
SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÓGÉPES IRÁNYÍTÁSI RENDSZEREK SZAK

**Objektumok észlelése légifelvételeken
mélytanuló algoritmusok
alkalmazásával**

MESTERI DISSZERTÁCIÓ

Témavezető:
Dr. Brassai Sándor Tihamér
egyetemi előadótanár

Végzős hallgató:
Hammas Attila

2023

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA SISTEME DE CONTROL INTELIGENTE**

**Detectarea obiectelor în imagini
aeriene folosind rețele neuronale
convoluționale**

LUCRARE DE DISERTAȚIE

**Coordonator științific:
Conf. dr. ing. Brassai Sándor Tihamér**

**Absolvent:
Hammas Attila**

2023

LUCRARE DE DIPLOMĂ

Coordonator științific:

Conf. dr. ing. Brassai Sándor Tihomér

Candidat: ing. Hammas Attila

Anul absolvirii: 2023

a) Tema lucrării de licență:

Detectarea obiectelor în imagini aeriene folosind rețele neuronale convoluționale

b) Problemele principale tratate:

- Studiu bibliografic privind detectarea obiectelor folosind rețele neuronale convoluționale
- Adaptarea algoritmilor de detectare a obiectelor pentru a se utiliza în prelucrarea imaginilor aeriene

c) Desene obligatorii:

- Arhitectura rețelelor neuronale convoluționale relevante utilizate pentru detectarea obiectelor
- Arhitectura sistemului
- Schema bloc a procesului de antrenare și testare a rețelelor neuronale convoluționale
- Figuri și grafice pentru compararea și analiza rezultatelor obținute prin diferite metode

d) Softuri obligatorii:

- Softuri pentru preprocesarea și prelucrarea datelor
- Softuri pentru analiza rezultatelor

e) Bibliografia recomandată:

1. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, „Ssd: Single shot multibox detector,” in Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, 2016.
2. Z. Tian, C. Shen, H. Chen and T. Ho, „Fcos: Fully convolutional one-stage object detection,” in Proceedings of the IEEE/CVF international conference on computer vision, 2019.
3. Zafrafi, J. Deng, R. Kumar, M. S. Khokhar, J. Chaves J. Kumar, „Multiscale and direction target detecting in remote sensing images via modified YOLO-v4,” IEEE Journal of Selected Topics in Appl. Earth Observations and Remote Sensing, 34(1), k30615, pp. 1039-1048, 2022

f) Termene obligatorii de consultații: săptămânal

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umansitate din Târgu Mureș

Primit tema la data de: 10.05.2023

Termen de predare: 27.06.2023

Semnătura Director Departament

Semnătura coordonatorului

Semnătura responsabilului
programului de studii

Semnătura candidatului

Detectarea obiectelor în imagini aeriene folosind rețele neuronale convoluționale

Extras

În zilele noastre, inteligența artificială se bucură de multă atenție, deoarece poate fi folosită într-o gamă foarte largă de aplicații, cum ar fi: procesarea imaginilor, imagistica, sisteme de suport pentru luarea deciziilor și multe altele. Rețelele neuronale convoluționale sunt un subset în domeniul inteligenței artificiale și sunt foarte potrivite pentru a rezolva probleme de procesare a imaginilor, cum ar fi detectarea obiectelor. În această teză, acești algoritmi de detectare a obiectelor vor fi folosite pe date colectate de sistemele UAV. Aceste sisteme UAV sau UAS sunt utilizate cu preferință, deoarece o astfel de dronă poate zbura și colecta date în câteva minute, acoperind o zonă mare. Algoritmii pentru detectare a obiectelor pot fi de mare ajutor la procesarea probelor colectate de drone, deoarece pot facilita evaluarea zonelor eșantionate. Vom analiza performanța mai multor algoritmi pe bază de rețele neurale convoluționale folosite pentru detectarea obiectelor în fotografiile aeriene. Vor fi examinate două abordări: utilizarea seturilor de date generale deja existente ca set de instruire și un set de antrenament bazat pe mostre care conțin mașini agricole sintetice compilate de noi. Cum reiese și din lucrarea de disertație ambele abordări au avantajele și limitele lor. Teza va examina 4 arhitecturi diferite de rețele neuronale convoluționale: Faster R-CNN, YOLO, SSD și FCOS. Toate aceste arhitecturi au avantajele și dezavantajele lor, unele necesită o capacitate de calcul mic, iar altele pot obține o precizie mai mare însă necesită capacitate de calcul mult mai mare.

Cuvinte cheie: Framework pentru implementarea rețelele neuronale artificiale, HLS, MLP, RBF, FPGA

Objektumok észlelése légifelvételeken mélytanuló algoritmusok alkalmazásával

Kivonat

A mesterséges intelligencia napjainkban nagy figyelmet élvez, hiszen nagyon széles körben lehet alkalmazni, mint például: képfeldolgozás, képalkotás, döntéshozatali támogató rendszerek és egyéb rendszerek kialakításában. A mély konvolúciós neuronhálók a mesterséges intelligencia egy kis alrészét képezik és ezek nagyon alkalmasak többek között képfeldolgozási feladatok elvégzésére, mint például objektumok detektálása. Ebben a dolgozatban UAV rendszerekkel gyűjtött képeken és videókon vannak alkalmazva az objektum észlelő algoritmusok. Ezek az UAV vagy UAS rendszerek előszeretettel vannak alkalmazva, hiszen egy ilyen drón nagy területet képes percek alatt berepülni és minta vételezni. A drónok által gyűjtött minták feldolgozásánál lehetnek nagy segítségre az objektum észlelő mélytanuló algoritmusok, hiszen megkönnyíthetik a vételezett területek felmérését. A dolgozatban különböző mély konvolúciós hálók lesznek alkalmazva légifelvételeken való objektum észlelésre. Két megközelítés lesz vizsgálva: már meglevő általános adathalmazok alkalmazása, mint tanítóhalmaz, illetve általunk összeállított szintetikus mezőgazdasági gépeket tartalmazó mintákon alapuló tanítóhalmaz. Amint a dolgozatban is kiderül mindkettőnek megvannak a maga előnyei és korlátjai. A dolgozatban 4 különböző konvolúciós neuronháló architektúra lesz vizsgálva: Faster R-CNN, YOLO, SSD és FCOS. Ezeknek az architektúráknak mind megvan az előnye és hátránya, van amelyik kis számítási kapacitást igényel, illetve van olyan, amely sebesség árán nagyobb pontosságot képes elérni.

Kulcsszavak: mesterséges intelligencia, mélytanulás, detektor, légifelvétel, UAV

Object detection using deep convolutional neural networks on aerial photography

Abstract

Nowadays, artificial intelligence enjoys a lot of attention, as it can be used in a very wide range of ways, such as: image processing, decision-making support systems, and many others. Deep convolutional neural networks are a small subset of artificial intelligence, and they are very suitable for image processing tasks such as object detection. In this thesis, we will try to use these object detection algorithms on data acquired by UAV systems. These UAS or UAV systems are widely used in the industry, since such a drone can fly in and sample a large area in minutes and for a fraction of the costs. Object detection deep learning algorithms can be of great help when processing the samples collected by drones, as they can facilitate the assessment of the sampled areas. In the thesis, different deep convolutional networks will be used for object detection on aerial photography. Two approaches will be examined: the use of already existing general data sets as a training set, and a training set based on samples containing synthetically generated agricultural machines. As it turns out in the thesis, both have their advantages and limitations. The thesis will examine 4 different convolutional neural network architectures: Faster R-CNN, YOLO, SSD and FCOS. All of these architectures have their advantages and disadvantages, some require a small computational resources, and some can achieve higher accuracy at the cost of speed.

Keywords: artificial intelligence, deep learning, object detection, UAV

Tartalomjegyzék

1 Bevezető.....	10
2 Célkitűzések.....	13
3 Rendszer specifikációk és architektúra.....	14
3.1 Felhasználói követelmények.....	14
3.2 Rendszer követelmények.....	14
3.2.1 Funkcionális követelmények.....	14
3.2.2 Nem funkcionális követelmények.....	14
4 Elméleti áttekintés.....	14
4.1 Kép osztályozási feladat.....	15
4.2 Objektum detektálási feladat.....	16
4.2.1 Objektum detektáló algoritmusok kiértékelésére.....	16
4.2.2 Kétlépcsős detektorok.....	17
4.2.3 Egylépcsős detektorok.....	19
4.2.3.1 YOLO.....	20
4.2.3.2 SSD.....	23
4.2.3.3 Horgonyok alkalmazásának előnyei és hátrányai.....	24
4.3 Szintetikus adatalapú tanítás.....	25
5 A megvalósítás során használt technológiák.....	26
5.1 Virtuálizáció.....	26
5.2 Címkező platform.....	26
5.3 Keretrendszerek.....	29
6 Metodológia.....	30
6.1 Szatellit alapú adathalmaz, mint kiindulási pont.....	30
6.2 Szintetikus minták alkalmazása az algoritmusok tanítására.....	34
7 Kísérletek és eredmények.....	37
7.1 Szatellit alapú adathalmaz, mint kiindulási pont.....	37
7.2 Szintetikus minták alkalmazása az algoritmusok tanítására.....	42
8 Összefoglalás.....	49
9 Bibliográfia.....	51

Ábrák jegyzéke

4.1. ábra: IoU metrika [16].	16
4.2. ábra: R-CNN algoritmus lépései [5].	17
4.3. ábra: Fast R-CNN architektúra [6].	18
4.4. ábra: Faster R-CNN architektúra [21].	19
4.5. ábra: A YOLO algoritmus működési alapelve [7].	20
4.6. ábra: SSD algoritmus működési alapelve [30].	24
5.1. ábra: CVAT platform tömbvázlata tömbvázlata.	28
6.1. ábra: A képek méretének normalizálása	32
6.2. ábra: A DOTA adathalmazon való tanításának és kiértékelés folyamata.	33
6.3. ábra: Betanított neuronhálók tesztelése.	33
6.4. ábra: A szintetikus mintákkal való tanítás folyamata.	35
6.5. ábra: Gyűjtött szintetikus mintákra példa.	36
6.6. ábra: Internetről gyűjtött példák a teszt adathalmazból.	36
6.7. Drónnal készített felvételekre példa a teszt adathalmazból.	37
7.1. ábra: Költségfüggvény változása a tanítás során (YOLOv4) [45].	37
7.2. ábra: Költségfüggvény változása a tanítás során (YOLOv4 Tiny) [45].	37
7.3. ábra: A validációs halmazon mért átlagpontosság fejlődése a tanítás során (YOLOv4) [45].	38
7.4. ábra: A validációs halmazon mért átlagpontosság fejlődése a tanítás során (YOLOv4 Tiny) [45].	38
7.5. ábra: A validációs halmazon mért átlagpontosság fejlődése (osztályonként) a tanítás során (YOLOv4) [45].	39
7.6. ábra: A validációs halmazon mért átlagpontosság fejlődése (osztályonként) a tanítás során (YOLOv4 Tiny) [45].	39
7.7. ábra: Szatellit minta és drón minta közötti különbség.	41
7.8. ábra: A bemenet mérete és az átlagpontosság.	41
7.9. ábra: Szintetikus mintákon való tesztelés [45].	42
7.10. ábra: Egy objektum különböző alakjai.	43
7.11. ábra: Faster R-CNN mérőszámai a tanítás során.	43
7.12. ábra: YOLOv3 mérőszámai a tanítás során.	44
7.13. ábra: SSD (VGG16 gerinccel) mérőszámai a tanítás során.	44

7.14. ábra: SSD (MobilNetv2 gerinccel) mérőszámai a tanítás során.	45
7.15. ábra: FCOS mérőszámai a tanítás során.	45
7.16. ábra: A mérőszámok összehasonlítása a validációs halmazon.	46
7.17. ábra: Példa a teszhalmazból részlegesen takart objektumokra.	46
7.18. ábra: Példa volumetrikus „zajra” a teszhalmazban.	47
7.19. ábra: Példa téves osztályozásra a teszt halmazon.	47
7.20. ábra: Példa korrodált objektumra a teszt halmazban.	48

Táblázatok jegyzéke

I. táblázat: YOLO architektúrák összefoglalása [16].	23
II. Táblázat: Mások eredményei a DOTA adathalmazon YOLO architektúrával [45].	32
III. táblázat: A „mAP@.50” az átlagpontosság, ahol a detektálási konfidencia nagyobb, mint 0.5. A „recall” a visszaidézőképesség [49].	34
IV. táblázat: Az általunk tanított YOLO algoritmus eredménye a különböző alhalmazokon [45].	40
V. táblázat Az általunk tanított YOLO algoritmus különböző osztályokon [45].	40
VI. táblázat: A „mAP@.50” az átlagpontosság, ahol a detektálási konfidencia nagyobb, mint 0.5. A „recall” a visszaidézőképesség.	48

1 Bevezető

A történelem során az emberiségben mindig meg volt az a törekvés, hogy megörökítse a fontosabb eseményeket, történeteket, illetve ezek tanulságait, gondoljunk akár a különböző művészettörténeti korszakokban keletkezett kezdetleges barlangrajzokra, épületeket díszítő freskókra, vagy különböző alapanyagokkal és hordozókra készült festményekre. Az 1800-as évek elején született felfedezéseknek köszönhetően lehetővé vált, hogy fényérzékeny anyagokon megörökítsünk tárgyakat, személyeket, pillanatokat. Ekkortól beszélhetünk analóg fotográfiáról. Az 1970-es években megjelent digitális alapú szenzorok lassan felváltották a film alapú képrögzítő eszközöket, a számítógépek, illetve az internet hálózat megjelenésének és elterjedésének köszönhetően pedig egyre több képi információt váltunk képessé összegyűjteni, tárolni és megosztani egymással. A technológia fejlődésével a feldolgozó egységek architektúrájának korszerűsödése is rohamossá vált, az eszközrendszer pedig specializálódott (megjelentek az általános célú mikrovezérlők, processzorok, majd a feladatspecifikus gyorsítók: GPU¹, TPU²), ami tovább növelte a képfeldolgozás hatékonyságát. Az infokommunikációs rendszerek tekintetében a fejlődés elsősorban az egyre nagyobb sávszélesség formájában jelentkezett, egyre távolabbra, illetve egyre több eszköz között voltunk (vagyunk) képesek jó minőségű, megbízható adatátvitelre. Ez lehetővé teszi számunkra például, hogy méréseket végezzünk veszélyes területeken szenzorokkal felszerelt drónok segítségével vagy hálózatba kapcsolt szenzorok segítségével IoT³ [1]. A különböző valós idejű szenzorhálózatokra épülő azon rendszerek, melyek feladata többek között, hogy megkönnyítsék a döntéshozatalt, különböző csatornákon keresztül kommunikálhatnak, ugyanakkor az egyes eszközöknek akkor is képesnek kell lenniük reagálni a környezeti változásokra/hatásokra és ellátni feladatukat, ha az adatátviteli út megszűnik a rendszer, illetve felhasználó felé. Ilyenek például önvezető autók, drónok, műholdak, vagy éppen az égitestek felderítésére készített robotok. Ezek a helyzetek komoly veszélyt jelenthetnek missziójukra, vagy akár környezetükre is abban az esetben, ha nem képesek megfelelően reagálni a környezetükből érkező befolyásoló tényezőkre.

A pilóta nélküli légi jármű (UAV⁴ vagy UAS⁵) olyan repülő rendszer, amelynek fedélzetén nem tartózkodik személyzet, mint például a távirányítású repülőgép vagy drón, de repülhet egy előre programozott repülési terv alapján, vagy akár önállóan is bonyolultabb dinamikus, automatizált

¹ GPU – Graphics Processing Unit

² TPU – Tensor Processing Unit

³ IoT – Internet of Things

⁴ UAV – Unmanned Aerial Vehicle

⁵ UAS – Unmanned Aerial System

algoritmusok alapján. Az UAV-kat leggyakrabban katonai tevékenységekkel lehetett asszociálni, azonban napjainkban előszeretettel vannak alkalmazva kutatásra, mentésre, megfigyelésre, forgalomfigyelésre, időjárás jelenségek vizsgálatára és tűzoltásra is, illetve egyre szélesebb körben terjed a szórakoztató ipari, illetve hobbi célú alkalmazásuk is. Az iparban is előszeretettel, vannak alkalmazva, hiszen egy ilyen drón nagy területet képes percek alatt berepülni és minta vételezni, valamint az általuk gyűjtött minták sokkal nagyobb felbontással rendelkeznek, mint mondjuk a szatellitokkal gyűjtött minták, ezért nagyon is alkalmasak lehetnek agrár területek felmérésére, infrastruktúra hálózatokban (magasfeszültségű vezetékek vagy gáz- és olaj csövek) keletkezett károk keresésére és felmérésére.

A mesterséges intelligencia a számítástechnika, és számítástudomány rohamosan fejlődő területe, amely olyan intelligens gépek, algoritmusok kifejlesztésére törekszik, amik képesek akár olyan feladatokban helyt állni ahol jelenleg emberi beavatkozásra vagy intuíciónak van szükség. A kutatók, fejlesztők olyan algoritmusok és modellek létrehozásán dolgoznak, amelyek a rendelkezésre álló információból képesek tanulni, felismerni a benne rejlő mintázatokat és azok alapján érvelni és döntéseket hozni, valamint kooperálni az emberekkel vagy a környezetükkel. Az mesterséges intelligencia kutatások egyik fő célja olyan gépek megalkotása, amelyek az emberekhez hasonló módon, összetett feladatokat ellátni, mint akár képfeldolgozás, a döntéshozatal és a problémamegoldás, viszont gyorsabban, pontosabban és költséghatékonyabban.

A mesterséges intelligencia kutatás egyik legizgalmasabb és leggyorsabban fejlődő területe a neurális hálózatok és a mély tanuló algoritmusok. A neurális hálózatok egymással összekapcsolt csomópontokból vagy neuronokból állnak, amelyek hierarchikus módon dolgozzák fel az információkat. Rétegről-rétegre a modellek egyre absztraktabb mintázatokat képesek felismerni a bementben, ezáltal egyre komplexebb problémákban nyújthatnak megoldást. Ez a hierarchikus feldolgozás lehetővé teszi a neurális hálózatok számára, hogy az emberek tanulásához hasonló módon fejlődjenek a tanítási folyamat során a rendelkezésre álló adathalmaz alapján [1].

A mélytanuló algoritmusok, olyan neurális hálók, amelyek a be és kimeneti réteg mellett rengeteg rejtett réteggel rendelkeznek. A mélytanulás lényeges előrehaladást hozott az AI⁶-kutatás területén, és áttörésekhez vezetett a számítógépes látás, a szöveg feldolgozás és a beszéd felismerés terén [1]. A mély tanulási modellek kínálják a legpontosabb megoldást

⁶ AI – Artificial Intelligence

számos kihívást jelentő feladatban. A technológiai fejlődés mára a mesterséges intelligencia (AI) reneszánsz időszakához vezetett, köszönhetően annak, hogy az elmúlt évtizedben különböző architektúrák és módszerek jelentek meg számos probléma megoldására, mint például az adatok (idősorok, képek, stb.) osztályozása, klaszterezése [2, 3, 4]; fontos jellemzők, szabályosságok keresése, detektálása nagy adathalmazokban vagy képeken [5, 6, 7]. Ugyanakkor manapság az AI már nem csak az adatfeldolgozással teszi könnyebbé életünket, de új tartalmakat is képes létrehozni úgy szöveg [8], mint kép [9, 10], vagy éppen emberi hang, zene, illetve videók formájában. Ezeknek a neurális hálózatoknak egy nagyon felkapott felhasználásimódja a különböző objektumok észlelése a képekben, ami alapján a rendszerek képesek megkönnyíteni a felhasználók dolgát: például, ahogy a mai digitális fényképezőgépek és okostelefonok javítják a fotós élményét azáltal, hogy detektálják és osztályozzák a fotózandó embereket, állatokat vagy objektumokat (szem, arc, ember, állat, mozgó tárgy) és az alapján válasszák meg a megfelelő beállításokat vagy a képet javító utófeldolgozó algoritmusokat. Fejlettebb és futurisztikus alkalmazás az autókban alkalmazott segédfunkciók, mint az önálló parkolás, a sávtartás, vész fékezés vagy akár teljesen autonóm vezetés (korlátozott körülmények között): Tesla béta fázisban levő Full Self Driving (FSD) nevű mélytanuló algoritmus komplexum, amely nagyon összetett kereszteződéseket és városi forgalmat képes kezelni a régió részletes feltérképezése nélkül. A mélytanuló algoritmusok egyik fő előnye, hogy képesek nagy mennyiségű adatból tanulni, ez különösen hasznos olyan alkalmazásokban, ahol sok adat áll rendelkezésre, például kép- vagy beszédfelismerés.

A rohamos fejlődés ellenére a mélytanulás kihívásokat is tartogat a kutatók és fejlesztők számára. Az egyre mélyebb és komplexebb neuronhálók nagyon sok számítási kapacitást igényelnek. A másik nagy probléma, a nagymennyiségű adatnak a beszerzése, ami kritikus lehet az alkalmazás szempontjából. Révén, hogy a tanítás során alkalmazott adathalmaz nagymértékben befolyásolja végső modell minőségét. Annak érdekében, hogy túlilleszkedés (amikor a modell túlságosan ráilleszkedik a tanító mintákra, és gyengén teljesít a még nem látott mintákon) vagy az alul tanítás (amikor a háló nem látott elég mintát, annak érdekében, hogy a legjobban teljesítsen). További kihívásokat jelent a megbízhatóság, avagy az értelmezhetőség hiánya, amikor nehezen vagy egyáltalán nem átlátható, hogy hogyan jutott el a modell az adott eredményhez. A neurális hálózatok és a mély tanulás az AI-kutatás izgalmas területei, amelyek számos iparágat alakítottak át és átalakíthatnak át a jövőben [11, 12]

A mesterséges intelligencia az elmúlt években kiváló lehetőséget kínál az UAV a rendszerek autonómia szintjének növelésére, valamint a gyűjtött információk feldolgozására, ezzel is megkönnyítve az operátorok, illetve az értékelő elemző munkát végzők feladatát.

A mai kereskedelmi drónok is rendelkeznek néhány fantasztikus funkcióval, amelyek háttérben mélytanuló algoritmusok vannak alkalmazva, mint például a DJI „smart track”, amely akkor is képes követni a kiválasztott mozgó célt, ha az eltűnik, és néhány képkockával később újra megjelenik a felvételen.

Azonban az egyre növekvő komplexitású AI modellek fejlesztéséhez, hangolásához egyre több erőforrás és nagy mennyiségű változatos tanítóminta szükséges. A nagy adathalmazok összeállítása az egy időigényes és költséges feladatnak bizonyulhat. Az UAV-k segítségével végzett AI alapú hatékony felderítés felmérés alapvető problémája, hogy a tanítandó mélytanuló algoritmusok számára rengeteg változatos (különböző szögek, fényviszonyok) mintára van szükség. Rendelkezésre álló észlelendő objektumok, eszközök esetén a szükséges felvételek elkészítése nem okoz különösebb problémát, bár idő és erőforrásigényes, így költséges feladat. De az igazi nehézséget az olyan eszközök fotódokumentációjának összeállítása jelenti, amelyekhez nincsen hozzáférésünk. A jelenlevő dolgozatban főként a rendelkezésre álló adat problémájára fogunk fókuszálni.

2 Célkitűzések

A dolgozat tárgyát képező kutatás célja megvizsgálni, hogy milyen nehézségekkel jár a légifelvételeken való objektumok észlelése és annak határoló kereteinek a behatárolása, milyen objektum detektáló mélytanuló algoritmusok és tanító mintákat tartalmazó adathalmazok állnak rendelkezésre a feladat elvégzéséhez, hogyan lehet a rendelkezésre álló fényképes adatbázist kiegészíteni augmentációval vagy akár mesterségesen előállított mintákkal. Annak a kutatásnak, amelynek keretében saját vizsgálataimat is végzem, egyik fontos célkitűzése különböző objektumok észlelése, valamint azoknak az elhelyezkedésének a meghatározása egy drón platformra telepített mesterséges intelligencia segítségével. Révén, hogy a mélytanuló algoritmusok számára nagy mennyiségű (akár több ezer), változatos tanító minta szükséges, felmerül a kérdés, hogyan lehetséges a felismerni tervezett eszközökről, lehetőleg minél több szögből mintát gyűjteni, a lehető legkisebb energiaráfordítással. Ugyanakkor a kutatás során figyelembe venni, hogy a különböző modellek pontossága és érzékenysége mellett milyen számítási költségekkel jár azoknak az alkalmazása. Révén, hogy drón platformról van szó figyelembe kell venni a különböző rendszerek üzem idejét is

3 Rendszer specifikációk és architektúra

3.1 Felhasználói követelmények

A kialakítandó rendszer fő feladata a drón segítségével vételezett, 3 csatornán (RGB⁷) rögzített képek vagy videófelvételek feldolgozása és elemzése, objektumok észlelése.

3.2 Rendszer követelmények

Révén, hogy a rendszert hangolás után más alrendszerekkel kell összekapcsolni, illetve más kutató társak kell tudják tesztelni, ellenőrizni és üzemeltetni, ezért a modellek hordozhatók kell legyenek, és amennyire lehet operációs rendszer függetlenek.

3.2.1 Funkcionális követelmények

A modellek futtatásához szükséges egy nagyobb számítási kapacitással rendelkező rendszer alkalmazása a neuronháló modellek futtatásához. Az alkalmazott keretrendszerek operációs rendszer függetlenek viszont a könnyeb hordozhatóság érdekében Docker virtualizációs technológia van alkalmazva, így annak a telepítésére lesz szükség. A Docker receptek megkönnyítik a rendszer üzembe helyezését.

Ha a felhasználó új konfigurációt szeretne létrehozni vagy a meglevőket szeretné módosítani, akkor szükség lesz egy szövegszerkesztőre (mint például: Visual Studio Code, PyCharm, stb.), amely lehetőleg tud csatlakozni a Docker konténerekhez a könnyedebb működtetéshez.

3.2.2 Nem funkcionális követelmények

Az alkalmazott keretrendszereknek van CPU optimalizált változatuk is viszont a nagyobb modellek több erőforrást igényelnek így javasolt egy nagyobb teljesítményű GPU (lehetőleg CUDA technológiát támogató Nvidia kártya) alkalmazása, amely szignifikánsan képes felgyorsítani modellek válasz idejét.

4 Elméleti áttekintés

A neurális hálózatok története egészen az 1940-es évekig nyúlik vissza, azonban a szélesebb körű alkalmazásukról komplexebb feladatokban, mint a számítógépes látás vagy a beszédfelismerés, csak az 1980-as és 1990-es években beszélünk. Abban az időben ugyanakkor a neurális hálózatok hangolása lassú folyamat volt, hiszen nem állt rendelkezésre a napjainkban elérhető számítási kapacitás, illetve utolsó generációs gyorsítók, így alkalmazhatóságuk korlátozott volt. A 2000-es években a nagy teljesítményű gyorsítók (grafikus és tenzor feldolgozóegységek) megjelenésével lehetővé vált olyan korábbi tanító algoritmusok

⁷ RGB – Red Green Blue

eredményes alkalmazása, mint például a „backpropagation”, ugyanakkor kifejlesztésre kerülhettek újabb és nagyobb architektúrák is, melyek ismét felkeltették a kutatók neurális hálózatok iránti érdeklődését. Az ezt követő években rohamos fejlődésről beszélhetünk, az egyre nagyobb méretű konvolúciós háló architektúrák a mélytanuló algoritmusok kialakulásához vezetett, ami a mesterséges neurális hálózatok egy olyan alterülete, amely nagyon sok rejtettréteget alkalmaz az adatok összetett kapcsolatainak modellezésére. A mélytanuló algoritmusok azóta forradalmasította az ipar számos területét, beleértve a kép- és beszédfelismerést, a természetes nyelvi feldolgozást és az autonóm vezetést.

4.1 Kép osztályozási feladat

A képosztályozás klasszikus képfeldolgozási feladat, amely egy teljes kép feldolgozását foglalja magába. A cél a kép osztályozása úgy, hogy egy adott címkéhez rendeli. Ezt a fajta feladatot leggyakrabban konvolúciós neuronhálókkal (CNN⁸) oldják meg, amelynek a bemenete maga a kép vagy annak a leskálázott változata, valamint a kimenete a különböző osztályokhoz rendelt konfidenciaszint [13]. A képosztályozó modellek mondhatni 2 részből épülnek fel: egy előfeldolgozás célját szolgáló konvolúciós rétegek sorozata, valamint egy teljesen összecsatolt osztályozó rétegek sorozata. A konvolúciós rétegek hangolásával a neuronháló megtanulja kinyerni a legfontosabb jellemzőket, amelyek alapján majd az osztályozó rész könnyedén el tudja dönteni, hogy a bemenet milyen osztályba kell sorolni.

Ennek a feladatnak a megoldására egyik megoldás mélyebbnél-mélyebb konvolúciós hálók alkalmazása, mint például a VGG [14] architektúrák voltak. Azonmban a mélyebb architektúrák egyre rosszabb konvergáláshoz vezettek a tanítás során. Erre egy jó megoldásnak bizonyult a „reziduális” blokkok alkalmazása, ami a ResNet [15] architektúrához vezetett. A „reziduális” blokkok egy fajta áthidalásként működnek a rétegek között, ami a gradiensek optimálisabb visszatérjesztésében segít, ami gyorsabb konvergáláshoz vezet. A mobil eszközök terjedésével szükség volt egy kisebb erőforrást igénylő architektúrára, mint a MobilNet [16], amely mélység szerinti konvolúciókat is alkalmaz a hagyományos konvolúciós kernelek mellett. Mindkét konvolúciós művelet szélességben megőrzi vagy csökkenti a bemenetére küldött adatot, viszont a hagyományos konvolúció esetében az adat az mélységben nő (az szerint, hogy hány kernelből tevődik össze a réteg), míg a mélység szerinti konvolúció esetében nem.

⁸ CNN – Convolutional Neural Network

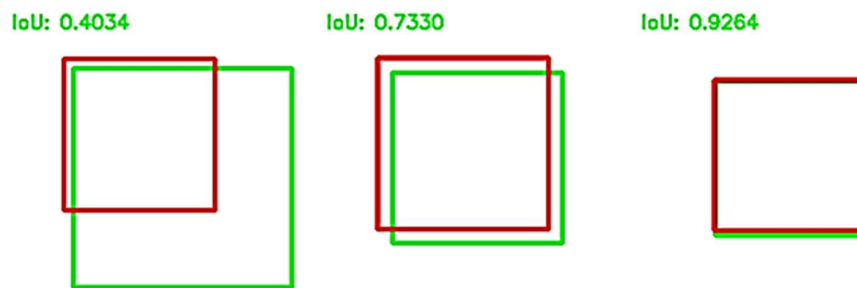
4.2 Objektum detektálási feladat

Az objektumdetektálás manapság alapvető probléma a gépi látásban, amely magában foglalja a képeken vagy videókon lévő objektumok észlelését és lokalizálását. Az objektum észlelő algoritmusok két nagy családba sorolhatók:

- Két lépcsős detektorok, mint a régióalapú konvolúciós neurális hálók (R-CNN⁹), ezek a problémát két lépésben oldják meg: régió javasoló rész és egy osztályozó rész; általában nagyobb pontosságot eredményezve magasabb számítási igénnyel
- Egylépcsős detektorok, mint például a YOLO¹⁰ és az SSD¹¹, ezek erőssége, hogy akár valós időben képesek az objektum észlelést elvégezni

Az objektumészlelési algoritmusok fejlődése számos gyakorlati alkalmazás kifejlesztését tette lehetővé, mint például az autonóm járművek és orvosi döntéshozatalt támogató diagnosztika.

4.2.1 Objektum detektáló algoritmusok kiértékelésére



4.1. ábra: IoU metrika [17].

Az objektumok észlelésének problémája lényegesen nehezebb, mint az egyszerű képosztályozási feladat. A detektorokat leginkább két adathalmazon szokták kiértékelni és összehasonlítani: MS COCO¹² [18], vagy a Pascal VOC¹³ [19]. A kiértékelés során az IoU¹⁴ mérőszámokra alapoznak. A Jaccard index, ismertebb nevén az IoU metrika, a leggyakoribb használt mérőszám két tetszőleges alakzat hasonlósága mértékének megállapítására. Az IoU metrika az összehasonlítandó felületek vagy objektumok paramétereit, mint például szélességét, magasságát és pozícióját, alkalmazza. Ezek alapján majd kiszámít egy normalizált

⁹ R-CNN – Region based Convolutional Neural Networks

¹⁰ YOLO – You Only Look Once

¹¹ SSD – Single Shot Detector

¹² MS COCO – Microsoft Common Objects in Context

¹³ VOC – Visual Object Classes

¹⁴ IoU – Intersection over Union

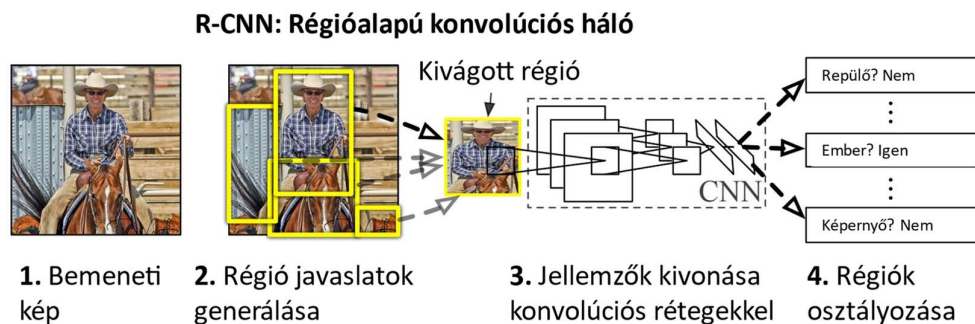
értéket, amely az egymást fedő illetve nem fedő területükre (vagy térfogatukra) összpontosít, amint a 4.1. ábra szemléltet. Az IoU metrikák átlaga fogja majd képezni a detektorunk osztályonkénti átlagpontosságát (AP^{15}), valamint azok átlaga a teljes detektor átlag pontosságát (mAP^{16}). A gyakorlatban a metrikának több variánsát is alkalmazzák a szakirodalomban [20, 21].

4.2.2 Kétlépcsős detektorok

A kétlépcsős objektumdetektorok működés szempontjából két fő szakaszból áll, a régiójavasoló szakasz és az objektum osztályozó szakasz. A régiójavaslat szakaszában az algoritmus generál egy régiókból álló halmazt, ahol potenciálisan egy objektum található. Az objektumosztályozási szakaszban az algoritmus az előző szakaszból kapott minden egyes régiót besorol egy objektum, vagy háttér osztályba.

Amint a 4.2. ábrán látható, az eredeti R-CNN [5] architektúra, amely három fő összetevőből áll: egy régiójavasoló háló (RPN^{17}), egy konvolúciós hálóból (CNN) a jellemzők kinyerésére, és az osztályozást támogató vektorgépekből (SVM^{18}) az objektumok osztályba való besorolására. A régiójavaslat szakaszában az RPN létrehoz egy régiókat tartalmazó halmazt. Az RPN egy keretet (bounding box) is generál minden egyes jelölt helyhez.

Az objektumok osztályozási szakaszában a CNN minden egyes jelölt régiót feldolgoz egymás után, kivonja a jellemzőket, és átadja azokat egy SVM-nek, ami a javasolt régiót besorolja egy osztályba vagy háttérként jelöli meg. Az SVM-ek nagy mennyiségű címkézett képek halmazán vannak tanítva, hogy fel tudják ismerni a különböző típusú objektumokat azok jellemzői alapján.



4.2. ábra: R-CNN algoritmus lépései [5].

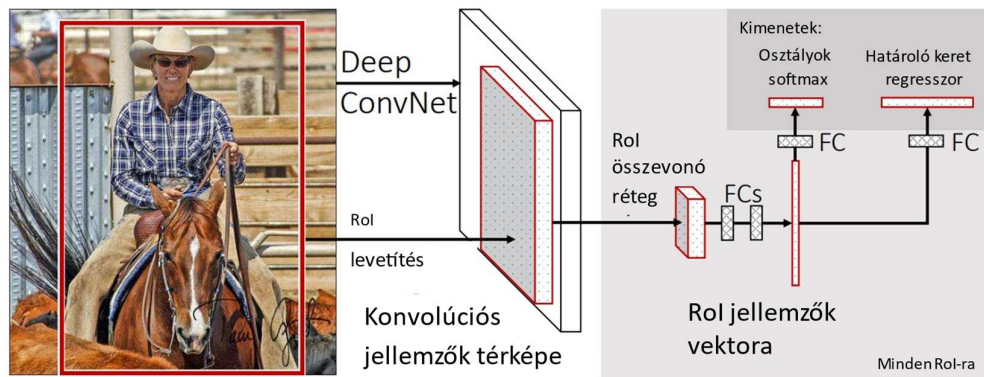
¹⁵ AP – Average Precision

¹⁶ mAP – mean AP (Average Precision)

¹⁷ RPN – Region Proposal Network

¹⁸ SVM – Support Vector Machine

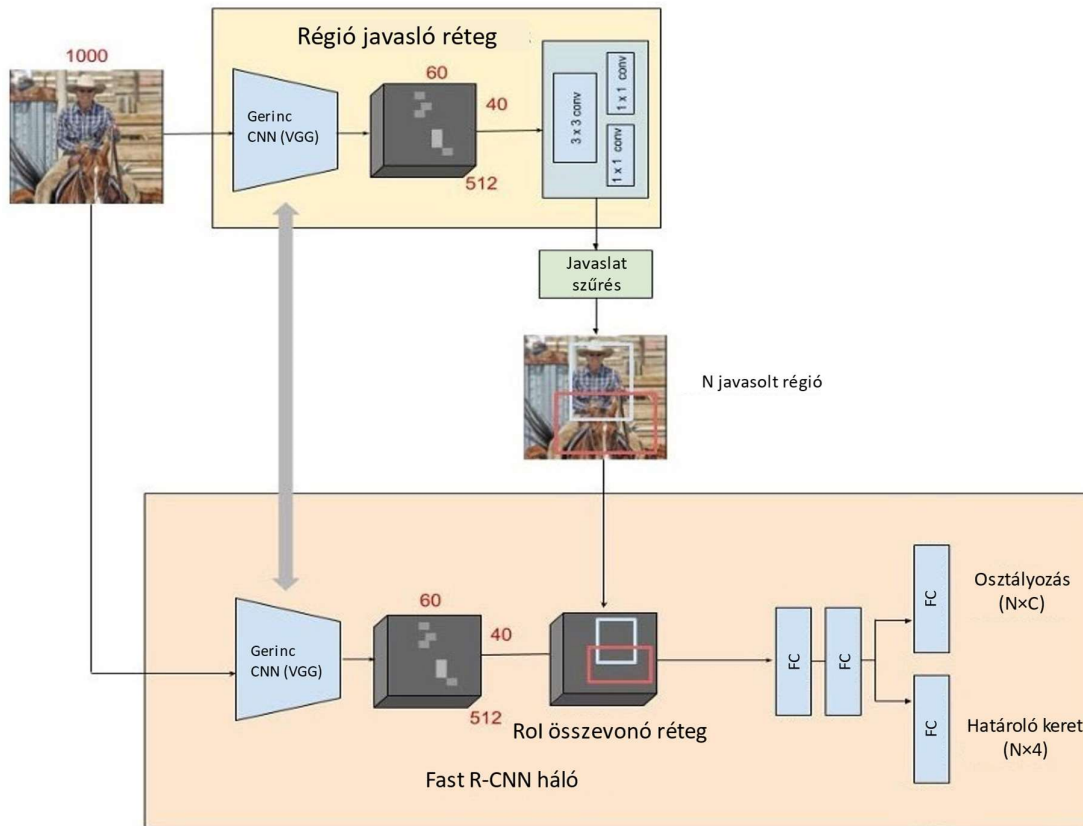
Ross Girshick, az R-CNN modell egyik fejlesztője, utólag javasolt egy javított, Fast R-CNN [6] nevű architektúrát, ami a 4.3. ábrán látható. Az új modellben az egyik fő továbbfejlesztése az, hogy egyszerre hajtja végre az összes javasolt régióban a jellemzők kivonását, ahelyett, hogy az egyes régiókhoz külön-külön kinyerné a jellemzőket, mint az R-CNN-ben. Ez csökkenti a számítási időt. Annak érdekében hogy a különböző régiók jellemző térképeit az osztályozó szakasz fel tudja dolgozni egy RoI¹⁹ összevonó (pooling) réteg használatára van szükség, amely lehetővé teszi a különböző méretű régiók fix méretű jellemzőtérképekké alakítását. Ez biztosítja a hálózat számára, hogy megtanuljon olyan jellemzőket, amelyek nem függenek az objektum méretétől, így robusztusabbá válik az objektum méretarányának változásainak hatásaival szemben. Ezen kívül a Fast R-CNN lecseréli az R-CNN-ben használt SVM osztályozókat teljesen összezsúrt rétegekre (az ábrán FC-ként van feltüntetve), illetve softmax osztályozókra, amelyek jobban teljesítenek osztályozási feladatokon. Összességében a Fast R-CNN által bevezetett javítások gyorsabbá, pontosabbá teszik a modellt, miközben a tanítás folyamatát is egyszerűsítik.



4.3. ábra: Fast R-CNN architektúra [6].

A Faster R-CNN [22] a Fast R-CNN algoritmus továbbfejlesztett változata, amely számos kulcsfontosságú változtatást vezetett be. A struktúra a 4.4 ábrán látható.

¹⁹ RoI – Region of Interest



4.4. ábra: *Faster R-CNN architektúra [22].*

Az egyik fő újítás egy olyan RPN használata, amely megosztja a konvolúciós jellemzőket az objektumészlelési hálózattal, ezáltal lényegesen felgyorsítva a bemeneti minta feldolgozását, révén, hogy a jellemzők csak egyszer vannak kinyerve. Ugyanakkor a Faster R-CNN horgonydobozokat (anchorbox) is alkalmaz, így gyakorlatilag egyszerre több méretű, illetve méretarányú régiókat is fel lehet térképezni egy átfutással, ami felgyorsítja a régió javasoló szakaszt. A detektor a Fast R-CNN mintájára, csak egyszer dolgozza fel a bemeneti képet és nyeri ki a jellemzőteket, és ezeket alkalmazva javasolja a régiót, illetve a RoI összevonó réteg ugyan csak ennek a jellemző térkép alapján a normalizált mintát az osztályozó szakasz számára.

Ezeknek a tovább gondolása a Mask R-CNN [23], ami nem egyszerű határoló kereteket generál, hanem az objektumokat pontosabban körvonalazó maszkokat.

4.2.3 Egylépcsős detektorok

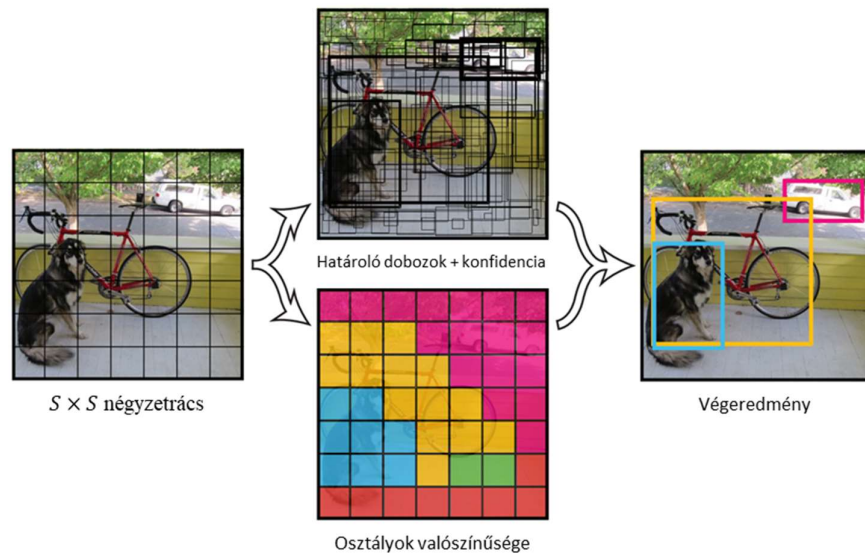
A kétlépcsős detektorok kialakításuk miatt számításigényesek és lassúak lehetnek. Ez gyorsabb egyfokozatú objektumészlelő architektúrák kifejlesztéséhez vezetett. Ilyenek például a YOLO

és az SSD, ezek egyetlen lépésben oldják meg objektum detektálást, anélkül, hogy több különálló alszakaszra lenne szükség. Az SSD és a YOLO gyorsaságukról és hatékonyságukról ismert révén, hogy akár valós időben képesek objektumészlelést végezni. Az egy lépcsős detektorok azonban általában kisebb pontosság jellemző, mint kétlépcsős társaik.

4.2.3.1 YOLO

A YOLO [7, 24, 25, 26] architektúra sorozat az egyik legnépszerűbb egy lépéses objektumdetektáló algoritmus. Felépítés szempontjából a YOLO modell más objektum észlelő architektúrákhoz hasonlóan egy konvolúciós és összevonó rétegekből álló gerincből, ugrásukat és reziduális kapcsolatokat tartalmazó nyakból és egy, illetve a későbbi iterációkban több, osztályozó fejből

A Joseph Redmon és társai által kifejlesztett algoritmus fő ötlete, hogy a bemeneti képre egy négyzetrácsként tekintenek, majd mindegyik cellához becsül több határoló keretet, illetve az adott osztályhoz való tartozását kimutató valószínűségi értéket. Amint az 4.5. ábrán is látható a becsült keretek száma az túlságosan nagy, ezeknek a számát vissza kell szorítani. Az algoritmus nem-maximális elnyomási (NMS²⁰) módszert alkalmaz, ami elnyomja a kis valószínűségekkel rendelkező kereteket a nagy valószínűségűekhez képest [7].



4.5. ábra: A YOLO algoritmus működési alapelve [7].

²⁰ NMS – Non-Maximum Suppression

A YOLO architektúra fő összetevői a következők:

- Az architektúra konvolúciós rétegei a bemeneti mintából kinyerik a jellemzőket. A hálózat több konvolúciós réteget használ, a kernelek méretei rétegenként nőnek, így a jellemzőket tartalmazó tenzorok térbeli mérete rétegről rétegre csökkenek, és mélységre nőnek. A jellemzőket tartalmazó tenzorok további méretbeli csökkenése érdekében összevonó rétegeket használ.
- A teljesen összezsúrolt rétegek végzik a képen az osztályhoz tartozási térkép felépítését és megbecsülik a határoló keretek koordinátáit.
- Miután a modell megjósolta a határolókeret-koordinátákat és az azokhoz tartozó objektumosztály-valószínűségeket, a rendszer egy NMS modul segítségével visszaszorítja a kisebb konfidenciaszinttel rendelkező észleléseket.

Redmon és társai szerint a YOLO algoritmus nagyon jó általánosítási tulajdonságokkal rendelkezik [7]. Ezt azzal igazolták, hogy valós képeken betanított hálót más stílusú képekkel rendelkező mintákon is tesztelték és azokon is jó eredményeket értek el, ugyanakkor az algoritmus azonban számos problémával rendelkezett [7]:

- Az algoritmus gyengén teljesít olyan mintákon, ahol kis méretű objektumok sűrűn vannak elhelyezve, hiszen gyakorlatilag cellánként csak két határoló keretet és egy osztályhoz tartozási valószínűséget rendel.
- Révén, hogy a háló a cellákban kinyert jellemzők alapján tanulja megbecsülni a kereteket, gyengén teljesít, amikor egy szokatlan méretarányú objektumokról beszélünk. Ugyanakkor mivel a jellemző térkép több összevonó rétegen is átmennek amíg a háló ezekből egy határoló dobozt becsülne, a háló egy durva textúrájú jellemző térkép alapján javasolja a határoló dobozt, ami sokszor hibához vezet.
- A veszteség függvény egyformán kezeli az eltérést a kisebb- és nagyobb méretű objektumok esetén, holott az abszolút értelemben vett kis eltérés sokkal nagyobb relatív hibát eredményez egy kisméretű objektum esetében.

Az említett hiányosságokat a következő iterációkban (YOLO9000 [24], YOLOv3 [25], YOLOv4 [26]) próbálták javítani. A YOLOv2 (a szerzők YOLO9000 néven mutatták be) ugyanakkor már jelentős javulást eredményezett eredeti modell korlátjainak lebontása során [24]:

- A YOLOv2 architektúrába elkezdtek alkalmazni a horgonydobozokat annak érdekében, hogy a modell jobban kezelje a különböző méretű és képarányú objektumokat. Ez javítja az objektumok lokalizációjának pontosságát.
- Nagyobb felbontású jellemző térképeket vezettek be. Az eredeti architektúrában 13×13 méretű jellemző térképek alapján becsülték az objektumok pozícióját és osztályát, ami elégséges a nagyobb objektumok detektálásához, viszont gyengén teljesített a kisebb objektumok esetén. A pontosság növelése érdekében egy 26×26 méretű jellemző térképet is bevezettek a már meglévő mellé, ezáltal a háló javult a kisebb objektumok detektálásában anélkül, hogy a nagy objektumok észlelése romoljon.
- Tanítás során normalizálták az egyes tételeket ezzel lényegesen növelve a modell stabilitását a tanítás során.
- Révén, hogy a háló csak konvolúciós és összevonó rétegekből épül fel, a háló bemenetének mérete módosítható, azaz a bemeneti réteg átskálázható anélkül, hogy a háló pontossága romoljon. Így a tanítás során időközönként változtatták a bemeneti réteg méretét ezzel is elkerülve a túltanulást.

A YOLO algoritmus második iterációja a javított modellek mellett olyan kísérletekről számolnak be a szerzők, amelyben több adathalmazon együttesen tanították azt, amelynek eredménye egy olyan betanított neuronháló modell volt, ami több mint 9000 osztályba volt képes sorolni az észlelt objektumokat, innen kapta a YOLO9000 nevet is [24].

A YOLOv3 inkrementális javításokat tartalmazott [25]:

- A YOLOv3-ban egy mélyebb darknet53-nak nevezett gerincet vezettek be, ezzel növelve a kinyerhető jellemzők komplexitását és javítva a háló pontosságát.
- A YOLOv3 FPN²¹-t használ a különböző térbeli felbontásban kinyert jellemzők kombinálására, ezzel tovább javítva a modell képességét, hogy nagy, illetve kisebb objektumokat is nagy pontossággal tudjon észlelni.
- A YOLOv3 az FPN három különböző szintjén végez objektumészlelést és add eredményt, lehetővé téve a különböző méretű objektumok nagyobb pontosságú észlelését.
- Révén, hogy a detekció több szinten kerül végrehajtásra, ezeket valamilyen módon egyesíteni kell, hogy ne legyenek ismétlődések. A YOLOv3 továbbfejlesztett NMS-

²¹ FPN – Feature Pyramidal Network

algoritmust használ, amely csökkenti az ismétlődő észleléseket és javítja az objektumok lokalizációjának pontosságát.

- Az általuk alkalmazott DarkNet keretrendszert is bővítették olyan funkciókkal, hogy robusztusabb hálókat lehessen vele tanítani, például különböző tanítási stratégiákkal, vagy adat argumentációs lehetőségekkel.

A YOLOv4 az előző iterációhoz hasonlóan tovább javított a háló különböző moduljain: kevesebb paraméterrel rendelkező viszont pontosabb CSPDarknet53 gerincet alkalmaz, továbbá egy pontosabb térbeli figyelemmechanizmust, és PANet²²-et a több szintes jellemzők egyesítésére, egy újabb MISH-nek nevezett aktivációs függvényt, valamint CIOU²³ alapú metrikát a háló kiértékelésnek a javítására.

Az ezt követő YOLO algoritmus verziók (YOLOv5, YOLOR [27], YOLOX [28], YOLOv6 [29], YOLOv7 [30], YOLOv8, stb.) már nem az eredeti kutatócsoportokhoz kapcsolódnak, így jó részük már PyTorch keretrendszerben került megvalósításra a DarkNet helyett. Elődjeikhez hasonlóan ezek is sok új funkcióval bővítették a YOLO algoritmust, mint például a szegmentáció, szkeleton meghatározás, és persze pontosság növelése [17]. A dolgozatírásnak idején az Ultralytics cég még nem publikálta az általuk fejlesztett YOLOv5 és YOLOv8 architektúra tapasztalatait.

I. táblázat: YOLO architektúrák összefoglalása [17].

Architektúra	Megjelenési év	Horgonydobozok használata	Keretrendszer	Gerinc	mAP
YOLO	2015	Nem	Darknet	Darknet24	63.4
YOLOv2	2016	Igen	Darknet	Darknet24	63.4
YOLOv3	2018	Igen	Darknet	Darknet53	36.2
YOLOv4	2020	Igen	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Igen	Pytorch	Modosított CSP	55.8
YOLOR	2021	Igen	Pytorch	CSPDarknet	55.4
YOLOX	2021	Nem	Pytorch	Modosított CSP	51.2
YOLOv6	2022	Nem	Pytorch	EfficientRep	52.5
YOLOv7	2022	Nem	Pytorch	RepConvN	56.8
YOLOv8	2023	Nem	Pytorch	?	53.9

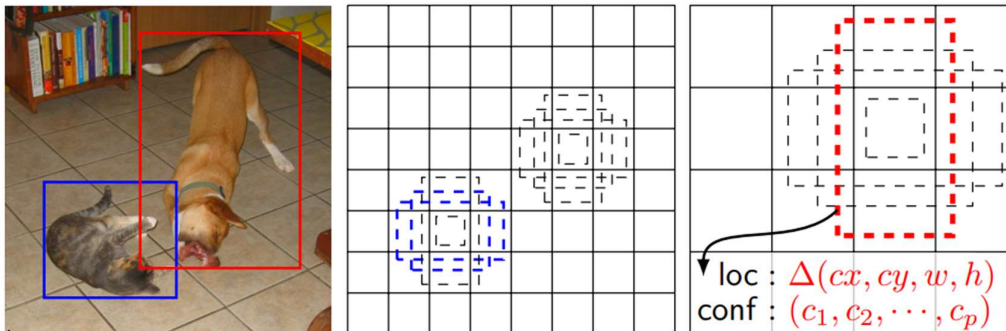
4.2.3.2 SSD

Egy a YOLO-hoz hasonló egy lépcsős objektumészlelő algoritmus az SSD [31]. Ez azonban lényegesen eltér a YOLO-ban alkalmazott megközelítéstől. Az algoritmus generál egy rend alapértelmezett határoló keretet, amely szisztematikusan van elosztva a bemeneten. A modell

²² PANet – Path Aggregation Network

²³ CIOU – Complet Intersection over Union

ezután konvolúciós rétegek sorozatát alkalmazza, hogy kivonja a jellemzőket a képből, és minden egyes alapértelmezett határolókerethez a modell megjósolja az osztályvalószínűségeket, valamint pontosítja a határoló kereteket a horgonydobozokhoz képest. Ez a modell általában pontosabban el tudja helyezni a határoló kereteket a YOLO architektúrához képest, viszont ez nagyobb késéssel jár. Nehézségekbe ütközhet olyan esetekben amikor kisméretű objektumok észlelésére van szükség, mivel az alapértelmezett keretek nem biztos, hogy alkalmasak arra, hogy teljes mértékben rögzítsék az apró objektumok részleteit.



4.6. ábra: SSD algoritmus működési alapelve [31] .

4.2.3.3 Horgonyok alkalmazásának előnyei és hátrányai

Sok objektum észlelő architektúra használ horgonydobozokat a becsült keretek pontosságának a javítására és visszahívó képeség növelése érdekében. Például már a YOLO első verziója nagyon gyorsan volt képes pontos határoló kereteket becsülni, viszont kevésbé teljesített jól a visszaidéző képesség (recall) terén. A YOLO következő iterációi (YOLO9000) alkalmaznak horgony dobookat, ami lényegesen javított a modell visszahívó képességén, anélkül, hogy lényegesen növelné az architektúra számításikapacitás igényét. A horgonyok segítenek csökkenteni a kiértékelendő háttér régiók számát, ami hatékonyabb és gyorsabb észlelési folyamatot eredményez.

Mint minden megközelítésnek ennek is megvannak az előnyei mellett a velejáró hátrányok, amelyek újabb kihívást jelenthetnek az objektum detektáló architektúrák tervezése és hangolása során. Ezek megfelelő számának és méretének kiválasztása gondos mérlegelést igényel az adatkészletben lévő objektumok méretére és képarányára vonatkozólag, hiszen nem megfelelő rögzítődobozok kiválasztása pontatlan észleléshez vagy lassabb tanításhoz és becsléshez vezethet [32]. Ugyanakkor a horgonyok feltételezik, hogy az objektumok rögzített képarányúak, ami nem minden esetben igaz. Például, ha a horgonydoboznak 1:1 arányt válasszuk, előfordulhat, hogy a detektor nem lesz képes észlelni a megnyúlt tárgyakat, mint

például oszlopokat vagy fákat. Természetesen ennek a problémának az áthidalására több horgony dobozt célszerű alkalmazni (a YOLO és az SSD is többet alkalmaz), amelyek különböző méretarányt fednek le, figyelve arra, hogy ne használjunk túl sokat és ezáltal fölöslegesen lassítsuk a detektort.

A horgonydobozokat gyakorlatilag úgy lehet kezelni, mint egyfajta hiperparaméter, hiszen legtöbb esetben ezek nincsenek hangolva a tanítási folyamat során, hanem a tanításiegyüttható, batch méret vagy az epochok számához hasonlóan a tanítás előtt rögzítésre kerülnek és lényegesen kihatnak a detektor működésére [32]. Tian és társai egy olyan egylépcsős objektumészlelési algoritmus javasoltak, amely kizárólag konvolúciós műveleteket használ, azaz nem tartalmaz horgonydobozokat arra, hogy objektumok helyét és osztályát meghatározza, amit közvetlen a bemeneti képről hajt végre. Az FCOS²⁴ [32] egy egylépcsős, horgonydoboz mentes objektumészlelési algoritmus, amely egy teljesen konvolúciós hálót használ az objektumok helyének és osztálypontoszámainak előrejelzésére, amit közvetlenül a képről hajt végre horgonydobozok igénybevétele nélkül. Más detektorokhoz képest, az FCOS kiküszöböli a horgonydobozok alkalmazásából eredő problémákat, és elkerüli a térinformáció elvesztését, amelyet a RoI-alapú módszerek (R-CNN architektúra sorozat) pooling művelete okoz. Lényegében az FCOS azáltal, hogy nem alkalmaz horgony dobozokat egyszerűbb optimalizálni és hangolni, mint detektort, viszont amint majd az összehasonlításból, az észlelés az lassabb mint az egylépcsős társainál.

4.3 Szintetikus adatalapú tanítás

A mélytanuló neurális hálózattok tanítása során általában nagy adatmennyiségre van szükség, azonban gyakorlatban sokszor előfordul, hogy a rendelkezésre álló tanító minták száma, változatossága korlátozott. Az ilyen esetekben a szakirodalomban különböző megközelítéseket alkalmaznak, például a meglévő mintákat zajjal augmentálják vagy a meglévő mintákat szintetikusán előállított adatokkal egészítik ki. Korábbi kutatások eredményei azt sugallják, hogy ha a tanítás során a szintetikus és valós adatok ötvözetét alkalmazzuk a mélytanuló algoritmusok hangolására, akkor robusztusabb modelleket kaphatunk [33, 34, 35]. Viszont, ha a cél az, hogy csak is szintetikusán előállított minták alapján hangoljuk a rendszert, akkor az nehézségekkel járhat [35], Viszont van rá példa a szakirodalomban, hogy csupán szintetikus adatok felhasználásával is lehetséges objektum észlelő modellek létrehozására [36, 37, 38]. Ingeborg Rasmussen és társai Unreal Engine 4-et hoztak létre fotórealisztikus képeket az objektum detektáló modellek tanítására [38]. Ennek a megközelítésnek számos előnye van,

²⁴ FCOS – Fully Convolutional One-Stage object detection

viszont a leg lényegesebb az, hogy egy jól megtervezett algoritmussal korlátlan számú tanító mintát tudnak előállítani illetve akár a gyűjtési folyamat is teljesen automatizálható. A fő kihívás azonban egy olyan kiegyensúlyozott adathalmaz létrehozása, amely áthidalja a valóság és a virtuális világ közötti tartományt, és jól általánosítható, annak érdekében, hogy valós világban is alkalmazható legyen [38]. Rasmussen és társai tanulmányának eredményei azt mutatják, hogy lehetséges egy valós objektumészlelő rendszert szintetikus adatokon betanítani, ugyanakkor a modellek számos más fejlesztési lehetőséget is kínálnak a következtetési eredmények stabilitása és megbízhatósága érdekében.

5 A megvalósítás során használt technológiák

5.1 Virtuálizáció

A Docker környezet egy operációs rendszer-szintű virtualizációt használó platform. Ennek segítségével a szoftvereket, valamint a futásukhoz szükséges függőségekkel (könyvtárak) együtt, konténereknek nevezett csomagokban való kitelepítést teszi lehetővé. A konténereket tároló szoftver neve Docker Engine. Előnye, hogy nincs szükség egy teljes operációs rendszerek és kernelek virtualizálására, mint a virtuális gépek esetén, hanem egy közös kernelt alkalmaznak az alkalmazások, viszont ugyan-úgy el vannak különítve egymástól. A dokumentáció illetve a telepítéshez szükséges segédanyagok elérhetők a Docker hivatalos oldalán: <https://www.docker.com>.

Általában automatizálásra (CI/CD²⁵) illetve alkalmazások platform független kitelepítésére alkalmazzák, az alkalmazások külön-külön is hatékonyan képesek működni különböző környezetekben. Sok szolgáltatás előre össze van csomagolva a számára szükséges függőségekkel és elérhető a DockerHub²⁶ oldalon, amely összetett programok könnyed kitelepítését tesz lehetővé.

A Docker Compose segítségével egymással kölcsönható, összetett rendszerek receptjei írhatók le, illetve 2-3 parancssoros utasítással telepíthetők és frissíthetők, lényegesen megkönnyítve például komplex fejlesztői környezetek kialakítását.

5.2 Címkéző platform

A CVAT²⁷ egy nyílt forráskódú interaktív videó- és képannotációs platform, amelyet gépilátás feladatok támogatására fejlesztettek. Felhasználók és cégek tízezrei használják szerte a világon. A platform elérhető, mint egy „freemium” (ingyenes alapsomag, fizetős funkcionálisok)

²⁵ CI/CD – Continuous Integration / Continuous Deployment

²⁶ DockerHub: <https://hub.docker.com>

²⁷ CVAT – Computer Vision Annotation Tool

szolgáltatásként, viszont az eszközrendszer kitelepíthető magán szerverekre is docker virtualizáció segítségével.

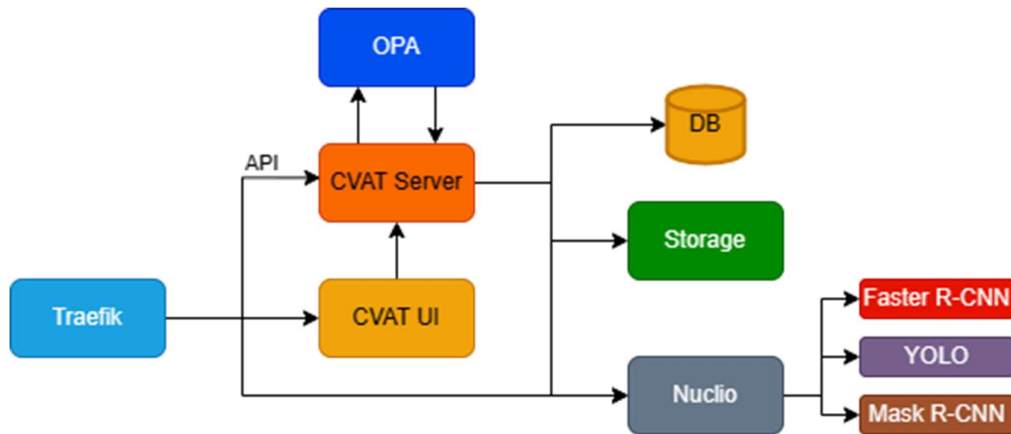
Mindaddig míg nem szeretnénk lényeges változtatásokat a rendszerbe a telepítés az nagyon egyszerű. A CVAT GitHub²⁸ oldalán elérhető, és pár kiinduló Docker-es állomány, amelyeket csupán minimális módosításával (URL, verziószám cserélése) a rendszer percek alatt telepíthetővé válik lokálisan. Amennyiben módosítani szeretnénk a jogosultság rendszert (milyen típusú felhasználó milyen műveleteket szabad elvégezzen), vagy új funkcionalitásokat szeretnénk hozzáadni (például egy saját címkézési formátum), akkor ezeket is könnyedén megtehetjük, hiszen az alrendszerek (Datumaro – címkéző formátumok menedzselésére, OpenPolicyAgent – a jogokszabályok menedzselésére, Nuclio – serverless függvények menedzselésére stb.) amiket a platform használ, azok is nyílt forráskódúak és jól dokumentáltak.

A platform segítségével a címkézési folyamat egyáltalán nem bonyolult, a címkézendő állományok egy „project”, „task” és „job” hierarchia rendszerben van szervezve. A munkafolyamatot egy projekt létrehozásával kell kezdeni, ahol definiálni lehet a különböző osztályokat, illetve az ezekhez tartozó attribútumokat. Ezt követően a projekten belül fel tudjuk tölteni a címkézendő képeket egy új „task”-ként. A címkézendő, illetve címkézett képek projekten belül „task”-okba vannak rendszerezve. A képek feltöltését egy „task” létrehozásával tehetjük meg. Feltöltéskor be lehet állítani a fontos paramétereket, mint például, hogy milyen minőségben kerüljenek tárolásra a képek, hány alfeladatra (job) legyen felosztva, melyik alhalmazba tartozik (tanító, validációs, teszt stb.), vagy milyen címkéző formátumot alkalmazzon a címkék tárolására. Fontos figyelembe venni a kiválasztott formátumot és az alkalmazandó címkék típusát. Azt a formátumot kell választani, ami képes eltárolni azt az információt, amit címkézés során beviszünk a rendszerbe (például a YOLO formátum csak vízszintes határoló kereteket képes tárolni így, ha valamit maszkolni akarunk azt nem lesz képes elmenteni), de az alapértelmezett CVAT formátum az minden fajta címkézési lehetőséget képes tárolni, amit a platform kínál.

A címkézés szempontjából sok lehetőség (ovális-, négyszög-, poligon alakú határoló keret stb.) van, viszont a dolgozat célja egy vízszintes négyszög határoló dobozokkal működő objektum detektáló algoritmusok tanítása. Ennek a bevitelére két lehetőség van, egy 2 pontos, illetve egy 4 pontos. Mindkét módszer ugyan azt eredményezi, így egyén és eset függő, hogy mikor melyiket érdemes használni.

²⁸ opencv/cvat – <https://github.com/opencv/cvat>

A CVAT platform sok más kiegészítő eszközt nyújt a felhasználó számára, mint például OpenCV alapú objektum követés, ami hasznos lehet például olyan képsorozatok esetén, ahol az adott objektum több egymás utáni képen is szerepel, viszont ezeken minimálisan változik a pozíciója.



5.1. ábra: CVAT platform tömbvázlata tömbvázlata.

A rendszer több különálló szolgáltatásból tevődik össze, mint ahogyan a 5.1. ábra szemlélteti:

- Traefik revers-proxy: mikró szerviz architektúrára kitervezett revers-proxy, ez a szolgáltatás gondoskodik arról, hogy a kintről érkező kérések a megfelelő szolgáltatásnak vannak továbbítva. Ez a szolgáltatás konfigurációját minimálisan volt változtatva az alapértelmezetthez képest annak érdekében, hogy a csapat számára más szolgáltatásokat is egy címen elérhetők legyenek.
- CVAT UI: ez egy Django alapú webes alkalmazás, amely segítségével egy vizuális felületen keresztül elérhetővé teszi a CVAT platform funkcióit.
- CVAT Server: Ez biztosítja az a címkéző platform funkciók készletét, menedzseli a címkézett és címkézendő mintákat minták. Datumaro könyvtárra építve rengeteg annotáció formátumot képes kezelni, valamint ezek közötti átjárást is biztosítja.
- OPA (Open Policy Agent): ez kezeli a jogosultsági köröket és az ezhez tartozó szabályrendszereket
- Nuclio: ennek segítségével lehet „serverless” függvényeket kitélepíteni, amin keresztül különböző funkciókkal lehet bővíteni a platformot, mint például automata címkézés.

A kutatás megvalósítása során a CVAT 2.1-es verziója volt alkalmazva.

5.3 Keretrendszerek

Manapság sok keretrendszer áll rendelkezésre azok számára, akik a mesterséges intelligencia területén szeretnének kutakodni: TensorFlow, PyThoech, Caffé, Theano, DarkNet, scikit-learn, és még sok más. A kutatás keretén belül 3 keretrendszert volt esélyem kipróbálni: DarkNet, PyThorch és MMDetection.

DarkNet leginkább a YOLO architektúrával asszociálható, hiszen az első 4 YOLO modell ebben a keretrendszerben volt megvalósítva. Ez C/C++ programozási nyelvben van implementálva, valamint egy konfigurációs állomány segítségével lehet a leimplementált modulokat egymás után fűzni, valamint különböző ugrásokat (mint például a reziduális blokkok) bevezetni.

PyTorch, jelenleg, nagy népszerűséget élvez a mélytanulással foglalkozó kutatók körében, mivel a használhatóságra való összpontosítást gondos teljesítménymegfontolásokkal kombinálja. Tervezés során 4 fő alapelve összpontosítottak a fejlesztők [39]:

- „Be Pythonic”: Sok adatelemző és magas szintű algoritmusokkal foglalkozó kutató választott nyelve a Python, ezért tervezés során odafigyeltek a arra, hogy a szintaktika ne távolodjon el a Python programozási nyelv ideológiáitól. A PyTorch-nak az ökoszisztéma első osztályú tagjának kell lennie. Az interfészek egyszerűek és konzisztensek, könnyedén integrálható a Python szabványos ábrázoló, hibakereső és adatfeldolgozó könyvtáraival.
- A kutatók vannak az első helyen: a modellek, adatbetöltők és optimalizálók írását a lehető legegyszerűbbé és hatékonyabbá tegye. A PyTorch arra törekszik, hogy a modellek, adatbetöltők és optimalizálók írását a lehető legegyszerűbbé és hatékonyabbá tegye. A gépi tanulásban rejlő összetettséget a PyTorch-könyvtárnak belül kell kezelnie, és el kell rejtenie az intuitív API-k mögé.
- Teljesítmény: annak érdekében, hogy a gyakorlatban alkalmazható legyen nagy teljesítményt és hatékonyságot kell biztosítson, ugyanakkor ez nem mehet az egyszerűség és a könnyű használat rovására. A fejlesztők a cikkben azzal példáztak, hogy elfogadható a sebesség 10%-áról lemondani egy lényegesen egyszerűbben használható architektúráért; viszont 100%-áról már nem. Ugyanakkor, ha olyan eszközöket biztosítanak a fejlesztőknek, amelyek segítségével manuálisan is paraméterezhetik kódjuk végrehajtását, ezáltal lehetővé válik számukra, hogy saját teljesítményjavításokat találjanak, függetlenül attól, amelyeket a könyvtár automatikusan biztosít.

- Nem a teljesség a fő cél: inkább legyen egy tiszta architektúrája, amelyet könnyedén lehet karban tartani, kibővíteni, és adaptálni a folytonosan fejlődő elvárásokhoz. Feltéve, hogy más külső befolyásoló tényezők nem változnak egy egyszerű keretrendszer, amely könnyedén tesztre szabható több időt és erőforrást spórol meg a kutatók, adatelemzők, mérnökök és fejlesztők számára, mint egy teljesebb, de kevésbé képlékeny rendszer.

A keretrendszer egyszerűségének és bővíthetőségének köszönhetően sok olyan könyvtár van, amely a keretrendszer funkcionalitását hivatott kibővíteni, mint például a torch-vision [40], ami kimondottan a gépíráshoz hasznos funkciókkal bővíti az alap keretrendszer funkcionalitását.

MMDetection [41] egy PyTorch alapjaira építkező keretrendszer, amely számtalan előre megvalósított adatbetöltő, előfeldolgozó modulokat, valamint sok neuronháló modell konfigurációt. Az MMDetection is számtalan kiegészítő modulút is tartalmaz, mint például MMRotation [42], amely olyan modellekre összpontosít, ahol a határoló keret az szögelfordulást is tartalmaz

6 Metodológia

Két különböző megközelítés volt alkalmazva:

- Egy már meglévő szatellites képeket tartalmazó DOTA tanító adathalmazból kiindulva
- egy általunk gyűjtött valós és szintetikus generált mintákból összeállított adathalmaz, amit kimondottan drónfelvételek elemzésére gyűjtöttünk.

6.1 Szatellit alapú adathalmaz, mint kiindulási pont

Felmerült a kérdés, hogy nem lehet-e kiindulni már meglévő adathalmazokból és ezeket adaptálni majd drónos felvételek feldolgozására. Például számos szatellites felvételeket tartalmazó tanító adathalmaz létezik már mélytanuló algoritmusok tanítására: DOTA, VEDAI HRSC2016 stb.

Az első lépés a tanítási mintákat tartalmazó adathalmaz kiválasztása. A DOTA adathalmaz az osztályok sokfélesége (18 osztály) és a minták számossága (több mint 10 ezer kép és 2 millió objektumpéldány) miatt került kiválasztásra. A DOTA adathalmaz több iterációval rendelkezik:

- DOTA-v1.0 [43]: 15 osztályt (repülőgép, hajó, tárolótartály, baseball pálya, tenispálya, kosárlabdapálya, futópálya, kikötő, híd, nagy autó, kis autó, helikopter, körforgalom, focipálya és úszómedence), 2806 képet és majdnem kétszázezer

szegmentált példányt tartalmaz, az adathalmaz javasolt felosztása: 50% tanítóminta, 17% validálásra alkalmazandó minta és 33% kiértékelésre.

- DOTA-v1.5 [44]: ugyanazokat a képeket tartalmazza, mint a DOTA-v1.0, de a példányok is címkézve vannak. Emellett egy új osztály, a „konténerdaru” is bekerült, így összesen több mint négyszázezer szegmentált példányt tartalmaz. A képek és az adatkészlet-felosztások száma megegyezik a DOTA-v1.0-val.
- DOTA-v2.0 [45]: lényegesen ki van bővítve az előző iterációkhoz képest. Több platformról (Google Maps, Google Earth, Baidu, stb.) készített légifelvételek sokaságát gyűjti össze. A DOTA-v2.0-ban 18 kategória (repülőgép, hajó, tárolótartály, baseball pálya, tenispálya, kosárlabdapálya, futópálya, kikötő, híd, nagy autó, kis autó, helikopter, körforgalom, focipálya, úszómedence, konténerdaru, reptér és helipad), több mint tizenegyezer kép és majdnem 1.8 millió szegmentált objektum példányt tartalmaz. A DOTA-v1.5-höz képest tovább bővíti az új „repülőtér” és „helikopter” kategóriákat. A DOTA 11 268 képe képzési, érvényesítési, tesztfejlesztő és teszt-kihívás készletekre van felosztva. A túlillesztés problémájának elkerülése érdekében a betanító és validáló készlet aránya kisebb, mint a teszt halmaz. A tanításra 1830 kép, amelyek 268627 objektum példányt tartalmaznak. Az validációs halmaz 593 képet és 81048 példányt tartalmaz. A teszt halmazok pedig a minták többi részéből állnak össze, viszont azokra nem elérhetők a címkék.

Az adathalmazban található minták alapján a YOLO architektúrát tanítottuk. A DOTA adathalmaz 1.0 és 1.5 verziójával több kiértékelés található a szakirodalomban, amely hasonló képen a YOLO architektúrát alkalmazza, viszont javarészüket a DOTAv1.0 vagy a DOTAv1.5 verziót alkalmazza, illetve sok esetben annak is egy alrészét, mint ahogy a III. táblázatban is látható:

II. Táblázat: Mások eredményei a DOTA adathalmazon YOLO architektúrával [46].

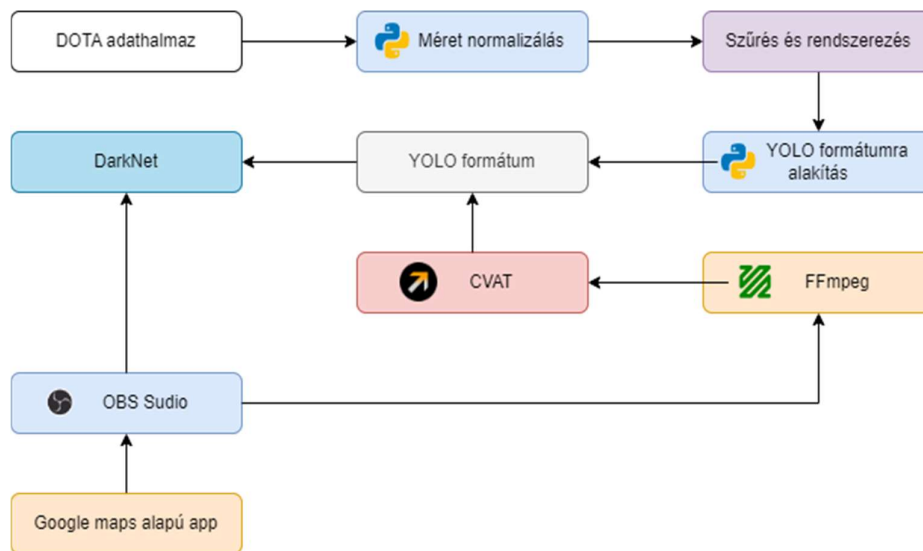
Pub.	Arch.	Adathalmaz	AP						mAP
			Kisautó	Teherautó	Repülő	Hajó	Reptér	Kikötő	
[47]	YOLOv2	DOTA 1.0	?	?	?	?	-	-	76.7%
	YOLOv3 Tiny		?	?	?	?	-	-	73.20%
	YOLOv3		?	?	?	?	-	-	88.31%
	szerzők megközelítése		?	?	?	?	-	-	88.63%
[48]	YOLOv4	DOTA 1.0 (kisautók)	76.7%	-	-	-	-	-	76.7%
[49]	YOLOv3	DOTA 1.0 (csak kis és nagy autók, repülő és hajók)	71.35%	53.82%	82.72%	79.00%	-	-	71.73%
	YOLOv4 Tiny		41.89%	46.89%	79.16%	65.50%	-	-	58.36%
	YOLOv4		72.65%	58.02%	87.27%	82.67%	-	-	75.15%
	YOLOv4-C1		68.30%	53.16%	86.21%	71.37%	-	-	69.83%
	YOLOv4-C2		63.19%	70.39%	85.03%	70.39%	-	-	68.77%
	YOLOv4-C3		70.97%	70.67%	88.57%	76.60%	-	-	74.22%

A DOTA osztályai nagyon változatosak úgy alakra, mint méretre. Ugyanakkor a képméretnek 800×800 és 20K×20K pixel között mozognak, ezért ezek a tanítás és tesztelés során normalizálva vannak. Ez a normalizálási folyamat a képek egyszerű újramintavételezése és egyenlő méretűre való felosztását foglalja magába.



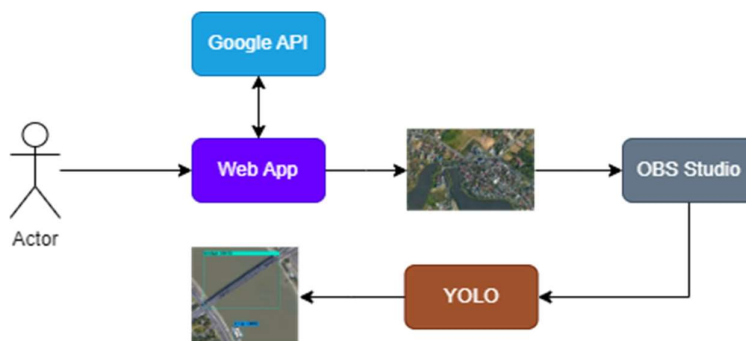
6.1. ábra: A képek méretének normalizálása

A normalizált tanítóminták a YOLOv4 architektúra tanítására vannak alkalmazva. A tanítás során 1024×1024 méretű minták volt alkalmazva, az újra méretezés során a képek méretaránya nem módosult. A YOLO architektúra két variánsa volt alkalmazva: YOLOv4 és YOLOv4 Tiny. A neuronháló konfigurálására a YOLO architektúrát tervező szerzők keretrendszere volt alkalmazva: DarkNet. Az egyik követelmény az volt, hogy a neurális hálózatot futtató környezet telepítése platformfüggetlen legyen. Ehhez a Docker virtualizációs környezet került kialakításra, így a neurális hálózatot tanításra, validálásra és tesztelésre, minimális módosításokkal és teljesítményvesztéssel lehetett működtetni Windows és Linux alapú operációs rendszereken egyaránt.



6.2. ábra: A DOTA adathalmazon való tanításának és kiértékelés folyamata.

A betanított architektúrák tesztelésére és valós időben való alkalmazására egy Google Maps platform alkalmazása tűnt célszerűnek. Azonban, ha a hivatalos Google Maps oldalt alkalmazzuk, akkor nehézkes tiszta mintavételezést készíteni, hiszen a platform leginkább navigáció támogatására volt tervezve, így rengeteg felírat, illetve információt átadó jelölés van a szatellites felvétel fölé kirajzolva. Ezeknek az eltüntetésére a Google Maps API-t felhasználó webes alkalmazás került implementálásra. Az API-hoz való hozzáférést a Google Cloud Console-on²⁹ keresztül lehet beállítani. A megfelelő projekt létrehozás után a Maps Static API és Maps JavaScript API aktiválása szükséges. Projekt konfigurálása a hozzáférési kulcsok generálásával használatra kész. A API kulcsok segítségével hozzáférhetünk az engedélyezett API hívásokhoz. Ezek az API hívások rendelkeznek egy havi kótával, ami alatt a szervízek ingyenesek, viszont a korlátok meghaladásához vezető API hívások kiszámlázásra kerülnek.



6.3. ábra: Betanított neuronhálók tesztelése.

²⁹ Google CCloud Console: <https://console.cloud.google.com>

A kialakított webalkalmazás segítségével tesztminták nyerhetők ki, amelyek alkalmazhatók a betanított modellek tesztelésére. OBS Studio-t alkalmazva vételezhető az alkalmazás által renderelt szatellit képek sorozata, amit V4L230 könyvtárcsomag segítségével a kinyert képkockák beadagolhatóak a neuronháló bemenetére, így a kinyert mintákat akár valós időben lehet feldolgozni. Az alkalmazást segítségével különféle eseteket lehet vizsgálni, mint például a betanított modellek hogyan teljesítenek a szatellit- és a drónfelvételeken.

A tanításra alkalmazott adathalmaz kiegészítésre szorul, főként olyan mintákkal, amelyeket drón segítségével készültek, azaz olyan képkockákra lenne szükség, amelyek az észlelendő objektumokat különböző szögekben láthatóak. Sajnos a DOTA adathalmaz osztályaira nehéz drónfelvételeket gyűjteni a légtér megszorítások miatt, így az általános légifelvételeken való objektumészlelést egyelőre leszűkítjük mezőgazdasági gépekre.

6.2 Szintetikus minták alkalmazása az algoritmusok tanítására

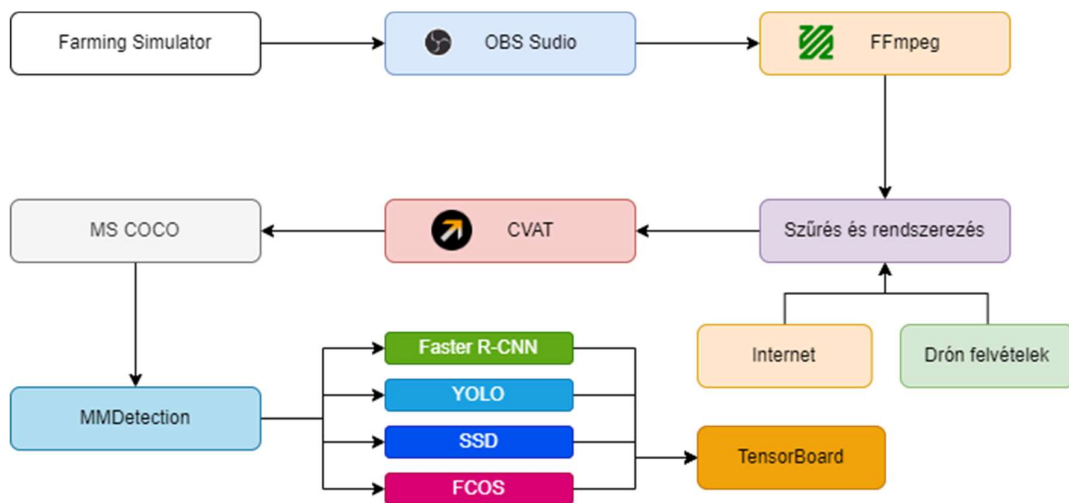
Hasonló módszert már alkalmaztunk más objektum detektálási feladatban. Ott is szintetikus adatok segítségével voltak hangolva különböző a mélytanuló algoritmusok. Amint a III. táblázat mutatja elég jó eredményeket értünk el [50]. A táblázatban fel vannak tüntetve a validációs halmazon (szintetikusan előállított minták) elért eredmény és össze van hasonlítva a teszt halmazon (drónok által gyűjtött valós minták) elért eredményekkel. Az eredményeken észlelhető egy kis pontosságvesztés, viszont az javítható, ha a modelleket tovább hangoljuk valós mintákon is.

III. táblázat: A „mAP@.50” az átlagpontosság, ahol a detektálási konfidencia nagyobb, mint 0.5. A „recall” a visszaidézőképesség [50].

Modell	mAP@.50 (valid.)	mAP@.50 (teszt)	recall	FPS	Késleltetés	CUDA mem.
Faster R-CNN (ResNet50)	0.817	0.657	0.400	30.4	32.9 ms	287 MB
Faster R-CNN (ResNet101)	0.817	0.731	0.440	22.7	44.1 ms	360 MB
YOLOv3 (320×320)	0.772	0.576	0.322	50.8	19.7ms	289 MB
YOLOv3 (608×608)	0.808	0.643	0.402	46.7	21.4 ms	347 MB
SSD (VGG16)	0.648	0.712	0.422	59.3	16.9 ms	138 MB
SSD (MobilNetv2)	0.483	0.534	0.344	51.5	19.4 ms	33 MB
FCOS (ResNet50)	0.782	0.633	0.421	34.1	29.2 ms	194 MB
FCOS (ResNet101)	0.671	0.549	0.386	24.3	41.2 ms	266 MB

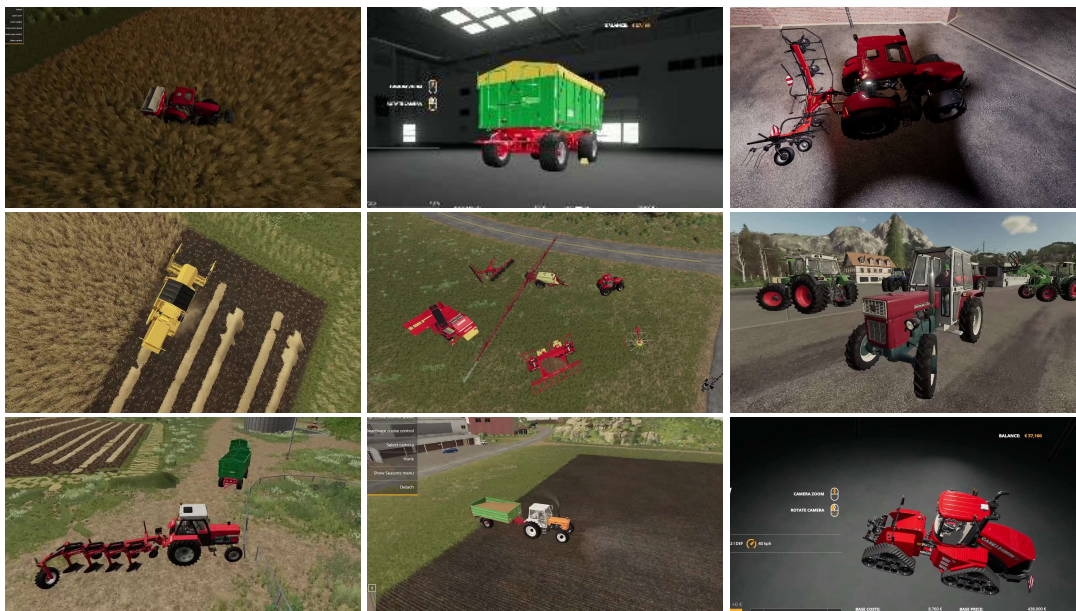
³⁰ V4L2 – Video For Linux v2

A továbbiakban fenti kutatás eredményei alapján szintetikusan előállított minták segítségével megvalósítható kellene legyen, hogy a különböző mezőgazdasági gépek detektálására alkalmas mélytanuló algoritmusokat hangoljunk. A detektálandó osztályok azok hasonlóak más általunk elvégzett meréséhez, ahol a feladat mezőgazdasági gépek észlelése légifelvételken volt [51]. A szintetikus képek előállítására több lehetőség áll rendelkezésre: egy már meglévő 3D szimulátor vagy videójátékból kinyerni őket, vagy egy játékmotor és 3D modellek segítségével összeállítani a jeleneteket és azokból kinyerni a tanítómintákat. Mindkét megközelítésnek megvan a maga előnye és hátránya: a már meglévő szimulátorok és videójátékok azok előre tartalmazzák a szükséges környezeteket, effekteket és 3D modelleket, viszont ezekből kinyerni az adatokat sokszor nehezebb lehet és limitáltak vagyunk a szoftver által nyújtott lehetőségektől [50], míg egy saját tervezésű és kivitelezésű szimulátor esetén a minta gyűjtési folyamat tesztre szabható és akár teljesen automatizálható, viszont ebben az esetben a modellek gyűjtése, effektek megírása az fejlesztőt terheli.



6.4. ábra: A szintetikus mintákkal való tanítás folyamata.

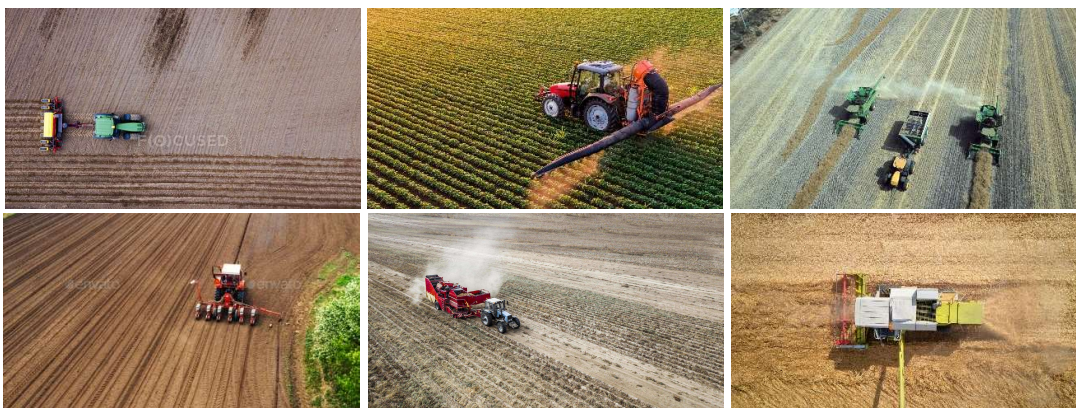
A kutatás során már meglévő 3D videójátékból gyűjtöttük a mintákat. A választás a Farming Simulator számítógépes játékra esett, ez rengeteg mezőgazdasági gépet tartalmazó 3D modellt tartalmaz. A képernyőfelvételek OBS Studio-val készültek, majd ezeket FFMPEG könyvtár segítségével másodpercenként 1-2 képkockával kerültek mintavételezésre. Mindkét eszköz nyílt forráskódú és ingyenesen letölthető, és számos beállítási lehetőséget biztosít a felvételezés minőségének szempontjából.



6.5. ábra: Gyűjtött szintetikus mintákra példa.

A kinyert képkockákat a kitelepített Computer Vision Annotation Tool nevű szoftverplatform segítségével voltak címkézve. Összesen 870 szintetikus minta került címkézésre, amely 6:1 arányban volt felosztva, azaz 745 tanításra és 125 validálásra volt alkalmazva.

A hálók tanítása, illetve a kiértékelése MMDetection keretrendszerrel volt elvégezve, ami egy PyTorch alapú megoldás, amelyben megtalálhatók különböző neuronháló modellek konfigurációja. A kísérletek 4 neuronháló modellen kerültek végrehajtásra, mint a Faster R-CNN, a YOLOv3, az SSD, és a FCOS. Mind a 4 modell esetén az alkalmazott bemeneti méret az 960×720 , illetve 30 cikluson keresztül volt tanítva. Minden második ciklus után a modellek ki voltak értékelve a validációs halmazon. Végül pedig egy valós mintákat tartalmazó adathalmazon volt kiértékelve.



6.6. ábra: Internetről gyűjtött példák a teszt adathalmazból.

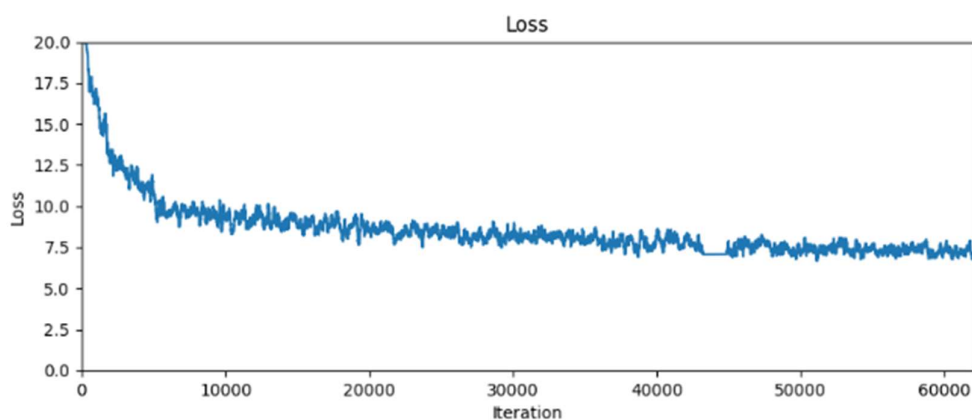


6.7. Drónnal készített felvételekre példa a teszthalmazból.

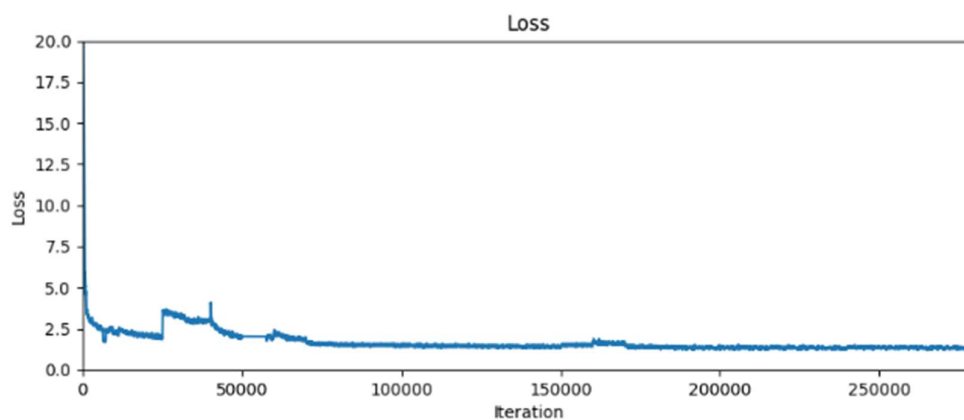
7 Kísérletek és eredmények

7.1 Szatellit alapú adathalmaz, mint kiindulási pont

A neurális hálózatok betanítása több napot vett igénybe. A forró napszakokban túlmelegedési elkerülése érdekében a tanítási folyamatot le kellett állítani. A veszteségfüggvény a neurális hálózat mindkét változatánál csökkenő tendenciát mutatott (7.1. és 7.2. ábra).



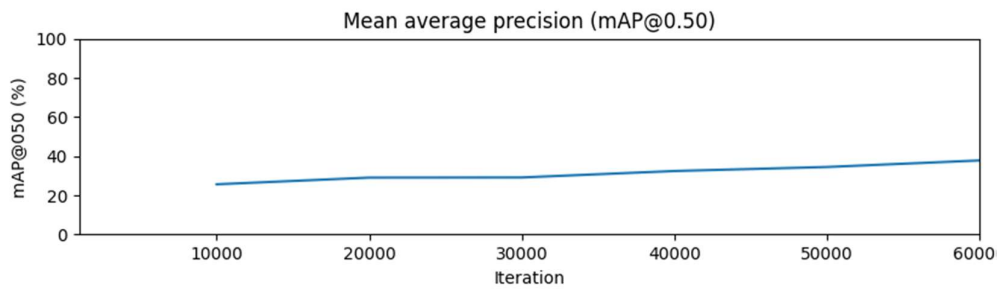
7.1. ábra: Költségfüggvény változása a tanítás során (YOLOv4) [46].



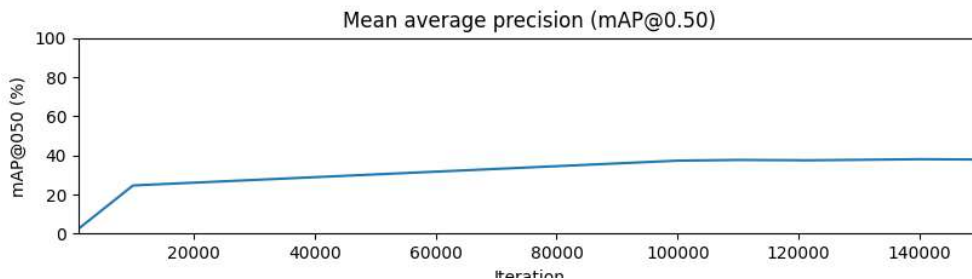
7.2. ábra: Költségfüggvény változása a tanítás során (YOLOv4 Tiny) [46].

Az egy batch feldolgozásához szükséges átlagos idő a YOLOv4 esetében 40 másodperc, a YOLOv4-tiny esetében pedig közel 4 másodperc volt, ez a különbség a neurális hálózat, a YOLOv4 tesztelése során is megfigyelhető volt. átlagosan 4-5 FPS volt, míg a YOLOv4-tiny megközelítette a 30 FPS-t. Mivel a YOLOv4-nek több számítási időre van szüksége, ezt a változatot nem volt idő ugyanannyi iteráción keresztül tanítani, mint a Tiny variánst, amint az az ábrákon látható iterációs számok eltéréséből. Annak ellenére, hogy az alapváltozatot kevesebb iteráción keresztül volt hangolva, az eredmények összehasonlíthatók.

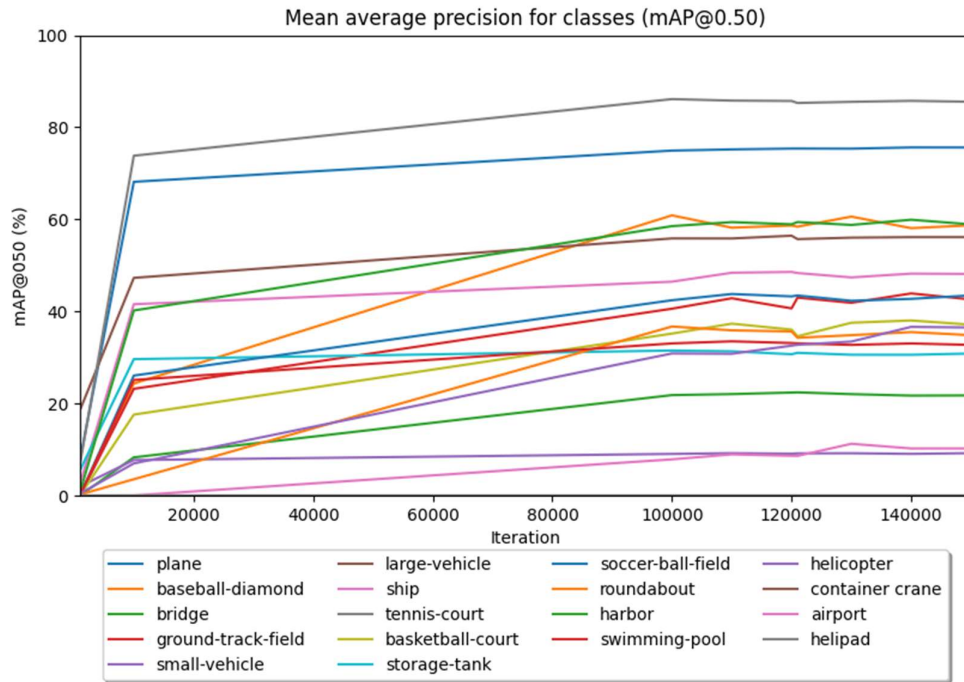
A betanított neurális hálózatokat külön validációs adatkészleten voltak ellenőrizve. Az átlagos pontosság (mAP) felfelé ível, így egyik változaton sem látszik túlillesztés. A két konfiguráció mAP-ját az 7.3. és 7.4. ábra mutatja be.



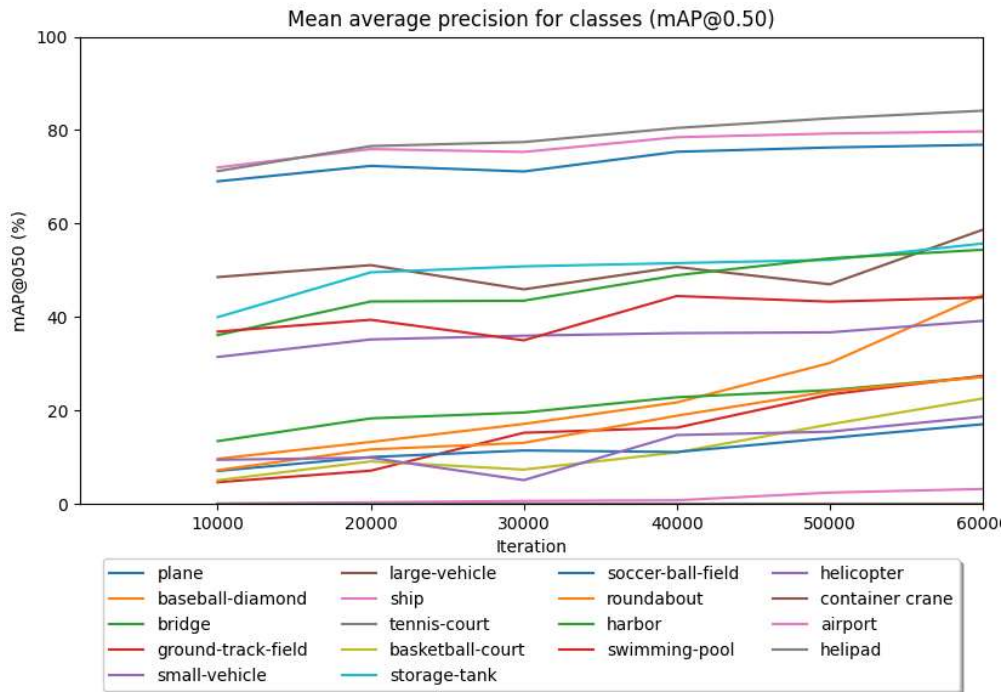
7.3. ábra: A validációs halmazon mért átlagpontosság fejlődése a tanítás során (YOLOv4) [46].



7.4. ábra: A validációs halmazon mért átlagpontosság fejlődése a tanítás során (YOLOv4 Tiny) [46].



7.5. ábra: A validációs halmazon mért átlagpontosság fejlődése (osztályonként) a tanítás során (YOLOv4) [46].



7.6. ábra: A validációs halmazon mért átlagpontosság fejlődése (osztályonként) a tanítás során (YOLOv4 Tiny) [46].

Az mAP értéket osztályonként elemezzük, akkor megállapítható, hogy vannak olyan osztályok, amelyeket az algoritmus könnyen felismer, mint például: repülőgépek, futballpályák, baseball

pályák, stb., és van néhány osztály, amivel az eredmények azok lényegesen rosszabbak, mint pl. : repülőterek, helikopterleszállók, kikötők, konténerdaruk stb. (7.5. és 7.6. ábra).

IV. táblázat: Az általunk tanított YOLO algoritmus eredménye a különböző alhalmazokon [46].

	mAP	
	YOLOv4	YOLOv4 Tiny
DOTA 2.0 (csak kisautók)	38.78%	9.21%
DOTA 1.0 (csak repülő, hajók kis- és nagyautók)	61.44%	47.27%
DOTA 2.0 (DOTT 1.0-ban használt osztályokra szűkítve)	44.21%	45.40%
DOTA 2.0	35.38%	37.91%

Az eredmények összehasonlítása más kutatási eredményekkel nem volt teljesen egyértelmű, mert sok esetben különbség van a tanítási minták között: a legtöbb kutatást továbbra is a DOTA-1.0 és DOTA-1.5 adathalmazokon értékelik, amelyek nem tartalmazzák a olyan osztályok, mint a repülőtér, helikopter-leszállóhely, konténerdaru, amelyeken az általunk használt YOLO algoritmus teljesített a legrosszabbul. Az összehasonlítások összefoglalását az IV. és V. táblázat foglalja össze. Az IV. táblázat tartalmazza az aladathalmazon számított eredményeink áttekintését, amelyeket más közleményekben is használnak. A V. TÁBLÁZAT az egyes osztályok eredményeit mutatja be, a DOTA 2.0-ban vannak olyan osztályok, mint a repülőtér, amelyekkel az algoritmus eléggé gyengén teljesít.

V. táblázat Az általunk tanított YOLO algoritmus különböző osztályokon [46].

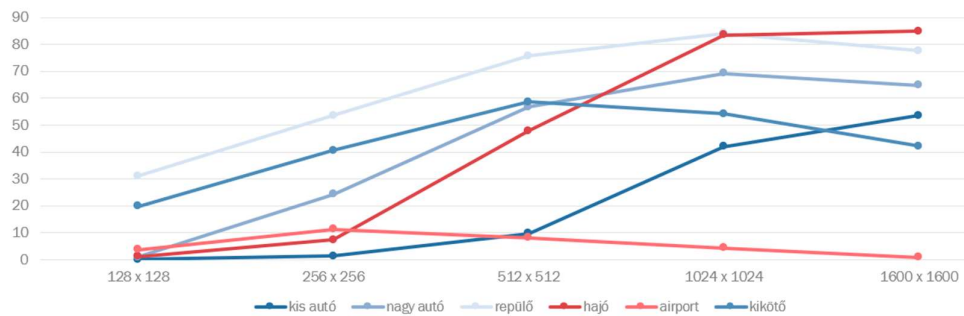
Pub.	Arch.	Adathalmaz	AP					
			Kisautó	Teherautó	Repülő	Hajó	Reptér	Kikötő
[48]	YOLOv4	DOTA 1.0 (csak kisautók)	76.7%	-	-	-	-	-
[49]	YOLOv3	DOTA 1.0 (csak repülő, hajók kis- és nagyautók)	71.35%	53.82%	82.72%	79.00%	-	-
	YOLOv4 Tiny		41.89%	46.89%	79.16%	65.50%	-	-
	YOLOv4		72.65%	58.02%	87.27%	82.67%	-	-
	YOLOv4-C1		68.30%	53.16%	86.21%	71.37%	-	-
	YOLOv4-C2		63.19%	70.39%	85.03%	70.39%	-	-
	YOLOv4-C3		70.97%	70.67%	88.57%	76.60%	-	-
kutatásban elért eredmények	YOLOv4	DOTA 2.0	38.78%	50.85%	76.29%	79.86%	0.33%	50.86%
	YOLOv4 Tiny		59.37%	54.89%	71.19%	81.79%	10.23%	58.91%

A betanított neurális hálózat teszteléséhez a Google Maps platformról beszerzett képeket használtuk. Megvalósítottunk egy minimális alkalmazást, amely egy előre meghatározott útvonalat követ a google maps-en, ezzel műholdképet rögzítettünk a neurális hálózat valós idejű teszteléséhez. Így megvizsgálhattunk néhány különleges esetet, például hogyan kezeli,

amikor a Google Maps műholdképekről drónfelvételekre vált át. Azt vettük észre, hogy révén, hogy tanítóminták szatellit felvételekből ál, gyengébben teljesít a drón felvételekből álló mintákon.



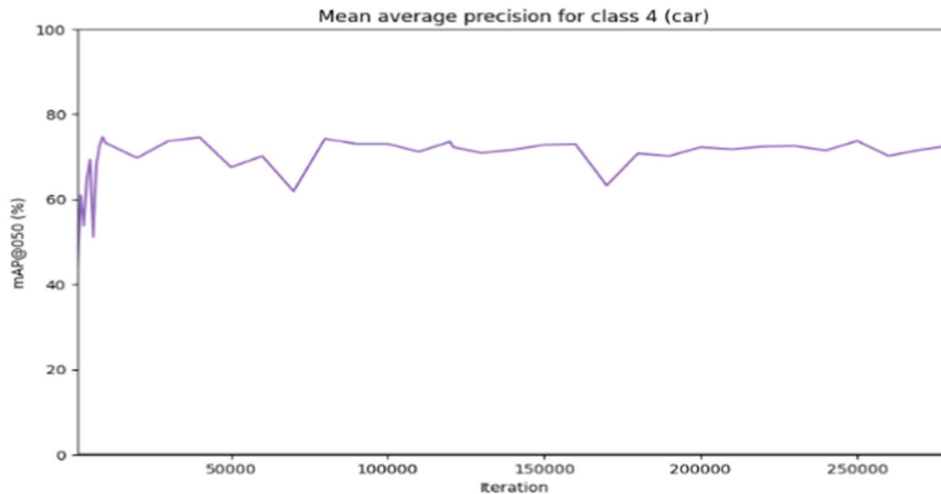
7.7. ábra: Szatellit minta és drón minta közötti különbség.



7.8. ábra: A bemenet mérete és az átlagpontosság.

Ugyanakkor a háló bemenetének a mérete lényegesen befolyásolja a becsült határolókeretek pontosságát (7.8. ábra), például a kis méretű objektumok, mint kis- és nagyautó azt pontosabban lehet megbecsülni a nagyobb felbontást alkalmazva, viszont a változatos és komplex objektumokat, mint a repterek azt kisebb felbontáson könnyebben tudja az algoritmus érzékelni.

Az YOLO szerzői mikor először bemutatták az architektúrát arra a következtetésre jutottak, hogy a modell képes általánosítani és különböző stílusú képeken objektumokat észlelni, annak ellenére, hogy a tanító minták nem tartalmaztak olyan stílust [7]. Ennek a tulajdonságnak a tesztelésére próbára tettük nem szintetikusán előállított mintákon is. A minták a GTA V játékmotorja állította elő, amelyeken hasonló képen felülnézetből voltak objektumok, leginkább kisautók. A szintetikus mintákon elért eredmények nagyon hasonlóak voltak a szatellites mintákon, ez elgondolkodtató, így felmerülhet az a kérdés, hogy lehetséges-e szintetikus minták segítségével betanítani a neuronháló modelleket úgy, hogy valós mintákon lehessen majd velük objektumokat észlelni.



7.9. ábra: Szintetikus mintákon való tesztelés [46].

7.2 Szintetikus minták alkalmazása az algoritmusok tanítására

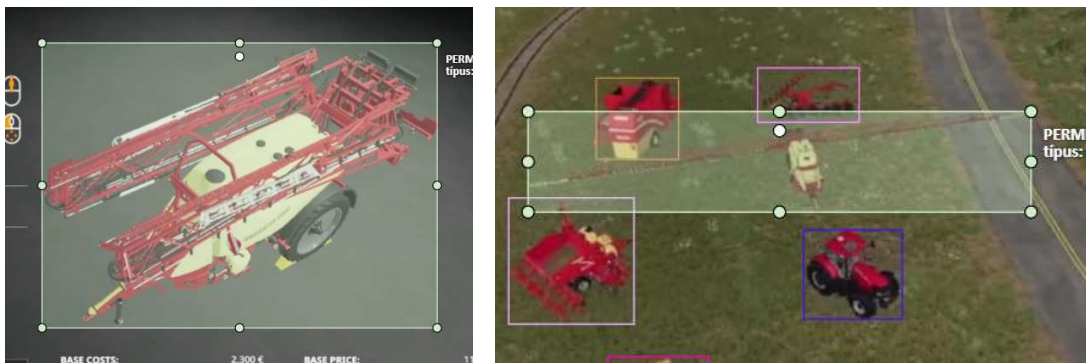
A hálók tanítást, illetve a kiértékelést MMDetection keretrendszerrel volt elvégezve. Négy különböző neuronháló architektúra volt alkalmazva: Faster R-CNN, a YOLOv3, az SSD, és a FCOS. Mindegyik háló tanítása során a minták 960×720 méretre voltak skálázva és 30 tanítási cikluson keresztül voltak tanítva. A tanítás során a rendelkezésre álló erőforrások miatt a batch méret különböző volt:

- Faster R-CNN (ResNet101): 8
- YOLOv3 (DarkNet53): 8
- SSD (VGG16): 16
- SSD (MobilNetv2): 16
- FCOS (ResNet50): 8

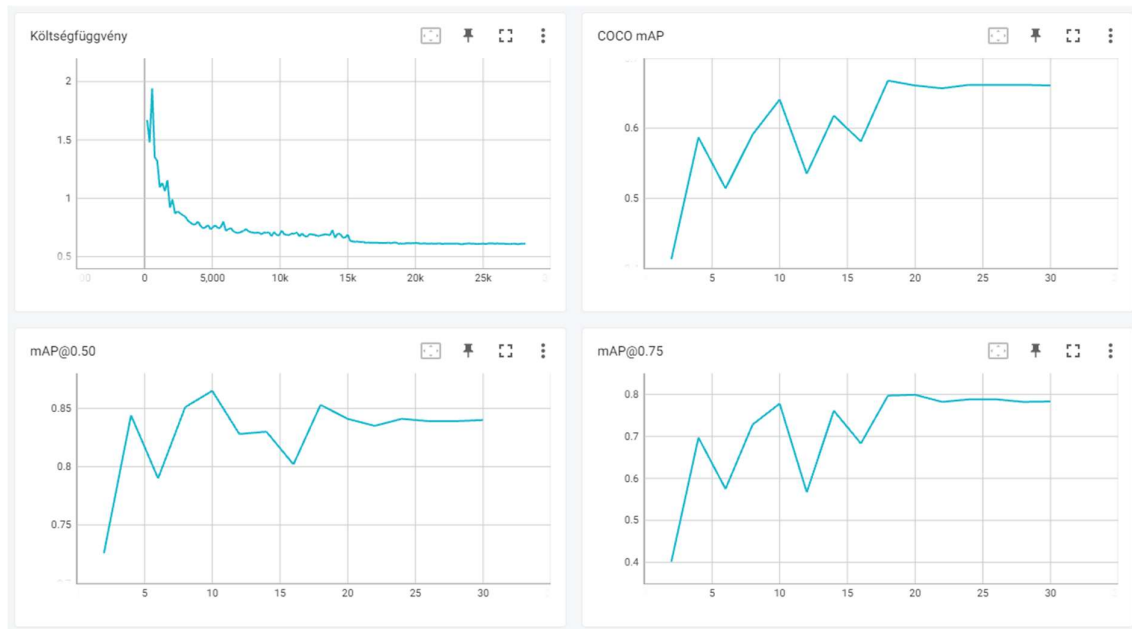
A tanítás során a hibából számolt költségfüggvénye tart a nullához. A költségfüggvényeket ábrázoló grafikonokon a függőleges tengelyen a költség függvény átlagos értéke lesz látható, a vízszintes tengelyen az interakciók száma. A tanítás során az iterációk száma eltérő hiszen a batch méret is különbözött az egyes modellekre. Nagyobb batch méret esetén kevesebb iterációs szükséges ahhoz, hogy ugyanannyi darab mintán keresztül hangoljuk a rendszert. Az átlag pontosságot ábrázol grafikonokon a függőleges tengelyen a pontosság van kirajzolva 0-1 skálán, illetve a vízszintes skálán az epoch-ok száma látható. A pontosság két konfidenciaszinten van kirajzoltatva: nagyobb mint 0,5 illetve nagyobb mint 0,75. Ez annyit jelent, hogy a háló pontosságának a kiszámítása során azok a megjelölt objektumok vannak

figyelembe véve, amelyeket a modell legalább 0,5 illetve 0,75 konfidenciával detektált és határolta be. Ugyanakkor a COCO mAP mérőszám is ábrázolva van.

Az elvárás az volt, hogy a Faster R-CNN teljesítsen a legjobban a validációs halmazon viszont amint a 7.11., 7.15. és a 7.16. ábrán látható az FCOS az pontosságban jobb eredményt ért el mint a Faster R-CNN. Ez annak tudható be, hogy a minták azok változó méretarányokkal rendelkeznek, így azok az architektúrák, amelyek nem alkalmaznak horgonydobozokat egy kis előnyben részesülhetnek. Egy példa, hogy miért jelenthet problémát a horgonydoboz azt a 7.10. ábra szemléltet, ahol egy permetező szállítási és üzem állapotában látható. Az üzem állapotban a permetező szélessége lényegesen nagyobb, mint a magassága így egy nagyon szélsőséges méretarányal rendelkezik.

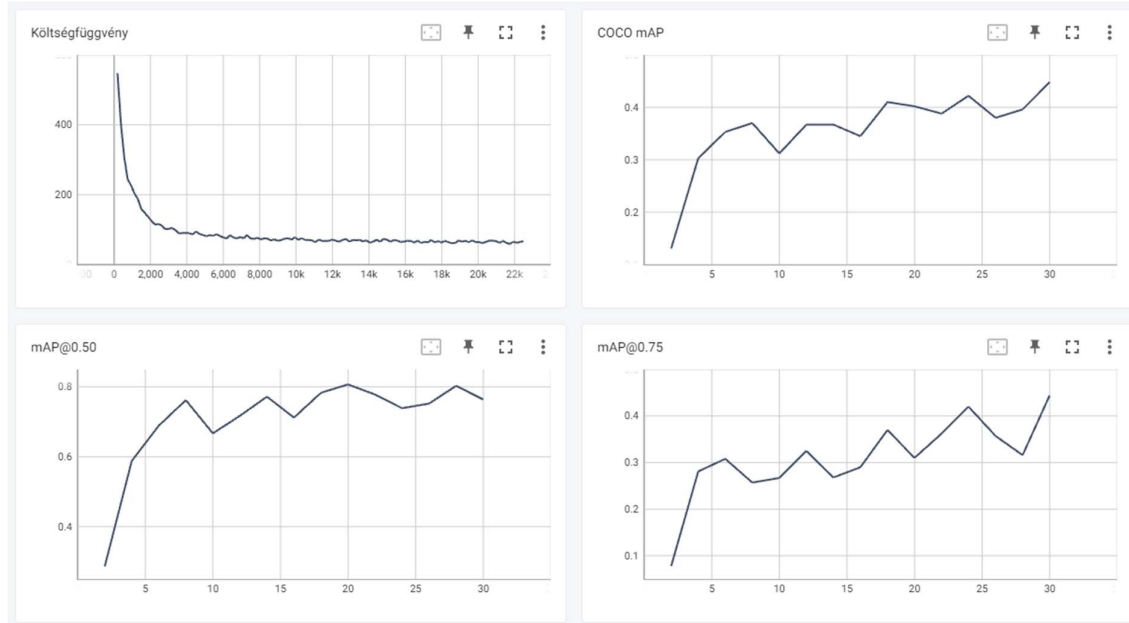


7.10. ábra: Egy objektum különböző alakjai.



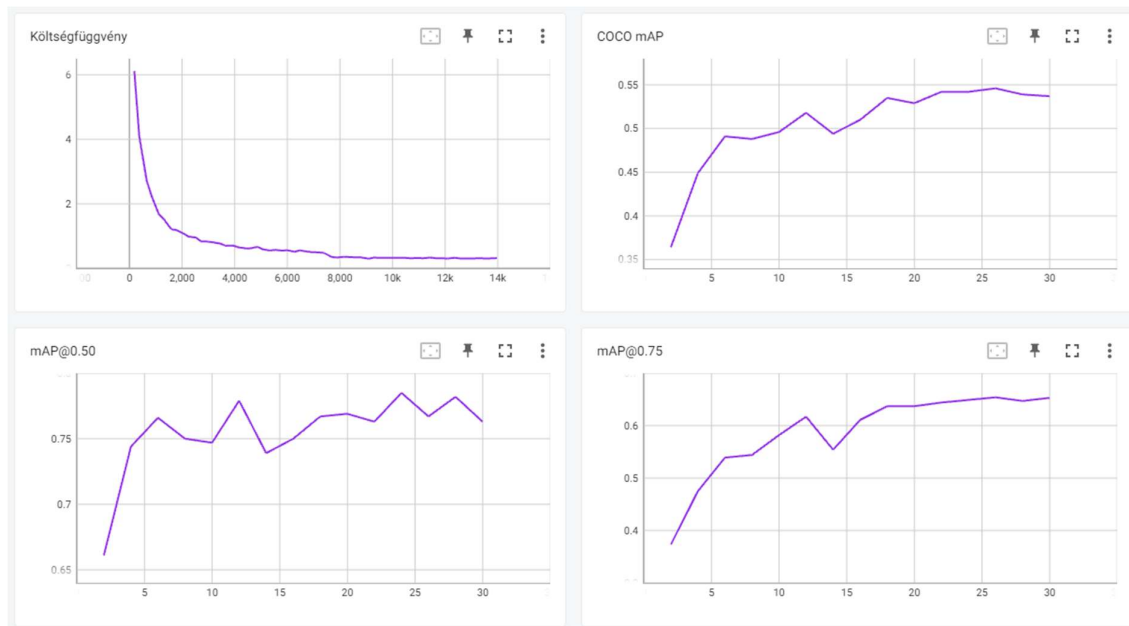
7.11. ábra: Faster R-CNN mérőszámai a tanítás során.

A YOLO architektúra nem teljesített a legjobban. Amint az 7.12. ábra mutatja a $mAP@0.50$ az a többi modellhez hasonlóan teljesített, viszont a $mAP@0.75$ és a COCO mAP mérőszámok alapján lemaradt a többi architektúrától. A COCO mAP azokat a modelleket részesíti előnyben, amely a nagyobb konfidenciaszinten nyújtanak pontosabb megoldást.

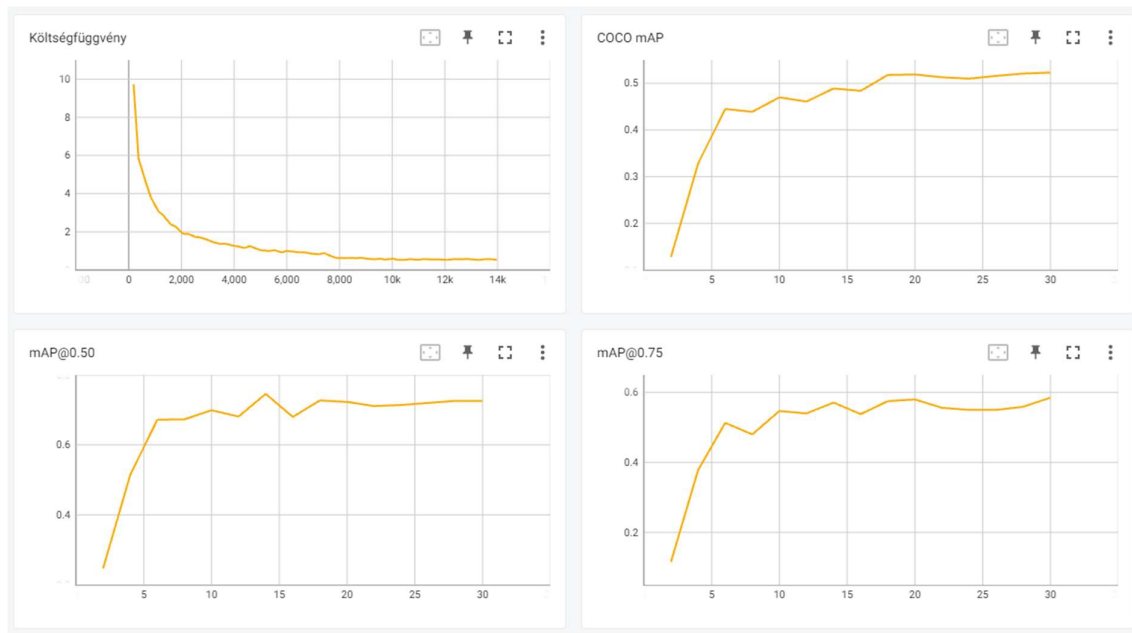


7.12. ábra: YOLOv3 mérőszámai a tanítás során.

Az SSD architektúra mindkét gerinc esetén nagyon jól teljesített amint a 7.13., 7.14. és 7.16. ábrák szemléltetik. Összességében jobban teljesített, mint a YOLO architektúra.

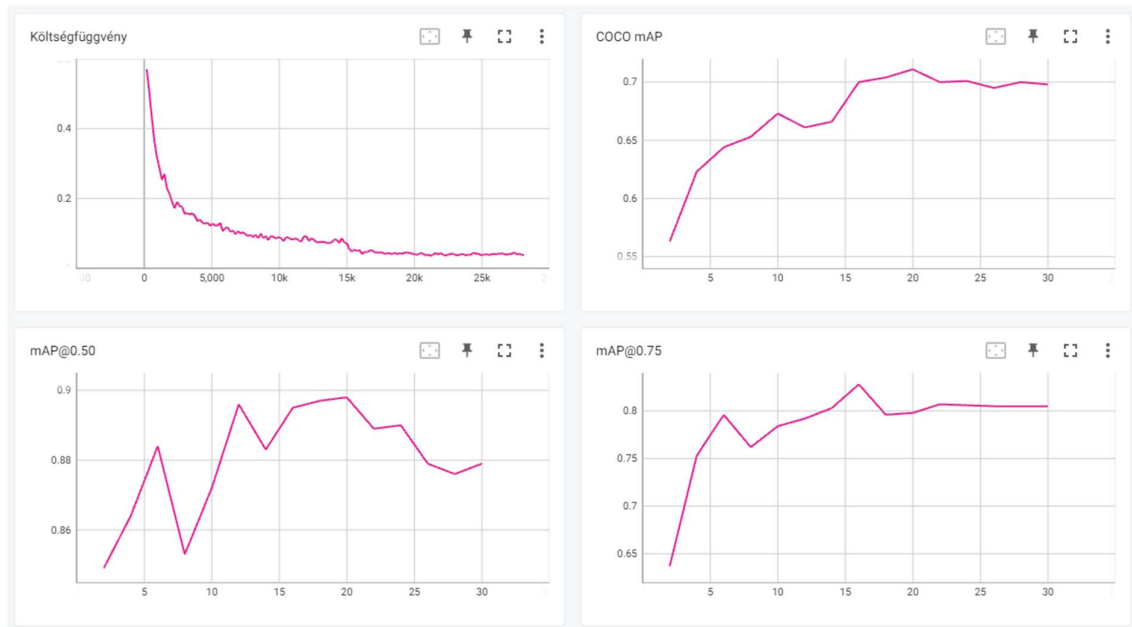


7.13. ábra: SSD (VGG16 gerinccel) mérőszámai a tanítás során.



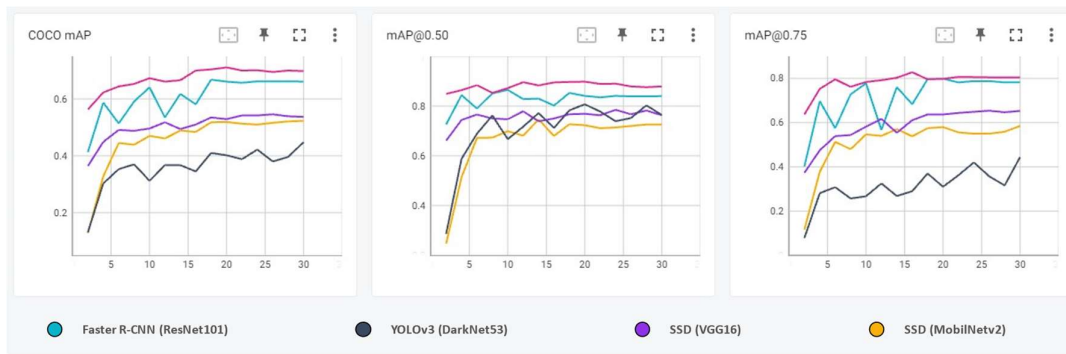
7.14. ábra: SSD (MobilNetv2 gerinccel) mérőszámai a tanítás során.

Meglepő módon ezen az adathalmazon az FCOS architektúra jobban teljesített az összes többi modellnél. Ez amint már szó volt róla a horgonydobozok kiiktatásának köszönhető.



7.15. ábra: FCOS mérőszámai a tanítás során.

Összességében az FCOS teljesített a legjobban, amit a Faster R-CNN modell követett. Az SSD (VGG16) és az SSD (MobilNetv2) áll a 3. illetve 4. helyen és a YOLO architektúra teljesített a leggyengébben.



7.16. ábra: A mérőszámok összehasonlítása a validációs halmazon.

Ha részletesebben megvizsgáljuk a teszt halmazban található mintákon az objektum észlelési eredményeket, akkor kijelenthetjük, hogy a detektorok alkalmasak valós körülmények között is felismerni az objektumokat, még akkor is, ha ezek részlegesen takarva vannak, mint a 7.17. ábra szemlélteti, ahol a baloldalon az elvárt kiment látható, illetve a jobb oldalon az objektum észlelő algoritmus által generált válasz.



7.17. ábra: Példa a teszhalmazból részlegesen takart objektumokra.

Viszont sok esetben a volumetrikus hatások, amelyekre a szintetikusán előállított mintákra nem igazán van jó minőségű példa, viszont a valós körülmények között az igen is jellemző, hogy a mezőgazdasági gépek működésük közben rengeteg por veszi őket körül. Az ilyen jellegű volumetrikus „zajt” a legtöbb játékmotor nem képes jó minőségben előállítani, ennek az egyik oka, hogy nehéz modellezni, valamint nagyon számítás igényes ezért a fejlesztők nem fektetnek erre nagy hangsúlyt. Viszont amint a következő mintákban (7.18. ábra) jól látható ez nagyon bezavarja a betanított objektum észlelő modelleket:



7.18. ábra: Példa volumetrikus „zajra” a teszhalmazban.

Egy másik visszatérő eset, hogy a mezőgazdasági gépek azok nagyon változatosak és lehetetlen az összes adott típusú gép modellről mintát gyűjteni. Illetve a legtöbb esetben a gazdák különböző módosításokat végeznek a gépeken annak érdekében, hogy számukra a legtöbbet nyújtsa az adott gép. Így, mint a 7.19. ábrán látható, akad olyan gép, amely lényegesen eltér azoktól, amik a tanítómintákban található, ezért van esély arra, hogy az algoritmus rossz osztályba sorolja az objektumot.



7.19. ábra: Példa téves osztályozásra a teszt halmazon.

Egy másik probléma, hogy révén, hogy a mezőgazdasági gépek éveken keresztül vannak alkalmazva, a környezet és strapás használat hatására idővel romlik az állapota: megsérül, megrozsdázik, kilikad stb. Ilyen szintű károokra nincs lehetőség a kiválasztót játékmotorban,

így amint a 7.20. ábrán látható van olyan objektum, amelyet nem képes felismerni a korróziók miatt.



7.20. ábra: Példa korrodált objektumra a teszt halmazban.

Az eredmények összefogásából az derül ki, hogy a III. táblázatban tapasztaltakhoz hasonlóan az MobilNet gerincet alkalmazó SSD architektúra teljesített igényelte a legkisebb számítási kapacitást, illetve ezzel több mint 50 FPS-el lehet objektum észlelést végezni. A III. táblázattal ellentétben a mezőgazdasági gépeket tartalmazó adathalmazon a YOLO architektúra teljesített a leggyengébben pontosság szempontjából, illetve az FCOS architektúra még Faster R-CNN modellt minimálisan felülmuta. Ez valószínűleg leginkább a szélsőséges méretarányoknak köszönhető, mint amilyenekre példát lathatunk a 7.10. ábrán. Az FCOS és a Faster R-CNN az pontosság és visszaidéző képesség szempontjából közel állnak egymáshoz, viszont az FCOS egy lépcsős detektor léte miatt lényegesen gyorsabb, illetve mivel sekélyebb gerincet alkalmaz, de nem csak sokkal kevesebb számítási kapacitást is igényel.

VI. táblázat: A „mAP@.50” az átlagpontosság, ahol a detektálási konfidencia nagyobb, mint 0.5. A „recall” a visszaidézőképesség.

Modell	mAP@.50 (valid.)	recall (valid)	mAP@.50 (teszt)	recall (teszt)	FPS	Késleltetés	CUDA mem.
Faster R-CNN (ResNet101)	0.863	0.570	0.678	0.570	14.5	69.0 ms	471 MB
YOLOv3 (DarkNet53)	0.766	0.554	0.648	0.463	27.2	36.8 ms	504 MB
SSD (VGG16)	0.744	0.575	0.608	0.437	19,7	50.8 ms	447 MB
SSD (MobilNetv2)	0.726	0.595	0.564	0.467	52.7	19.0 ms	159 MB
FCOS (ResNet50)	0.879	0.726	0.677	0.595	21.4	46.7 ms	276 MB

A IV. táblázatban látható értékek egy RTX3060 videokártyán volt mérve. Az alkalmazott feldolgozó egység lényegesen befolyásolja a késleltetést, viszont a szükséges memória, valamint a kapott eredmények azok feldolgozó egységtől nem függenek.

Végső soron a 7.18. 7.19. és 7.20. példák tapasztalataira építkezve vannak esetek amikor csak szintetikus minták alkalmazása a tanítóhalmazban nem elegendő. Annak érdekében, hogy a detektorok valós körülmények között is a lehető legjobban teljesítsenek valós minták alkalmazása is célszerű. Ez megvalósítható úgy, hogy valós mintákat is alkalmazunk a szintetikusan előállítottak mellett a tanítás során vagy utólag hangoljuk az modelleket valós mintákon.

8 Összefoglalás

A dolgozatban objektum detektáló algoritmusok légifelvételeken való alkalmazása van bemutatva. Tárgyalva vannak a szatellit és drón alapú felvételek közötti különbségek. Amint láthattuk ezek között elegendő különbség van ahhoz, hogy a szatellitfelvételeken betanított mélytanuló algoritmusok ne legyenek képesek objektumészlelésre. Ugyanakkor a minták gyűjtése drónok segítségével egy hosszadalmas folyamatnak bizonyulhat a drónok limitált üzemidejének köszönhetően és nehezen kivitelezhetőnek olyan célobjektumok esetében, amelyekhez légi szabályozás vagy egyéb okok miatt nincs hozzáférésünk. Ezeknek az akadályok áthidalására alkalmas lehet szintetikus minták alkalmazása, amely segítségével hosszútávon a ráfektetett időt és humánerőforrást lehet minimalizálni. A pályamunkában több mesterséges intelligencia alapú objektum észlelő architektúra van kipróbálva: Faster R-CNN, YOLOv3, SSD, illetve FCOS. Két megközelítés van tárgyalva: egy már meglévő adathalmazból (jelen esetben DOTA) való kiindulás, illetve az ezen betanított modellek tesztelése Google Maps alapú platform segítségével. Az itt elért eredmények alapján arra lehet következtetni, hogy mind addig míg a minták azok megközelítőleg 90 fokos szöget zártak be a horizonttal, addig az objektum észlelés az nagy pontossággal valósítható meg úgy szatellit felvételeken, mint szintetikusan előállított mintákon. Viszont abban az esetben, ha a vételezési szög a kezd eltávolodni a 90 foktól, azaz a fénykép nem felülnézetből, hanem a felülnézet és oldalnézet közötti átmeneten történik, a detektorok visszahívó képessége és pontossága lényegesen romlik. Egy másik megközelítés volt a tanítóminták egy 3D grafikus motorból kinyerni, majd drónnal vételezett és világhálóról gyűjtött valós mintákon kiértékelni. A tanítómintákat Farming Simulátor játékból kinyert képernyőfelvételekből voltak mintavételezve, szűrve, és címkézve; majd az ezzel betanított neuronháló modellek kiértékelésre kerültek egy valós mintákat tartalmazó adathalmazon. A vizsgálat legfontosabb eredménye, hogy az algoritmusok pontossága, illetve visszaidéző képessége kis eltérést mutatott a teszhalmazon a validációs halmazhoz képest, így ez egy jó megközelítésnek bizonyulhat olyan esetekben, ahol kis mennyiségben állnak csak rendelkezésre tanítóminták.

A bemutatott módszernek egyelőre vannak hiányosságai, mint a minták szűrésének igénye, vagy a minimális pontosság-, illetve visszaidézőképesség vesztes. Ennek ellenére a szintetikus minták segítségével betanított modellek alkalmazhatók UAV rendszerek által gyűjtött valós mintákon is. A pontosság az tovább növelhető abban az esetben, ha a tanítási folyamat során valós és szintetikus minták ötvöze van alkalmazva.

A TKP2021-NVA-16 számú projekt, a Magyar Innovációs és Technológiai Minisztérium által a Nemzeti Kutatási, Fejlesztési és Innovációs Alapból, a TKP2021-NVA támogatási konstrukció keretében finanszírozott támogatásával valósult meg a kutatás.

9 Bibliográfia

- [1] A. Németh és K. Virágh, „Mesterséges intelligencia és haderő – A mesterséges intelligencia területei III. rész,” *Haditechnika*, %1. kötet56, %1. szám3, pp. 2-7., 2022.
- [2] Y. LeCun, L. Bottou, Y. Bengio és P. Haffner, „Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, %1. kötet86, p. 2278–2324, 1998.
- [3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke és A. Rabinovich, „Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.
- [4] A. Krizhevsky, I. Sutskever és G. E. Hinton, „Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, %1. kötet60, p. 84–90, 2017.
- [5] R. Girshick, J. Donahue, T. Darrell és J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.
- [6] R. Girshick, „Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015.
- [7] J. Redmon, S. Divvala, R. Girshick és A. Farhadi, „You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser és I. Polosukhin, „Attention is all you need,” *Advances in neural information processing systems*, %1. kötet30, 2017.
- [9] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu és M. Chen, „Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.
- [10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser és B. Ommer, „High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [11] A. Németh és K. Virágh, „Mesterséges intelligencia és haderő - Polgári alkalmazási lehetőségek V. rész,” *Haditechnika*, %1. kötet56, %1. szám5, pp. 2-7., 2022.
- [12] A. Németh és K. Virágh, „Mesterséges intelligencia és haderő - További polgári alkalmazási lehetőségek VI. rész,” *Haditechnika*, %1. kötet56, %1. szám6, pp. 2-7, 2022.
- [13] S. T. Brassai, *Neurális hálózatok és fuzzy logika*, Cluj-Napoca: Scientia, 2019.
- [14] K. Simonyan és A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015.
- [15] K. He, X. Zhang, S. Ren és J. Sun, „Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto és H. Adam, „Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [17] J. Terven és D. Cordova-Esparza, „A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond,” *arXiv preprint arXiv:2304.00501*, 2023.
- [18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár és C. L. Zitnick, „Microsoft coco: Common objects in context,” in *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 2014.
- [19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn és A. Zisserman, „The pascal visual object classes (voc) challenge,” *International journal of computer vision*, %1. kötet88, p. 303–338, 2010.
- [20] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid és S. Savarese, „Generalized intersection over union: A metric and a loss for bounding box regression,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.
- [21] X. Wang és J. Song, „ICIou: Improved Loss Based on Complete Intersection Over Union for Bounding Box Regression,” *IEEE Access*, %1. kötet9, pp. 105686-105695, 2021.
- [22] S. Ren, K. He, R. Girshick és J. Sun, „Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, %1. kötet28, 2015.
- [23] K. He, G. Gkioxari, P. Dollár és R. Girshick, „Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017.

- [24] J. Redmon és A. Farhadi, „YOLO9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.
- [25] J. Redmon és A. Farhadi, „Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [26] A. Bochkovskiy, C.-Y. Wang és H.-Y. M. Liao, „Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [27] C.-Y. Wang, I.-H. Yeh és H.-Y. M. Liao, „You only learn one representation: Unified network for multiple tasks,” *arXiv preprint arXiv:2105.04206*, 2021.
- [28] Z. Ge, S. Liu, F. Wang, Z. Li és J. Sun, „Yolox: Exceeding yolo series in 2021,” *arXiv preprint arXiv:2107.08430*, 2021.
- [29] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie és others, „YOLOv6: A single-stage object detection framework for industrial applications,” *arXiv preprint arXiv:2209.02976*, 2022.
- [30] C.-Y. Wang, A. Bochkovskiy és H.-Y. M. Liao, „YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” *arXiv preprint arXiv:2207.02696*, 2022.
- [31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu és A. C. Berg, „Ssd: Single shot multibox detector,” in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 2016.
- [32] Z. Tian, C. Shen, H. Chen és T. He, „Fcos: Fully convolutional one-stage object detection,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.
- [33] B. Alvey, D. T. Anderson, A. Buck, M. Deardorff, G. Scott és J. M. Keller, „Simulated photorealistic deep learning framework and workflows to accelerate computer vision and unmanned aerial vehicle research,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.
- [34] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi és others, „Unity perception: Generate synthetic data for computer vision,” *arXiv preprint arXiv:2107.04259*, 2021.
- [35] A. Prakash, S. Bochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira és S. Birchfield, „Structured domain randomization: Bridging the reality gap by context-aware synthetic data,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.
- [36] M. Martinez, C. Sitawarin, K. Finch, L. Meincke, A. Yablonski és A. Kornhauser, „Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars,” *arXiv preprint arXiv:1712.01397*, 2017.
- [37] K. A. Özfuttu, „Generating synthetic data with game engines for deep learning applications,” 2022.
- [38] I. Rasmussen, S. Kvalsvik, P.-A. Andersen, T. N. Aune és D. Hagen, „Development of a Novel Object Detection System Based on Synthetic Data Generated from Unreal Game Engine,” *Applied Sciences*, %1. kötet12, p. 8534, 2022.
- [39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai és S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, 2019.
- [40] T. maintainers és contributors, *TorchVision: PyTorch's Computer Vision library*, GitHub, 2016.
- [41] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy és D. Lin, „MMDetection: Open MMLab Detection Toolbox and Benchmark,” *arXiv preprint arXiv:1906.07155*, 2019.
- [42] Y. Zhou, X. Yang, G. Zhang, J. Wang, Y. Liu, L. Hou, X. Jiang, X. Liu, J. Yan, C. Lyu, W. Zhang és K. Chen, „MMRotate: A Rotated Object Detection Benchmark using PyTorch,” in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022.
- [43] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo és L. Zhang, „DOTA: A Large-Scale Dataset for Object Detection in Aerial Images,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018.
- [44] J. Ding, N. Xue, Y. Long, G.-S. Xia és Q. Lu, „Learning RoI Transformer for Detecting Oriented Objects in Aerial Images,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [45] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo és L. Zhang, „Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1-1, 2021.

- [46] A. Hammas, S. T. Brassai, A. Németh és S. L. Gábor, „Training YOLO v4 on DOTA dataset,” in *2023 IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2023.
- [47] M. Ju, J. Luo, P. Zhang, M. He és H. Luo, „A simple and efficient network for small target detection,” *IEEE Access*, %1. kötet7, p. 85771–85781, 2019.
- [48] J. Lu, C. Ma, L. Li, X. Xing, Y. Zhang, Z. Wang és J. Xu, „A vehicle detection method for aerial image based on YOLO,” *Journal of Computer and Communications*, %1. kötet6, %1. szám11, pp. 98-107, 2018.
- [49] Z. Zakria, J. Deng, R. Kumar, M. S. Khokhar, J. Cai és J. Kumar, „Multiscale and direction target detecting in remote sensing images via modified YOLO-v4,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, %1. kötet15, pp. 1039-1048, 2022.
- [50] A. Hammas, „Harcjárművek észlelése szintetikusán előállított mintákon tanított mélytanuló algoritmusok segítségével,” in *NKE ITDK*, Budapest, 2023.
- [51] S. T. Brassai, A. Németh, A. Hammas és S. L. Gábor, „Deep learning based object detection for agricultural machinery,” in *2022 IEEE 22nd International Symposium on Computational Intelligence and Informatics and 8th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (CINTI-MACRo)*, 2022.

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA SISTEME DE CONTROL INTELIGENTE

Vizat decan
Conf. dr. ing. Demekos József

Vizat director departament
Șl. dr. ing. Szabó László Zsolt