Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

Tárnok Márton

ANCHOR-FREE JÁRMŰDETEKTÁLÁS KONTRASZTÍV OBJEKTUM SZŰRŐVEL

KONZULENS

Dr. Szemenyei Márton

BUDAPEST, 2023

Tartalomjegyzék

Összefoglaló

A járműdetektálás az elmúlt években a modern forgalomirányítási és -felügyeleti rendszerek fontos alapköve, amely a biztonsági intézkedéseket, a forgalom optimalizálását és az autonóm járművek irányítását segíti. A valós idejű analitikára való egyre nagyobb támaszkodás és az edge eszközök jelentős térnyerése miatt a járművek gyors és pontos észlelése a legkülönfélébb helyzetekben kiemelkedő fontosságúvá válik. A hagyományos képfeldolgozási módszereket félrevezethetik a környezet dinamikus változásai, a járművek különböző megjelenési formái, valamint a tájolás vagy a méretarányok eltérései, így valós környezetben nem megbízhatóak. A mélytanulás, mivel képes tanulni és általánosítani számtalan helyzetben, megoldásként szolgálhat ezekre a kihívásokra. A legtöbb mélytanulás-alapú detektor azonban anchor-boxokra - előre meghatározott térbeli szűrőkre - támaszkodik, amelyek korlátozhatják alkalmazkodóképességüket és sebességüket. Emellett időkritikus és helyszíni rendszerekben gyakran nincs lehetőség a kiterjedt számítási erőforrásokra, kevésbé toleránsak a késleltetésre. Ezen nehézségek megoldására dolgozatom elsősorban az anchor-free detektálás világában merül el, mely egy olyan paradigma, amely megkerüli az előredefiniáltság miatt keletkezett korlátokat, rugalmasabb és potenciálisan gyorsabb detektálást tesz lehetővé. Az anchor-free architektúrák számos benchmark-on state-of-the-art teljesítményt érték el mean Average Precision (mAP) tekintetében. Ezek a módszerek azonban jellemzően többfejes architektúrát alkalmaznak, ami a paraméterek számának jelentős növekedését eredményezi. Következésképpen kihívást jelent alacsony késleltetésű környezetben való alkalmazásuk.

Kutatásom során kidolgoztam egy egy-fejes, alacsony paraméter számú, anchor-free architektúrát, amely lehetővé teszi a szigorú időkorlátokkal és korlátozott erőforrásokkal rendelkező rendszerekben való alkalmazását. Ezt az architektúrát egy példánykódolási technika egészíti ki, melyben egy asszociatív embedding réteg és a kontrasztív tanulás kombinálásával a modell képes megkülönböztetni és elvetni az azonos objektumot detektáló, egymást átfedő dobozokat. Ez a technika felfogható úgy, mint egy tanulható, Non-Maxima Supression (NMS) mechanizmus. Az implementált eljárás teljesítményét a népszerű járműdetektálásra használt KITTI adathalmazon értékeltem ki. Az eredményeket részletekbe menően összevetettem a korábbi megoldásokkal sebesség és pontosság tekintetében, hogy teljes képet alkothassak az új eljárás teljesítményéről.

Abstract

Vehicle detection has long been an important cornerstone of modern traffic management and surveillance systems, aiding safety measures, traffic optimization and autonomous vehicle management. With the increasing reliance on real-time analytics and the significant proliferation of edge devices, fast and accurate detection of vehicles in a wide variety of situations is becoming paramount. Traditional image processing methods can be misled by dynamic changes in the environment, different vehicle appearances, and differences in orientation or scale, making them unreliable in real-world environments. Deep learning, with its ability to learn and generalise in a myriad of situations, can provide a solution to these challenges. However, most deep learning-based detectors rely on anchor-boxes - predefined spatial filters - which can limit their adaptability and speed. In addition, time-critical and on-site systems often lack extensive computational resources and are less tolerant to latency.

To address these difficulties, this thesis primarily dives into the world of anchor-free detection, a paradigm that circumvents the limitations due to predefinition of anchor-boxes, enabling more flexible and potentially faster detection. Anchor-free architectures have achieved state-of-the-art performance in terms of mean Average Precision (mAP) on several benchmarks. However, these methods typically use a multi-head architecture, which results in a significant increase in the number of parameters. Consequently, their application in low latency environments is challenging.

In this thesis, I have developed a single-head, low parameter count, anchor-free architecture suitable for use in systems with strict time constraints and limited resources. This architecture is complemented by an instance coding technique, in which, by combining an associative embedding layer and contrastive learning, the model is able to discriminate and discard overlapping boxes that detect the same object. This technique can be conceived as a learnable Non-Maxima Supression (NMS) mechanism.

The performance of the implemented technique was evaluated on the KITTI dataset used for popular vehicle detection. I compared the results in detail with previous solutions in terms of speed and accuracy to get a complete picture of the performance of the new technique.

1 Bevezetés

A számítógépes látás területén az elmúlt években a mély neurális hálózatok jelentős áttöréseket értek el, így radikálisan átalakítva a “látás" és "értelmezés" paradigmatikus kereteit. Mivel hatalmas adathalmazokban bonyolult minták és finom különbségek felismerésére alkalmasak, számos gyakorlati alkalmazás során kiemelkedő módszernek tekinthetőek. A modern GPU-k megjelenésével jelentősen megnőtt e hálózatok komputációs komplexitása. A legújabb architektúrák (DETR [1], DETA [2], YOLONAS [3]) a kiváló teljesítményre törekedve gyakran a pontosságot helyezik előtérbe, néha a hatékonyság rovására. Azonban, mint minden technológiai fejlesztés esetében, a pontosság és a számítási kapacitás között is fennáll egy alapvető kompromisszum. Olyan területeken, ahol a valós idejű analitika kiemelkedő fontosságú, a nagy bonyolult modern architektúrákkal hálózatok által igényelt számítási erőforrások jelentős szűk keresztmetszetet jelenthetnek.

Az egyre népszerűbb okos-város infrastruktúra erőteljesen tendál az intelligens rendszerek használata felé, melyek középpontjában a helyszíni (on-site) más néven edge computing áll. Ahelyett, hogy a nyers adatokat (kamerafelvételeket) egy központi szerverre továbbítanák, az edge eszközök helyben dolgozzák fel azokat. Ez a helyi feldolgozás megkerüli a sávszélesség korlátait, amelyek egyébként akadályoznák a nyers adatok továbbítását. Ugyanakkor ez a lokalizált technológia számítási szempontból hatékony, alacsony futási idejű neurális hálózatok kialakítását is megköveteli, biztosítva, hogy a valós idejű feldolgozás akadálytalan maradjon. Az anchor nélküli (anchor-free) (CornerNet [4], CenterNet [5], FSAF [6], FCOS [7]) detektorok alkalmazása lehetőséget ad a gyorsaságot előtérbe helyező objektumdetektálási módszerek felfedezésére. Ezek az architektúrák nem támaszkodnak előre definiált térbeli szűrőkre vagy anchor boxokra, a hagyományos konvolúciós modellekkel ellentétben (YOLO [8], RCNN [9]). Az anchor boxok kötöttségei és bonyolultsága nélkül az észlelési mechanizmus egyszerűbb lesz, ami gyorsabb előrejelzéseket tesz lehetővé, mely különösen előnyös az edge computing esetén.

Dolgozatomban egy újszerű architektúrát mutatok be, amely kihasználja az anchor nélküli észlelés erősségeit. Az egy-fejű (single-head) és alacsony paraméterszámú megoldás szigorú időbeli korlátokkal és korlátozott számítási erőforrásokkal rendelkező környezetre van szabva. Ez az architektúra a Fully Convolutional One-Stage Object Detection (FCOS [7]) modell elvein alapul és kiegészítem Centernet [5], és a Cornernet

[4], két kiemelkedő anchor-free hálózat koncepcióival, hogy robusztus és hatékony járműérzékelési mechanizmust kínáljon.

Bár az anchormentes architektúrák Non-Maxima Suppression (NMS) mentességet is ígérnek, a gyakorlatban ezek a megoldások gyakran esnek áldozatul dupla detekciónak. Bár a hagyományos NMS-technika potenciálisan enyhítheti ezeket a problémákat, számítási igénye miatt nem ideális időkritikus helyzetekben. Az elmúlt években egyre nagyobb figyelem övezte a tanulható NMS-módszerek detektáló rendszerekbe való beépítését. Ami azonban megkülönbözteti ezt a munkát a korábbi kutatásoktól, hogy kontrasztív tanulás segítségével váltja ki az NMS használatát. Ez a módszer önfelügyelt módon használja ki a kontrasztív tanulás erősségeit az azonos objektumhoz tartozó detekciók hatékony és eredményes megkülönböztetése és kiküszöbölése érdekében. Ezáltal a javasolt architektúra nem csak az NMS-mentes működés ígéretét valósítja meg, hanem mindezt számítás szempontjából hatékony módon teszi.

A munkám eredményei:

- Létrehoztam egy új erősen optimalizált anchor-free hálót, mely időkritikus környezetben is képes magas teljesítményre, pontosságra.
- Egy új kernel-alapú dekódolási módszert alkottam, amely képes kiszűrni a legmagabiztosabb predikciókat és kihasználja azok közvetlen környezetében, szomszédos cellák által eltárolt információkat is a pontosabb előrejelzés érdekében.
- Egy újszerű kontrasztívan tanulható NMS módszert hoztam létre, mely képes az azonos objektumot kódoló dobozok (dupla detekciók) szűrésére, és elhanyagolható számítási többlettel jár.

A dolgozat felépítése:

A leírást az elméleti háttér, azon belül a fontosabb alapfogalmak, mint a neurális hálózatok, anchoralapú és anchor-mentes objektumdetektálás, edge computing bemutatásával kezdem. A vonatkozó szakirodalom áttekintéseként bemutatom a munkám során releváns korábbi kutatásokat, módszereket. Ezt követően részletesen bemutatom az általam javasolt megoldásokat, a kernel-alapú dekódolást, illetve a kontrasztívan tanulható NMS módszerét. A modelltanítás leírása után az eredményeket értelmezem, majd egy összefoglalóval zárom a sort.

2 Elméleti háttér

2.1 Neurális hálózatok

A neurális hálózatok számítási paradigmája az agy biológiai felépítésén alapul. Legkisebb egységei a neuronok, melyek egy matematikai függvényt valósítanak meg a bemenetük és kimenetük között. A neuronok minden egyes bemenetéhez tartozik egy súly érték, melyek lineáris kombinációján valamilyen nem-lineáris, úgynevezett aktivációs függvény hívódik. A neuronok rétegekbe rendeződnek. A legelső réteget nevezzük bemeneti rétegnek, az utolsó réteget kimeneti rétegnek, míg a közöttük elhelyezkedő rétegeket rejtett rétegeknek hívjuk. A neurális hálózatok különböző felépítésűek lehetnek attól függően, hogy a rétegek milyen kombinációban helyezkednek el. Két domináns rétegtípust is megkülönböztetünk: a lineáris, illetve a konvolúciós réteget. A lineáris, más néven összekötött rétegekben az összes neuron kapcsolatban áll egymással, míg a konvolúciós rétegeknél a kimeneti neuronok csak közvetlen szomszédaikkal kapcsolódnak [10] [11].

A neurális hálózatok tanításának egyik módja a felügyelt tanítás. A felügyelt tanítás során az adatok címkézettek, tehát minden bemeneti tanítóadathoz meg van adva az elvárt kimenet. Minden adatra a háló predikciót ad, majd veszteség függvény segítségével megvizsgálja, hogy a kiszámított kimenet mennyire tér el az elvárt kimenettől. A tanítás során a cél a veszteség minimalizálása, így a háló pontosabb eredményt tud adni. A veszteség kiszámítása után a háló módosítja a súlyokat a hiba visszaterjesztésével (back propagation). A validáció során a súlyokat fixáljuk, nem történik visszaterjesztés, és azt vizsgáljuk, hogy a rendszer, hogy teljesít olyan adatokkal, amiken még nem tanult, ezzel szimulálva a működését valós adatokon. A neurális hálózatok használhatóak regressziós és osztályozási feladatokra egyaránt.

A számítógépes képfeldolgozásban jellemzően konvolúciós neurális hálózatokat (Convolutional Neural Network, röviden CNN) alkalmaznak, melyek konvolúciós rétegekből épülnek fel, mert ezek kevesebb paraméterrel dolgoznak az összekötött hálózatoknál [12], illetve hatékonyan kezelik a képek térbeli jellemzőit. A bemeneti kép háromdimenziós formában jelenik meg, a képelemek a neuronokhoz köthetőek. Az olyan alapvető képi jelenségeket, mint a kontúrok és sarkak, az első rétegek jelölik ki, az összetettebb tulajdonságok azonosításáért már mélyebb rétegek felelősek [11]. A konvolúció során egy kernel (általában 3x3-as mátrix), csúszóablakszerűen végigmozog

a kép felett. A réteg feladata valamilyen absztrakt tulajdonság megtanulása. Az egyes tulajdonságokat reprezentáló térkép (activation map) egy új eleme az eredeti kép és a kernel pontonként vett szorzatának összegéből határozható meg. A konvolúciós neurális hálózatokba le- és felskálázó (pooling) rétegek is beépíthetőek. A MaxPooling réteg célja a bemeneti réteg mintavételezése, csökkentve annak dimenzionalitását. Itt is egy kernel segítségével határozzuk meg az adott terület legnagyobb értékét, ezt fogjuk a következő rétegben reprezentálni. A MaxPooling csökkenti a számítási költségeket, illetve magasabb szintű absztrakcióra van lehetőségünk általa.

2.2 Objektum detektálás

Az objektumdetektálás a számítógépes látás egyik kulcsfontosságú területe, amely a képen vagy videón belül az egyes objektumok helyének meghatározását és azonosítását jelenti. Ez túlmuta a felismerésen, meghatározza az objektumok elhelyezkedését és határait a képkockán belül. Az objektumdetektálást számos területen széleskörűen alkalmazzák. Használják többek között autonóm járművekben a gyalogosok és az akadályok felismerésére, biztonsági rendszerekben megfigyelésre, betörések felismerésére, a kiskereskedelemben leltárkezelésre, az egészségügyben pedig orvos diagnosztikai eszközökben is.

A tárgydetektálás területén alkalmazott neurális hálózatok különféle architektúrák széles skáláját ölelik fel, melyek specifikus előnyökkel rendelkeznek. A konvolúciós neurális hálózatok (CNN) kivételes képességükkel tűnnek ki pixel-alapú adatok feldolgozásában és komplex képi jellemzők extrakciójában. A régióalapú CNN modellek (R-CNN [9]) és annak evolúciós variánsai, mint a Fast R-CNN[13] és a Faster R-CNN[14], a régiójavaslati módszerek integrálásával bővítenek az alap CNN architektúrákon, így növelve a detektálási hatékonyságot.

A Single Shot MultiBox Detector (SSD) [15] és a You Only Look Once (YOLO) [8] további népszerű neurális hálózatok, amelyeket objektumfelismerésre használnak. Ezeket a modelleket valós idejű objektumdetektálásra tervezték, mivel céljuk, hogy a kép egyetlen átfutásával detektálják az objektumokat, ami gyorsabbá és alkalmasabbá teszi őket valós idejű alkalmazásra.

2.2.1 Anchorok

Az anchor boxok, más néven prior boxok, az objektum detektáló algoritmusok szerves részét képezik. Ezek előre meghatározott alakzatokat biztosítanak, amelyeket a modell alapul használ a képen lévő objektumok helyének és méretének előrejelzéséhez. Ezeket az anchor dobozokat kiindulási pontként használva a modell azt tanulja meg hatékonyan, hogy hogy igazítsa őket a valódi objektumhatárokhoz. Az "anchor boxok" koncepciója eredetileg a Faster R-CNN publikációban [14] került bemutatásra, mint egy eszköz a diverz méretű és formájú objektumok effektív kezelésére. Az anchor boxok alkalmazásának elsődleges motivációja a tárgydetektáló algoritmusok detektálási teljesítményének javítása volt. Az optimális anchor boxok megtalálása a végeredmény szempontjából kritikus feladat. Ez magában foglalja a helyes számú, méretű és arányú anchor box kiválasztását, amelyek meghatározása erősen összefügg az adott alkalmazási terület jellemzőivel. Az anchor boxok ideális paramétereinek meghatározására gyakran alkalmaznak klaszterezési technikákat, például a K-means módszert [16], annak érdekében, hogy olyan anchorokat azonosítsanak, melyek a lehető legpontosabban illeszkednek az adathalmazban szereplő dobozok eloszlásához.

Az objektumdetekciós modellek regressziós technikákat használnak az anchor boxok finomítására, hogy azok minél közelebb illeszkedjenek a valóságos, ún. ground truth boxokhoz. Azáltal, hogy a neurális hálózat tanulási folyamat során képes adaptálódni ezekhez az eltolásokhoz, javítja annak képességét, hogy különböző méretű és formájú objektumok pozícióját pontosabban becsülje meg.

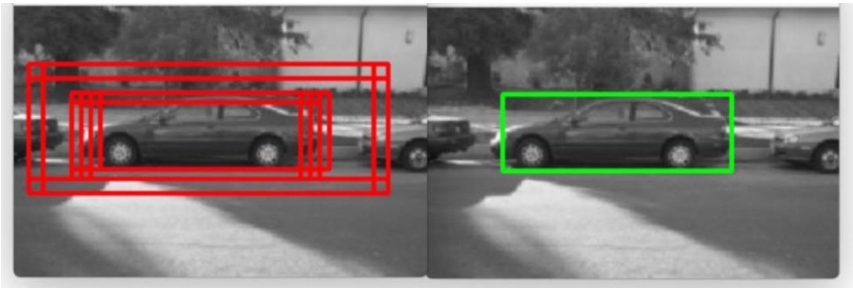
Az anchor boxok alkalmazásának előnyeivel és korlátaival kapcsolatosan számos megfontolás merül fel. Egyik fontos előnyük, hogy strukturált keretet nyújtanak a változó méretű és arányú objektumok lokalizációs problémáinak kezelésére. A modell ezen inicializációs pontok segítségével képes növelni a detektálási teljesítményt. Azonban az anchor boxok használatával számos technikai kihívással is szembe kell nézni. Egy kiemelkedő probléma az optimális anchor paraméterek – mint a mennyiség, méret és képarány – meghatározásának komplexitása. Egy nem optimálisan választott anchor konfiguráció degradálhatja a detektálási teljesítményt. Továbbá, a több anchor használata növeli a számítási igényeket, és priori információt injektál a modellbe. Ráadásul, azokban az esetekben, amikor a valósághű (ground truth) bounding boxok jelentősen eltérnek az előre definiált anchor boxoktól, az anchor-alapú modellek pontossága és lokalizációs képessége kompromitálódhat [5].

2.3 Anchor-free objektum detektálás

Az anchor-mentes objektumdetektálás jelentős változást jelent a hagyományos anchor-alapú módszerektől. Az előre meghatározott anchorok- elhagyásával az anchor-free technikák természetesebb megközelítést alkalmaznak, és közvetlenül detektálják az objektumokat. Az anchor-free detektorok két csoportra bonthatók: a sűrű predikciós (dense predict) és a kulcspont-alapú módszerek (key-point based) [17]. Míg az előbbi a kép diszkrét cellákra bontásával minden egyes cellára predikciót készít, az utóbbi specifikus karakterisztikus pontokra, mint az objektum középpontja vagy annak sarokpontjai, koncentrál. Az anchor-mentes stratégiák számos előnnyel rendelkeznek, többek között adaptív képességekkel a diverz méretű objektumok precíz azonosítására. Az előre meghatározott anchorok hiánya ésszerűbb és gyorsabb észlelési folyamatokat is eredményezhet, csökkentve a számítási ráfordítást. Ugyanakkor ezek a módszerek saját specifikus kihívásokkal is rendelkeznek, mint például a redundáns detekciók, ami további post-processzálási lépések bevezetését teszi szükségessé.

2.3.1 Non-maxima suppression

A Non-Maxima Suppression (NMS) az objektumdetektálás egyik alaptechnikája, amelyet a felesleges, egymást átfedő dobozok kiküszöbölésére alkalmaznak. Az Intersection Over Union (IOU) metrika felhasználásával az NMS értékeli a dobozok közötti átfedést, megtartva a legnagyobb detektálási megbízhatóságú dobozokat és kizárva a kisebbeket. Bár a pontosság eléréséhez elengedhetetlen, az NMS természeténél fogva számítási időveszteséget okoz. Ez a késedelem elsősorban abból adódik, hogy az NMS alkalmazása előtt dekódolni kell a dobozokat, így az eljárás iteratív jellegű. Továbbá, ha GPU-n hajtják végre, a folyamat párhuzamosításának nehézsége miatt még kevésbé lesz hatékony.



1. ábra: Az NMS működése [18]

2.4 Önfelügyelt tanulás

Az önfelügyelt tanulás (self-supervised learning [19] egy olyan paradigma a gépi tanuláson belül, ahol a modellt explicit kézi címkék nélkül tanítják, kihasználva az adatok inherens struktúráját. A számítógépes látás területén az önfelügyelt tanulás egyre nagyobb teret nyert [20], mivel hatalmas mennyiségű címkézetlen vizuális adat áll rendelkezésre, és lehetőség van arra, hogy ezeket az adatokat értékes modelltanításhoz használjuk fel. A manuális annotációkra alapozott módszerek helyett a képfeldolgozásban alkalmazott önfelügyelt tanulási technikák képesek térbeli kontextus, színinformáció és időbeli szekvencia alapján tanítani a modelleket. Az egyik legkiemelkedőbb módszer a kontrasztív tanulás [21]. A kontrasztív tanulás során a modellt úgy képezzük ki, hogy különbséget tegyen a hasonló és a különböző adatminták között, a hasonló reprezentációkat összehúzza, a különbözőket pedig szétválasztja a látens térben. Ez a megkülönböztetés, amelyet gyakran pozitív párok (hasonló minták) és negatív párok (eltérő minták) összehasonlításával érnek el, gazdag képzési jelet biztosít, amely lehetővé teszi a modell számára, hogy címkézetlen adatokból robusztus reprezentációkat tanuljon meg.

2.5 Edge computing

Az edge computing [22] egy számítási paradigma, melyben a számításokat a központosított adatközpontoktól a hálózat perifériájára, az adatgenerálás helyszínéhez és a felhasználók közvetlen közelébe mozgatja. Ezt a paradigmaváltást a valós idejű feldolgozás, a késleltetés minimalizálása és a sávszélesség hatékony kihasználásának szükségessége motiválja. Az alkalmazások gyorsaságát és reszponzivitását is javítja, így az edge computing elengedhetetlen az olyan helyzetekben, ahol azonnali visszajelzésre van szükség, anélkül, hogy hatalmas mennyiségű nyers adatt küldjenek a hálózaton keresztül a központi szerverekre.

3 Korábbi munkák

3.1 Anchor-free objektum detektálók

3.1.1 Cornernet és Centernet

A CornerNet [4] egy egylépéses, end-to-end, anchor-mentes objektumdetektáló modell, mely a dobozok definiálását azok bal felső és jobb alsó sarkainak prediktív helyzetével valósítja meg. E célból a neurális architektúra minden objektumosztály számára két prediktív sűrűségi térképet hoz létre: egyiket a bal felső, a másikat a jobb alsó sarkok azonosítására. Ezek a térképek kifejezik a sarkok jelenlétének valószínűségét az egyes pixel pozíciókban, az adott kategóriák kontextusában. Továbbá, a rendszer minden egyes sarokhoz egy embedding vektort is hozzárendel, amely segíti az azonos objektumhoz tartozó sarkok összepárosítását. Azok a sarkok tartoznak össze, melyeknek az embedding-jeik közötti távolság minimális.

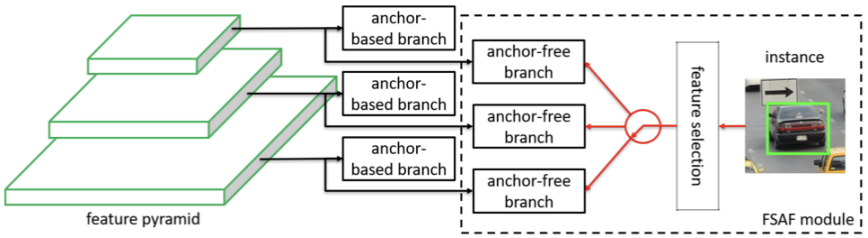
A CornerNet alkotói Newell és munkatársainak mintájára (2017) [23] 1 dimenziós embeddingeket használtak. Az embedding teret kialakító veszteségfüggvény két komponensből állt: egy húzótényezőből (pull) és egy tolótényezőből (push): A "húzó" hibafüggvényt a sarkok csoportosítására alkalmazták, a "toló" hibafüggvényt pedig a sarkok megkülönböztetésére [23]. A CornerNet 42,2 \% AP-t ért el az objektum detektálás kiértékelésére gyakran használt MSCOCO [24] adatset-en.

$$L_{pull} = \frac{1}{N}\sum_{k=1}^{N}\left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2\right] \quad L_{push} = \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{\substack{j=1 \\ j\neq k}}^{N}\max\left(0, \Delta - |e_k - e_j|\right)$$

A Centernet alapötlete hasonló a Cornernet-hez, de két pont helyett egyetlen középponttal írja le az objektumot, és ennek a pontnak a segítségével jelzi előre a tárgy szélességét, magasságát és kategóriáját. A pontalapú reprezentáció csökkenti a modellhez szükséges paraméterek számát, így a modell hatékonyabban és gyorsabban tanítható és tesztelhető. A modell két predikciós fejet alkalmaz. Az egyiket a konfidencia valószínűségi térkép becslésére használják, a másik fejet pedig - más detektorokhoz hasonlóan - a box koordinátáinak és offszetjeinek előrejelzésére tanítják be, melyek az első fej által prediktált box középponthoz tartoznak.

3.1.2 Az FSAF módszer

Az FSAF [6], avagy Feature Selective Anchor-Free, egy olyan komponens, amelyet a feature-pyramid network-kel (FPN) [25] rendelkező, egylépéses objektumdetektorokban használnak. Az FSAF modul a tanítás során dinamikusan kiválasztja a legmegfelelőbb piramis szintet minden egyes példányhoz. Minden egyes példány minden egyes szinten számításon esik át. Ez a számítás magában foglalja az osztályozási (focal) hibafüggvény és a regressziós (IoU) hibafüggvény kiértékelését. Ezt követően a két veszteség minimális összegét mutató szint kerül kiválasztásra az adott példány feletti felügyeletre. A megfelelő piramis szint dinamikus kiválasztása, amely a méreten túl a példány tartalmától is függ, lehetővé teszi a pontos és hatékony objektumfelismerést különböző méretarányok és képarányok esetén.



2. Ábra: Az FSAF koncepciója

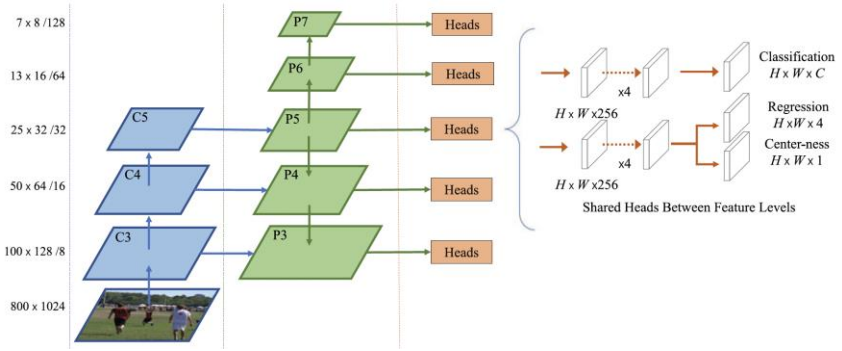
Az ábra az FSAF általános koncepcióját szemlélteti, ahol egy anchor-free ágat helyezünk el a piramis minden szintjén, amely egy osztályozási és egy regressziós alhálózatból áll. Amelyik szint a legkisebb hibát adja, arra lesz visszapropagálva a veszteség. Az FSAF modul dolgozhat függetlenül vagy párhuzamosan az anchor-alapú ágakkal.

3.1.3 Az FCOS módszer

Az FCOS (Fully Convolutional One-stage Object Detection) [7] egy anchor-mentes objektumdetektálási architektúra, amely a sűrű predikcióra (dense prediction) támaszkodik, tehát minden kimeneti gridcella egy lehetséges dobozt határoz meg. Feature Pyramid Network-öt (FPN) használ, kompatibilis ismertebb alternatíváival is, mint például a PANnet [26], a NAS-FPN [27] vagy a BiFPN [28]. Az FCOS architektúra két elkülönített ággal rendelkezik a fej moduljában: az egyiket a lokalizációs feladatokra, míg a másikat az osztályozási műveletek végzésére használja. Az eredeti FCOS publikációban [7] további konvolúciós szűrőket építettek be a fejbe, lehetővé téve az osztályok és a lokalizáció külön tanulását. Több fejet is alkalmaztak, és a dobozokat

különböző fejméretekre kódolták, hatékonyan felosztva a különböző méretű dobozok felismerésének felelősségét.

Az FCOS egyik legfontosabb újítása a center-ness score bevezetése, amely a lokalizációs fejből ered, és célja, hogy megoldja az objektum középpontjától távol eső, gyenge előrejelzések problémáját. Megfigyelték ugyanis, hogy a célpont határoló boxának középpontjához közelebbi helyek általában megbízhatóbb előrejelzéseket eredményeznek. A centerness scoret arra használják, hogy a tárgy középpontjától távolodó előrejelzéseket kevésbé vegyék figyelembe, és így javítsák a felismerési teljesítményt. A center-ness score előnye abban nyilvánul meg, hogy a felismerési eredmények jelentősen javulnak, miközben a számítási ráfordítás csak kis mértékben növekszik.



3. Ábra: FCOS felépítése

3.2 Tanulható NMS

Az elmúlt években számos megközelítés alapján próbálkoztak egy tanulható NMS kifejlesztésére az objektumdetektálás javítása érdekében.

A Soft-NMS [29] a hagyományos NMS-technika továbbfejlesztése, amely megtartja az eredeti NMS érdemeit, de bevezet egy soft supression-t a fals pozitív eredmények csökkentése érdekében, javítva ezzel a detektálási teljesítményt. A modellük adaptív mechanizmust vezet be, amely felülmúlják a tradicionális NMS stratégiát, ezáltal jobb általános észlelési pontosságot eredményez. A GossipNet [30] kollektív objektumkészlet-feldolgozást alkalmaz a duplikátumok kiküszöbölésére, lehetővé téve az észlelések közötti kommunikációt, a reprezentációk finomítása érdekében. Bár a pontossága az NMS-t idézi, bonyolult hálózati felépítése magas számítási igényeket eredményez futásidőben, ellentétben a mi egyszerűsített megközelítésünkel. A RelationNet [31] attention [32] mechanizmust használ az objektumok közötti kapcsolatok

megértésére a megjelenési és geometriai jellemzők kollektív feldolgozásával. Bár innovatív, a felépítése jelentősen komplex. Az én megoldásom ezzel szemben egyszerűbb, amely a pontosság feláldozása nélkül biztosítja a hatékony feldolgozást.

4 Javasolt megoldások bemutatása

4.1 A feladat megfogalmazása

A kutatás célja egy olyan detektáló hálózat kifejlesztése volt, amely alacsony késleltetésű környezetben is megbízhatóan teljesít, és jobb eredményeket képes produkálni, mint a sokszor használt YOLO. Ehhez az kell, hogy a hálózatban csökkentsük a súlyszorzásokon kívüli plusz számításokat úgy, hogy azok minél kevesebb időt vegyenek el. A tudományos munka fókuszában tehát olyan univerzális megoldások kidolgozása áll, melyek a backbone-tól függetlenül, általánosan képesek fokozni a hálózat teljesítményét.

Összességében a dolgozat bemutat egy új, anchor-free FCOS alapú hálózatot, amely modifikált fejjel, új dekódolási módszerrel és teljesen neurális alapon tanulható NMS-el rendelkezik, ígérve gyors és pontos járműdetektálást alacsony késleltetésű környezetekben.

4.2 Megoldások

A fejezetben a következők során bemutatom az alacsony futáskörnyezetre tervezett hálót, annak tervezési döntéseit, és az általam javasolt megoldásokat, a munka során felmerült nehézségeket.

A feladat megoldására az 'anchor-free' hálózatok kínálnak ígéretes megoldást, hiszen ezek mentesek az anchorok plusz számításaitól. Ezen belül az FCOS architektúrára esett a választás, mivel ez sűrű predikciót használ, ennek köszönhetően több cella is tartalmazza az információkat ugyanazon objektumról. Jelentős lehetőségeket láttam a hálózat predikciós idejének javítására a módosíthatósága miatt.

Az első célkitűzés egy olyan dekódoló kialakítása volt, amely minimalizálja a hálózat terhelését és szűrni enged a boxok között. Ennek keretei között egy másik, anchor-free eljárások során már bevált dekódolási módszert próbálok ki az FCOS-on.

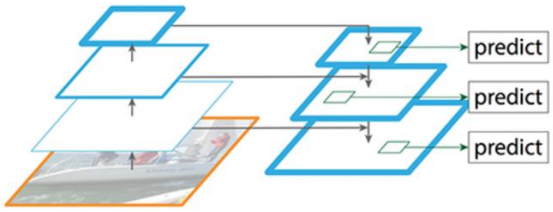
Második célom egy tanulható NMS kialakítása, mely szintén csökkenti a hálózat futás idejét. A dolgozatomban elsőként sikerült kontrasztív tanulási megközelítéssel egy tanulható NMS megoldást kifejleszteni. Ennek lényege, hogy a grid teret szeparálni kívánja, és tanulhatóvá teszi, hogy egyetlen objektumot több box is detektálhasson. A dolgozat bemutatja az új hálózatokat, és elemzi azokat sebességük, pontosságuk, a paraméterszámuk szempontjából, valamint vizsgálja összefüggéseiket.

4.3 Hálózat felépítése

Az objektumdetektáló konvolúciós neurális hálózatok általában három fő komponensből épülnek fel: a feature extractor-ból (más néven "backbone"), a nyakból (neck) és a predikciós fejből (head).

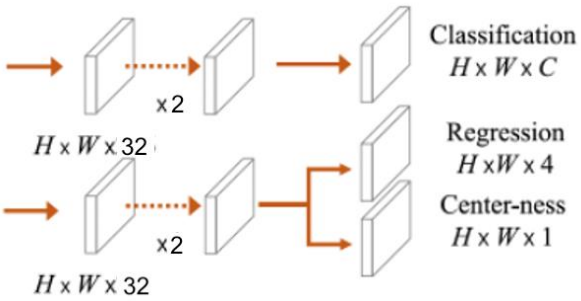
Backbone: A backbone a bemeneti képből kivonja a legfontosabb jellemzőket. Fontos, hogy elegendő számítási kapacitással rendelkezzen a feladathoz, de ne legyen túlságosan nagyméretű, hogy ne növelje túlzottan a számítási költségeket és az időigényt. Ebben az esetben egy kb. 500000 paraméteres tisztán konvolúciós rétegekből és max pooling rétegekből álló architektúrát választottam.

Nyak: A nyak a backbone által kinyert jellemzőket dolgozza fel és továbbítja a predikciós fejnek. Hierarchikusan összekapcsolja a különböző mélységű jellemzőket, lehetővé téve a hálózat számára, hogy aggregálja az információt a különböző méretű és tulajdonságú objektumokról. Számos változata létezik, mint például az FPN (Feature Pyramid Network), vagy BiFPN vagy a PanNET. Ezek közül a sima FPN-t választottam, mivel gyors és viszonylag egyszerűen megvalósítható. A modellben összesen négy szintű FPN-t alkalmaztam.



5. Ábra: Az FPN felépítése. Két ágával hierarchikusan összekapcsolja a jellemzőket [33]

Fej (Head): A predikciós fej a hálózat azon része, amely közvetlenül előállítja a kívánt kimenetet. Az eredeti FCOS architektúrában több predikciós fejet alkalmaznak, mindegyiket egy külön FPN kimenethez kötve, ahogy a 3. ábrán látható. Amikor ezt a megközelítést teszteltem, azt tapasztaltam, hogy a hálózat jelentősen lelassul, mivel a különböző szintű detekciók kezelése többletmunkát jelent mind a tanítás, mind kiértékelés során. Az FCOS fejében található súlyok szintén hozzájárulnak a lassuláshoz, ezért a saját modellben csökkentettem a fejparaméterek számát: a 4x64x64 mérettől a kompaktabb 2x32x32 méretre váltottam. Ezzel a változtatással a hálózat jelentősen gyorsabb lett, miközben továbbra is képes volt hatékonyan detektálni az objektumokat.



6. Ábra: Az általam használt átparaméterezett fej struktúrája

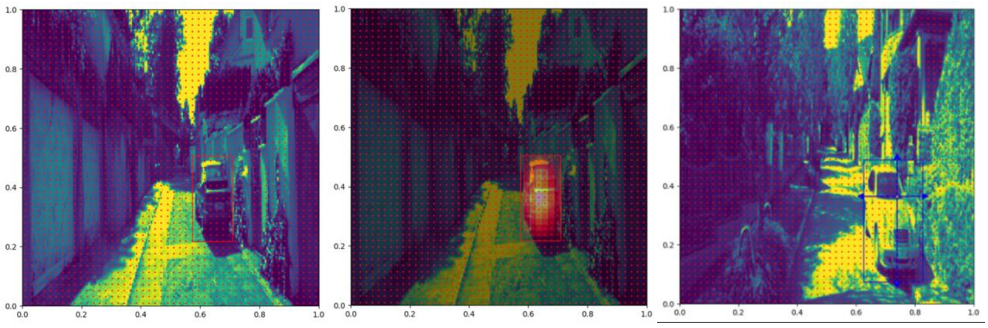
A fej működési elve: A predikciós fej grides kimenetet használ, amely a képet 48x48 részre osztja. Minden cella egy objektumot reprezentáló dobozt prediktál. Ha egy cella egy doboz területére esik, az adott cella felelős lesz az objektum detektálásáért. Mivel egy objektumot több cella is kódol (one-to-many), többféle predikció készülhet ugyanarra az objektumra, viszont ezzel együtt jár a kihívás is: a sok készült predikció közül ki kell választani a legmegfelelőbbet.

4.3.1 Bemenet előfeldolgozása

Míg az eredeti megközelítés több fej használatán alapult, én csak egyetlen fejet alkalmazok, ezért módosítottam az adatok előkészítési módszerét. Minden cellának kódolnia kell a saját dobozához tartozó négy koordinátát. Az FCOS-ban ez a távolságot jelenti a cella közepétől a doboz négy oldaláig (ezek az úgynevezett ltrb értékek, melyek a left, top, right, bottom koordinátákat jelentik). Emellett a celláknak az objektum osztályát is kódolniuk kell, ami one-hot kódolással történik. Továbbá az úgynevezett "centerness" értéket tárolják még, mely a következő képlettel számolható ki:

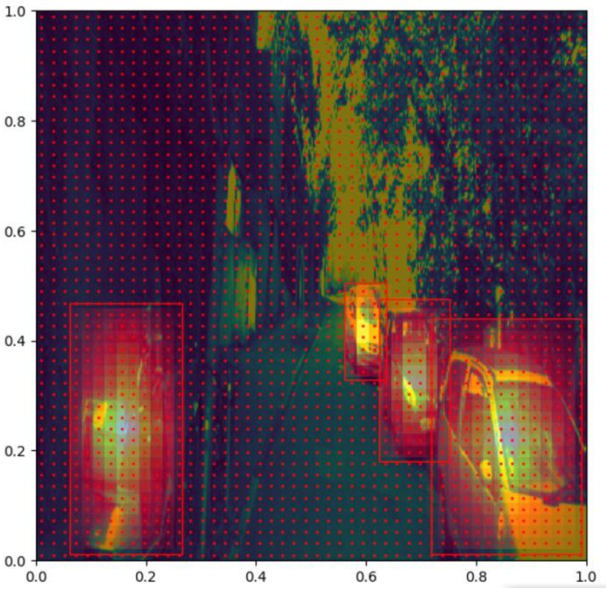
$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$

Vizualizálva ez azt jelenti, hogy egy cella annál magasabb "centerness" értékkel rendelkezik, minél közelebb helyezkedik el az objektum középpontjához.



7. Ábra: Elkódolt értékek vizualizációja. Balra az eredeti doboz látható, középen a centerness érték, jobbra a ltrb érték vizualizációja.

A sűrű predikció használatából fakadó kihívás, hogy az átfedő dobozoknál melyikre kódoljuk a cellát. Az eredeti publikáció szerint átfedés esetén a kisebb objektumot preferálták, ez viszont problémákat vet fel. Ha egy kisebb doboz például elfedi egy nagyobb doboz közepét, akkor a nagyobb dobozt elveszítjük a dekódolás során. Ezt a problémát orvosolja az általam bemutatott módszer: a cella mindig azt a dobozt kódolja, amelyiknél a "centerness" értéke a legmagasabb. A "centerness" érték megmutatja, mennyire játszik központi szerepet egy cella egy adott objektumon belül. Az ilyen módon történő kódolással nemcsak precízebb predikciókat érhetünk el, hanem minimalizáljuk az objektumok elvesztésének kockázatát is.

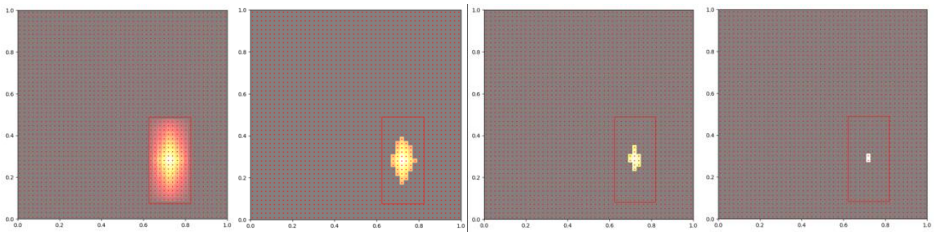


8. Ábra: Átfedő objektumok kódolása centerness-szel

4.4 Kerneles lokális maximum keresés

Az FCOS kimenetén minden egyes grid cella tartalmaz egy-egy predikciót, amelyek az objektumok koordinátáira, osztályára, és centerness értékére vonatkoznak. A dekódolás során azt szeretnénk elérni, hogy mindegyik objekumhoz pontosan egyetlen cellát bontsunk ki. Amennyiben több cellából dekódolunk bounding boxokat ugyanarra az objektumra, redundáns és pontatlan eredmények keletkeznek, ami az NMS számára további műveleteket igényel. Az objektumdetektáló rendszerekben - ahogyan az FCOS eredeti tanulmányában is - konfidencia-küszöbértékeket alkalmaznak a nem kívánt predikciók kiszűrésére. Ez a módszer nem megbízható, mivel nincs olyan univerzális

küszöbérték, ami minden modell esetében optimális lenne, hisz ez az érték a modell architektúrájától, a tanítási folyamattól, sőt, akár az azonos modell különböző tanulási fázisaitól is függhet. Továbbá, még egy gondosan kiválasztott limit sem garantálja, hogy egy adott objektumhoz csak egyetlen grid cella kerül dekódolásra, a dobozok változó mérete miatt.



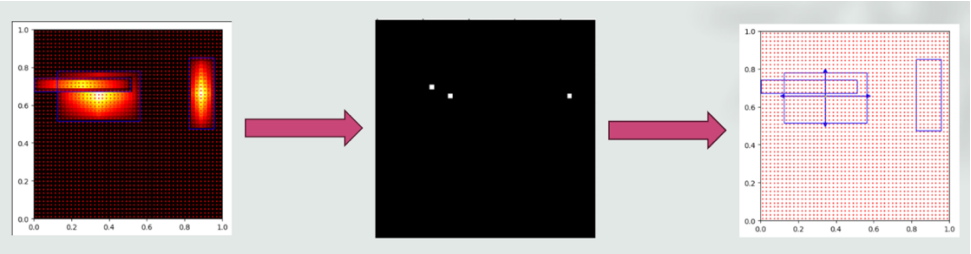
9. ábra: Dekódolás 0-ás, 0.5-ös, 0.7-es és 0.9-es thresholdal. Minden esetben több cella marad egyetlen objektumhoz.

Erre a problémára nyújt megoldást az általam implementált kernel-alapú lokális maximum kereső algoritmus. A Centernet publikációjából [5] átemelt metódust elsősorban a keypoint-alapú detekciós hálózatoknál használják, mivel azok pixel-pontos predikciókat kínálnak, így szükséges a kimenetek szűrése, ugyanakkor ez a megközelítés a grid alapú modellekre is kiterjeszthető.

Az algoritmus öt lépésből áll:

1. Egy (k x k)-ás (Centernet esetén k = 3) maxpool művelet kerül alkalmazásra a grid kimeneten. Ennek eredményeképpen minden cella értéke, amely nem reprezentál lokális maximumot, átalakul a közvetlen környezetében lévő maximumértékre, miközben a lokális maximum értéke változatlan marad.
2. A maxpool művelet bemenete és kimenete közötti elemenkénti összehasonlításra kerül sor. Amennyiben egy cella értéke változatlan marad, az összehasonlítás kimeneti értéke 1, ellenkező esetben 0. Ezen lépéssel létrejön egy bool maszk, amely kiemeli a lokális maximumok helyeit.
3. A bool maszk segítségével maszkolásra kerül a grid.
4. A maszkolt adatokon konfidencia-küszöbértéket alkalmazunk, mivel a lokális maximumok között is előfordulhatnak kevésbé magabiztos predikciók.

5. A megmaradt cellák értékei kerülnek dekódolásra, így alkotva detekciókat.

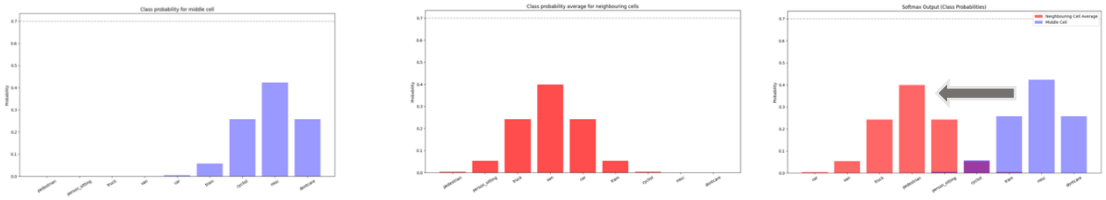


10. ábra: A kerneles lokális maximum keresés segítségével pontosan annyi doboz jön létre, ahány objektum kódolásra került.

A TensorFlow lehetővé teszi az algoritmus effektív és párhuzamos implementációját, így a műveletek rövidebb idő alatt futnak le. Fontos azonban átlátni a módszer hozadékait a későbbi elemzések szempontjából. A kernel méretének megválasztása a detektálási pontosság szempontjából kulcsfontosságú. Amennyiben a kernel mérete túl nagyra van állítva, az átfedő vagy egymáshoz közeli detekciókat elveszíthetjük. Esetünkben ez azt jelenti, hogy 382 x 382 bemeneti képméret, 48 x 48-as kimeneti grid és 3 x 3 -as kernel méret alkalmazásakor a modell nem képes megkülönböztetni azokat az objektumokat, melyek középpontjai közötti távolság kevesebb, mint 12 pixel. Gyakorlati szempontból ez főként távoli és egymáshoz nagyon közel eső objektumoknál okoz problémát, ám az összteljesítményre csak elenyésző hatással van. Túl kis kernel méret alkalmazása esetén a $\min\left(\frac{k}{2}+1\right)$ távolságra lévő, ugyanazt az objektumot reprezentáló magas konfidenciájú cellák nem kerülnek kiszűrésre, ami duplikált detekciót eredményez. Elméletileg egy ideális modellben ez nem fordulna elő, így NMS-re se lenne szükség, de a gyakorlati tapasztalatok ezzel ellentétesek (lásd 5.3).

4.4.1 Osztály konfidencia támogatása kernellel

Bár a regressziós érték a detektált bounding boxok közepe felé a legpontosabb, ez nem mondható el minden esetben az osztály konfidenciákról. Különösen akkor merülhet fel probléma, mikor a predikciós modell két osztály között nehezen tud dönteni, az osztályok valószínűsége hasonlóan magas. Ilyen esetekben értékes információt tartalmazhat a cella közvetlen környezete, mely segíthet a helyes osztály meghatározásában. Felmerül az egyszerű többségi szavazás ötlete, de ez nem veszi figyelembe az osztályok közötti eloszlást, értékes információt dob el. Az általam javasolt módszer kifinomultabb megoldást javasol: a középső cella osztály valószínűségi eloszlását a szomszédos cellák kumulált valószínűségi eloszlása felé lépteti.



11. Ábra: Középső cella osztály konfidencia eloszlásának léptetése a szomszédos cellák eloszlása felé

A módszer leírása a következő: Legyen $P(C_i)$ annak a valószínűsége, hogy a középső cella i osztályba tartozik, legyen N a szomszédos cellák halmaza és legyen $P_n(C_i)$ annak a valószínűsége, hogy az n szomszédos cella osztály i osztályhoz tartozik. Ekkor a szomszédos cellák osztályonkénti átlaga és szórása felírható a következőképpen:

$$\mu(C_i) = \frac{\sum_{n \in N} P_n(C_i)}{|N|} \qquad \sigma(C_i) = \sqrt{\frac{\sum_{n \in N} \left(P_n(C_i) - \mu(C_i)\right)^2}{|N|}}$$

Ekkor a középső cella frissített értékei a saját és szomszédos cellák eloszlásából a a következőkppen írhatóak fel:

$$P_{updated}(C_i) = \alpha P(C_i) + (1 - \alpha)\mu(C_i)$$

Ahol α a súlyozási tényező, mely a szórásból számolódik:

$$\alpha = e^{-\sigma(C_i)}$$

Ez garantálja, hogyha az átlagtól való átlagos eltérés nagy, akkor ez az együttható is nagy, így több felelősséget kap a középső cella az oszály meghatározásában, mivel szomszédai nem határozottak. Ha a szórás kicsi, akkor jobban bízhat a szomszédos értékekben, azok kapnak nagyobb felelősséget az osztály meghatározásában.

Végül, annak érdekében, hogy az osztályonkénti valószínűségek összege 1 legyen a frisített értékeket normalizálni kell:

$$P_{normalized}(C_i) = \frac{P_{updated}(C_i)}{\sum_{j=1}^{K} P_{updated}(C_j)}$$

A módszer eredményeképp magabiztosabb osztály meghatározást kapunk. A műveletek effektíven elvégezhetőek és jól párhuzamosíthatóak, a korábban bemutatott kernel tolóablakszerű futtatásával a griden. A módszernél jelenesetben is kritikus a megfelelő kernelméret választása, hisz negatív hatást tud gyakorolni a predikcióra, ha túl távoli, más osztályú objektumhoz tartozó cellák is beleszólhatnak az osztály meghatározásába. A kernel méretének helyes megválasztása az objektumok méretétől, átfedettségétől és a kimeneti grid nagyságától is függ.

4.5 Kontrasztív NMS

Bár a centerness koncepcióját annak érdekében vezették be, hogy növelje az objektum közepéhez közelebb eső gridcellák konfidenciáját, és a kernel-alapú dekódoló művelet is azt célozza, hogy szűrje a dupla detekciókat, mégis előfordulhat, hogy a hálózat bizonytalanná válik az objektum közepének azonosításában. Emiatt továbbra is szükség van egy Non-Maximum Suppression alkalmazására.

12. ábra: Többszörös detekció ugyanazon objektumra, centerness és kerneles dekódolás esetén is

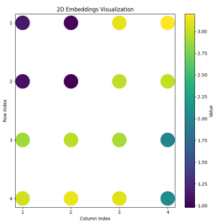
Ennek megoldására egy tanulható NMS eljárást javaslok, amelynek segítségével a neurális hálózat képes megtanulni mely grid cellák tartoznak egy adott objektumhoz. Az így szerzett információ alapján a dekódolás során ezek a cellák szűrhetővé válnak. A módszert a továbbiakban **asszociatív (ANMS) vagy kontrasztív NMS**-ként említem.

Ahhoz, hogy a neurális hálózatnak megtanítsuk, mely cellák tartoznak egy objektumhoz, valamilyen maszkolási stratégiát kell alkalmaznunk. Kulcsfontosságú, hogy a kódolás sorrendfüggetlen legyen: nem célszerű egyszerű számokkal azonosítani a grid cellákat, mivel ez nem kínál optimális tanulási lehetőséget. Ehelyett a cellák közötti relációra kell összpontosítanunk. Ezt a megközelítést az önfelügyelt tanulás paradigmáján keresztül is értelmezhetjük, ahol a hálózat az adatok közötti kapcsolatokat tanulja meg, anélkül, hogy explicit címkézésre lenne szükség.

Kódoláskor rögzítjük azt az információt, hogy mely grid cellákat mely objektumokhoz rendeltük. Azok a cellák, melyek nem tartoznak egy adott objektumhoz, kapcsolódjanak egy ún. "háttér objektumhoz". Ennek eredményeképpen minden celláról eldönthető, hogy mely más cellákkal alkot egy közös objektumot, illetve melyekkel nem oszt meg hasonló összetartozást.

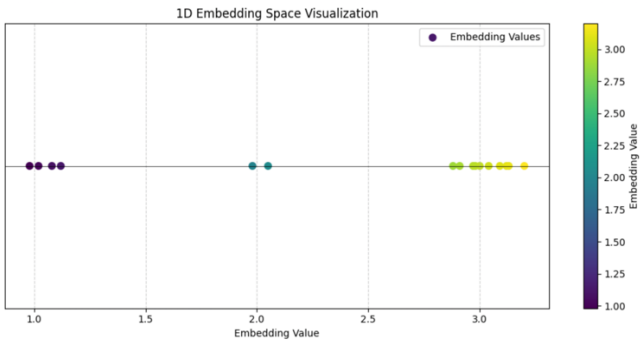
A	A	Z	Z
A	A	Z	Z
Z	Z	Z	B
Z	Z	Z	B

1,12	1,02	3,12	3,20
1,08	0,98	2,98	3,00
2,88	2,97	2,91	1,98
3,04	3,13	3,09	2,05



13. ábra: Példa az asszociativ NMS helyes működésére.

14. Ábra: Az egybe tartozó cellák egymáshoz közeli kimenetet adnak, míg a nem összetartozók távolabb kerülnek.



15. ábra: A prediktált értékek kiterítése egy dimenzióba

Az általam kidolgozott kontrasztív hibafüggvényt (loss) alkalmazó módszer pont erre az elképzelésre épül, amit specifikusan a grid struktúrára adaptáltam. Ezt a kontrasztív veszteségfajtát korábban elsősorban pontpárosítási problémákra alkalmazták, például a Cornernet [4] megközelítésben. A veszteség két komponensből áll: a pull és push loss-okból.

A pull loss célja, hogy minimalizálja az adott cella kimeneti értékének és a hozzátartozó klasszter középpontjának (amelyet a klasszter átlagaként határozunk meg) közötti távolságot. Matematikailag a pull loss a következőképpen írható le.

$$\text{pull\_loss} = \sum_{i=1}^{N} \| \frac{\sum_{j=1}^{K} x_j}{K} - x_i \|_2^2$$

ahol x_i az i-edik cella kimeneti értéke, K az objektumhoz tartozó cellák száma, a szumma pedig az azonos objektumhoz tartozó cellák kimeneti értékeinek összegét írja le. Ennek a veszteségnek az a célja, hogy elősegíti az azonos klaszterbe tartozó cellák közötti kohéziót. Viszont önmagában nem elegendő, mivel nem garantálja, hogy a különböző klasszterek megfelelően elkülönülnek egymástól. Ezt a problémát oldja meg a push loss.

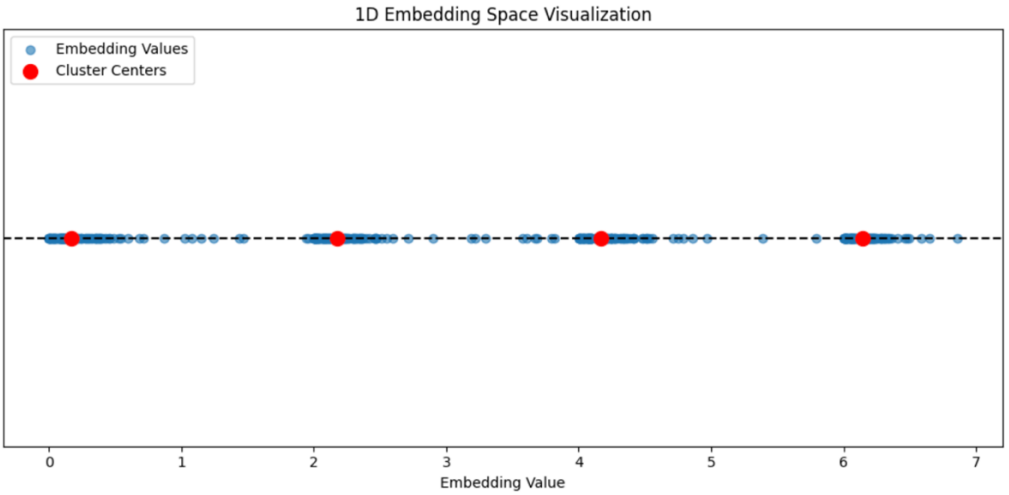
A push loss célja, hogy maximalizálja a különböző klaszterekhez tartozó cellák közötti távolságot. Matematikailag a push loss a következőképpen írható le:

$$\text{push\_loss} = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{K_i} \max(0, m - \|x_i - x_j\|_2^2)$$

Ahol N az összes cella, K_i azon cellák száma, melyek nem tartoznak az i cellával egy objektumhoz. Az m a margin-t jelöli, ami garantálja, hogy a különböző klaszterek középpontjai közötti távolság egy minimális értéknél nagyobb legyen. Bár elsőre úgy tűnhet, hogy ez a módszer nehezen tanulható, a valóságban hatékonyan működik. Ennek az az oka, hogy a térben egymáshoz közeli cellák gyakran azonos objektumhoz tartoznak a pozíciójukból adódóan. Ráadásul kimenetük is erősen összefügg, mivel gyakran ugyanazt a dobozt kódolják le.

A pull loss tehát a szomszédsági mátrixban összetartozó cellák kimeneti értékeit hozza közelebb egymáshoz, míg a push loss célja, hogy a nem összetartozó cellák kimeneti értékei legalább egy előre meghatározott küszöbérték (margin) távolságra kerüljenek egymástól. A tanítási folyamat lényege így az, hogy a kimeneti értékek (esetünkben egy 48x48-as lebegőpontos mátrixon), melyek egydimenziós embedding teret alkotnak, szeparálhatóak legyenek. Az optimálisan betanított modell kimenetén jól elkülöníthető klasszterek jelennek meg, pont annyi klaszterközépponttal, ahány objektumot azonosítunk (ideértve a háttérobjektumot is). A valóságban gyakran előfordulnak átfedések az objektumok között, így a határon lévő cellák klaszterbe sorolása kihívást jelenthet. Ugyanakkor ez a kihívás nem releváns a mi esetünkben, mivel a "centerness" értékek biztosítják, hogy az objektumok középpontjához tartozó cellák kerüljenek kiemelésre, ezekről kell eldönteni, hogy egy klaszterbe tartoznak-e. Dekódolás

során tehát csak azt kell vizsgálni, hogy a lokális maximum értékek embedding-jei a tanítás során használt küszöbértéken belül esnek-e.



16. ábra: Embedding tér vizualizáció 48x48-as grid esetén.

5 Modellek tanítása

5.1 A környezet bemutatása

A futtatási környezetben a TensorFlow keretrendszert használtam, a modelleket NVIDIA RTX 3060 GPU-n tanítottam, mely 12 GB GDDR6 memóriájával rendelkezik, lehetővé téve a kód gyors és hatékony futtatását. A konzisztens futtatási környezet biztosítása érdekében minden modellt ugyanebben a környezetben futtattam, a különbségek csak a modell architektúrális specifikációiban mutatkoztak meg. A KITTI [34] adatbázis a közlekedésben alkalmazott detektorok tanítására jött létre. Az adatbázis 9 különböző kategóriát foglal magában ('car', 'van', 'truck', 'pedestrian', 'person_sitting', 'cyclist', 'tram', 'misc' és 'dontcare'.), összesen mintegy 7500 képpel. Az adatbázist úgy osztottam fel, hogy a teljes adatkészlet 60%-a tanító, 20%-a validációs és további 20%-a teszt adatokból álljon. Ezt az elosztást az egyes osztályokon belül is arányosan végeztem el, így biztosítva az adatkészlet egyensúlyát a különböző kategóriák között.

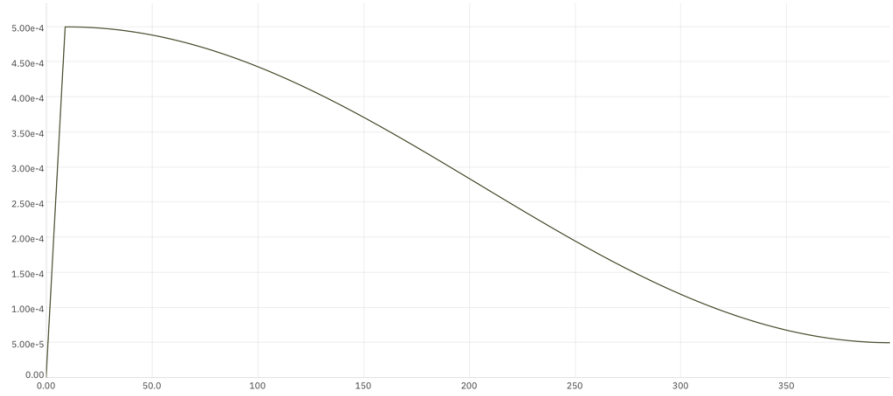
5.1.1 A választott hiperparaméterek indokoltsága

A választott hiperparaméterek kulcsfontosságúak a mélytanulási modellek hatékonyságának és pontosságának növelésében. A képméret (384x384) optimális választásnak bizonyult: kisebb méretnél már az objektumok is nehezen különíthetőek el, míg a növelés nem javított jelentősen az eredményeken, csak a futási időt növelte. A csatorna szempontjából a szürkeárnyalatos képek használata lehetőséget teremtett a modell egyszerűsítésére, a színes képek háromszoros bemeneti adatot igényelnek, ezért a futási idő jelentősen nő. A gyakorlatban az Adam optimizer a legtöbbször választott megoldás a mély tanulási modellekben, a kipróbált optimizáló (RMSprop és SGD) közül ennél a problémánál is ez bizonyult a leghatékonyabbnak. A batch méret kiválasztása (32) az aktuális képméret mellett a GPU memóriájának hatékony kihasználása miatt volt fontos. Az epoch szám tekintetében a 400 epochos tanítás végére a tanulási görbe kilaposodott, jelezve, hogy a modell optimális teljesítményre tanult rá.

Size	Channels	Optimizer	Batch Size	Epochs
384x384	1	Adam	32	400

17. ábra: Választott hiperparaméterek

5.1.2 Tanítási ráta ütemező



18. ábra: A tanításokhoz használt learning rate

A warmup cosine annealing learning rate scheduler [35] egy kifinomult stratégia a neurális hálózatok tanítási folyamatának optimalizálására. A módszer lényege a kétfázisú megközelítésben rejlik: a bemelegítési (warmup) fázist cosinus-os lágyítás követ. A bemelegítési fázis alatt, amely esetünkbe 10 epochot ölel fel, a tanulási sebesség lineárisan növekszik, elkerülve a rendkívül nagy gradiensfrissítéseket a tanítás kezdetén, amikor a súlyok még véletlenszerűek. Egy ilyen stratégia megakadályozhatja a divergenciát, és segíthet a modellnek gyorsabban elérni a hibafelszín egy megfelelő régióját. A bemelegítés után az ütemező cosinusos lágyítást alkalmaz, amely a tanulási sebességet egy fél-cosinusos ciklus szerint modulálja. Ez biztosítja, hogy a tanulási ráta egy maximális értékről indul, idővel egyenletesen csökken, ami a cosinus függvény viselkedését idézi.

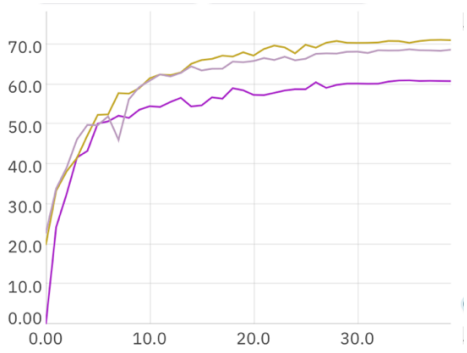
Kutatásomban ez azért fontos, mert a veszteség sok kompensű, főleg a tanulható NMS esetén, egy ilyen bemelegítéssel rendelkező tanulási ráta megkönnyíti a “loss landscape” feltárását. A 400 epoch-os teljes tanulási intervallumot figyelembe véve ez az ütemező dinamikusan állítja be a tanulási sebességet, és kihasználja mind a kezdeti szakaszok gyors feltárásának, mind a későbbi szakaszok célzott finomításának előnyeit.

5.1.3 Tanítás menete

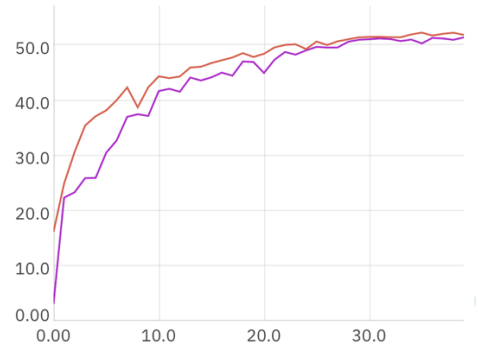
Tanítás során azt tapasztaltam, hogy az anchor-free modelleknek több időre volt szükségük a tanuláshoz, mint a YOLO modellnek megegyező epoch szám mellet. Míg a YOLO tanítási ideje körülbelül 2.5 óra volt, addig az FCOS változatok tanítási ideje meghaladta a 6 órát. Ezt a jelentős időkülönbséget a komplex veszteségfüggvény okozza.

Ezzel egy kompromisszumot kötünk: a hosszabb tanítási időért cserébe rövidebb inferencia időt kapunk.

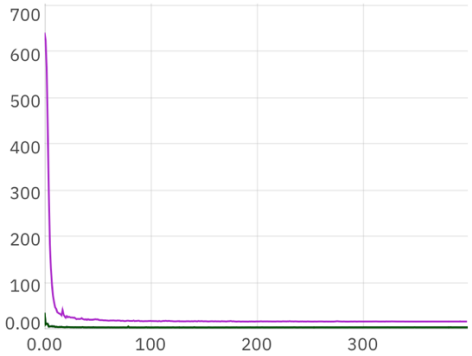
A tanítási folyamat legnagyobb kihívása a veszteségek összehangolása volt. A lokalizációs négyzetes hiba (Mean Squared Error), a klasszifikációs fokális [36] veszteség, a centerness bináris keresztentrópia vesztesége és a kontrasztív NMS két komponense közötti egyensúlyt megtalálni különösen nagy kihívást jelentett. Fontos volt, hogy minden komponens megfelelően betanuljon, anélkül, hogy az egyik veszteségkomponens túlságosan elnyomná a másikat. Összességében több mint 150 különböző tanítási iterációra került sor.



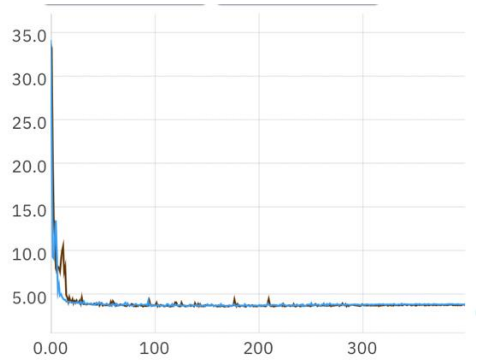
19. Ábra: FCOS tanítási görbe



20. Ábra: YOLO tanítási görbe



21. Ábra: FCOS veszteséggörbe



22. Ábra: YOLO veszteséggörbe

6 Eredmények

A következő részben a tesztek során kapott eredményeket és az értékelési kritériumokat mutatom be. Emellett felfedem a fejlesztés alatt adódott problémákat, azokra adott válaszokat és levont következtetéseket is.

6.1 Mérési szempontok és metrikák

A kritikus rendszerekben alkalmazott neurális hálózatok tervezésekor és finomításakor kiemelt fontosságú, hogy az architektúrák gyorsan, minimális késleltetéssel működjenek, ugyanakkor minél pontosabbak legyenek. Ebben a kontextusban a hálózat reakcióideje és a pontosság közvetlenül hatással lehet a rendszer teljesítményére és a biztonságra, ami kritikus alkalmazások esetén létfontosságú. Ennek megfelelően az értékeléshez a fenti kritériumok figyelembevételével határozzuk meg a metrikákat.

Munkám keretében egy detektor időkritikus jellemzőit vizsgálom, ezért először is fontos áttekinteni a detektorok teljesítményének mérésére alkalmazott kulcsfontosságú mértékegységeket:

1. **Precision [%]:** A háló által helyesen detektált objektumok százalékos arányát jelzi az összes detektált objektumhoz képest.
2. **Recall [%]:** Azt mutatja, hogy az összes tényleges objektum közül hány százalékot ismert fel a hálózat. Más szavakkal, mennyire képes az összes lehetséges pozitív mintát detektálni.
3. **F1 score**: A precision és a recall harmonikus átlagaként adódik. A tökéletes F1 score értéke 1 (vagy 100%), míg a legrosszabb érték 0. Ez egy olyan metrika, amely mind a precisiont, mind a recallt figyelembe veszi, és azok egyensúlyát próbálja megtalálni.
4. **mAP (mean Average Precision):** Az összes osztályra vonatkozó átlagos precízió értéket adja meg. A mAP gyakran használt metrika az objektum-detektáló rendszerek teljesítményének értékelésére, mivel egyetlen számmal képes kifejezni a rendszer teljesítményét.
5. **Paraméter szám:** A neurális hálózatban található paraméterek (súlyok és biasok) összessége. Egy kisebb paraméterszám gyakran kevesebb memóriát és rövidebb futásidőt jelent, de a hálózat komplexitása és teljesítménye is csökkenhet.

6. **Futásidő [FPS]:** Frames Per Second. A háló másodpercenkénti képkocka-feldolgozási képességét mutatja. Magasabb FPS gyorsabb válaszidőt és kisebb késleltetést jelent.

6.2 Eredmények értelmezése

Jelen fejezetben az FCOS detektor különböző változatai és alkalmazott módszerei kerülnek összehasonlításra a baseline YOLO detektorral. Azért a YOLO detektort választottam viszonyítási alapnak, mert széleskörben használják az objektum detektálás világában, jól teljesít alacsony futásidejű rendszerekben, így erős kihívó. Mindegyik modell esetén a backbone és nyak azonos, annak érdekében, hogy igazságos összehasonlítást végezzünk. Elsőként kvantitatív szempontból vizsgálom az eredményeket: a korábban ismertetett metrikák alapján bemutatásra kerül a modellek teljesítménye, valamint sebessége. Ezt követően kvalitatív elemzésre kerül sor, hogy képek segítségével vizuálisan is szemléltetve legyenek az egyes módszerek eredményei.

6.2.1 Kvantitatív eredmények

6.2.1.1 Pontosság

MODEL	F1 SCORE	RECALL	PRECISION	mAP
YOLO+NMS	61,23	61,81	60,66	60,92
FCOS (limit 0.05)	58,63	53,41	65,02	59,27
FCOS *3x3* (ours)	66,98	71,91	63,46	67,21
FCOS *5x5* (ours)	**69,2**	65,16	73,77	69,12
FCOS *7x7* (ours)	67,29	58,74	78,76	68,18
FCOS+NMS *3x3* (ours)	**72,7**	69,01	76,8	72,32
FCOS+NMS *5x5* (ours)	71,03	64,35	79,27	71,45
FCOS+NMS *7x7* (ours)	67,95	58,33	81,37	68,41
FCOS+ANMS *3x3* (ours)	70,3	68,65	75,41	70,12
FCOS+ANMS *5x5* (ours)	**70,6**	63,28	78,97	70,32
FCOS+ANMS *7x7* (ours)	67,05	57,31	80,21	67,13

23. ábra: Modellek teljesítménye

A táblázat az FCOS detektor különböző változatainak és a baseline YOLO detektorral való összehasonlítását prezentálja.

A baseline YOLO modell, amely NMS alkalmazásával futott, 61,23% F1 score-t ért el. NMS nélküli futtatás során a modell értelmezhetetlen eredményeket produkált, túlzottan sok bounding box detektálásával. A baseline FCOS modell, az eredeti publikációban meghatározott 0.05-ös konfidencia limit alkalmazásával, kevésbé teljesített jól (58,63% F1), ami arra enged következtetni, hogy az osztálykonfidencia és a centerness értékek önmagában nem elegendőek a pontos dekódoláshoz.

Az általam javasolt maxima search metódus jelentősen növelte a modell teljesítményét, különösen az 5x5-ös kernel alkalmazásával, amelynél az F1 pontszám 69,2%-ra nőtt. A táblázatban látható, hogy a kernel méretének csökkentésével a modell több duplán detektált bounding boxot eredményez, azaz azonos objektumokat többször is kódol. Ennek következtében csökken a pontosság (Precision) és nő a Recall értéke. A kernel méretének növekedésével ezek a dupla detekciók megszűnnek, ugyanakkor a modell a griden belül egymáshoz közeli, nagy mértékben átfedő, de eltérő objektumokat kódoló bounding boxokat is eldob. Az 5x5-ös kernel méret ezen viselkedések közötti jó kompromisszumot kínál, egyensúlyozva a precízió és recall közötti hatást, így minimalizálva az esetleges hibákat és javítva a teljesítményt.

A hagyományos NMS alkalmazása a kernelek mögött szintén jelentős javulást eredményezett, a modell F1 score-ja 72,7%-ra emelkedett 3x3-as kernel alkalmazásával. Ez szignifikáns majdnem 10 %-os emelkedést jelent a baseline YOLO-hoz képest. Ez azt mutatja, hogy a 3x3-as kernel használatával még mindig maradnak dupla detekciók, azonban az NMS képes ezt orvosolni és javítani a modell teljesítményén. Nagyobb kernel méretek alkalmazása esetén ugyanakkor objektumvesztés figyelhető meg, az NMS nem képes teljes mértékben kompenzálni a nagyobb kernel méret hozta negatív hatásokat.

A dolgozatban javasolt asszociatív NMS bár nem érte el a hagyományos NMS teljesítményét, mégis megközelítette azt. Különösen az NMS nélküli változatokhoz képest mutatott jelentős javulást, 3x3-as kernel alkalmazásával az F1 pontszám 70,3%-ot ért el.

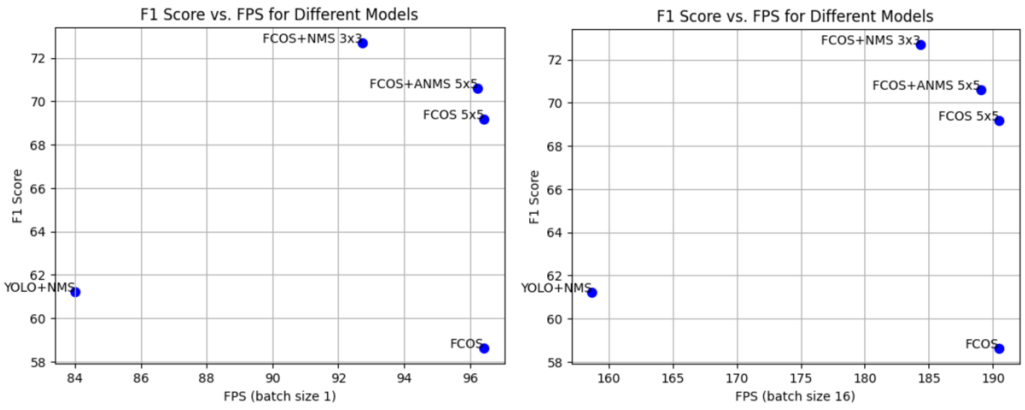
Az elemzés rámutat arra, hogy az anchor-free megoldások saját magukban nem mentesek az NMS-től. A hagyományos NMS és a tanulható NMS egyaránt szignifikáns teljesítménynövekedést indukáltak a vizsgált modellekben.

6.2.1.2 Futásidő

Runtimes (FPS)					
Model	**Batchsize**				
	1	2	4	8	16
YOLO+NMS	84,01	87,63	95,29	129,21	158,70
FCOS	**96,43**	106,18	**113,41**	**122,37**	**190,49**
FCOS+NMS	92,74	103,21	106,32	114,61	184,37
FCOS+ANMS	96,24	**106,67**	112,48	122,12	189,04

24. Ábra: A modellek sebessége a batch mérettől függően

A YOLO modellel elért sebesség a legalacsonyabb (84,01 FPS), ami magyarázható az anchor-ok dekódolási folyamatával és az NMS összetettségével. Az FCOS a leggyorsabb, mert nem használ NMS-t. A dekódoló kernel mérete és a küszöbérték nem befolyásolja érdemben a futásidőt, mivel ezek a műveletek nagy hatékonysággal hajthatók végre tensor műveletekkel. Az NMS alkalmazása az FCOS modell esetében lassulást eredményez, ahol batch mérettől függően átlagosan 5-tel csökkennek az FPS értékek. A batch-enkénti eltérések a GPU és CPU közötti párhuzamos működés variabilitásából adódhatnak. A tanulható NMS használatakor a futásidő nagyon hasonló a kiindulási FCOS modellhez, 2-es batch esetén még javít is rajta. Ez azzal indokolható, hogy elhanyagolhatóan kevés hozzáadott számítással jár, felépítésük az utolsó réteget leszámítva azonos, paraméterszámukban minimális az eltérés.



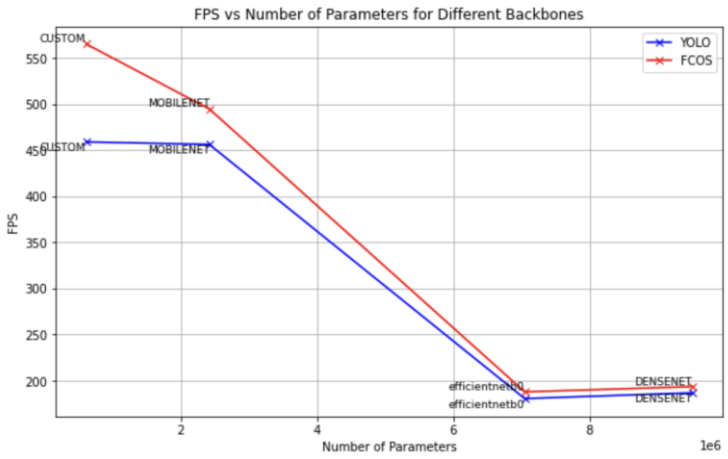
25. Ábra: Legjobban teljesítő modellek pontosságának és sebességének vizualizációja

Megfigyelhető, hogy az FCOS + NMS 3x3 modell mutatja a legjobb F1 score értéket, ami a pontosságot tükrözi. Eközben az FCOS 5x5 a leggyorsabb FPS értékkel rendelkezik, azonban pontossága alacsonyabb. Ugyanakkor az FCOS + ANMS 5x5 egyensúlyt teremt a sebesség és a pontosság között, ami azt jelenti, hogy viszonylag magas F1 Score értéket ér el, miközben a sebessége is elfogadható szinten marad. Az

eredeti baseline YOLO modell lassabb működésű, azonban magasabb pontossággal rendelkezik a méréseim alapján, mint az FCOS, mely gyorsabban fut, de a teljesítménye kicsit elmarad az YOLO-tól.

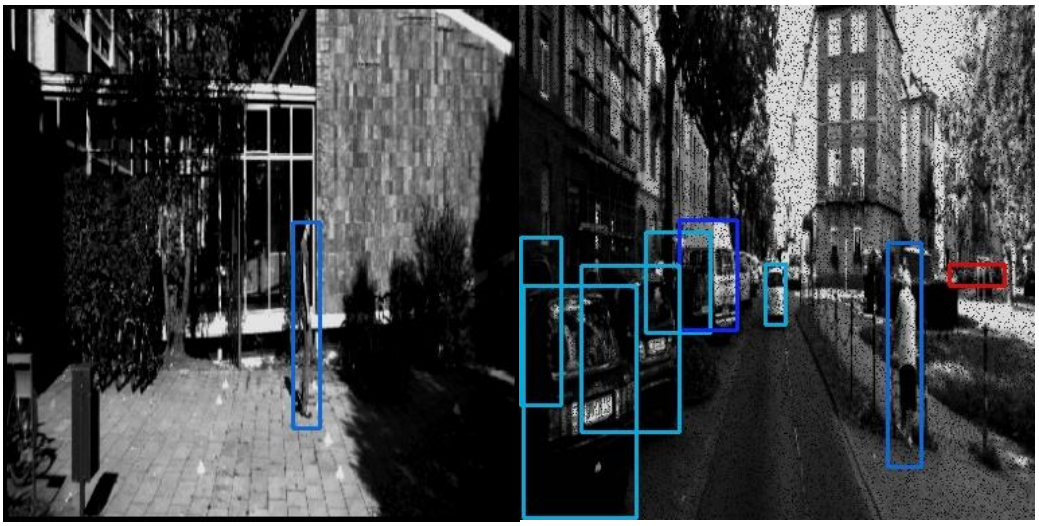
6.2.1.3 Paraméterszám

A 26. ábrán látható diagram a modellek futási idejét ábrázolja különböző paraméterszámú feature extractorok használatával (Mobilenet[38], Efficientnet[39], Densenet[40], Custom: az általam javasolt alacsony paraméterszámú variáns). A YOLO és FCOS közötti eltérés főként a dekódolással eltöltött többlet időből adódik, mely a GPU-n nem jól párhuzamosítható művelet. Mivel ez az időbeli többlet konstans, a paraméterszám növekedésével egyre kevésbé érvényesül a teljes futási időben, és így a nagyobb paraméterszámú modellek esetén egyre kevésbé érzékelhető. Különösen gyors válaszidőt igénylő alkalmazások esetén ez jelentős hatással lehet a modell teljesítményére. Ezen túlmenően az is megfigyelhető, hogy nem csak a paraméterszám, hanem az architekturális sajátosságok is befolyásolják a modell futási idejét, így a vizsgált modellek között tapasztalt futási idő és a paraméterszám közötti összefüggés nem mutat lineáris trendet. Ez alátámasztja az állítást, hogy az anchor-free módszerek alacsony futási idő igényű környezetekben előnyös választásnak bizonyulnak.



26. Ábra: Modellek futásideje különböző méretű backbone esetén

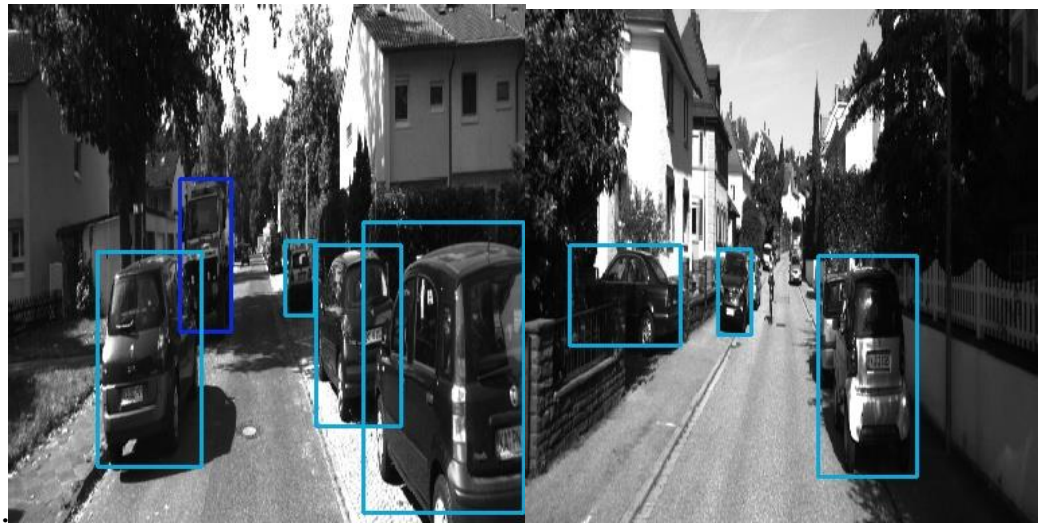
6.2.2 Kvalitatív eredmények



27. és 28. Ábra: Szabad szemmel nehezen észrevehető és átfedő objektumok esetén is jól teljesít az FCOS+ANMS

29. Ábra: ANMS nélkül dupla detekció kapott kép, ANMS-sel jól működik

30. Ábra: A távoli apró objektumok nehézséget jelentenek a modellnek



31. és 32.Ábra: Zsúfolt utcán is eredményes, a távoli aprók jelentik a nehézséget.

7 Összefoglaló

A modern forgalomirányítási és -felügyeleti rendszerek alapköve a járműdetektálás, hiszen a különböző biztonsági intézkedéseket, a forgalom optimalizálását és az autonóm járművek irányítását segíti. Manapság egyre inkább a valós idejű analitikára támaszkodik az ipar, a járművek gyors és pontos észlelése a legkülönfélébb helyzetekben kiemelkedően fontos. A legtöbb mélytanulás-alapú detektor azonban anchor-boxokra - előre meghatározott térbeli szűrőkre – és bonyolult architektúrális elemekre támaszkodik, amelyek korlátozhatják alkalmazkodóképességüket és sebességüket. Emellett időkritikus és helyszíni rendszerekben gyakran nincs lehetőség kiterjedt számítási erőforrásokra, kevésbé toleránsak a késleltetésre.

Kutatásom célja egy olyan detektáló hálózat kifejlesztése volt, amely alacsony késleltetésű környezetben is megbízhatóan teljesít, és jobb eredményeket képes produkálni, mint a sokszor használt YOLO. A hálózatban csökkentettem a súlyszorzásokon kívüli plusz számításokat úgy, hogy azok minél kevesebb időt vegyenek el. A tudományos munka fókuszában tehát olyan univerzális megoldások kidolgozása állt, melyek a backbone-tól függetlenül, általánosan képesek fokozni a hálózat teljesítményét.

Az implementált eljárás teljesítményét a népszerű járműdetektálásra használt KITTI adathalmazon értékeltem ki. Az eredményeket részletekbe menően összevetettem a korábbi megoldásokkal sebesség és pontosság tekintetében, hogy teljes képet alkothassak az új eljárás teljesítményéről.

Összességében a dolgozat bemutat egy új, anchor-free FCOS alapú hálózatot, amely modifikált fejjel, új dekódolási módszerrel és teljesen neurális alapon tanulható NMS-el rendelkezik, ígérve gyors és pontos járműdetektálást alacsony késleltetésű környezetekben. A javasolt modell mind futásidő, mind pontosság tekintetében jelentősen felülmúlja a baseline YOLO modellt. A kerneles dekódolási metódus több mint 10 %-os javulást eredményez F1-score tekintetében, miközben a futás időt nem változtaja a baseline FCOS-hoz képest. Az eredmények alapján látható, hogy az anchor-free megoldások önmagukban nem NMS mentesek. Erre a problémára fejlesztettem ki a kontrasztívan tanulható NMS eljárást, melynek pontossága megközelíti a hagyományos NMS-ét, miközben átlagosan 5 FPS-el gyorsabb lesz a modell, ami kritikus futás idejű környezetben jelentős.

Az eredmények biztatóak az alacsony számítási kapacitással rendelkező, edge computingban használt hálózatok számára és ösztönzi az anchor-free modellek felhasználását a jól bevált anchorosokkal szemben.

Az eredmények azt mutatják, hogy az anchor-mentes modellek potenciálisan előnyösek lehetnek alacsony számítási kapacitású rendszerekben, különösen az edge computing területén. Ezen munka ösztönzi a további kutatásokat folytatását az anchor-mentes architektúrák irányában, összehasonlítva azokat az évek során jól teljesítő anchor-alapú megoldásokkal.

Irodalomjegyzék

[1] Zhang, Hao, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. “DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection.” arXiv, July 11, 2022. https://doi.org/10.48550/arXiv.2203.03605.

[2] Ouyang-Zhang, Jeffrey, Jang Hyun Cho, Xingyi Zhou, and Philipp Krähenbühl. “NMS Strikes Back.” arXiv, December 12, 2022. https://doi.org/10.48550/arXiv.2212.06137.

[3] “Introducing YOLO-NAS: One of The Most Efficient Object Detection Algorithms | by Luís Fernando Torres | LatinXinAI | Medium.” Elérés dátuma: November 2, 2023. https://medium.com/latinxinai/introducing-yolo-nas-one-of-the-most-efficient-object-detection-algorithm-d24303de542.

[4] Law, Hei, and Jia Deng. “CornerNet: Detecting Objects as Paired Keypoints.” arXiv, March 18, 2019. https://doi.org/10.48550/arXiv.1808.01244.

[5] Kaiwen Duan et al. “CenterNet: Keypoint Triplets for Object Detection”. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Oct. 2019.

[6] Zhu, Chenchen, Yihui He, and Marios Savvides. “Feature Selective Anchor-Free Module for Single-Shot Object Detection.” arXiv, March 1, 2019. https://doi.org/10.48550/arXiv.1903.00621.

[7] Tian, Zhi, Chunhua Shen, Hao Chen, and Tong He. “FCOS: Fully Convolutional One-Stage Object Detection.” arXiv, August 20, 2019. https://doi.org/10.48550/arXiv.1904.01355.

[8] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection.” arXiv, May 9, 2016. https://doi.org/10.48550/arXiv.1506.02640.

[9] Girshick, Ross, Jeff, Donahue, Trevor, Darrell, Jitendra, Malik. „Rich feature hierarchies for accurate object detection and semantic segmentation.” arXiv, Nov, 11, 2013. https://doi.org/10.48550/arXiv.1311.2524

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.

[11] M. A. Nielsen, "Neural Networks and Deep Learning," Determination Press, 2015. [Online]. Elérhető: http://neuralnetworksanddeeplearning.com/index.html. [Elérés dátuma: 25. 05. 2020.].

[12] Y. LeCun, K. Kavukcuoglu, and C. Farabet, „Convolutional networks and applications in vision.,” Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp. 253-256, 2010.

[13] Girshick, Ross. “Fast R-CNN.” arXiv, September 27, 2015. https://doi.org/10.48550/arXiv.1504.08083.

[14] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” arXiv, January 6, 2016. https://doi.org/10.48550/arXiv.1506.01497.

[15] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single Shot MultiBox Detector,” 9905:21–37, 2016. https://doi.org/10.1007/978-3-319-46448-0_2.

[16] Zhumabay, Yerdaulet. “Anchor Boxes Generation Using K-Means Clustering.” *Medium* (blog), December 26, 2022. https://medium.com/@yerdaulet.zhumabay/generating-anchor-boxes-by-k-means-82f11c690b82.

[17] Zand, Mohsen, Ali Etemad, and Michael Greenspan. “ObjectBox: From Centers to Boxes for Anchor-Free Object Detection.” arXiv, July 14, 2022. http://arxiv.org/abs/2207.06985.

[18] 2.8 Non-maxima suppression. 2020. [Online]. Elérhető: https://cvexplained.wordpress.com/2020/07/15/2-8-non-maxima-suppression/ [Elérés dátuma: 04 september 2023].

[19] Ericsson, Linus, Henry Gouk, Chen Change Loy, and Timothy M. Hospedales. “Self-Supervised Representation Learning: Introduction, Advances and Challenges.” *IEEE Signal Processing Magazine* 39, no. 3, May, 2022: 42–62. https://doi.org/10.1109/MSP.2021.3134634.

[20] Zhang, Hao, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. “DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection.” arXiv, July 11, 2022. https://doi.org/10.48550/arXiv.2203.03605.

[21] Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. “A Simple Framework for Contrastive Learning of Visual Representations.” arXiv, June 30, 2020.

[22] Liu, Dongqi, Haolan Liang, Xiangjun Zeng, Qiong Zhang, Zidong Zhang, and Minhong Li. “Edge Computing Application, Architecture, and Challenges in Ubiquitous Power Internet of Things.” *Frontiers in Energy Research* 10, 2022. https://www.frontiersin.org/articles/10.3389/fenrg.2022.850252.

[23] Newell, Alejandro, Zhiao Huang, and Jia Deng. “Associative Embedding: End-to-End Learning for Joint Detection and Grouping.” arXiv, June 9, 2017. http://arxiv.org/abs/1611.05424.

[24] “COCO - Common Objects in Context.” Accessed November 2, 2023. https://cocodataset.org/#home.

[25] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. “Feature Pyramid Networks for Object Detection.” arXiv, April 19, 2017. https://doi.org/10.48550/arXiv.1612.03144.

[26] Liu, Shu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. “Path Aggregation Network for Instance Segmentation.” arXiv, September 18, 2018. https://doi.org/10.48550/arXiv.1803.01534.

[27] Ghiasi, Golnaz, Tsung-Yi Lin, Ruoming Pang, and Quoc V. Le. “NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection.” arXiv, April 15, 2019. https://doi.org/10.48550/arXiv.1904.07392.

[28] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. “EfficientDet: Scalable and Efficient Object Detection.” arXiv, July 27, 2020. https://doi.org/10.48550/arXiv.1911.09070.

[29] “Anchor-Free Object Detection with Contrastive Filter,” n.d. Bodla, Navaneeth, Bharat Singh, Rama Chellappa, and Larry S. Davis. “Soft-NMS -- Improving Object Detection With One Line of Code.” arXiv, August 8, 2017. https://doi.org/10.48550/arXiv.1704.04503.

[30] Hosang, Jan, Rodrigo Benenson, and Bernt Schiele. “Learning Non-Maximum Suppression.” arXiv, May 9, 2017. https://doi.org/10.48550/arXiv.1705.02950.

[31] Hu, Han, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. “Relation Networks for Object Detection.” arXiv, June 14, 2018. https://doi.org/10.48550/arXiv.1711.11575.

[32] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob et al. „Attention is all you need” arXiv, June 12, 2017. https://doi.org/10.48550/arXiv.1706.03762

[33] Hui, Jonathan.: *Understanding Feature Pyramid Networks for object detection (FPN)* Medium.com, 2018. [Online]. Available: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c [Access date: 04 september 2023].

[34] „The KITTI Vision Benchmark Suite,” [Online]. Available: http://www.cvlibs.net/datasets/kitti/.

[35] Z. Liu, "Super Convergence Cosine Annealing with Warm-Up Learning Rate," CAIBDA 2022; 2nd International Conference on Artificial Intelligence, Big Data and Algorithms, Nanjing, China, 2022, pp. 1-7.

[36] Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. “Focal Loss for Dense Object Detection.” arXiv, February 7, 2018. https://doi.org/10.48550/arXiv.1708.02002.

[37] Blanka, Bencsik. „Mély neurális hálózatok hatékony ritkítása modell alapú megerősítéses tanulás alkalmazásával.” Tudomány Diákköri Konferencia. 2021.

[38] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” arXiv, April 16, 2017.

[39] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. “EfficientDet: Scalable and Efficient Object Detection.” arXiv, July 27, 2020. https://doi.org/10.48550/arXiv.1911.09070.

[40] Huang, Gao, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. “Densely Connected Convolutional Networks.” arXiv, January 28, 2018. https://doi.org/10.48550/arXiv.1608.06993.

[41] Newell, A. and Deng, J. „Pixels to graphs by associative embedding. In Advances in Neural Information Processing Systems.” 2017, pages 2168–2177.