabadkezű annotációk átalakítása, képek újra méretezése, data augmentáció, valamit maga a tanítás, a Google Colab felhőalapú szolgáltatás segítségével zajlott. Nagy számítási kapacitásának köszönhetően az adatfeldolgozás és a modell tanítása hatékonyan végre hajtható volt.

A fejlesztés során több előre telepített Python csomagot is igénybe vettem, mint például a NumPy-t, Pandas-t és OpenCV-t.

Az adathalmaz tárolására a Google Drive-ot választottam, mivel könnyen integrálható a Google Colabbal.

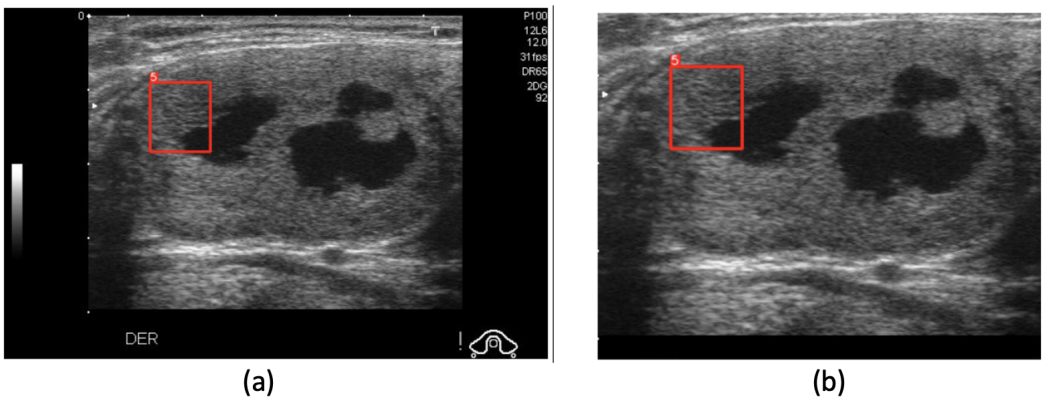
Az adathalmaz a különböző kísérletek során folyamatosan változott, így minden módosítást rögzítettem és mentettem a Google Drive-ra, hogy biztosítsam az adatok integritását. Emellett a modell fejlesztése során a különböző állapotokat is ide mentettem el, hogy biztonságban tartsam azokat, és későbbi összehasonlítási metrikák alapján értékelhessem a fejlesztést. Ezzel a módszerrel biztosítható volt, az adatok és modellek hatékony kezelhetősége, valamint, hogy biztonságban tarthatóak legyenek a fejlesztés során.

Az első tanításnál az adatbázisban található képek szerkezetén, méretén semmit sem változtattam. Azonban hamar rájöttem, hogy a hatékony tanításhoz jóval több előfeldolgozás alá kell vetni a bemeneti adatokat. A predikciók magabiztossága alacsony volt, illetve nem csak a releváns objektumokat tekintette fontosnak.

Ezután a második próbálkozásnál a képek átméretezésével és a fölösleges részek levágásával jobban kiemeltem a hasznos információkat. Ehhez az Albumentations

Python könyvtárat használtam fel, amely számos képfeldolgozási technikát kínál az adatok előkészítéséhez deep learning algoritmusok tanításához. Az átméretezést során a határoló dobozokat is arányosan kellet transzformálnom, amiről a felhasznált könyvtár megvalósított nekem.

Az így előállított képek 350 pixel magassággal, valamint 400 pixel szélességgel rendelkeznek.




Ábra 7.1: Kép az újraméretezésről (a - előtte, b - utána)

A képeken jól látható, hogy az újra méretezés sikeresen megtörtént úgy, hogy a határoló dobozoknak a helyzete nem változott.

Az újraméretezés végrehajtása után, a tanítás eredménye már sikeresnek volt mondható.

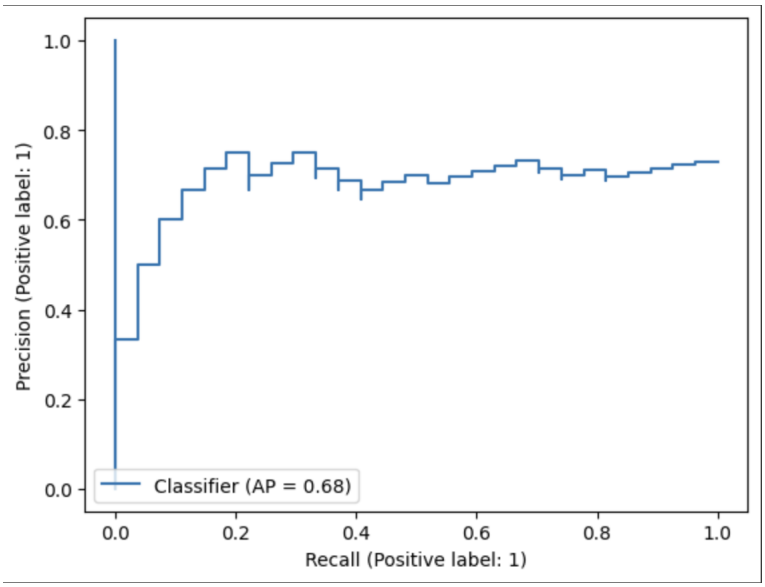
A használt adatbázis számossága, eloszlása a 7.1 táblázatban látható.

| | **Esetek** | **Pozitív** | **Negatív** |
|---|---|---|---|
| Tanító halmaz | 274 | 232 | 42 |
| Validáló halmaz | 35 | 26 | 9 |
| Teszt halmaz | 35 | 25 | 10 |
| Össz halmaz | 344 | 283 | 61 |

Táblázat 7.1: Az adatbázis számossága az adatbővítés előtt

| **IoU threshold** | **mAP érték** |
|---|---|
| 0.5 | 0.68 |
| 0.75 | 0.65 |
| 0.9 | 0.63 |

Táblázat 7.2: Elért mAP az adatbővítés előtt




Ábra 7.2: Pontosság-felismerési görbe az adatbővítés előtt

Az adathalmaz számossága kevés, és a csomók mérete, alakja és textúrája jelentősen eltér, ezért az elért eredmény az adathalmaz összetételére való tekintettel jónak ítélhető.

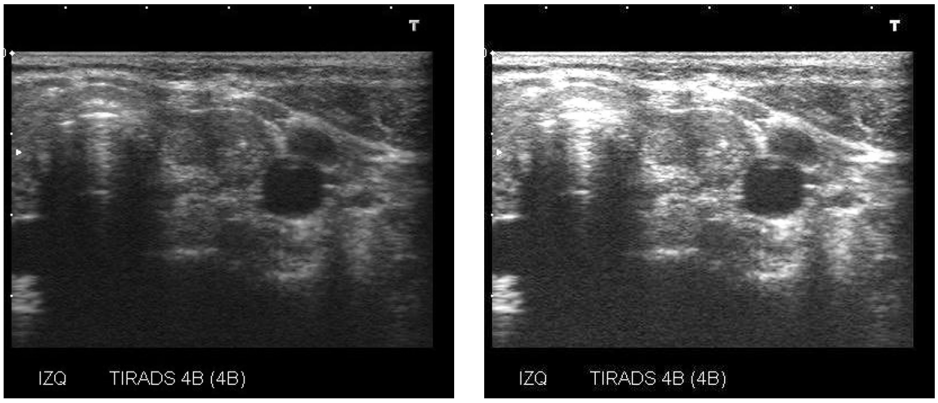
Következő mérföldkőnek az adathalmaz bővítést tűztem ki. Az adatbővítés (data augmentation) a gépi tanulási modell tanításhoz rendelkezésre álló adatok mennyiségének mesterséges növelését jelenti, a már meglévő adatok új, módosított változatainak létrehozásával. Az adatok növelésének leggyakoribb módszerei közé tartozik a képek átfordítása, elforgatása, méretezése, kivágása és zaj hozzáadása. Segíthet a modellnek abban, hogy megtanuljon változatosabb mintákat és jellemzőket felismerni.

A modell általánosításának javítására és az osztályok kiegyensúlyozatlanságának kezelésére a meglévő adatokat bővítettem, új, kissé módosított változatok létrehozásával.

Ehhez a BBAug (Bounding Box Augmentation) [26] python csomagot használtam, amely kifejezetten az objektumdetektálási és lokalizációs feladatokra szabott eszközöket biztosít az adatok bővítéséhez.

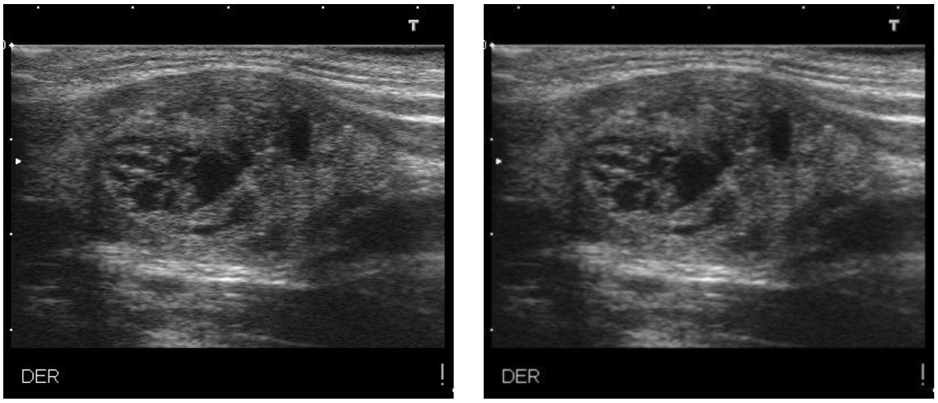
A BBAug az augmentációs technikák széles skáláját kínálja, beleértve a geometriai transzformációkat, mint a forgatás, méretezés és átfordítás, valamint a fotometriai transzformációkat, mint a fényerő és kontraszt beállítások, színtér átalakítások és zaj hozzáadása. Ezek a transzformációk mind a bemeneti képre, mind a képen belüli objektumokat jelölő határoló dobozokra is alkalmazhatók. A határoló dobozok az alkalmazott transzformációkhoz képest helyesen lokalizáltak maradnak.

Az adathalmaz bővítésekor az alábbi transzformációkat használtam: a kép elforgatása, homályosítása, illetve kontraszt növelése. Az alap adathalmaz méretét dupláztam minden esetben.




(a) (b)

Ábra 7.3: Példa a kép kontrasztjának növelésére ( a - eredeti, b - módosított)




(a) (b)

Ábra 7.4: Példa a kép homályosítására ( a - eredeti, b - módosított)

A bővített adatbázis számossága, illetve elosztása a 7.3 táblázatban látható.

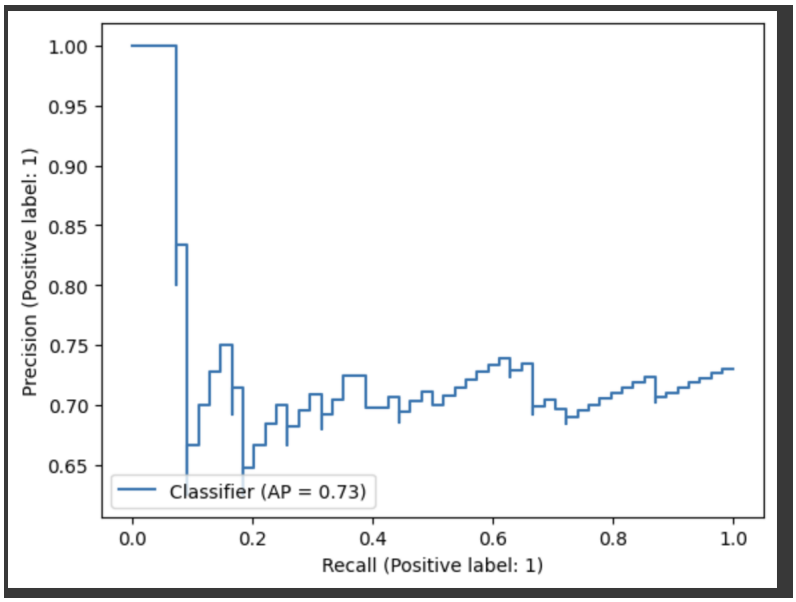
| | **Esetek** | **Pozitív** | **Negatív** |
|---|---|---|---|
| Tanító halmaz | 548 | 464 | 84 |
| Validáló halmaz | 70 | 52 | 18 |
| Teszt halmaz | 70 | 50 | 20 |
| Össz halmaz | 686 | 566 | 122 |

Táblázat 7.3: Az adatbázis számossága az adatbővítés után

Az adatbővítést követően elért eredmények ábrázolva vannak a 7.5 grafikonon.

Apró javulás látható az adatok bővítése után. Az adatbővítés elérte a célját, hiszen a modell pontosságát sikerült - ha nem sokkal - de javítani.

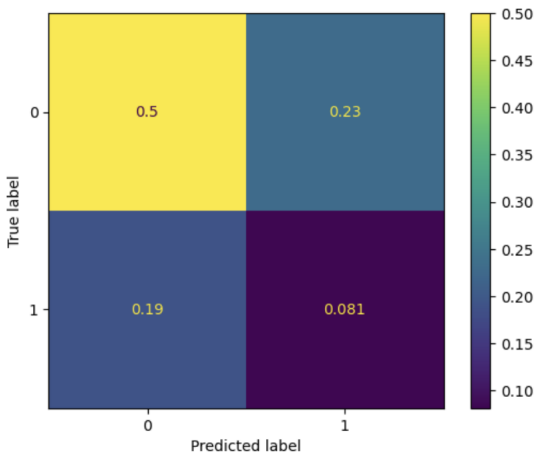
Az osztályok pontosabb klasszifikálására vonatkozó részletek a rendszer által használt neurális modellről az alábbi 7.4 táblázatban, illetve 7.6 ábrán láthatók.




Ábra 7.5: Pontosság-felismerési görbe az adatbővítés után

| | **precision** | **recall** | **f1-score** |
|---|---|---|---|
| jóindulatú | 0.50 | 0.15 | 0.23 |
| rosszindulatú | 0.75 | 0.94 | 0.84 |
| **accuracy (pontosság)** | | | 0.73 |

Táblázat 7.4: Osztályonként elért pontosság




Ábra 7.6: Konfúziós mátrix (0 - rosszindulatú, 1 - jóindulatú)

| IoU threshold | mAP érték |
|---|---|
| 0.5 | 0.73 |
| 0.75 | 0.71 |
| 0.9 | 0.71 |

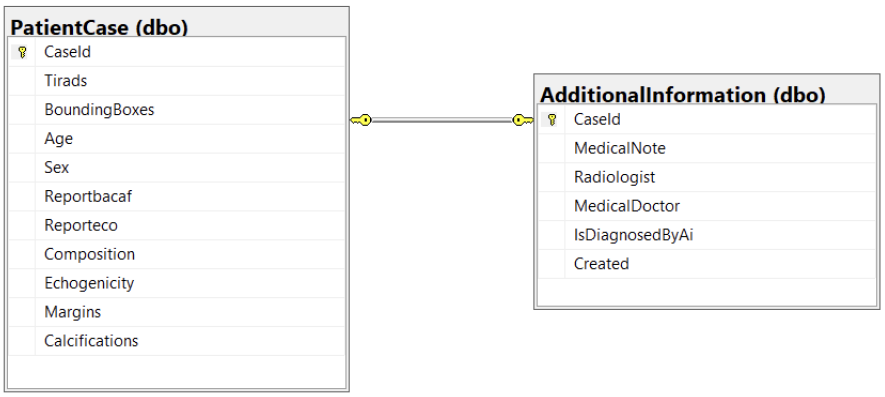
Táblázat 7.5: Elért mAP az adatbővítés után

A kapott eredmények alapján megállapítható, hogy a neurális modell hatékonyan teljesít a rosszindulatú csomók osztályozásában, ugyanakkor a jóindulatú eseteknél alacsonyabb pontosságot ér el. Ennek oka valószínűleg az, hogy a tanító adatbázisban a pozitív esetek aránya magasabb, ami miatt a modell nem tudta kellően megtanulni a negatív esetekre vonatkozó fő jellemzőket és különbségeket.

### 7.1.2 Webalkalmazás

A webalkalmazás implementálásához az MVC (Model-View-Controller) tervezési mintát használtam. Ennek a mintának a segítségével könnyen megvalósítható volt az alkalmazás különböző részeinek az elkülönítése. Ezzel növelve az alkalmazás modularitását, tesztelhetőségét és karbantarthatóságát.

Az MVC tervezési minta tehát három fő részből áll. Először a Modell rész került kialakításra, azaz az alkalmazás adattárolására szolgáló része. Ehhez első sorban szükségem volt egy adatbázis szerver létrehozására. A Microsoft Azure Cloud nevezetű felhőszolgáltatás mellett döntöttem, ahol létrehozott adatbázis szerver az alábbi diagrammon látható táblákat tartalmazza.




Ábra 7.7: Adatbázis diagramm

A PatientCase tábla szolgál a beteg adatok tárolására, aminek mezői megegyeznek a használt adatbázisban található xml fájlok felépítésével. Az AdditionalInformation tábla pedig eltárol számos plusz információt a betegekhez, miszerint a neurális hálózat által lett diagnosztizálva, emellett megjegyzéseket is tárol, amit a felhasználók adhatnak meg az adott pajzsmirigy csomóról.

Majd a Vezérlő (Controller) rész, ami tartalmazza az alkalmazás logikáját. A modellt és a nézetet kapcsolja össze, és feldolgozza a felhasználói interakciókat. Ezeket a kommunikációs pontokat REST API interfész segítségével alakítottam ki. Amiket aztán Swagger-el manuálisan teszteltem.

Amik közül egyedül a statisztikai lekérdezéseket szolgáló folyamatot emelném ki, ahol a bemeneti paraméterek a különböző szűrőfeltételek lehetnek. Ilyen feltétel lehet például a beteg kor intervalluma (ageFrom - ageTo), maga a Tirads pontszám, vagy akár a nemi alapú lekérdezés.

```
StatisticsQuery ⌄ {
    ageFrom          integer($int32)
                     nullable: true
    ageTo            integer($int32)
                     nullable: true
    tiradsScore      string
                     nullable: true
    gender           string
                     nullable: true
}
```

Ábra 7.8: Staisztikai bemeneti objektum felépítése

```
PatientCaseDTO ⌄ {
    caseId           string
                     nullable: true
    tirads           string
                     nullable: true
    age              integer($int32)
    sex              string
                     nullable: true
    medicalNote      string
                     nullable: true
    created          string
                     nullable: true
    medicalDoctor    string
                     nullable: true
    radiologist      string
                     nullable: true
    isDiagnosedByAi  integer($int32)
                     nullable: true
}
```

Ábra 7.9: Staisztikai lekérdezés kimeneti objektum felépítése

Majd legutoljára a felhasználói felület azaz a Nézet rész került megvalósításra.

A front-end fejlesztése Vue.js keretrendszerben zajlott. A különböző HTTP kéréseket a szerver felé, az Axios nevezetű Javascript könyvtárral valósítottam meg, amely intuitív API-t biztosított a HTTP kérésekhez ebből fakadóan könnyedén kezelhetők voltak a hibák, kérések (response) illetve a válaszok (request) is.

Az adatok megjelenítéséhez Vue.js keretrendszer adatkötési lehetőségeit használtam, amelynek köszönhetően a felhasználónak megjelenő adatok állapotának automatikus szinkronizációját sikerült megvalósítanom. Emellett az alkalmazásban használtam a v-model, v-show és v-for beépített direktívákat is, amelyek segítségével összekapcsoltam a különböző komponenseket, lekérdezési formokat és a válaszként kapott iteratív elemeket, például listákat is megjelenítettem.

A különböző komponensek kialakításához egy másik népszerű front-end webfejlesztési keretrendszert is használtam, amelynek neve Bootstrap. Előre tervezett HTML, CSS és JavaScript komponenseket kínál, így könnyedén implementálhatóak voltak a különböző oldalak.

A oldalak kinézeteinek ismertetése a 7.2 fejezetben részletes bemutatásra kerül.

### 7.1.3 Objektum detektáló alkalmazás fejlesztése

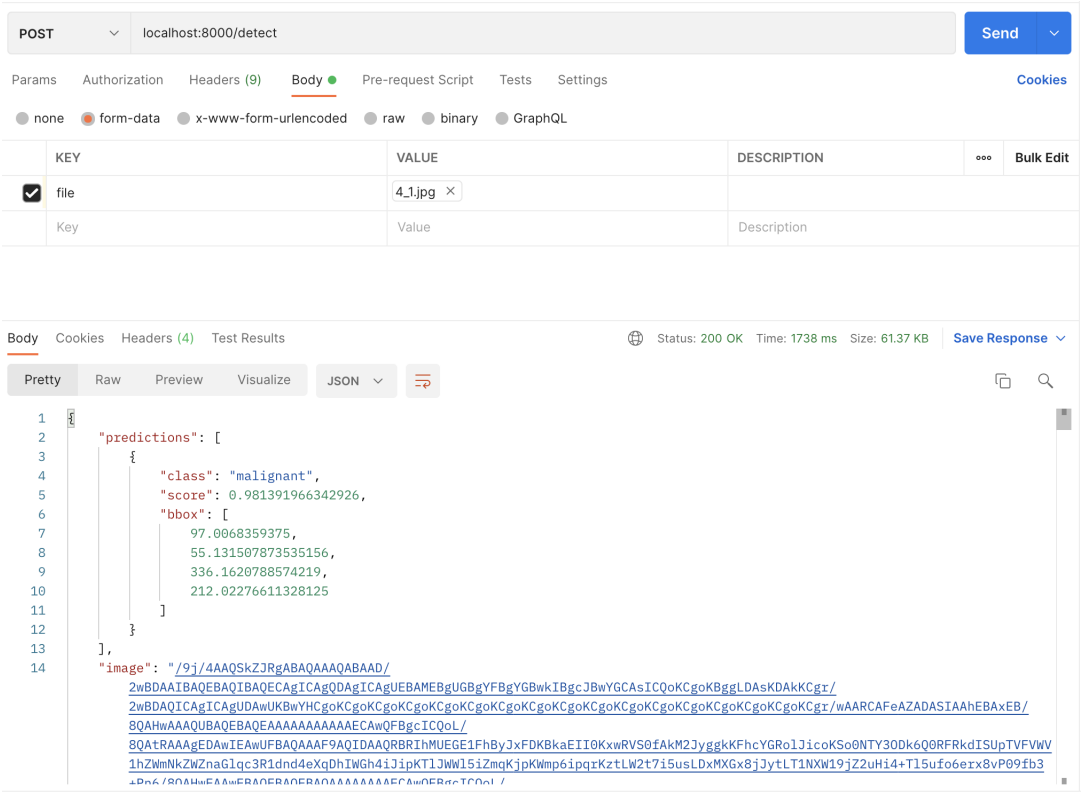
Az alkalmazásban a kommunikációs folyamatokat, mint például az adatbázis elérése és a frontend-backend közötti kérések, REST API interfészekkel valósítottam meg, ahogy már korábban említettem. Az objektumfelismerésre tervezett alkalmazásban is ugyanezt a megközelítést használtam, melyhez a FAST API-t alkalmaztam.

A FAST API [27] egy gyors és modern Python webkeretrendszer, amely segít az API-k könnyű és hatékony kialakításában. Könnyedén lehet ennek segítségével definiálni az API-k útvonalait, validálni a bemeneti, valamint kimeneti adatokat.

A végpont a pajzsmirigy csomóról készült ultrahangos képet várja bemenetként, amelyet a felhasználó ad meg a fent említett webalkalmazáson keresztül. Ezután a kimentett neurális modell elvégzi a kép feldolgozását, majd a detektált eredményt, beleértve az osztályt, a predikció pontosságát, a határoló doboz helyét, és magát a detektáció alapján felannotált képet Base64 kódolással, JSON formátumban küldi vissza a felhasználói felületnek.

A Base64 kódolás során a kép bináris reprezentációját ASCII karakterek sorozataként ábrázoltam, annak érdekében, hogy HTTP kérésen keresztül továbbítani tudjam a webalkalmazásnak. A felhasználói felületen ezen karakterlánc dekódolásával a kép megjeleníthető.

A végpont működésének tesztelését Postman segítségével végeztem. Az alábbi 7.15 ábrán látható módon.




Ábra 7.10: Objektum felismerés Postman teszt

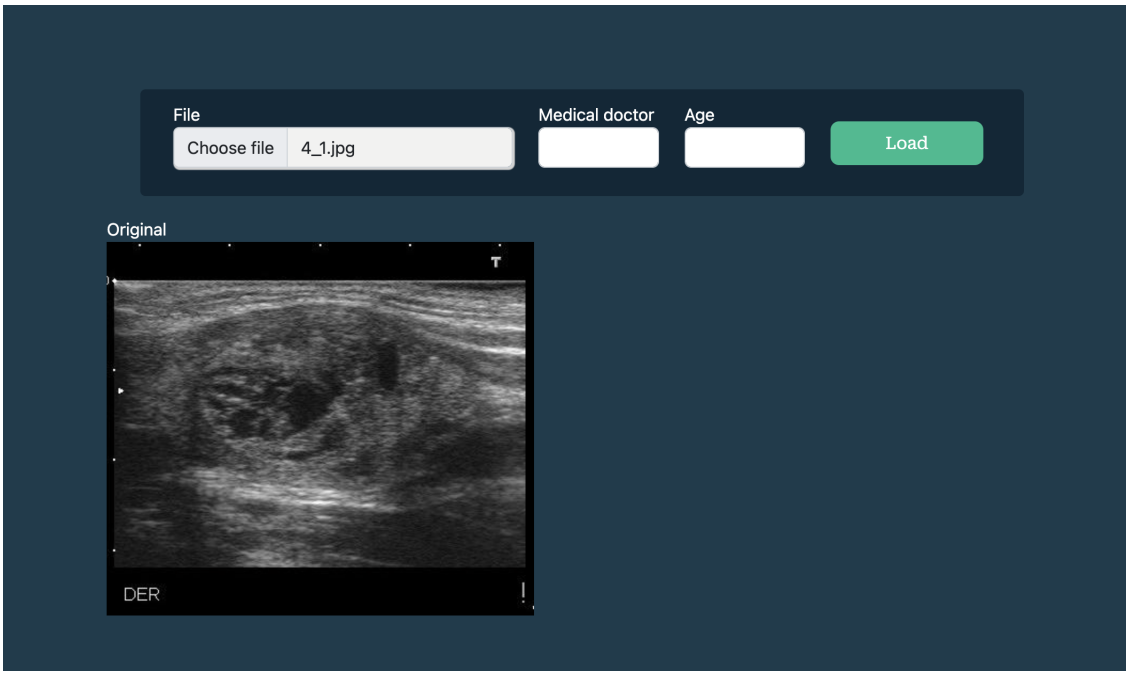
Az ábrán jól látszik, hogy a kérés sikeres volt, és a visszatérő predikció tömb értéke is látható.

## 7.2 Tesztelés

A rendszer tesztelése manuális történt.

Az elsődleges tesztelési folyamat során a szerverkérések tesztelésére került sor a fejlesztés során, amelyeket a már említett Swagger és Postman eszközökkel segítségével végeztem el.

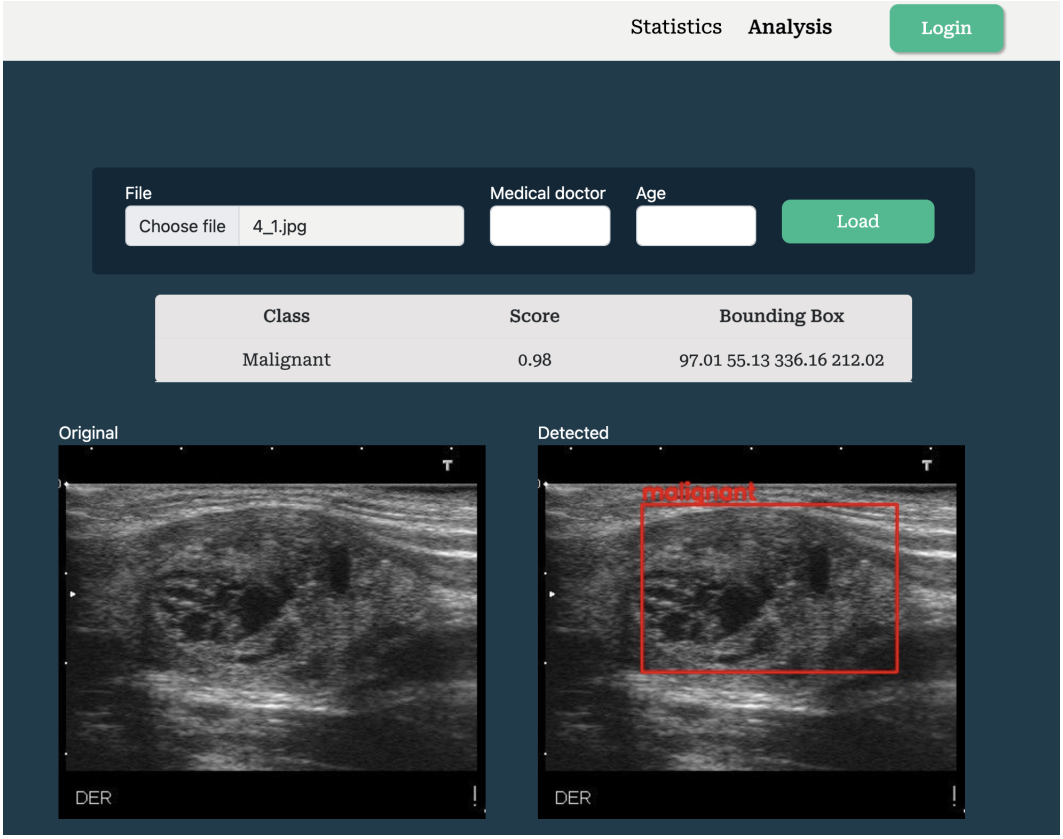
Az alkalmazás különböző részeinek integrációja után az egész rendszer működésének átfogó tesztelését pedig az alábbiakban szeretném ismertetni. Az elsődleges tulajdonság és egyben legfontosabb, amit bemutatnék a rendszerben, az az objektumdetektálás. A felület igen felhasználóbarát, lehetővé teszi a felhasználó számára, hogy feltöltsön egy képet, amelyen az objektumdetektálás azonnal elvégezhető és eközben az eredeti kép is megjelenik.




Ábra 7.11: Eredeti kép megjelenítése

A felismerés lefutása következtében egy táblázatban jelenik meg a klasszifikált objektumok listája, valamint a predikciókhoz tartozó magabiztossági értékek és a határoló dobozok lokációi. Ezenfelül az eredeti kép mellett megjelenik a predikció által felannotált kép, amelynek segítségével könnyen összehasonlítható a prediktált és az eredeti állapot.

A biztosított vizuális megjelenítés elősegíti a felhasználókat az objektumok könnyű azonosításában és a hibák korrigálásában. Egyértelműen látható, hogy melyik objektumot milyen magabiztossággal detektált a model, valamint hogy hol helyezkedik el a határoló doboz, amely további pontosságot és érthetőséget biztosít az adott pajzsmirigy csomóról a felhasználó számára.



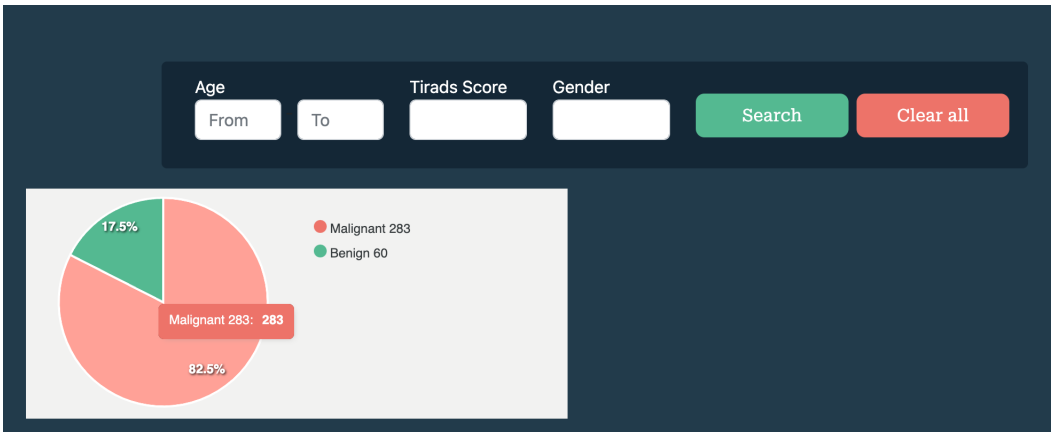

Ábra 7.12: A predikció megjelenítése

Az elvégzett detektálások eredményei elmentésre kerülnek az adatbázisba annak érdekében, hogy azok rendelkezésre álljanak a későbbi lekérdezésekhez.

A lekérdező felület egyszerű és átlátható, de lehetőséget biztosít számos további kiegészítésre is, azonban ezen dolgozat keretei között nem volt lehetőségem ezeknek a kiegészítéseknek megvalósítására. Azonban, ha az alkalmazás későbbi verzióiban további funkciókat terveznék hozzáadni, a felület bővítése viszonylag könnyen megvalósítható lenne.

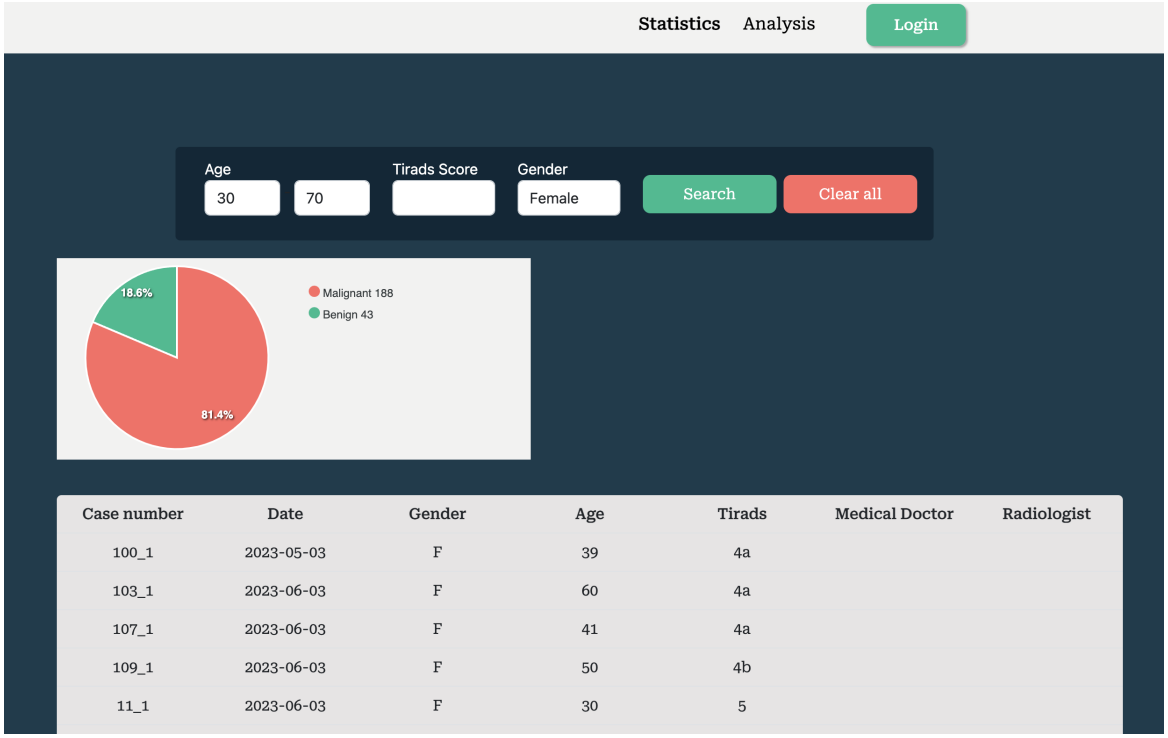
Egyszerűsége ellenére a felhasználó számos lekérdezés végre hajhat, és láthatja a beállított szűrőfeltételek alapján az adathalmaz rosszindulatú, illetve jóindulatú eloszlását.

Kiindulásként a diagramm az egész adathalmazra vonatkozó megosztás szolgáltatja, majd a különböző szürőfeltételeknek beállított értékek alapján jeleníti meg az adatokat.



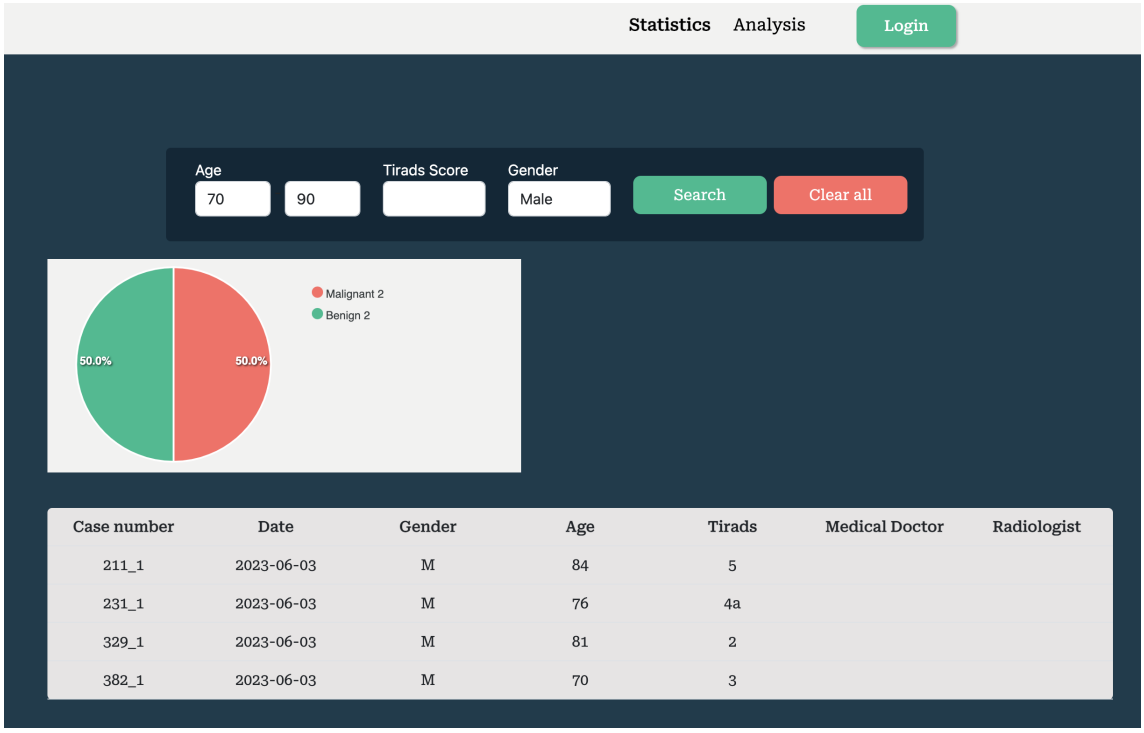

Ábra 7.13: Lekérdező felület alap beállítással

Ahogyan az 7.14 ábrán is látható, a felhasználó számos lehetőséggel rendelkezik a lekérdezések paramétereinek megadásához. A kor tartomány megadása mellett lehetősége van megadni egy Tirads pontszámot, ami alapján a rendszer azonosítja azokat az eseteket. Továbbá kiválaszthat egy nem alapján történő megosztást is, hogy pontosabb eredményeket kapjon.




Ábra 7.14: Lekérdező felület "nő" szűrő feltétellel

A lekérdező felület funkcionalitásának bemutatására készítettem további teszteseteket, amelyek segítségével részletesebben reprezentálni tudtam az alkalmazás működését.




Ábra 7.15: Lekérdező felület "férfi" szűrő feltétellel

# 8 Eredmények

## 8.1 Eredmények összevetése a szakirodalommal

Kutatásom során talált, néhány hasonló adatbázison végzett kísérletek eredményét használtam fel referencia értékként. Az egyik tanulmányban [22], ahogy a 4.4.1.fejezetben is említettem, a Faster R-CNN tanításának performanciáját különböző küszöb értékek (IoU threshold) mellet megvizsgálva, az alábbi adathalmaz elosztással (8.1 táblázat), a következő eredményeket érték el (8.2 ábra).

| | **Esetek** |
|---|---|
| Tanító halmaz | 400 |
| Validáló halmaz | 50 |
| Teszt halmaz | 50 |
| Össz halmaz | 500 |

Táblázat 8.1: A referencia tanulmány adatbázisának számossága

Ezek az eredmények kizárólag rosszindulatú elváltozások detektálását jellemzik.

| IoU threshold | mAP érték |
|---|---|
| 0.5 | 0.68 |
| 0.75 | 0.53 |
| 0.9 | 0.36 |

Táblázat 8.2: A referencia tanulmány elért eredménye

Az általam elért pontosság a korábban részletezett tanítóhalmazon és egyéb adatfeldolgozási módszereken alapulva került meghatározásra. Az eredményeket a 8.3 táblázatban részleteztem ugyanúgy az átlagos pontossági mérőszám alapján, csak a rosszindulatú elváltozásokra nézve.

| IoU threshold | mAP érték |
|---|---|
| 0.5 | 0.73 |
| 0.75 | 0.71 |
| 0.9 | 0.71 |

Táblázat 8.3: Elért eredmények

Az értékek alapján jól látszik, hogy sikerült a referenciaként vett tanulmányban használt módszer precizitását, hanem sokkal, de felülmúlni.

Természetesen számos tanulmány került publikálásra, amelyek során sikeresebb eredményeket értek el, mint amilyeneket nekem sikerült a saját kutatásom során. Érdemes megemlíteni, hogy ezek a kutatások általában nagyobb méretű és változatosabb adatbázisokkal dolgoztak, amelyek lehetővé tették a modell számára, hogy jobban megtanulja a főbb jellemzőket és különbségeket az adott osztályok között.

Azonban a referenciaként használt tanulmány két kisebb adatbázison alapult, amelyek közül az egyik az általam felhasznált adatbázis volt. Ennek köszönhetően releváns összehasonlítási adatokat sikerült elérnem.

## 8.2 Eredmények értékelése

Dolgozatom munkájának értékelése pozitívnak mondanám, amelyet az előbbiekben ismertetett fejlesztési és tesztelési eredmények alátámasztanak.

Sikerült elérni a kitűzött célokat, ugyanis az általam fejlesztett rendszer egy átfogó alkalmazás, amely egy felhasználóbarát felületen keresztül biztosítja a betegek adatainak hozzáférhetőségét. Emellet, amit szeretnék hangsúlyozni, a rendszer hatékonyan detektálja a pajzsmirigy csomók helyét az ultrahangos képek segítségével, valamint képes azon nodulusoknak a pozitív malignitásának a meghatározására. Továbbá az alkalmazásban elemzési funkciók is találhatóak, amelyek segítségével a szakemberek könnyedén értékelhetik a betegek állapotát.

A fejlesztett rendszernek azonban van egy hátránya, amelyet érdemes megemlíteni. A tanítás során felhasznált adathalmaz többségében rosszindulatú csomókról származó adatokat tartalmazott, ami azt eredményezte, hogy a rendszer jobban teljesít a rosszindulatú esetek felismerésében, mint a jóindulatú esetekében. Ennek köszönhetően a jóindulatú esetek diagnózisa nem olyan pontos, mint amilyen pontos lehetne, ha a tanítás során több jóindulatú esetről is rendelkeznék adatokkal.

# 9 Továbbfejlesztési lehetőségek

Az orvosi terület folyamatosan változik és fejlődik. Az egészségügyben való alkalmazása a neurális hálózatoknak még számos új megközelítést, illetve kihívást vet fel önmagában, amelyekkel a kutatóknak szakembereknek foglalkozniuk kell. Az efféle rendszerek számára kiemelten fontos, hogy mindig megbízható eredményt szolgáltassanak, mivel az egészségügyi döntéshozók és a betegek életét érintik.

Az egészségügyi alkalmazások fejlesztése során felmerülő problémák, többek között a nagy mennyiségű és változatos adatok kezelése, az algoritmusok optimalizálása, és nem utolsó sorban a megbízhatóság biztosítása. A hatékonyság és megbízhatóság bizonyítása érdekében állandó ellenőrzésre van szükség, és számos tesztet, valamint validációt kell elvégezni.

Ennek megfelelően, a szakdolgozat témájaként kiválasztott területen is még számos lehetőség rejlik, amelyekkel javítható a pajzsmirigy csomók detektálásának pontossága és hatékonysága. Azonban szeretnék ismertetni néhány, általam kigondolt gyakorlati kiegészítési lehetőséget a fentebb hangsúlyozott problémák alapján, amelyek konkrétan a dolgozatomként megvalósított alkalmazás kiegészítési lehetőségeire fókuszálnak.

Az egyik továbbfejlesztési lehetőség, egy olyan megoldás, amiben egy nagyobb adathalmazon tanítanám a neurális hálózati modellt, amely több jóindulatú esetet is tartalmazna, és összeségében több adott pontszámhoz hozzárendelhető esetet is. Az ilyen tanító adathalmaznak nagyban hozzá kellene járulnia a Tirads pontszám helyes meghatározásához. Így a modell nem csak azt tudja megállapítani, hogy az adott nodulus rossz vagy jóindulatú, hanem képes lenne magára a Tirads pontszám helyes megállapítására is.

A nagyobb és megosztóbb adatbázis létrehozása azonban nem könnyű feladat, és jelenleg az ilyen jellegű kutatások nem jellemzőek az adott területen. Ha azonban sikerülne ilyen adathalmazt létrehozni, az a pajzsmirigy csomók diagnosztizálásában és kezelésében jelentős előrelépést jelentene, és segítené az orvosokat a hatékonyabb és pontosabb kezelési terv kialakításában.

Mivel a diagnózis felállítása a pajzsmirigy csomók esetében kizárólag magasan képzett patológusok és radiológusok szaktudásán alapul, aminek megszerzése hosszú időt vesz igénybe, ezért az alkalmazás egy másik ígéretes bővítési lehetősége lehetne egy tanuló programmal való kiegészítése. Ez alatt egy olyan felületre gondolok, ahol a tanuló képes lenne, ugyanúgy különböző esetek megvizsgálására a neurális modell segítségével. A kapott detektált eredmények alapján a tanuló képes lenne saját tudásának helyességét ellenőrizni, vagy esetlegesen a modell által vétett lokalizációs, klasszifikációs hibákat kijavítani. A módosított, helyesen felannotált képek pedig elmentésre kerülnének, ezzel is bővítve az adatbázist. Továbbá így egy folyamatos ellenőrzés kerülne kialakításra, ami a modell hatékonyságát javítaná.

# 10 Összefoglaló

A lakosság számottevő többsége szenved valamilyen endokrin betegségtől, amelynek hátterében a legtöbb esetben valamilyen pajzsmirigy rendellenesség áll. Ilyen pajzsmirigy rendelleneségeket tud okozni akár egy apró fizikai elváltozás, mint a pajzsmirigy csomó. Ebből következően dolgozatom célja egy orvosi döntést támogató rendszer létrehozása volt.

Pajzsmirigy diagnosztika felállítása első sorban a nyak képalkotó vizsgálatával történik. Az így készült ultrahangos képek alapján különböző információk nyerhetők ki a pajzsmirigy csomókról, mint például összetétel, echogenitás, alak, marginok/szélek, és echogén gócok. Ezen jellemzők osztályozására definiálva van egy egységes pontozási módszer, a TI-RADS rendszer. A táblázat minden egyes tulajdonsághoz hozzárendel egy pontszámot, majd azok összege megadja egy 1-től 5-ig tartó skálán a malignitás gyanújára való esélyt. A négyes, illetve ötös pontszám nagy valószínűséggel rosszindulatú csomó, míg a hárommal egyenlő vagy kisebb, jóindulatút jelent. Az így kapott pontszámok nagyon jól alkalmazhatók egy neurális hálózati problémára.

A területen már több kutatás is történt különböző adathalmazokon, eltérő módszerekkel. Az eltérő adathalmazok számossága, illetve az alkalmazott módszerek sokfélesége miatt a kutatási eredmények összehasonlítása és a neurális hálózati modell kiválasztása nehéznek bizonyult.

Azonban sikerült találni egy tanulmányt, amely összehasonlít két korszerű objektumdetektálási módszert, a Faster R-CNN és YOLO alkalmazását. A modellek tanítására, valamint validálására az általam felhasznált nyílt hozzáférésű, előre felannotált ultrahangos képeket tartalmazó adatbázist (Digital Database of Thyroid Ultrasound Images, DDTI) vették igénybe, kiegészítve egy 200 képből álló belső adatbázissal. Az elért különböző küszöbértékeknél vizsgált átlagos pontosság alapján kijelenthető, hogy megfelelően finomhangolt Faster R-CNN modellel teljesebb csomópont lokalizációs eredményt kaphatunk, amely jobban közelít az alapigazsághoz, mint a YOLO architektúrán tanított modell. Ezen tanulmányra alapozva döntésemet egy előre tanított finomhangolt Faster R-CNN modellt használtam fel.

A sikeres tanítás érdekében számos feldolgozást igényeltek a nyers adatok. Először is a XML fájlokban szabadkézzel felrajzolható annotációk jelölik a csomók helyét, viszont nekem a tanításhoz határoló dobozokra, azaz egy négyszög koordinátáira volt szükségem. A megadott pontok halmazából előállítható volt a négy pont, azonban a teljes méretű képek felesleges információkat tartalmaztak. Így az első tanításnál a predikciók magabiztossága alacsony volt, és nem csak a releváns objektumokat tekintette fontosnak. További átalakítások révén az 560x360 méretű képeket lecsökkentettem 400x350 felbontásúvá, ezzel sikerült redukálni a kinyerhető,

megtanulható jellemzők aranyát az ultrahangos felvételekről. Már a modell magasabb valószínűséggel detektálta a pajzsmirigy pontos helyét, azonban alultanítottnak bizonyult, mivel a rosszindulatú csomókat nagyobb arányban detektálta sikeresen, mint a jóindulatúakat. Ez következhet abból, hogy az adathalmaz számossága kevés volt, és a csomók mérete, alakja és textúrája jelentősen eltér. Ezért a modell általánosításának javítására és az osztályok kiegyensúlyozatlanságának kezelésére a meglévő adatokat bővítettem, új, kissé módosított változatok létrehozásával. Transzformálások révén a képek kontrasztját növeltem, illetve homályosítottam, amivel a meglévő halmaz a duplájára növekedett. Adat augmentációval sikerült javítani a modell teljesítményén, ami 73%-os átlagos pontosságot jelentett már a rosszindulatú csomók detektálása terén, azonban a kapott pontosság a jóindulatú esetek terén nem mérvadó.

Hasonló tanulmányokkal összevetve, ahol ugyanazon DDTI adatbázist használták, az általam elért eredmények pozitívnak mondhatók, hiszen sikerült reprodukálni kutatásokban elért eredményeket. A tanító halmaz összetétele miatt tényleges bináris klasszifikáció ezen adatbázissal nem érhető el, mivel a pozitív esetek aránya magasabb, ami miatt a modell nem tudja kellően megtanulni a negatív esetekre vonatkozó fő jellemzőket és különbségeket. Ebből kiindulva egyik ígéretes tovább fejlesztési lehetőség, a modell tanítása egy nagyobb, változatosabb mintákat tartalmazó adatbázison. A megvalósított döntést támogató rendszer több rétegből áll. Az adatok tárolásáért egy Microsoft Azure Cloud-ban létrehozott adatbázis felel, majd a felhasználói interakciók lebonyolítására egy ASP.NET backenddel, illetve Vue.JS frontend keretrendszerben implementált webalkalmazással történik. A különböző alkalmazások REST API interfészeken keresztül kommunikálnak egymással. Maga az objektum detektálást végző alkalmazás FAST API segítségével hajtja végre a kérések lebonyolítását, mint például a kép küldése, majd detektált eredmény továbbítása a felhasználónak. Ezen rendszerben egyszerű, de különböző statisztikai adatok megjelenítésekre is képesek, akár különböző korosztályokban, vagy nemi alapú megkülönböztetés szerint.

Összeségében dolgozatom munkáját pozitívnak mondanám, hiszen sikerült elérni a kitűzött célokat. Az általam fejlesztett rendszer egy átfogó alkalmazás, amely egy felhasználóbarát felületen keresztül biztosítja a betegek adatainak hozzáférhetőségét. A rendszer hatékonyan detektálja a pajzsmirigy csomók helyét az ultrahangos képek segítségével, valamint képes azon nodulusoknak a pozitív malignitásának a meghatározására.

# 11 Summary

Most of the population suffers from some endocrine disease, which is in many cases caused by some form of thyroid disorder. Such thyroid disorders can be caused by even a small physical lesion such as a thyroid nodule. Consequently, the aim of my thesis was to create a medical decision support system.

Thyroid gland diagnosis is first established by imaging of the neck. The resulting ultrasound images provide various information about the thyroid nodules, such as composition, echogenicity, shape, margins/edges, and echogenic nodules. A unified scoring method, the TI-RADS system, has been defined to classify these features. The table assigns a score to each feature and the sum of these scores gives the probability of suspected malignancy on a scale of 1 to 5. A score of four or five indicates a high probability of a malignant nodule, while a score of three or less indicates a benign nodule. The resulting scores are very applicable to a neural network problem.

Several studies have been done in this area on different data sets using different methods. Due to the large number of different data sets and the variety of methods used, comparing the research results, and selecting a neural network model has proved difficult.

However, it was possible to find a study comparing two state-of-the-art object detection methods, Faster R-CNN and YOLO. The open access Digital Database of Thyroid Ultrasound Images (DDTI), an annotated database of ultrasound images, was used to train and validate these models, extended with an internal database of 200 images. Based on the mean average precision score at different thresholds achieved, it can be stated that a properly fine-tuned Faster R-CNN model can be used to obtain more complete node localization results, which are closer to the ground truth than the model trained on the YOLO architecture. I have based on my decision on this study to use pre-trained fine-tuned Faster R-CNN model.

The raw dataset required a lot of preprocessing to ensure successful training. First of all, in XML files, free-hand annotations are used to indicate the location of the nodules, but I need bounding boxes, i.e. the coordinates of a rectangle. The four points could be generated from the set of points provided, but the full-size images contained unnecessary information. Thus, for the first training, the confidence of the predictions was low, and not only the relevant objects were considered important. Through further transformations, I reduced the 560x360 images to 400x350 resolution, thus reducing the ratio of extractible, learnable features from the ultrasound images. Already, the model had a higher probability of detecting the exact location of the thyroid nodules but proved to be under-trained. It can successfully detect malignant nodules at a higher rate than benign ones. This may due to the fact that the number of data sets was small and the size, shape and texture of the nodules differed. Therefore, to improve the generalization of the model and to deal with the

unbalanced treatment of classes, I extended the full data by creating new, slightly modified versions. Transformations were used to increase the contrast of the images and blur them, which doubled the size of the existing dataset. Data augmentation is improved the performance of the model, resulting in an average accuracy of 73% already for the detection of malignant nodules, but the resulting accuracy is not relevant for benign cases.

Compared to similar studies using the same DDTI database, my results are positive, as I was able to reproduce the results obtained in other studies. Due to the composition of the training set, an actual binary classification cannot be achieved with this database, as the ratio of positive cases is higher, which makes the model unable to learn sufficiently the main features and differences for negative cases. From this point of view, one promising avenue for further development is to train the model on a larger database with more diverse samples. The implemented decision support system consists of several layers. A database created in the Microsoft Azure Cloud is responsible for storing the data, followed by a web application implemented in an ASP.NET backend and Vue.JS frontend framework to handle user interactions. The different applications communicate with each other via REST API interfaces. The object detection application itself uses the FAST API to perform requests such as sending an image and then forwarding the detected result to the user. In this system, simple but different statistical data can be displayed, either by different age groups or by gender.

Overall, I would say that the work of my thesis is positive, as I have succeeded in achieving the goals I set. The system I developed is a comprehensive application that provides access to patient data through a user-friendly interface. The system effectively detects the location of thyroid nodules using ultrasound images and can determine the positive malignancy of those nodules.

# Ábrajegyzék


3.1 Pajzsmirigy elhelyezkedése . . . . . 5
3.2 Pajzsmirigy betegségek . . . . . 6
3.3 Pajzsmirigy csomó elhelyezkedése . . . . . 7
3.4 TIRADS pont kalkulátor [2] . . . . . 7
4.1 Szigmoid és tanH függvény összehasonlítása . . . . . 10
4.2 ReLU függvény ábrázolása . . . . . 10
4.3 A IoU kiszámításának ábrázolása . . . . . 11
4.4 Detektálás pontosságának reprezentálása . . . . . 12
4.5 Konfúziós mátrix felépítése . . . . . 12
4.6 Konvolúciós neurális hálózat felépítése . . . . . 14
4.7 Jellemzők kinyerése . . . . . 14
4.8 Pooling módszerek . . . . . 15
4.9 RCNN működése . . . . . 16
4.10 SVM mükődése . . . . . 16
4.11 Fast RCNN működése [6] . . . . . 17
4.12 Faster RCNN működése . . . . . 18
4.13 A halózat mélysége és a performancia kapcsolata [7] . . . . . 19
4.14 A reziduális blokk felépítése [7] . . . . . 19
4.15 Darknet-53 [9] . . . . . 20
4.16 YOLOv3 architektúra részletei [9] . . . . . 21
4.17 Két módszer összehasonlításának eredménye [23] . . . . . 25
5.1 Arhitektúra felépítése . . . . . 28
5.2 Folyamatábra . . . . . 28
6.1 Annotációs eszköz bemutatása . . . . . 29
6.2 Adathalmaz felépítése . . . . . 30
6.3 Eredeti és annotált kép bemutatása . . . . . 30
6.4 Jóindulatú pajzsmirigy csomó . . . . . 30
6.5 Rosszindulatú pajzsmirigy csomó . . . . . 31
6.6 Átalakítás illusztrációja tanulmány szerint . . . . . 31
6.7 Átalakítás illusztrációja általam . . . . . 32
6.8 JSON fájl felépítése . . . . . 32
7.1 Kép az újraméretezésről (a - előtte, b - utána) . . . . . 34
7.2 Pontosság-felismerési görbe az adatbővítés előtt . . . . . 35
7.3 Példa a kép kontrasztjának növelésére ( a - eredeti, b - módosított) . 36
7.4 Példa a kép homályosítására ( a - eredeti, b - módosított) . . . . . 36
7.5 Pontosság-felismerési görbe az adatbővítés után . . . . . 37
7.6 Konfúziós mátrix (0 - rosszindulatú, 1 - jóindulatú) . . . . . 37
7.7 Adatbázis diagramm . . . . . 38

7.8 Staisztikai bemeneti objektum felépítése . . . . . 39
7.9 Staisztikai lekérdezés kimeneti objektum felépítése . . . . . 39
7.10 Objektum felismerés Postman teszt . . . . . 41
7.11 Eredeti kép megjelenítése . . . . . 42
7.12 A predikció megjelenítése . . . . . 43
7.13 Lekérdező felület alap beállítással . . . . . 44
7.14 Lekérdező felület "nő" szűrő feltétellel . . . . . 44
7.15 Lekérdező felület "férfi" szűrő feltétellel . . . . . 45

# Táblázatjegyzék

4.1 Frontend keretrendszerek összehasonlítása . . . . . 24
4.2 A mAP értékek összehasonlítása [23] . . . . . 25
7.1 Az adatbázis számossága az adatbővítés előtt . . . . . 34
7.2 Elért mAP az adatbővítés előtt . . . . . 34
7.3 Az adatbázis számossága az adatbővítés után . . . . . 36
7.4 Osztályonként elért pontosság . . . . . 37
7.5 Elért mAP az adatbővítés után . . . . . 38
8.1 A referencia tanulmány adatbázisának számossága . . . . . 45
8.2 A referencia tanulmány elért eredménye . . . . . 46
8.3 Elért eredmények . . . . . 46

# Irodalomjegyzék

[1] "Development of the thyroid gland". In: *Mikael Nilsson,Henrik Fagman* (2017). DOI: 10.1242/dev.145615.

[2] "Online TI-RADS Calculator". In: *Jimmy Tanche Wang,Tasha Ellchuk,Gary Groot ,Rob Otani* (2018). DOI: 10.4236/ojrad.2018.83020.

[3] Hinton G. LeCun Y Bengio Y. "Deep learning." In: (2015). DOI: 10.1038/nature14539.

[4] Wang Zhiqiang and Liu Jun. "A review of object detection based on convolutional neural network". In: (2017), pp. 11104–11109. DOI: 10.23919/ChiCC.2017.8029130.

[5] "An Introduction to Convolutional Neural Networks". In: *Keiron O'Shea, Ryan Nash* (2015). DOI: 10.48550/arXiv.1511.08458.

[6] Ankita Pramanik Puja Bharati. *Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey.* Advances in Intelligent Systems and Computing. Springer Singapore, 2019. ISBN: 978-981-13-9041-8.

[7] "Deep Residual Learning for Image Recognition". In: *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun* (2015). DOI: abs/1512.03385.

[8] "YOLOv3: An Incremental Improvement". In: *Joseph Redmon, Ali Farhadi* (2018). DOI: 10.48550/arXiv.1804.02767.

[9] "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications". In: *Hong-Mei Sun,Qi-Chao Mao,Yan-Bo Liu,Rui-Sheng Jia* (2019). DOI: 10.1109/ACCESS.2019.2941547.

[10] *Python*. URL: https://www.python.org/ (visited on 10/21/2022).

[11] *Tensorflow*. URL: https://www.tensorflow.org/ (visited on 10/21/2022).

[12] *Pytorch*. URL: https://pytorch.org/ (visited on 10/21/2022).

[13] *Google Colab*. URL: https://colab.research.google.com/ (visited on 10/21/2022).

[14] *ASP.NET Core*. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0 (visited on 10/21/2022).

[15] *Spring Boot*. URL: https://spring.io/projects/spring-boot (visited on 10/21/2022).

[16] *Vue*. URL: https://vuejs.org/ (visited on 10/21/2022).

[17] *React*. URL: https://reactjs.org/ (visited on 10/21/2022).

[18] *Angular*. URL: https://https://angular.io/ (visited on 10/21/2022).

[19] "Artificial Intelligence-Based Thyroid Nodule Classification Using Information from Spatial and Frequency Domains". In: *Dat Tien Nguyen, Tuyen Danh Pham, Ganbayar Batchuluun, Hyo Sik Yoon, Kang Ryoung Park* (2019). DOI: 10.3390/jcm8111976.

[20] "An open access thyroid ultrasound-image Database". In: *Lina Pedraza,Carlos Vargas, Fabián Narváez, Oscar Durán, Emma Muñoz, Eduardo Romero* (2015). DOI: 10.1117/12.2073532.

[21] "Automated location of thyroid nodules in ultrasound images with improved YOLOV3 network". In: *Ling Zhang, Yan Zhuang, Zhan Hua, Lin Han, Cheng Li, Ke Chen, Yulan Peng, Jiangli Lin* (2021). DOI: 10.3233/XST-200775.

[22] "Detection of Thyroid Nodules with Ultrasound Images Based on Deep Learning". In: *Xia Yu, Hongjie Wang, Liyong Ma* (2020). DOI: 10.2174/1573405615666191023104751.

[23] "Nodule Localization in Thyroid Ultrasound Images with a Joint-Training Convolutional Neural Network". In: *Ruoyun Liu,Shichong Zhou,Yi Guo, Yuanyuan Wang, Cai Chang* (2020). DOI: 10.1007/s10278-020-00366-6.

[24] "The Bethesda System for Reporting Thyroid Cytopathology: A Cytohistological Study". In: *Bakiarathana Anand,Anita Ramdas, Marie Moses Ambroise, Nirmal P. Kumar* (2020). DOI: 10.1155/2020/8095378.

[25] "Thyroid Nodule Ultrasound Image Classification Through Hybrid Feature Cropping Network". In: *RUONING SONG, LONG ZHANG ,CHUANG ZHU, JUN LIU, JIE YANG, TONG ZHANG* (2020). DOI: 10.1109/ACCESS.2020.2982767.

[26] *BBAug*. URL: https://github.com/harpalsahota/bbaug (visited on 04/21/2023).

[27] *FAST API*. URL: https://fastapi.tiangolo.com/ (visited on 04/21/2023).