

37. Országos Tudományos Diákköri Konferencia
Had- és Rendészettudományi Szekció
Budapest, 2025. április 22–24.
Nemzeti Köszolgálati Egyetem
Hadtudományi és Honvédtisztképző Kar

Harcjárművek észlelése szintetikusan előállított mintákon tanított mélytanuló algoritmusok segítségével

Jelige: *Medve Mester*

Pályamunka lezárásának időpontja: 2024. június

2025.

Tartalomjegyzék

1. Bevezetés.....	3
2. Elméleti áttekintés	6
2.1. Kétlépcsős detektorok	7
2.2. Egylépcsős detektorok	10
2.2.1. YOLO	10
2.2.2. SSD.....	14
2.2.3. Horgonyok alkalmazásának előnyei és hátrányai.....	15
2.3. Szintetikus adatalapú tanítás	16
3. Harcjárművek detektálásához szükséges adathalmaz	17
3.1. Tanító minták gyűjtése.....	17
3.2. CVAT – címkéző platform	18
3.3. Címkézés.....	19
3.4. Adathalmaz összeállítása	21
4. Mesterséges neuronhálók tanítása	23
5. A vizsgálatok eredményei	30
5.1. Az eredmények értelmezése.....	30
5.2. Eredmények gyakorlati jelntőssége.....	31
5.3. A kutatás korlátai	34
6. Összegzés	36
7. Felhasznált irodalom	37

1. Bevezetés

A 20. század végére ma már sokan nosztalgiával emlékeznek, amikor a Vasfüggöny leomlása egy békésebb világ délibábját vetítette Európa horizontjára. A kétpólusú, tartós szembenállás, elrettentés időszakát látszólag felváltotta egy együttműködésen, párbeszédén, kölcsönös előnyökön alapuló biztonságpolitikai környezet, amelyet „csak” korlátozott kiterjedésű fegyveres konfliktusok árnyaltak. A 21. század azonban gyorsan kijózanította a nyugati civilizációt rövid életű komfortzónájából, amikor azok a bizonyos repülőgépek a WTC ikertornyaiba csapódtak. Ennek hatására megváltozott az ellenségkép, ezzel együtt pedig a stratégiák és doktrínák, az emberiség új korszakra virradt. Az elhúzódó, széleskörű összefogáson alapuló nemzetközi terrorizmus elleni harc árnyékában azonban számos országban hatalomra kerültek, illetve megerősödtek olyan vezetők, akik a „nyugatitól” eltérő módszerekkel kezdték el formálni először közvetlen környezetüket, majd fokozatosan terjesztették ki hatalmukat különböző ideológiák mögé bújva. Ennek egyenes következménye, hogy mindennapjainkat ma újra a háború árnyékában kell élnünk, és még nem tudhatjuk, mit is hoz a holnap. Nem csoda, hogy egy ilyen környezetben az államok ismét haderejünk megerősítésén fáradoznak, annak érdekében, hogy garantálni tudják határaik biztonságát, országuk területének integritását, állampolgáraik védelmét. Ma már messze nem számít újdonságnak, az orosz-ukrán háború azonban egyértelműen bizonyította létjogosultságát, sőt előre vetítette a jövő háborúiban való dominanciáját a pilóta nélküli légijárművekkel, drónokkal végrehajtott katonai műveleteknek. Napjaink technikai színvonala és eszközrendszere számos más területen történő alkalmazás mellett, az információszerzésben, a katonai felderítésben történő felhasználást már a legszélesebb körűen képes támogatni számos olyan diszruptív megoldást ötvözve, mint például a mesterséges intelligencia, a 3D nyomtatás, vagy az alternatív valóság technológiák.

A pilóta nélküli légi jármű (UAV¹) egy olyan repülő eszköz, amelynek fedélzetén nincs személyzet, így ez lehet akár távirányítású repülőgép, de repülhet önállóan is egy előre programozott repülési terv, vagy bonyolultabb dinamikus, automatizált algoritmusok alapján. Bár az UAV-kat első megjelenésük óta leggyakrabban katonai tevékenységekkel lehetett összefüggésbe hozni, ma már többek között kutatásra, mentésre, megfigyelésre, forgalomfigyelésre, időjárás-jelenségek vizsgálatára és tűzoltásra is használják őket, de egyre szélesebb körben terjed hobbi célú alkalmazásuk is.

¹ UAV – Unmanned Aerial Vehicle

Katonai műveleti környezetben az UAV-k hasznosak lehetnek különböző felderítési feladatok ellátására, az ellenérdeket felek helyzetéről, személyi állományuk és haditechnikai eszközeik mennyiségéről, képességeiről, tevékenységéről információk összegyűjtésére, célpontok felkutatására, megjelölésére. Az ilyen eszközökkel történő tevékenységek a hagyományos repülő eszközöknél egyrészt lényegesen hatékonyabban végezhetők, másrészt nem veszélyeztetik közvetlenül saját katonáink életét [1] [2]. A mesterséges intelligencia (AI²) 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 mesterséges intelligencia a számítástechnika, és számítástudomány gyorsan bővülő területe, amely olyan intelligens gépek, megoldások kifejlesztésére törekszik, amik képesek akár emberi intelligenciát igénylő feladatokat is hatékonyan ellátni. Az ezen a területen dolgozók olyan algoritmusok és modellek létrehozásával foglalkoznak, amelyek képesek tanulni a rendelkezésre bocsátott információkból, érvelni és döntéseket hozni, felismernek mintákat, valamint természetes és intuitív módon kölcsönhatásba lépnek az emberekkel és környezetükkel. Az AI-kutatás végső célja olyan gépek létrehozása, amelyek az emberekhez hasonló módon, de gyorsabban és pontosabban képesek olyan összetett feladatokat ellátni, mint akár nyelvi fordítás, a vizuális felismerés, a döntéshozatal és a problémamegoldás.

A mesterséges intelligencia kutatás egyik legizgalmasabb és leggyorsabban fejlődő területe a neurális hálózatok és a mély tanuló algoritmusok. Ezek olyan gépi tanulási algoritmusok, amelyeket az emberi agy szerkezete és működése ihletett. A neurális hálózatok egymással összekapcsolt csomópontokból vagy neuronokból állnak, amelyek hierarchikus módon dolgozzák fel az információkat, és a neuronok minden rétege egyre összetettebb jellemzőket von ki a bemeneti adatokból. Ez a hierarchikus feldolgozás lehetővé teszi a neurális hálózatok számára, hogy az emberek tanulásához hasonló módon fejlesszék magukat az adatok felhasználásával [3].

A mélytanulás a neurális hálózati algoritmusok egy részhalmaza, amelyek több rétegű idegsejteket használnak az adatok összetett kapcsolatainak modellezésére. A mélytanulás forradalmasította az AI-kutatást, és áttörésekhez vezetett a számítógépes látás, a természetes nyelvi feldolgozás és a beszédfelismerés terén. [3] A mély tanulási modellek kínálják a leghatékonyabb megoldást számos kihívást jelentő feladatban, mint például a képosztályozás, az objektumészlelés és a gépi fordítás.

² AI – Artificial Intelligence

A mélytanulás egyik fő előnye, hogy képes nagy mennyiségű adatból tanulni. Ez különösen hasznos olyan alkalmazásokban, ahol sok adat áll rendelkezésre, például kép- vagy beszédfelismerés. A mélytanulási modellek hasznos információkat szintetizálhatnak az adatokból, és megtanulják felismerni még az emberek számára is nehezen észlelhető mintákat. A számos eredmény ellenére a mélytanulás kihívásokat is tartogat a fejlesztők számára. Az egyik fő probléma az úgynevezett túlillesztés, amikor a modell túlságosan ráilleszkedik a tanító mintákra, és gyengén teljesít az új adatokon. További kihívást jelent az értelmezhetőség hiánya, amikor nehezen érthető meg, hogyan jutott el a modell a döntéseihez.

A neurális hálózatok és a mély tanulás az AI-kutatás izgalmas területei, amelyek számos iparágat átalakíthatnak át a jövőben. [4] [5] Ahogy a terület folyamatosan fejlődik, valószínű, hogy még sok áttörést és újítást fogunk látni az elkövetkező években.

Az UAV-k segítségével végzett AI alapú hatékony felderítés alapvető követelménye, hogy eszközönként közel azonos számban és szögekből, valamint fényviszonyok és hátterek előtt készített, megfelelő mennyiségű, illetve minőségű fényképet tartalmazzon a tanítóhalmaz. Saját eszközök esetén a szükséges felvételek elkészítése nem okoz különösebb problémát, bár idő és erőforrásigényes, így költséges feladat. De az igazi nehézséget az olyan eszközök fotodokumentációjának összeállítása jelenti, amelyek nincsenek a saját haderőben rendszeresítve, és amelyekről a világhálón sem érhető el kellő mennyiségű forrásanyag.

A dolgozatom tárgyát képező kutattatás célja megvizsgálni, hogyan lehet a rendelkezésre álló fényképes adatbázist kiegészíteni mesterségesen generált képekkel. Annak a kutatásnak, amelynek keretében saját vizsgálataimat is végzem, egyik fontos célkitűzése számos katonai objektum mellett, harcjárművek detektálása. 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. Dolgozatban a szintetikus előállított harcjárművekről készült mintákból felépülő tanítóhalmazokon, mélytanuló algoritmusok segítségével fejlesztett és betanított modellek hatékonyságát vizsgálom valós mintákon végzett objektumdetekció során.

2. Elméleti áttekintés

A neurális hálózatok története egészen az 1940-es évekig nyúlik vissza, de a gyakorlatban csak az 1980-as és 1990-es években kezdődött meg szélesebb körű használatuk olyan alkalmazásokban, mint a beszédfelismerés, vagy a számítógépes látás. Abban az időben ugyanakkor a neurális hálózatok betanítása lassú és számításigényes folyamat volt, amihez nem állt rendelkezésre megfelelő technikai hattár, így hatékonyságuk korlátozott volt. A 2000-es években a nagy teljesítményű grafikus feldolgozóegységek (GPU³) megjelenésével lehetővé vált olyan korábbi tanító algoritmusok eredményes alkalmazása, mint például a backpropagation, de kifejlesztésre kerülhettek olyan új algoritmusok is, melyek ismét felkeltették a kutatók érdeklődését a neurális hálózatok iránt. Az ezt követő fejlődés a konvolúciós hálókön keresztül a mélytanulás kialakulásához vezetett, ami a neurális hálózatok egy olyan részterülete, amely több neuronréteget használ az adatok összetett kapcsolatainak modellezésére. A mélytanulás azóta forradalmasította az mesterséges intelligencia számos területét, beleértve a kép- és beszédfelismerést, a természetes nyelvi feldolgozást és az autonóm vezetést.

Az objektumdetektálás alapvető probléma a gépi látásban, amely magában foglalja a képeken vagy videókon lévő objektumok észlelését és lokalizálását. Az objektumészlelési algoritmusok fejlődése szorosan követte a mélytanulás fejlődését. A korai objektumdetektálási algoritmusok, mint például a Viola-Jones és a HOG⁴, kézzel készített funkciókra és egyszerű osztályozókra támaszkodtak az objektumok észlelése során. A 2010-es évek elején a mélytanulási modellek, például a konvolúciós neurális hálózatok (CNN⁵-ek) fejlődése jelentős javulást eredményezett az objektumészlelési pontosság terén. A régióalapú konvolúciós neurális hálók (R-CNN⁶), például a Fast R-CNN és a Faster R-CNN fejlesztése pedig még tovább javította a pontosságot és a sebességet egyaránt azáltal, hogy régiójavasoló neurális hálókat használtak a feldolgozandó objektumhelyek számának csökkentésére. A közelmúltban az egylépcsős objektumészlelési modellek, mint például a YOLO⁷ és az SSD⁸ a legjobb teljesítményt érték el ezen a területen, miközben sokkal gyorsabbak is voltak, mint a korábbi módszerek. Az objektumészlelési

³ GPU – Graphics Processing Unit

⁴ HOG – Histograms of Oriented Gradients

⁵ CNN – Convolutional Neural Network

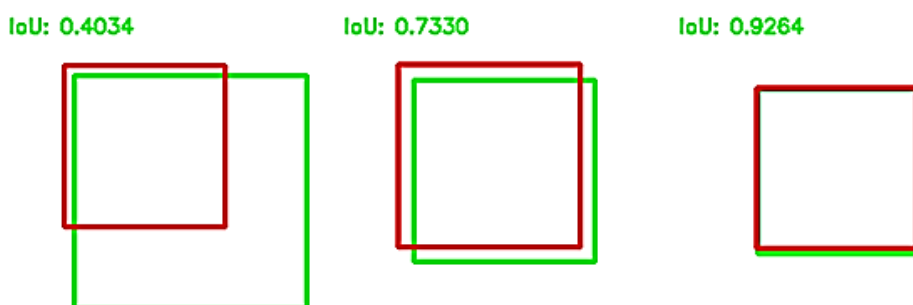
⁶ R-CNN – Region based Convolutional Neural Networks

⁷ YOLO – You Only Look Once

⁸ SSD – Single Shot Detector

algoritmusok fejlődése számos gyakorlati alkalmazás kifejlesztését tette lehetővé, mint például az autonóm vezetés, a megfigyelés és orvosi döntéshozatalt támogató diagnosztika.

Aszerint, hogy milyen megoldással történik a képek feldolgozása, az objektum detektáló neuronháló modellek két fő kategóriába sorolhatók, mint a kétlépcsős-, illetve az egylépcsős detektorok.



1. ábra: IoU metrika [6].

Az objektum észlelő algoritmusok kiértékelésére leginkább az MS COCO, vagy a Pascal VIOC adathalmazokat szokták alkalmazni, mint viszonyítási alap, ahol a mérőszámok azok IoU⁹-ra alapoznak. Az IoU, más néven Jaccard index, a leggyakoribb használt metrika két tetszőleges alakzat hasonlósága mértékének megállapítására. Az IoU kódolja az összehasonlítandó objektumok alaktulajdonságait, mint például két határolókeret szélességét, magasságát és elhelyezkedését a régió tulajdonságba, majd kiszámít egy normalizált értéket, amely a területükre (vagy térfogatukra) összpontosít, amint a 1. ábra szemléltet. Az IoU metrikák átlaga fogja majd képezni a detektorunk osztályonkénti átlagpontosságát (AP¹⁰), valamint azok átlaga a teljes detektor átlag pontosságát (mAP¹¹). Ennek a metrikának több variánsáról is találhatók informatív leírások a szakirodalomban [7, 8].

2.1. Kétlépcsős detektorok

A kétlépcsős objektumdetektorok működése két fő szakaszból áll, a régiójavaslatból és az objektumok osztályozásából. A régiójavaslat szakaszában az algoritmus létrehozza a képen a jelölt objektumhelyek készletét, míg az objektumosztályozási szakaszban az algoritmus minden egyes jelölt helyet besorol egy objektum, vagy háttér osztályba.

Amint a 2. ábra is mutatja, az eredeti R-CNN [9] architektúra három fő összetevőből áll: egy régiójavaslató hálózattól (RPN¹²), egy konvolúciós neurális hálózattól (CNN) a jellemzők

⁹ IoU – Intersection over Union

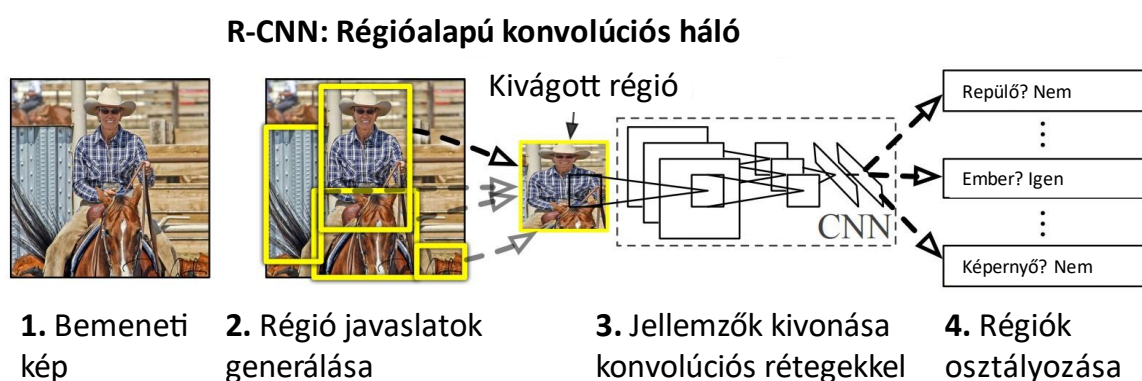
¹⁰ AP – Average Precision

¹¹ mAP – mean AP (Average Precision)

¹² RPN – Region Proposal Network

kinyerésére, és az osztályozást támogató vektorgépekből (SVM¹³) az objektumok besorolására. A régiójavaslat szakaszában az RPN létrehoz egy régiókat tartalmazó halmazt úgy, hogy egy kis ablakot csúsztat a kép fölött, és megjósolja, hogy minden ablak tartalmaz-e objektumot, vagy sem. Ezt az algoritmust az angol irodalomban „sliding window” néven ismerik. Az RPN egy keretet (bounding box) is generál minden egyes jelölt helyhez.

Az objektumok osztályozási szakaszában a CNN minden egyes jelölt régióból kivonja a jellemzőket, és átadja azokat egy SVM-nek, ami a helyet objektumként vagy háttérként osztályozza. Az SVM-ek nagy mennyiségű címkézett képek halmazán vannak tanítva, hogy fel tudják ismerni a különböző típusú objektumokat azok jellemzői alapján.



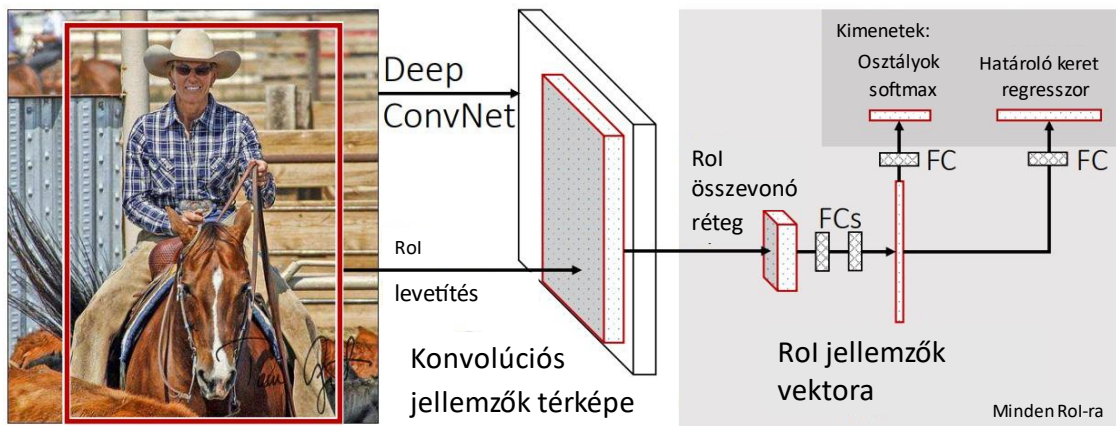
2. ábra: R-CNN algoritmus lépései [9].

Ross Girshick, az R-CNN algoritmus egyik fejlesztője, később javasolt egy javított Fast R-CNN [10] architektúrát, ami a 3. ábrán látható. Az új modellben az egyik fő módosítás 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 és lehetővé teszi a jellemzők pontosabb megjelenítését. Egy másik fejlesztés az érdekes régió RoI¹⁴ összevonó(pooling) réteg használata, amely lehetővé teszi a változó 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. Ezeken kívül a Fast R-CNN lecseréli az R-CNN-ben használt SVM osztályozókat teljesen összezsugorított rétegekre (az ábrán FC-ként van feltüntetve) illetve softmax osztályozókra, amelyek jobban megfelelnek a többosztályos osztályozási feladatoknak. Összességében a Fast R-CNN által

¹³ SVM – Support Vector Machine

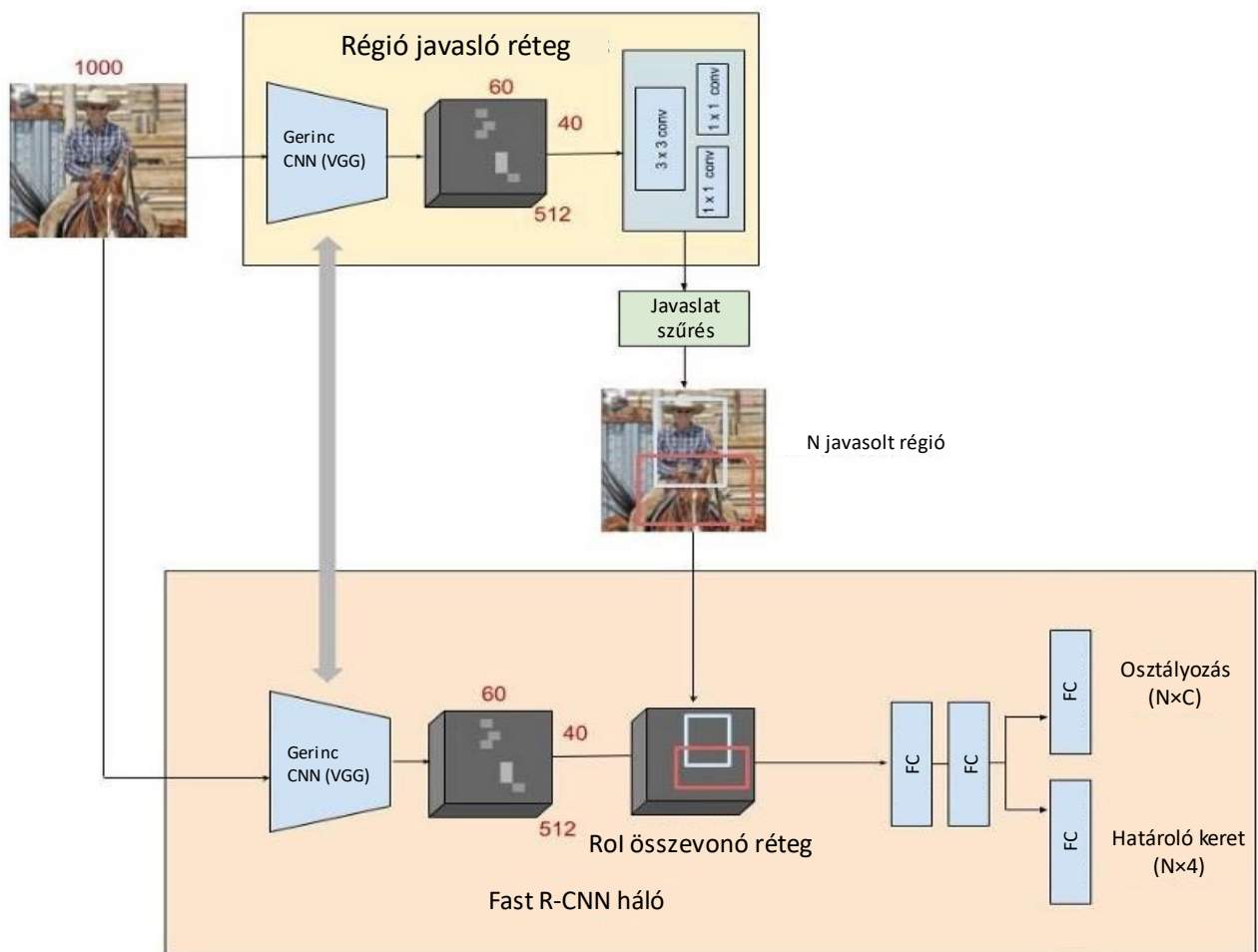
¹⁴ RoI – Region of Interest

bevezetett javítások gyorsabbá, pontosabbá teszik a modellt, miközben a tanítás folyamatát is egyszerűsítik.



3. ábra: Fast R-CNN architektúra [10].

A Faster R-CNN [11] a Fast R-CNN algoritmus továbbfejlesztett változata, amely számos kulcsfontosságú változtatást vezetett be. A struktúra a 4. ábrán látható.



4. ábra: Faster R-CNN architektúra [11].

Az egyik fő újítás az RPN használata, amely megosztja a konvolúciós jellemzőket az objektumészlelési hálózattal, lehetővé téve a régiójavaslatok gyorsabb generálását. Ez csökkenti a számítási időt a csúsztatott ablak megközelítéshez képest, és lehetővé teszi, hogy a hálózat közvetlenül a bemeneti képből tanuljon meg régiókat javasolni. Ugyanakkor a Faster R-CNN bevezeti a horgonydobozok (anchorbox) fogalmát is, amelyeket referenciapontként használnak az RPN számára a régiójavaslatok generálásához, így ezek segítségével egyszerre több méretű, illetve méretarányú régiókat is fel lehet térképezni egy átfutással. A detektor a Fast R-CNN-hez hasonlóan csupán egyszer nyeri ki a jellemzőket a képből, és ezeket alkalmazva javasolja a régiót, illetve végzi el a régiók osztályozását. Ezeknek a tovább fejlesztése a Mask R-CNN [12], ami egyszerű objektum detektálás mellett szegmentációra is képes.

2.2. Egylépcsős detektorok

A kétlépcsős detektorok nagy pontosságukról ismertek, de kialakításuk miatt számításigényesek lehetnek. Ez gyorsabb egyfokozatú detektorok kifejlesztéséhez vezetett, mint például a YOLO és az SSD, amelyek egyetlen lépésben végeznek objektum detektálást, anélkül, hogy külön régiójavaslati szakaszra lenne szükség. Mind az SSD, mind a YOLO gyorsaságukról és hatékonyságukról ismert, és arról, hogy különféle hardverplatformokon valós időben képesek objektumészlelést végezni. Ezek a detektorok azonban jellemzően kisebb pontossággal rendelkeznek, mint kétlépcsős társaik.

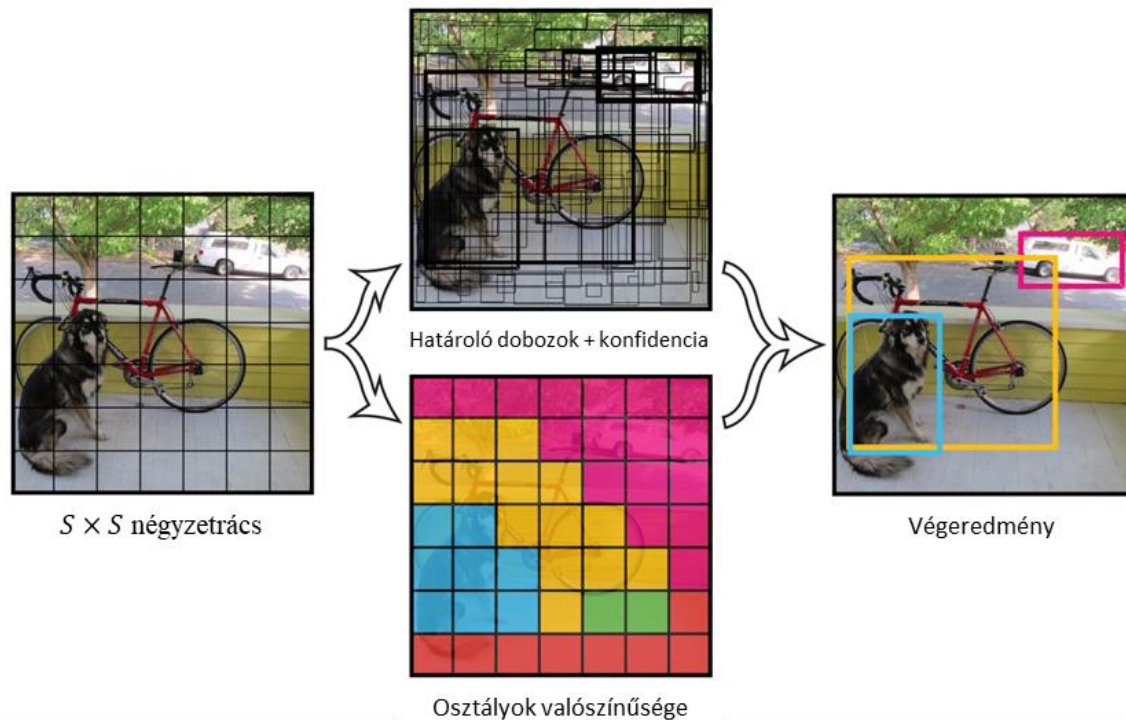
2.2.1. YOLO

A YOLO [13, 14, 15, 16] architektúra sorozat egy népszerű, egy lépéses objektumdetektáló algoritmus. Az architektúra több kulcsfontosságú komponensből áll, köztük egy sor konvolúciós és pooling rétegből, majd teljesen összekapcsolt rétegekből, amelyek eredménye az objektumosztály valószínűsége és a határoló doboz koordinátái.

A Joseph Redmon és társai által bevezetett algoritmus fő ötlete, hogy a bemeneti képet egy $S \times S$ méretű négyzetrácsra bontja, majd mindegyik cellához rendel B darab keretet, illetve C darab valószínűségi értéket, hogy az észlelt objektum melyikbe tartozik. Azonban mint az 5. ábrán is látható ez a megközelítés nagyon sok javasolt keretet eredményez, ezért ezeknek a számát vissza kell szorítani, amire egy nem-maximális elnyomási (NMS¹⁵) modul

¹⁵ NMS – Non-Maximum Suppression

alkalmaznak, ami elnyomja a kis valószínűségekkel rendelkező kereteket a nagy valószínűségűekhez képest [13].



5. ábra: A YOLO algoritmus működési alapelve [13].

A YOLO architektúra fő összetevői a következők:

- A YOLO neuronháló bemenete – az R-CNN hálókhoz hasonlóan – egy képet tartalmazó kétdimenziós mátrix, aminek a mérete általában rögzített.
- A YOLO hálózat konvolúciós rétegei a bemeneti képből vonják ki a jellemzőket. A hálózat több konvolúciós réteget használ, a szűrőméretek rétegenként nőnek, így a jellemzőket tartalmazó tenzorok térbeli méretben rétegről rétegre csökkennek, valamint mélységre nőnek. A jellemzők méretbeli csökkenése érdekében pooling (összevonó) rétegek is alkalmazásra kerülnek.
- A hálózat teljesen összekapcsolt rétegei határozzák meg az objektumosztály valószínűségét és a határolódoboz koordinátáit a bemeneti kép minden rácscellája számára. Ezek a rétegek konvolúciós és gyűjtőrétegek kombinációját használják a jellemzőtérképek fix méretű tenzorral alakításához.
- Miután a hálózat megjósolta az objektumosztály-valószínűségeket és a határolókeret-koordinátákat, a rendszer egy NMS algoritmust alkalmaz az ismétlődő észlelések eltávolítására és a legbiztosabb észlelések kiválasztására.

Redmon és társai cikkükben arról számolnak be, hogy a YOLO algoritmus nagyon jó általánosítási tulajdonságokkal rendelkezik, azaz a valós képeken betanított hálót festményeken tesztelve is jó hatásfokkal működik.

Az eredeti YOLO algoritmus azonban számos korláttal rendelkezett [13]:

- Az egyik lényeges megszorítás, hogy az algoritmus cellánként csak 2 keretet becsült. Ennek hatására a háló által észlelt sűrűn elhelyezett objektumok száma limitálva van.
- Révén, hogy a háló a cellákban kinyert jellemzők alapján tanulja megbecsülni a kereteket, gyengén teljesít, amikor egy szokatlan méretarányú objektumokról beszélünk. Ugyanakkor mivel a jellemzők több összevonó rétegen is átmennek amíg a háló ezekből egy határoló dobozt becsülne, a háló egy durva textúrájú jellemző alapján javasolja a határoló dobozt, ami sokszor vezet hibához.
- 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 [14], YOLOv3 [15], YOLOv4 [16]) próbálták javítani. A YOLOv2 (a szerzők YOLO9000 néven mutatták be) ugyanakkor már jelentős javulást eredményezett eredeti modell korlátjainak lebontása során [14]:

- 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 [14].

A YOLOv3 inkrementális javításokat tartalmazott [15]:

- A YOLOv3-ban egy mélyebb darknet53-nak nevezett gerincet vezettek be, ezzel növelve a kinyerhető jellemzők komplexitását és javítva a háló pontosságát.
- A YOLOv3 FPN¹⁶-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-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 PANet¹⁷-et a több szintes jellemzők egyesítésére, egy újabb MISH-nek nevezett aktivációs függvényt, valamint CIOU¹⁸ alapú metrikát a háló kiértékelésnek a javítására.

Az ezt követő YOLO algoritmus verziók (YOLOv5, YOLOR [17], YOLOX [18], YOLOv6 [19], YOLOv7 [20], YOLOv8, stb.) már nem az eredeti kutatócsoportokhoz kapcsolódnak, így jó részük már PyTorch keretrendszerben került megvalósításra a DarkNet helyett. Elődjeikhez

¹⁶ FPN – Feature Pyramid Network

¹⁷ PANet – Path Aggregation Network

¹⁸ CIOU – Complet Intersection over Union

hasonlóan ezek is sok új funkcióval bővítették a YOLO algoritmust, mint például a szegmentáció, szkeleton meghatározás, és persze pontosság növelése [6]. A dolgozat írásnak idején az Ultralytics cég még nem publikálta az általuk fejlesztett YOLOv5 és YOLOv8 architektúra tapasztalatait.

I. táblázat: YOLO architektúrák összefoglalása [6].

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

2.2.2. SSD

Egy másik népszerű egylépcsős objektum detektálásra alkalmazott algoritmus a korábban is említett SSD [21], ami úgy működik, hogy először létrehoz egy rend alapértelmezett határolókeretet különböző méretarányokkal a bemeneti képen. A modell ezután konvolúciós rétegek sorozatát alkalmazza, hogy kivonja a jellemzőket a képből, és minden egyes alapértelmezett határolókerethez a modell megjósolja az osztályvalószínűségeket és az eltolásokat. Az SSD egyik előnye a sebessége, amivel képes akár valós idejű objektumészlelést is elérni, valamint nagy pontossággal képes detektálni a különböző méretű és képarányú objektumokat, így ez egy sokoldalú objektumészlelési modell. Nehézségekbe ütközhet ugyanakkor a kisméretű objektumok detektálásánál, mivel az alapértelmezett keretek nem biztos, hogy képesek teljes mértékben rögzíteni ezen objektumok részleteit. Ennek a korlátnak a kiküszöbölése érdekében a szakértők az SSD néhány változatát, például az SSD-t Feature Fused-el (SSDFF) és az SSD piramisokkal (SSDP) megoldást javasolták, amivel lehetőség nyílik a kis tárgyak észlelési pontosságának javítására.

2.2.3. Horgonyok alkalmazásának előnyei és hátrányai

Sok objektum detektáló algoritmus használ horgony dobozokat a pontosság és visszahívó képesség növelésére. Például a YOLOv1 nagyon kis késleltetéssel volt képes pontos határoló kereteket becsülni, viszont gyenge volt a visszaidéző képessége (recall). A YOLO következő iterációjában (YOLO9000) a szerzők horgonydobozokat kezdtek el alkalmazni annak érdekében, hogy növeljék a visszahívó képességét a modellnek anélkül, hogy sokkal több erőforrást igényelne a rendszer. A horgonyok segítenek csökkenteni a kiértékelendő háttér régiók számát, ami hatékonyabb és gyorsabb észlelési folyamatot eredményez.

A horgonydobozok azonban újabb kihívást jelentenek az objektum észlelő algoritmusok tervezése során. Ezek megfelelő számának és méretének kiválasztása gondos mérlegelést igényel az adatkészletben lévő objektumok méretére és képarányára vonatkozólag. A nem megfelelő rögzítődobozok kiválasztása pontatlan észleléshez vagy lassabb tanításhoz és becsléshez vezethet. Ugyanakkor a horgonyok feltételezik, hogy az objektumok rögzített képarányúak, ami nem minden esetben igaz. Például, ha a horgonydobozok rögzített képaránya 1:1, előfordulhat, hogy nem képes észlelni a megnyúlt tárgyakat, például oszlopokat vagy fákat, ennek kiküszöbölésére több horgonydobozt alkalmaznak egy adott ponton.

A horgonydobozokra úgy tekinthetünk, mint egyfajta hiperparaméterekre, amelyek lényegesen kihatnak a detektor működésére [22]. Tian és társai egy olyan egylépcsős objektumészlelési algoritmus javasoltak, amely kizárólag konvolúciós műveleteket használ az objektumok helyének becslésére és osztályba sorolására, amit közvetlen a bemeneti képről hajt végre. Az FCOS¹⁹ [22] egy egylépcsős objektumészlelési algoritmus, amely egy teljesen konvolúciós hálót használ az objektumok helyének és osztálypontoszámainak előrejelzésére, amit közvetlenül a képről hajt végre. Más detektorokhoz képest az FCOS kiküszöböli a horgonydobozok használata által okozott hátrányokat is, és elkerüli a térinformáció elvesztését, amelyet a RoI-alapú módszerek pooling művelete okoz. Ez egyszerűbbé, hatékonyabbá és könnyebben optimalizálhatóvá teszi az FCOS-t, mint detektort. Ezenkívül az FCOS nagy pontosságot tud elérni az objektumészlelési feladatokban, különösen kisméretű objektumok esetén.

¹⁹ FCOS – Fully Convolutional One-Stage object detection

2.3. Szintetikus adatalapú tanítás

A mélytanuló neurális hálózatok tanítására rengeteg adatra van szükség, viszont vannak esetek, amikor a rendelkezésre álló adatmennyiség limitált. A szakirodalomban vannak arra próbálkozások, hogy a valós adatok mellett alkalmazzunk zajjal argumentált, vagy akár szintetikusan előállított mintákat. Korábbi vizsgálatok eredményei azt mutatják, hogy a szintetikus és valós adatok kombinálása hogyan javíthatja a betanított konvolúciós neuronháló modellek teljesítményét [23, 24, 25]. Kihívást jelenthet ugyanakkor az a feladat, amikor egy olyan valós objektum detektáló modellt szeretnénk létrehozni, amely tanítását pusztán szintetikus adatokon végeztük [25]. Viszont van rá példa a szakirodalomban, hogy csupán szintetikus adatok alkalmazásával hogyan készíthetünk objektum észlelő modelleket [26, 27, 28]. Ingeborg Rasmussen és társai például Unreal Engine 4 segítségével generáltak fotórealisztikus szintetikus mintákat az objektum detektáló algoritmusok tanítására [28]. A szintetikus adatok segítségével korlátlan számú tanítási adatot tudnak biztosítani automatikus generálással. 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 [28]. 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. Ezenkívül a cikk értékes betekintést nyújt abba, hogy az eszközök száma és a tanításra alkalmazott minták hogyan befolyásolják a kapott modellt.

A további fejezetekben a fenti kutatások eredményei alapján vizsgálom a szintetikus objektumok felhasználásának lehetőségeit harcjárművek detektálása során.

3. Harcjárművek detektálásához szükséges adathalmaz

A tanulmány alapját képező kutatás célja tehát, hogy megvizsgáljuk, lehetséges-e katonai objektumok detektálására alkalmas neuronháló modellt készíteni úgy, hogy minimális számú valós képet alkalmazunk, vagy akár egyáltalán nem alkalmazunk valós képeket a tanítás során. A katonai objektumok közül is elsősorban a harcjárművekre fókuszáltam, így célszerű megközelítés volt, hogy egy olyan játékot alkalmazzak adatforrásként, amelyben sok harcikocsi típus megtalálható. A modellek kiértékeléséhez pedig célszerű valós mintákat alkalmazni, hiszen ezáltal vizsgálható, hogy a modelleknek mennyire jó az általánosító képessége, illetve hogy a szintetikus tanítóhalmazon tanult hálók mennyire alkalmazhatók hatékonyan valós minták esetén.

3.1. Tanító minták gyűjtése

A minták gyűjtésére a „World of Tanks” (WoT) nevű online számítógépes játékot használtam, hiszen ez rengeteg harcikocsi modellt tartalmazza az első világháborús időktől egész a hidegháború végéig alkalmazott, illetve tervezett eszközökről. Nagyon hasznos volt, hogy a játék rendelkezik egy garázsnak nevezett jelenettel, ahol a tankok körül szabadon tudunk körbe forogni, így nagyon sok szögből lehetett gyűjteni mintát különböző típusú harcikocsikról. Hátrány azonban, hogy a környezet nem változtatható, így van esély arra, hogy a fejlesztett detekciós modellek nem általános jellemzőket tanulnak, hanem túltanulnak, azaz csak az adott jelenetben lesznek képesek felismerni a célobjektumokat. A 6. ábrán egy 122TM (1970-es évekből) harcikocsi látható különböző szögekből a garázs jelenetben.

A háttér diverzifikálása érdekében nem csak ebben a jelenetben végeztem mintavételezést a játékból, hanem különböző játékmeneteken (csatákon) keresztül is készült képernyőfelvétel. Ennek köszönhetően a több típusú háttérben is találhatók tankok, ezzel robusztusabbá téve a mintákon tanított modellt. (7. ábra)

A képernyőfelvételek OBS Studio-val készültek, majd ezeket FFMPEG könyvtár segítségével másodpercenként 1-2 képkockával mintavételeztem. Mindkét eszköz nyílt forráskódú és ingyenesen letölthető, és számos beállítási lehetőséget biztosít a felvételezés minőségének szempontjából.



6. ábra: Közeli minták egy harcikocsiról különböző szögekből garázs jelenetben.



7. ábra: Tankok különböző háttérrel és helyzetekben csata közben.

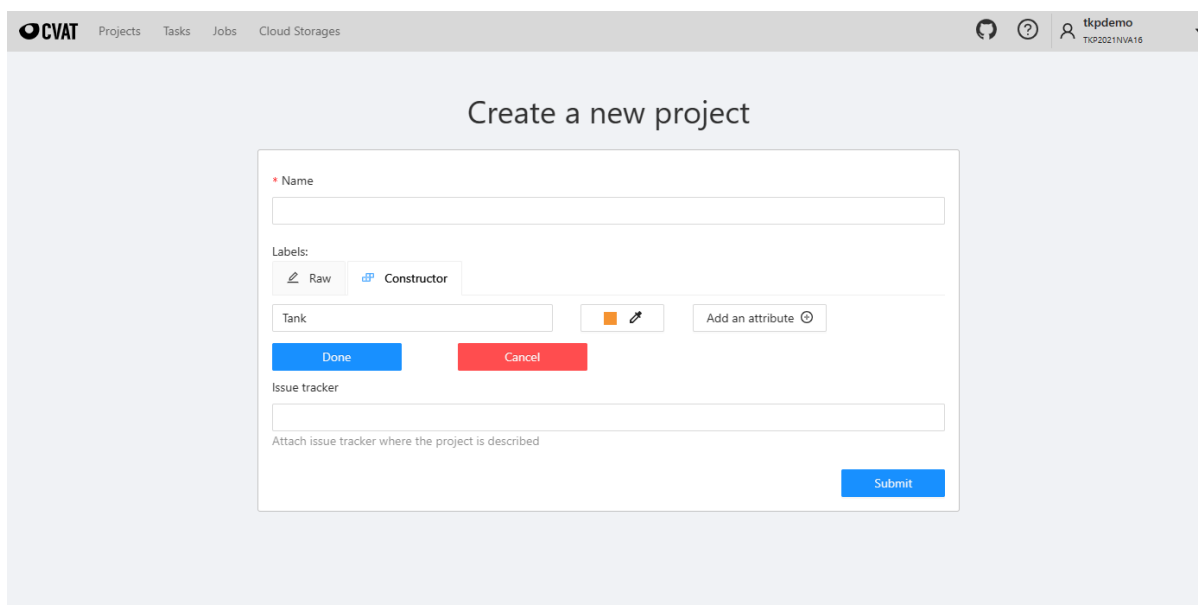
3.2. CVAT – címkéző platform

A kinyert képkockákat a Computer Vision Annotation Tool (CVAT) nevű szoftverplatform segítségével címkéztem, ami egy ingyenes, nyílt forráskódú, webalapú kép- és videóannotációs eszköz, amit kifejezetten számítógépes látási algoritmusok adatainak címkézésére fejlesztettek. Az eredetileg az Intel által fémjelzett CVAT-ot egy professzionális alkalmazásokban való

használatra tervezték, a felhasználói felületet pedig számítógépes látásannotációs feladatokra optimalizálták. Nemrég a platform fejlesztése egy külön jogi entitás kezébe került, aminek a csapatát többnyire egykori Intel, illetve OpenCV fejlesztők alkotják.

A szoftvernek elérhető már egy telepíthető változata (<https://app.cvat.ai>) is, amelynek bizonyos megkötésekkel ingyenes a használata. Én a címkézés során a lokálisan telepített verziót használtam. A telepítési folyamat nem bonyolult mindaddig, amíg nincs szükség lényeges módosításokra. A CVAT GitHub²⁰ oldalán elérhető, és alapértelmezett Docker-es állományok minimális módosításával (URL, verziószám cserélése) a rendszer percek alatt telepíthetővé válik lokálisan. Amennyiben módosítani szeretnénk a jogi státuszt, vagy a funkcionalitásokat, akkor ezeket is 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.

3.3. Címkézés



8. ábra: Projekt létrehozása CVAT platformon.

A címkézési folyamat egyáltalán nem bonyolult, egyszerűen csak létre kell hozni a projektet, ahol definiálni lehet az osztályokat, illetve ezek attribútumait. A 8. ábra a projekt létrehozását szemlélteti.

²⁰ opencv/cvat – <https://github.com/opencv/cvat>

Ezt követően a projekten belül fel tudjuk tölteni a címkézendő képeket. A címkézendő, illetve címkézett képek CVAT-n belül „task”-okba vannak rendszerezve. A képek feltöltését egy „task” létrehozásával tehetjük meg (9. ábra).

CVAT Projects Tasks Jobs Cloud Storages tkpdemo TKP2021NVA16

Create a new task

Basic configuration

* Name

Project: WoT project

Subset: Input subset

Labels: Project labels will be used

* Select files

My computer Connected file share Remote sources Cloud Storage

Click or drag files to this area
Support for a bulk images or a single video

Advanced configuration

* Sorting method

☒ Lexicographical ☐ Natural ☐ Predefined ☐ Random

Specify how to sort images. It is not relevant for videos.

☒ Use zip chunks

Force to use zip chunks as compressed data. Actual for videos only.

☒ Use cache

Using cache to store data.

* Image quality: 70 %

Overlap size

Segment size

Start frame

Stop frame

Frame step

Chunk size

Dataset repository URL

e.g. <https://github.com/user/repos> [annotation/<anno_file_name>.zip]

Attach a repository to store annotations there

Choose format

CVAT for video 1.1

9. ábra: Task létrehozása CVAT platformon.

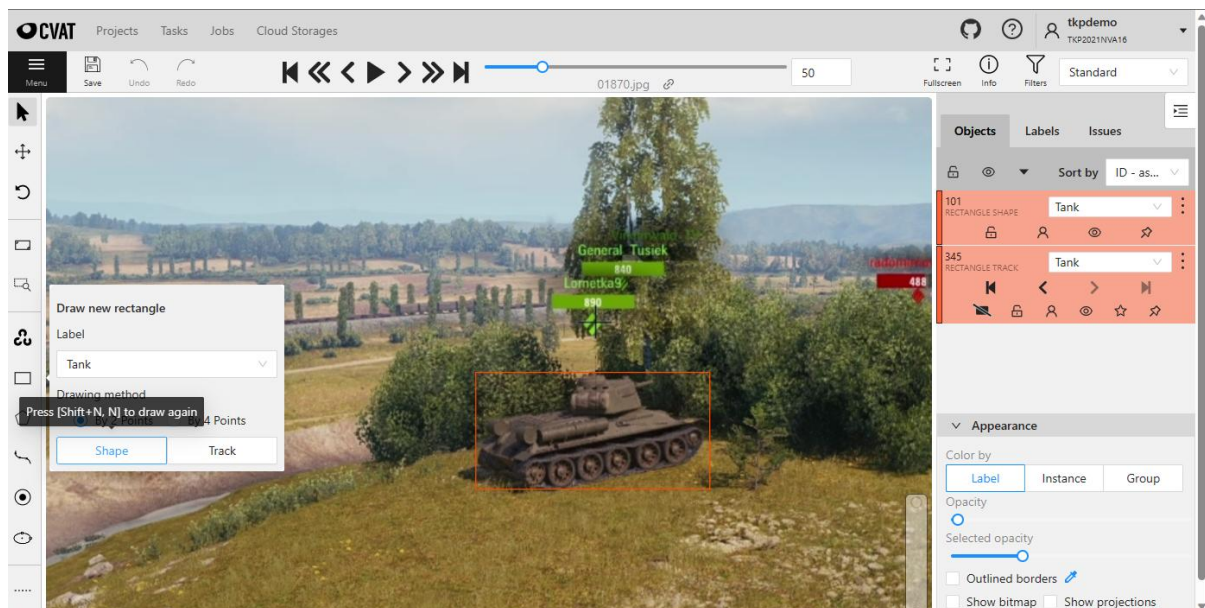
Feltöltéskor be lehet állítani a fontos paramétereket, mint például, hogy milyen minőségben kerüljenek tárolásra a képek, vagy milyen címkéző formátumot alkalmazzon a címkék

tárolására. Fontos, hogy olyan címkéző formátumot válasszunk, ami képes eltárolni mindazt az információt, amit mi szeretnénk (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, ezért a címkézés során én is azt alkalmaztam. 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.

A CVAT platform sok más kisegítő 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 változik a pozíciója.

A határolódobozok elhelyezést követően azok megjelennek egy listában, ahol ki lehet őket választani, valamint módosítani is lehet azokat. A címkéző felületről látható egy képernyőfelvétel a 10. ábrán.



10. ábra: Címkézés a CVAT platformon.

3.4. Adathalmaz összeállítása

Az adathalmazt háromféle képből állítottam össze:

- Garázsjelenetben körbeforgatott harckocsik: itt közelről szerepeltek az eszközök, sok részlet kivehető a képekből.

- Csatajelenetekből mintavételezett képek: itt a tankok már nagyobb távolságban helyezkedtek el a virtuális kamerától, így az apróbb részletek már nem kivehetők.
- Valós képek: gyakorlatokon rögzített felvételek harckocsikról.

A szintetikus képek (garázsjelenet és csatajelenetek) a tanító, illetve validáló halmazban voltak alkalmazva, míg a valós képek a teszt halmazt alkották. Összesen 846 kép került felhasználásra, melyből 616 a tanításra, 50 a validációhoz, valamint 80 a háló kiértékeléséhez. Az adathalmaz COCO formátumban került kimentésre a CVAT címkéző platformról, amelyben alkalmaztam azokat a tanítás, illetve a kiértékelés folyamán.

4. Mesterséges neuronhálók tanítása

A hálók tanítást, illetve a kiértékelést MMDetection keretrendszerrel végeztem, 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.

Az objektum észlelő algoritmusok esetén egy bevált módszer a már előre betanított modellek alkalmazása. Ezzel elérhető, hogy a tanítandó modell már egy optimális kiindulási állapotból kezdjen el tanulni, vagy akár a sok paraméteres mély neuronhálónak kevesebb paraméterét kelljen hangolni a tanítás során. Leggyakrabban a gerincet szokták vagy előre tanítani, vagy már más feladatokon betanított súlyokkal alkalmazni. Ilyen előre betanított gerincekre jelentenek példát a mélytanuló neuronhálók jellemzőkinyerő konvolúciós rétegei: VGG16 [29], ResNet [30], MobilNet [31]. Ezt a folyamatot a szakirodalomban „Transfer Learning” néven említik, amikor egy feladatra korábban betanított hálót, egy másik feladatra alkalmaznak. Ebben az esetben egy objektumosztályozó hálót alkalmazunk objektum észlelésre. A kutatás során betanított hálók a fentiekhez hasonló módon egy előre betanított gerincből indultak ki.

A kísérletek során a modellek több konfigurációban kerültek futtatásra:

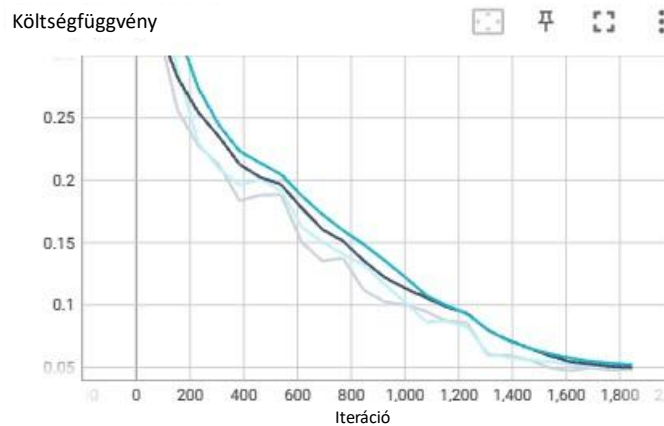
- Faster R-CNN 640×640 bemenettel:
 - ResNet50 gerinccel
 - ResNet101 gerinccel
- YOLOv3 darknet53 gerinccel:
 - 608×608 bemenettel
 - 320×320 bemenettel
- SSD 300×300 bemenettel:
 - VGG16 gerinccel
 - MobilNetv2 gerinccel
- FCOS 640×640 bemenettel:
 - ResNet50 gerinccel
 - ResNet101 gerinccel

A tanítás során a hibából számolt költségfüggvénye az tart a nullához. Abban az esetben, ha a modell teljesen rá tud illeszkedni a tanító mintákra, akkor tanítás végén a rendszer költségfüggvénye zéró. Viszont ilyenkor felmerül a kérdés, hogy a modell képes-e egy általa meg nem látott mintára is helyes becslést adni. Annak érdekében, hogy a tanítás végén egy

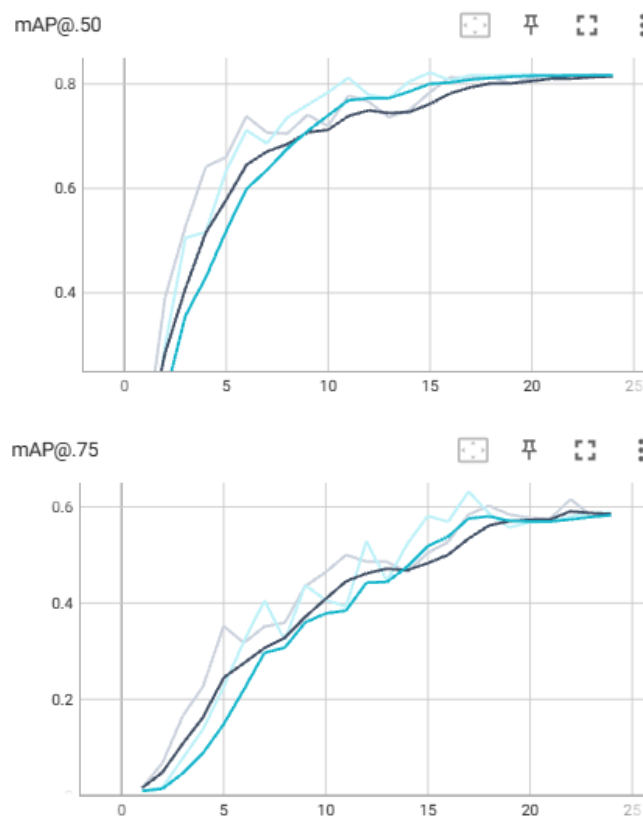
általános modellt kapjunk, a tanítás során időközönként ki kell értékeljük egy olyan minta halmazon, ami nem része a tanító minta halmaznak, ezt validációs halmaznak nevezik. Így lényegében két fontos mérő számról beszélünk, a költség függvény, amit minimalizálni kell, illetve egy validációs halmazon mert pontosság, amit értelemszerűen maximalizálni kell. A következőkben ezek a mérőszámok lesznek kirajzolva. Ezek az értékek tanítás során nem folytonosan változnak, hanem ugrálhatnak, ezért az ablakon majd az elsimított változatuk lesz kiemelve, de halványan látható a valós érték is.

A költség függvényeket ábrázoló grafikonokon a függőleges tengelyen a költség függvény átlagos értéke lesz látható, a vízszintes tengelyen az interakciók száma. Ez sok esetben el fog térni a modell variánsok között, abban az esetben, ha a modell nem ugyanakkora batch mérettel volt tanítva (egy iteráció alatt hány mintát tudunk egyszerre feldolgozni), hiszen nagyobb batch méret esetén kevesebb iterációs szükséges a arra, hogy ugyanannyi darab mintát feldolgozzunk. Az átlag pontosságot ábrázol grafikonokon a függőleges tengelyen a pontosság lesz kirajzolva 0-1 skálán, illetve a vízszintes skálán az epoch-ok száma lesz látható. A modellek minden epoch végén ki voltak értékelve a validációs halmazon. A fontos mérőszámok, mint például a költségfüggvény értéke és az átlagpontosság, TensorBoard könyvtár segítségével grafikonok formájában kerültek megjelenítésre. 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.

A Faster R-CNN hálók 24 cikluson keresztül kerültek tanításra. A költségfüggvények változását a 11. ábra, míg a validációs halmazon elért eredményeket a 12. ábra szemlélteti. A háló bemenetén a képek 640×640 méretre voltak leskálázva. Látható, hogy a modell nagyon nagy pontossággal volt képes észlelni az objektumokat 0,5-ös megbízhatósági szint mellett meghaladta a 80%-os, 0,75-ös ráta esetén pedig megközelítette a 60%-os pontosságot.



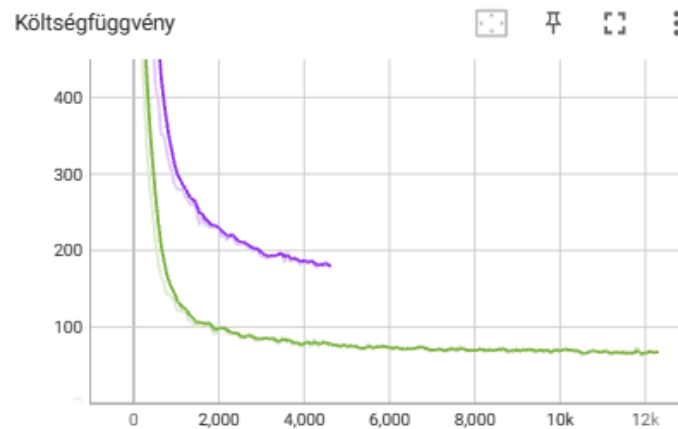
11. ábra: ResNet50 (világos kék) ResNet101 (sötét kék) gerinceket alkalmazó Faster R-CNN háló költségfüggvényének változása a tanítás során.



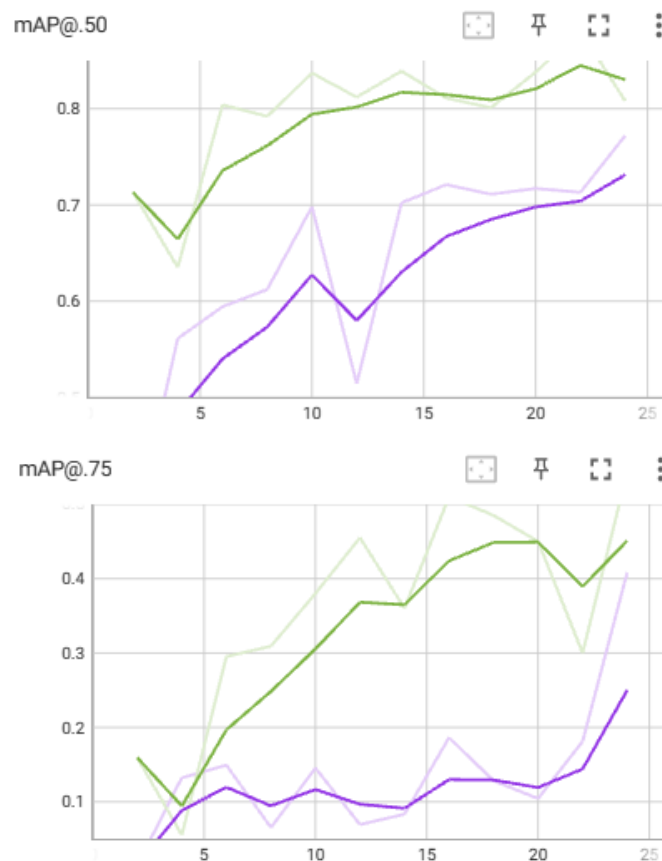
12. ábra: ResNet50 (világos kék) ResNet101 (sötét kék) gerinceket alkalmazó Faster R-CNN háló átlag pontosságának változása a tanítás során.

A YOLOv3 instabilnak bizonyult a tanítás során, ezért kisebb tanítási együtthatóval próbálkoztam a tanítás során, így hosszabb tanítási ciklusok kerültek alkalmazásra, azaz többször választott véletlenszerű mintát ciklusonként. A kisebb bemeneti felbontással rendelkező modell lényegesen kevesebb memóriát igényelt, mint a nagyobb felbontású változat, ezért egyszerre több mintát volt képes ciklusonként feldolgozni. A 320×320 bemenettel rendelkező modell tanítása során így 32, míg a 608×608 bemenettel rendelkező

modell 12 mintát kapott iterációkként. Mindkét háló 128 cikluson keresztül került tanításra. A költségfüggvények változását a 13. ábra, míg a validációs halmazon elért eredményeket a 14. ábra szemlélteti.



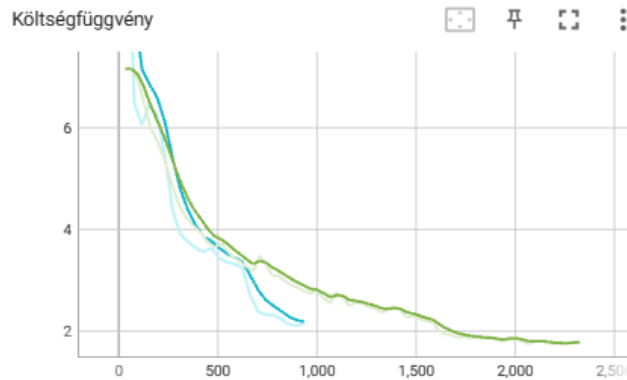
13. ábra: 320×320 (lila) 608×608 (zöld) bemenettel rendelkező YOLOv3 háló költségfüggvényének változása a tanítás során.



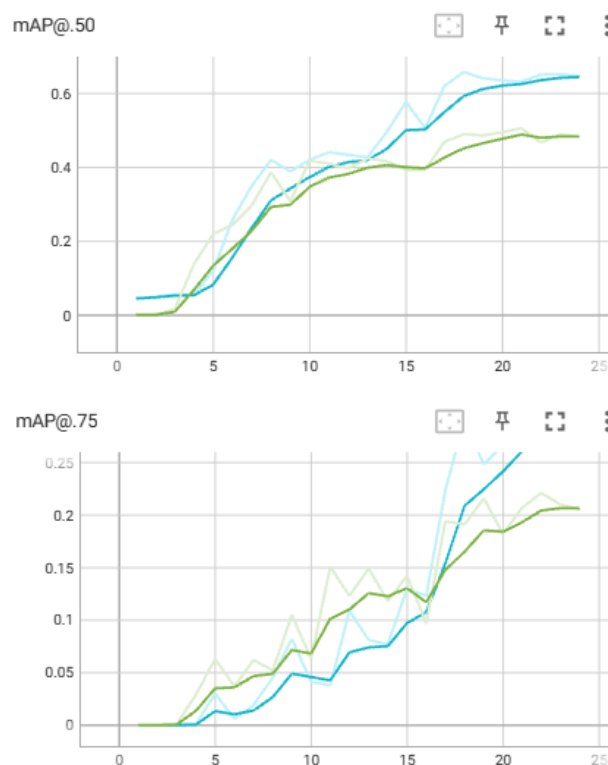
14. ábra: 320×320 (lila) 608×608 (zöld) bemenettel rendelkező YOLOv3 háló átlag pontosságának változása a tanítás során.

Az SSD háló tanítása során a MobilNet gerincet alkalmazó variáns lassabban volt képes megtanulni helyesen becsülni az objektumok határoló keretét, ezért hosszabb tanítási ciklusokat

kellett alkalmazni. Ugyanakkor a VGG16 gerinc több konvolúciós réteget tartalmaz, mint a MobilNet, ezért ebben az esetben kisebb batch méret került alkalmazásra, azaz 32 a VGG16, illetve, 64 a MobilNetv2 esetén. Az SSD hálót leginkább kis erőforrású eszközökön alkalmazzák, ezért legtöbbször kisebb felbontású bemenetet alkalmaznak ebben az esetben. Én is az eredeti kutatási eredményhez igazodtam és 300×300 nagyságú bemenetet alkalmaztam. Mindkét variáns 24 cikluson keresztül került tanításra. A költségfüggvények változását a 15. ábra, míg a validációs halmazon elért eredményeket a 16. ábra mutatja.

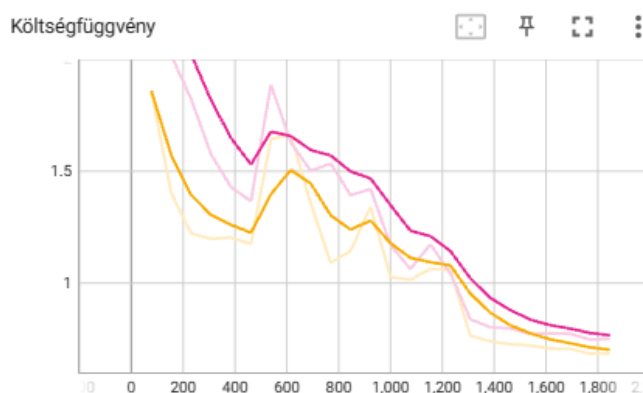


15. ábra: VGG16 (világos kék) MobilNetv2 (zöld) gerinceket alkalmazó SSD háló költségfüggvényének változása a tanítás során.

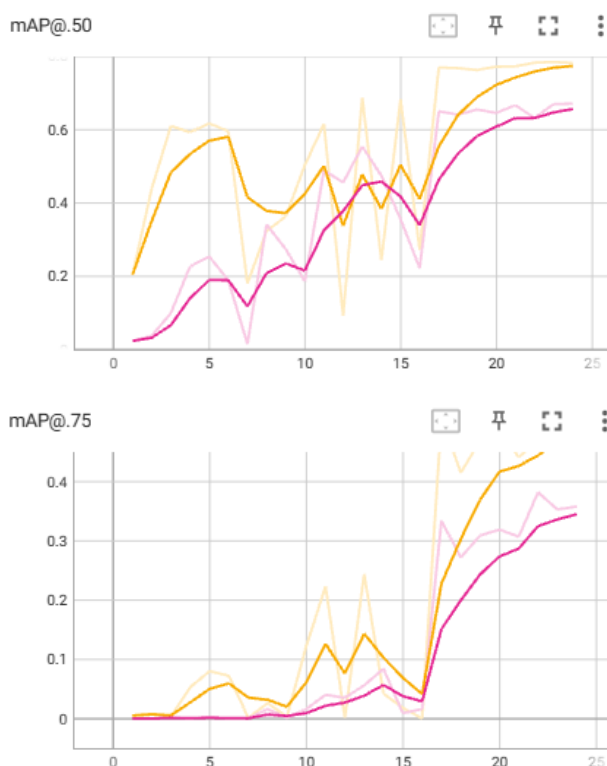


16. ábra: VGG16 (világos kék) MobilNetv2 (zöld) gerinceket alkalmazó SSD háló átlag pontosságának változása a tanítás során.

Az FCOS hálók 24 tanítási cikluson keresztül kerültek tanításra. A költségfüggvények változását a 17. ábra, míg a validációs halmazon elért eredményeket a 18. ábra mutatja. A háló bemenetén a képek 640×640 méretre kerültek leskálázva.



17. ábra: ResNet50 (sárga) ResNet101 (rózsaszín) gerinceket alkalmazó FCOS háló költségfüggvényének változása a tanítás során.



18. ábra: ResNet50 (sárga) ResNet101 (rózsaszín) gerinceket alkalmazó FCOS háló átlag pontosságának változása a tanítás során.

A kiértékelés során az a súlytényező került alkalmazásra, amelyik a legjobban teljesített a validációs halmazon. A kiválasztott súlyozók és az általuk elért eredmények a II. táblázatban kerültek összefoglalásra. A táblázat tartalmazza a hálók tanítható paramétereinek számát egész millióra kerekítve.

Az összes neuronháló közül a Faster R-CNN teljesített a legjobban, hiszen 0.50 konfidenciaszinten mindkét R-CNN variáns 0.817 (81,7%) átlag pontosságot tudott elérni. Pontosság szempontjából az R-CNN hálókat követte YOLOv3 architektúra, ami 0.772 (77,2%) (320×320-as bementtel) és 0.808 (80,8%) (608×608-as bementtel), illetve az FCOS, ami 0.782 (78,2%) (ResNet50 gerinccel) és 0.671 (67,1%) (ResNet101 gerinccel) átlagpontosságot ért el a validációs halmazon. A négy architektúra közül az SSD két variánsa teljesített a legrosszabbul ez csak 0.684 (68,4%) (VGG16 gerinccel), illetve 0.483 (48,3%) (MobilNet v2 gerinccel) átlagpontosságot ért el. Viszont azt is szem előtt kell tartani, hogy ennek az architektúrának van a legkevesebb paramétere, valamint amint a későbbiekben is kiderül a legtöbb mintát is képes feldolgozni másodpercként, ami különös előnyt jelent abban az esetben, ha egy beágyazott rendszeren szeretnénk alkalmazni.

II. táblázat: A modellek átlagpontossága a validációs halmazon. A „mAP@.50” és a „mAP@.75” az átlagpontosság, ahol a detektálási konfidencia nagyobb, mint 0.5, illetve 0.75. A „recall” a visszaidézőképesség.

Modell	Tanítási ciklusok	Tanítható param.	mAP@.50	mAp@.75	recall
Faster R-CNN (ResNet50)	24	41M	0.817	0.588	0.561
Faster R-CNN (ResNet101)	24	60M	0.817	0.582	0.555
YOLOv3 (320×320)	24	62M	0.772	0.408	0.454
YOLOv3 (608×608)	24		0.808	0.543	0.529
SSD (VGG16)	24	24M	0.648	0.286	0.416
SSD (MobilNetv2)	24	3M	0.483	0.206	0.327
FCOS (ResNet50)	24	32M	0.782	0.491	0.520
FCOS (ResNet101)	24	51M	0.671	0.358	0.443

5. A vizsgálatok eredményei

5.1. Az eredmények értelmezése

A III. táblázatban összefoglalt adatok alapján az objektum észlelő algoritmusok átlagpontossága megközelítőleg 0.11-el, azaz 11%-kal csökken a teszhalmazon a validációs halmazhoz képest. Ez egy természetes jelenség, hiszen a valós képek eltérő karakterisztikákkal rendelkeznek a szintetikus előállított képekhez képest. Ugyanakkor a teszhalmaz tartalmaz olyan harckocsi modelleket is, amelyek újabbak, így a WoT játékban nem találhatók meg, hiszen az csak a hidegháborús időszakig tartalmaz tankokat.

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

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

A validációs halmazon elért eredményekhez hasonlóan a Faster R-CNN háló teljesített a legjobban, az ResNet101 gerincet alkalmazó variáns 0.731 (73,1%), illetve a ResNet50 gerincet alkalmazó az 0.657 (65,7%) átlagpontosságot ért el. A YOLOv3 architektúrán tapasztalható a legnagyobb pontosságvesztés, a 608×608, illetve 320×320 méretű bemenetű variáns validációs halmazon 0.808 (80,8%), illetve 0.772 (77,2%) átlag pontosságot ért el, míg a teszt halmazon mindössze 0.576-t (57,6%), illetve 0.673-t (67,3%). Az FCOS háló ResNet50 gerinccel 0.633 (63,3%), ResNet101 gerinccel 0.549 (54,9%), illetve az SSD modell VGG16 gerinccel 0.712 (72,1%), MobilNetv2 gerinccel 0.534 (53,4%) átlagpontosságot ért el.

A III. táblázatban van egy rendhagyó eset, még pedig a MobilNetv2-t alkalmazó SSD háló, ugyanis a modell jobban teljesített a valós képeket tartalmazó halmazon, mint a validációs halmazon. Révén, hogy a validációs- és a teszhalmaz mintáinak a számossága nem jelentős, ebből az eredményből még nem lehet messzemenő következtetéseket levonni, ugyanakkor később érdemes lenne egy alaposabb vizsgálatnak is alávetni ezt a konfigurációt.

A III. táblázat tartalmazza a modellek sebeségét is, melyeket egy Nvidia RTX 3060 12 GB RAM-mal rendelkező videokártyán mértem. Mint ahogy korábban már szó volt róla, az SSD architektúra rendelkezik a legkevesebb tanítható paraméterrel és a legkevésbé komplex a vizsgált architektúrák közül. Látható, hogy ennek köszönhetően ezek voltak a leggyorsabb modellek, késleltetésük átlagosan 17 ms volt a VGG, illetve 19 ms a MobilNetv2 gerinccel végzett vizsgálatok esetén. Memóriaigényük is lényegesen kisebb volt, mint a többi architektúrának, a kisebbik SSD modellnek például csupán 33 MB, ami szintén az alacsonyabb számú tanítható paraméternek volt köszönhető. Ez az architektúra, főleg a MobilNetv2 gerincet alkalmazó, eredetileg mobileszközös alkalmazásokra került kifejlesztésre, ahol fontos szempont a csökkentett erőforrás igény. Ez hasznos lehet például az UAV felderítő rendszerek esetén is, főleg olyan konfigurációkban, ahol a detektálás közvetlenül a drón fedélzetén történik, ahol az erőforrások korlátozott mennyiségben állnak csak rendelkezésre, valamint fontos az üzemidő is. Ugyanúgy látható, hogy a horgonydobozos egylépéses hálók (YOLOv3, SSD) előnyt élveznek az FCOS modellel szemben, ami nem alkalmaz ilyen megoldást.

Fontos lehet még megemlíteni az, hogy ezeknek a detektoroknak a pontosság tovább növelhető, ha a szintetikus generált minták mellé valós mintákat is keverünk, vagy a már betanított modelleket finoman tovább hangoljuk egy kisebb méretű valós mintákat tartalmazó adathalmazon. A hangolásnak köszönhetően a modellek alkalmazkodnak a valós minták karakterisztikájához, mint a színvilág, felvételi zajok stb.

5.2. Eredmények gyakorlati értékelése

Amint a III. táblázatban látható a Faster R-CNN teljesített a legjobban. Olyan helyzetekben is képes volt objektumokat észlelni, és egy határoló keretet adni ezekre, ahol az objektum túlnyomó részben takarva volt, mint például a 19. ábra, ahol a felvételen a harcikocsit részben füst takarja. Azonban ahogy a többi teszt mintából kiderül a háló olyan objektumokat is harcikocsiként osztályoz, amelyek nem azok. Ilyen például a 20. ábra, ahol az előtérben található T-72-es harckocsi mellett a háttérben parkoló személygépkocsit is harcjárműnek azonosította a modell. Ez arra utal, hogy több ellen példára lenne szükség a tanító minták között, amelynek hatására kevesebb számban lennének jelen a hamis észlelések.

Egy másik észrevétel az R-CNN hálókkal kapcsolatban, hogy ezeknek a régió javasó hálójára még további tanításra szorul, hisz sok esetben egy objektumot több régióra bontott, aminek eredménye többszörös észlelés, erre példát a 21. ábra szemléltet.



19. ábra: Példa egy félig kitakart minta detektálására. Fent az elvárt kimenet, lent pedig a detektor kimenete látható.



20. ábra: Példa a Faster R-CNN érzékenységére. Fent az elvárt kimenet, lent pedig a detektor kimenete látható.



21. ábra: Példa többszörös detektálásra Faster R-CNN esetén. Fent az elvárt kimenet, lent pedig a detektor kimenete látható.

Ami ugyanakkor minden háló esetén megfigyelhető volt, hogy gyengén teljesítettek amikor az azonosítandó objektumok felüről kerültek észlelésre. Mivel ez nem csak egy hálóra volt jellemző, így a jelenség feltételezhetően a tanító adathalmaz hiányosságra utal. A tanító minták között ugyanis kevés volt az olyan kép, amelyeken az objektumok felülnézetből vannak megörökítve, illetve olyan esetekben, amikor felülnézetből szerepelnek, a fény, illetve környezeti elemek mintázata nem változik jelentősen. Ez annak köszönhető, hogy a WoT játék garázsjelenetében többnyire szabadon lehet forgatni a virtuális kamerát, de nem lehet változtatni a hátteret, és a fényviszonyokat, illetve az időjárás effekteket. A csatajelenetben változik ugyan a környezet, de az objektumot ilyenkor különböző vetített kijelző-elemek (HUD²¹) takarják ki részlegesen, így ezeknek a mintáknak a nagy részét is kellett szűrni a tanítóhalmaz összeállításakor. Ennek köszönhetően a neuronháló modellek gyengébben teljesítettek, amikor például felülnézetben mások voltak a fényviszonyok. Erre látható példa a 22. ábrán, egy harcikocsiról

²¹ HUD – Head Up Display (Ezek olyan kirajzolt elemek, amelyek nem a jelenethez tartoznak, hanem információkat közölnek, mint például sebesség, orientáció stb.)

szikrázó napsütésben készült felvételen. Megfigyelhető, hogy az alsó képen nem történt meg az objektum észlelése



22. ábra: Harckocsi felülnézetből napsütésben. Fent az elvárt kimenet, lent pedig a detektor kimenete látható.

5.3. A kutatás korlátai

Az elvégzett kutatás bizonyította egyrészt, hogy komoly létjogosultsága lehet a mélytanulósos neuronhálóknak a katonai felderítés támogatásában, a harcjárművek AI alapú detekciójában, ugyanakkor a fenti hibajelenségek arra is rávilágítanak, hogy hatékony modellek kidolgozása elképzelhetetlen az algoritmusok tanítóhalmazainak nagymennyiségű és megfelelő minőségű rendezett mintákkal való feltöltése nélkül.

A dolgozatban végzet mérések egyik fő hiányossága, az volt, hogy nem állt rendelkezésre elegendő forráskép. Míg a publikusan elérhető adathalmazok (MS COCO, PASCAL VOC, DOTA stb.), amiken ki szokták értékelni a neuronháló modelleket, több tízezres, vagy akár százezres nagyságrendben tartalmazznak mintákat, addig az általunk alkalmazott adathalmaz az csupán 846 képből tevődött össze, amiből mindössze 80 kép volt a valós minta. Így az egyik legfontosabb jövőbeli feladat az adathalmazok bővítése lesz.

A kutatás másik fontos tapasztalata, hogy a szintetikus generált minták is alkalmasak a tanító adathalmazban történő alkalmazásra, ugyanakkor a WoT játékszoftver az nem arra volt tervezve, hogy melytanuló algoritmusok számára generáljon tanító adathalmazokat. Ezért a minták gyűjtése egyrészt nem volt a leggyorsabb, másrészt azok minősége a játéktérre vetített HUD elemek miatt nem optimális, hiszen a játékosok nevei vagy a célzókereszt sokszor az objektumok fölé rajzolódik ki (lásd a 23. ábrán). Amiatt a gyűjtött mintákat szűrni kellett, illetve sokszor utólag módosítani, hogy az objektumok fölé rajzolt elemek ne zavarják a tanítást. Mindez rengeteg időt és emberi erőforrást igényel, ezért érdemes elgondolkodni olyan alternatív platform felkutatásában, vagy megoldás kidolgozásában, ami képes megkönnyíteni és jelentősen felgyorsítani a szintetikus minták gyűjtési folyamatát. Másik előnye lenne egy ilyen célra optimalizált 3D virtualizált környezetnek, hogy az akár magát a címkézési fázist is felgyorsíthatja, vagy akár teljesen automatizálhatja. Ezzel lényegesen fel lehet gyorsítani a mintagyűjtési folyamatot. A TKP2021-NVA-16 projekten belül, a „Mesterséges intelligencia, alternatív valóságok és rádiófrekvenciás eljárások védelmi célú alkalmazása” kiemelt kutatási terület keretében tevékenykedő, mesterséges intelligencia alapú, drónplatformra telepített felderítőrendszer fejlesztésén dolgozó kutatócsoport keretein belül már dolgoznak egy ilyen mintagenerálásra alkalmas szimulátor megvalósításán.



23. ábra: Zavaró játékelemek a mintákban.

A mérések során kizárólag színes RGB képeket használtam úgy a tanítás, mint a tesztelés során, ugyanakkor az UAV rendszerekben a kamerák más konfigurációkban is megjelenhetnek, mint például a monokróm (fekete-fehér), vagy akár az infra tartományban működő kamerák, illetve ezek kombinációja. Érdekes lehet ilyen eseteknek a vizsgálata is egy későbbi kutatás keretében, mind valós, mind szintetikus generált képeket tartalmazó, illetve vegyes tanítóhalmazok esetére.

6. Összegzés

Dolgozatban a mesterséges intelligencia alapú objektum észlelő algoritmusok szintetikus minták segítségével történő tanítására és az eredmények vizsgálatára fókuszáltam. A mérésekre Faster R-CNN, YOLOv3, SSD, illetve FCOS neuronháló modelleket használtam. A tanítómintákat WoT játékból kinyert képernyőfelvételekből állítottam össze, majd az ezzel betanított neuronháló modellek kiértékelését végeztem el egy valós mintákat tartalmazó adathalmazon. A vizsgálat legfontosabb eredménye, hogy az algoritmusok pontossága, illetve visszaidéző képessége minimális 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.

Ugyanakkor ezek megoldását követően a szintetikus minták segítségével betanított modellek akár alkalmazhatóvá válnak olyan UAV rendszerekben is, amelyeknek a feladatkörébe tartozik katonai felderítés, információszerzés, vagy éppen a célpontok autonóm felkutatása.

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.

7. Felhasznált irodalom

- [1] A. Németh, „Technical Dimensions of the Development of Unmanned Aerial Systems and Their Impact on Public Service Uses,” *AARMS*, pp. 149-163., 2018.
- [2] G. Udeanu, A. Dobrescu és M. Oltean, „Unmanned aerial vehicle in military operations,” *Sci. Res. Educ. Air Force*, %1. kötet18, p. 199–206, 2016.
- [3] A. Németh és K. Virágh, „Mesterséges intelligencia és haderő – A mesterséges intelligencia területei III. rész,” *Haditechnika*, %1. kötet56, %1. szám3, pp. 2-7., 2022.
- [4] A. Németh és K. Virágh, „Mesterséges intelligencia és haderő - Polgári alkalmazási lehetőségek V. rész,” *Haditechnika*, %1. kötet56, %1. szám5, pp. 2-7., 2022.
- [5] A. Németh és K. Virágh, „Mesterséges intelligencia és haderő - További polgári alkalmazási lehetőségek VI. rész,” *Haditechnika*, %1. kötet56, %1. szám6, pp. 2-7, 2022.
- [6] J. Terven és D. Cordova-Esparza, „A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond,” *arXiv preprint arXiv:2304.00501*, 2023.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] R. Girshick, „Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015.
- [11] 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.
- [12] K. He, G. Gkioxari, P. Dollár és R. Girshick, „Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017.
- [13] 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.
- [14] J. Redmon és A. Farhadi, „YOLO9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.
- [15] J. Redmon és A. Farhadi, „Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [16] A. Bochkovskiy, C.-Y. Wang és H.-Y. M. Liao, „Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [17] 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.
- [18] Z. Ge, S. Liu, F. Wang, Z. Li és J. Sun, „Yolox: Exceeding yolo series in 2021,” *arXiv preprint arXiv:2107.08430*, 2021.

- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] K. A. Özfuttu, „Generating synthetic data with game engines for deep learning applications,” 2022.
- [28] 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.
- [29] K. Simonyan és A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015.
- [30] 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.
- [31] 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.

Ábra jegyzék

1. ábra: IoU metrika [4].	7
2. ábra: R-CNN algoritmus lépései [5].	8
3. ábra: Fast R-CNN architektúra [6].	9
4. ábra: Faster R-CNN architektúra [7].	9
5. ábra: A YOLO algoritmus működési alapelve [9].	11
6. ábra: Közeli minták egy harcikocsiról különböző szögekből garázs jelenetben.	18
7. ábra: Tankok különböző háttérrel és helyzetekben csata közben.	18
8. ábra: Projekt létrehozása CVAT platformon.	19
9. ábra: Task létrehozása CVAT platformon.	20
10. ábra: Címkézés a CVAT platformon.	21
11. ábra: ResNet50 (világos kék) ResNet101 (sötét kék) gerinceket alkalmazó Faster R-CNN háló költségfüggvényének változása a tanítás során.	25
12. ábra: ResNet50 (világos kék) ResNet101 (sötét kék) gerinceket alkalmazó Faster R-CNN háló átlag pontosságának a változása a tanítás során.	25
13. ábra: 320×320 (lila) 608×608 (zöld) bemenettel rendelkező YOLOv3 háló költségfüggvényének változása a tanítás során.	26
14. ábra: 320×320 (lila) 608×608 (zöld) bemenettel rendelkező YOLOv3 háló átlag pontosságának a változása a tanítás során.	26
15. ábra: VGG16 (világos kék) MobilNetv2 (zöld) gerinceket alkalmazó SSD háló költségfüggvényének változása a tanítás során.	27
16. ábra: VGG16 (világos kék) MobilNetv2 (zöld) gerinceket alkalmazó SSD háló átlag pontosságának a változása a tanítás során.	27
17. ábra: ResNet50 (sárga) ResNet101 (rózsaszín) gerinceket alkalmazó FCOS háló költségfüggvényének változása a tanítás során.	28
18. ábra: ResNet50 (sárga) ResNet101 (rózsaszín) gerinceket alkalmazó FCOS háló átlag pontosságának a változása a tanítás során.	28
19. ábra: Félig takart mintára példa. Bal oldalt az elvárt kimenet, jobb oldalt a detektor kimenete.	32
20. ábra: Faster R-CNN érzékenységre példa. Bal oldalt az elvárt kimenet, jobb oldalt a detektor kimenete.	32

21. ábra: Többszörös detektálásra példa Faster R-CNN esetén. Bal oldalt az elvárt kimenet, jobb oldalt a detektor kimenete.	33
22. ábra: Harckocsi felülnézetből napsütésben. Bal oldalt az elvárt kimenet, jobb oldalt a detektor kimenete.	34
23. ábra: Zavaró játékelemek a mintákban.....	35

Táblázat jegyzék

I. táblázat: YOLO architektúrák összefoglalása [4].	14
II. táblázat: A modellek átlagpontossága a validációs halmazon. A „mAP@.50” és a „mAP@.75” az átlagpontosság, ahol a detektálási konfidencia nagyobb mint 0.5, illetve 0.75. A „recall” a visszaidézőképesség.	29
III. táblázat: A „mAP@.50” az átlagpontosság, ahol a detektálási konfidencia nagyobb mint 0.5. A „recall” a visszaidézőképesség.	30