Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmozott Informatikai Tanszék

Szabó Attila Dániel

Valósidejű intrúder felismerés UAV környezetben

Konzulensek:

Ács Judit,

Daróczy Bálint Zoltán

Budapest, 2017

Tartalomjegyzék

Kivonat

Egyre több pilóta nélküli légi jármű (Unmanned Aerial Vehicle, UAV) vagy más néven drón jelenik meg a légtérben, melyek ön- vagy távirányítással közlekednek. Előbbiek esetében a megfelelő integráció egyik fontos eleme az érzékelés (pl.: látás) és az azt követő esetleges reakció. TDK munkámban a (fedélzeti) kamerán megjelenő UAV egységek automatikus azonosításával foglalkozom, ezen feladat megoldására konvolúciós neurális hálózatokat (Convolutional Neural Network, CNN) használok.

A dolgozatom témája a Magyar Tudományos Akadémia (MTA), Számítástechnikai és Automatizálási Kutatóintézet (SZTAKI), Informatikai Kutató Laboratórium egyik projektjének része. A kutatás célja egy monokuláris kamera alapú fedélzeti beágyazott rendszer fejlesztése, amelynek feladata, hogy valós időben dolgozza fel a képi adatokat és indítsa el az elkerülési folyamatot, ha szükséges.

A megoldásom mély neurális hálózatokon alapul (Deep Neural Network, DNN), e tanulómodellek közös tulajdonsága, hogy nagy számítási kapacitást igényelnek, de ezzel ellentétesen a beágyazott környezetben jelentősen kevesebb erőforrás érhető el. A tanítást előre el lehet végezni erős grafikus processzorok (Graphical Processing Unit, GPU) segítségével, ám a valós idejű feldolgozáshoz a kész hálózat predikciójának is gyorsnak kell lennie. További kihívás, hogy kevés adat áll rendelkezésre az ilyen rendszerek tanításához szükséges adatmennyiséghez képest.

Kutatásom során megvizsgálom a nemzetközi szakirodalomban elérhető különböző általános lehetőségeket a problémák megoldására és az adott rendszerre optimalizált tanuló modelleket dolgozok ki. Továbbá különböző tanulási technikákat hasonlítok össze és optimalizálom a hálózatokra a korlátozásokat figyelembe véve. Dolgozatomban bemutatom az eredményeket, összemérve a különböző modellek képességeit.

Abstract

More and more unmanned aerial vehicles (UAV), also known as drones, are present in the national airspace either with remote control or with fully autonomous on-board system. In the latter case, the sense (based on e.g. visual, radar or multi-modal input) and avoid capability is crucial in the integration of such devices. In my research I focus on the automatic identification of the UAVs with convolutional neural networks (CNN).

This topic is part of the project called Real Flight Demonstration of Monocular Image-Based Aircraft Sense and Avoid from the Computer and Automation Research Institute (SZTAKI) of the Hungarian Academy of Sciences (MTA). The main goal is to develop a monocular camera based on-board system, which processes image data in real-time and initiates collision avoidance if required.

The core of my solution for the task are deep neural networks (DNN). These artificial intelligence models require quite a lot of computational capacity. In contrast to this, there are significantly less resources available in embedded environments. The learning part could be done previously on strong graphical processing units (GPU), but the deployment phase of the net should also be fast for real-time processing. Further requirement for machine learning models is data. In this case the small size of the available dataset is an other challenge to solve.

In my research I review the potential techniques for the problems recommended by the literature and I develop an adequate method for this specific system. Furthermore, I compare different learning techniques for convolutional neural architectures with respect to the given restrictions. Last but not least the evaluation of the solution is also part of my work. The results are set side by side for the different models.

1. Bevezetés

Az autonóm járművek területe napjaink egyik legizgalmasabb témájának tekinthető mind az ipari, mind a tudományos világban. Az összes nagy technológiai cég képviseli magát valamilyen szinten ezeknek a rendszereknek a fejlesztésében. Az önvezető autók [4,5,14,24,33], az UAV gépek több különböző területet is érintenek, például szenzorok, "dolgok internete" (Internet of Things, IoT), Big Data vagy éppen mesterséges intelligencia.

Utóbbihoz tartozik a drónok önállóságának mértéke, mely magába foglalja mind a távoli irányítású jármű, mind pedig a teljesen önálló UAV kérdéskörét. Az ilyesfajta működés eléréséhez sokféle hardveres és szoftveres megoldás lehetséges. Az MTA SZTAKI-ban 2016-ban több kutatócsoport együttműködésével indított projekt [35] célja egy évek óta fejlesztett UAV kiegészítése egy látható színtartományú kamerával és a hozzá tartozó intrúder detektáló integrálása. A projekt célja, hogy a robotok döntéshozó képességét bővítsük egy automatikus ütközéselkerülő rendszerrel.

A monokuláris kamera rögzíti a felvételt, eközben három fő feladatot kell végrehajtani rövid idő alatt. Ezeknek a műveleteknek a sebessége meghatározó, hiszen valós időben kell reagálni adott esetben. Ez az egyik legfontosabb megszorítás bármely autonóm rendszerre nézve, mely függ a hardveres, a szoftveres és a kommunikációs komponensektől. Az első feladat tehát, hogy a képi adatot feldolgozzuk, és detektáljunk minden esetleges érzékelhető veszélyforrást magas visszahívási arány mellett. Ezt követi a konkrét felismerés, egy osztályozás, hogy a képen van-e másik repülő eszköz vagy sem. Végül a kiértékelés alapján egy esetleges elkerülő manővert kell végrehajtani.

Az én feladatom az UAV egységek felismerését segítő tanuló rendszer elkészítése. Ehhez rendelkezésre áll egy előre felcímkézett adathalmaz, így alapvetően kínálkozott, hogy felügyelt tanuláson (supervised learning) alapuló megoldást válasszak.

A tanuló rendszerek között manapság a neurális hálózatok [1,3,15] a legelterjedtebbek vizuális szenzorok kimenetének feldolgozására. Az egyre mélyebb és mélyebb architektúrák alkalmazásának legfőbb oka, hogy ma már sok esetben léteznek megfelelő méretű adathalmazok [9], és a feladathoz szükséges számítási kapacitás is rendelkezésre áll [18]. A drónok beágyazott rendszerei technológiai korlátaik okán (pl. fogyasztás, hűtés, tömeg és környezeti hatások) jelentős mértékben elmaradnak az egyébként széles körben alkalmazott asztali vagy szerver GPGPU-k teljesítményétől. Felmerül a kérdés, lehetséges-e s ha igen milyen formában alkalmazni beágyazott rendszerekben mély neurális hálózatokat.

Az utóbbi években mind a neurális hálózatok, mind beágyazásuk egyre nagyobb és nagyobb figyelmet kapott. Célhardverek jelentek meg gépi tanulási algoritmusok tanítására és/vagy felhasználásra. Ilyen célú chipeket fejleszt például a Nvidia és a Google is, illetve már az Apple új okostelefonjaiban is megtalálható hasonló. Szoftveres oldalon nagyon sok keretrendszer érhető el (pl.: Tensorflow[1], Caffe[2]), melyeket a gyártók is támogatnak különböző, a tanítás gyorsítását elősegítő natív program könyvtárakkal. Grafikus gyorsítókon való tanításhoz a CUDA programozási nyelvhez elérhető a külön a neurális hálózatokra és a mély tanulásra optimalizált cuDNN[3] csomag.

A kutatásom célja, hogy bemutassam, miként lehet konvolúciós neurális hálózatokat alkalmazni az adott korlátozások mellett.

[1] https://www.tensorflow.org
[2] https://caffe.berkeleyvision.org
[3] https://developer.nvidia.com/cuda-zone

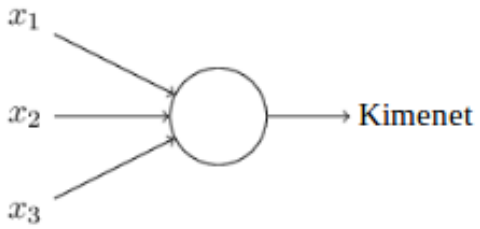
2. Irodalmi áttekintés

A dolgozatomban taglalt rendszerek a gépi tanulás területéhez tartozónak tekinthetőek. Szemben a klasszikus programozási módszerekkel, nem bizonyos szabályok és utasítások mentén hajtja végre a program a feladatot, hanem saját maga sajátítja el az adatok statisztikai tulajdonságai alapján a kívánt működést. Ezáltal olyan problémákat is meg tudunk oldani, amelyeket más módszerekkel nem, vagy nagyon nehezen lehet abszolválni. Ide tartoznak a különböző osztályozási feladatok, felismerési problémák, például képfelismerés. Miután maga a feladat során az előre kiválogatott jelölt területek osztályozása a feladat, sem tradicionális Boosting [2,8,32] sem Support Vector Machine [6,30,22] vagy éppen Bag-of-features [7,23] alapú módszereket nem alkalmaztam.

Neuron

A mesterséges neurális hálózatok olyan tanuló rendszerek, melyeket az agyban található biológiai neuronokról és azok hálózatáról mintáztak. Ezért a biológiai megfelelőjéhez hasonlóan a neuronok tekinthetőek a rendszer alapvető alkotóelemének [1,3].

A modell lényege, hogy egy súlyozott összeget képzünk a bemenetekből, és erre alkalmazunk egy aktivációs függvényt.



1. ábra: Perceptron három bemenettel[4]

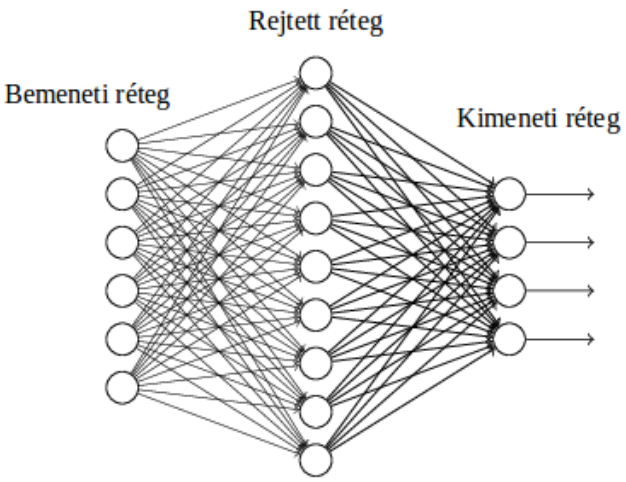
Multilayer Perceptron

Egy neuronnak korlátozottak a képességei (véges a kapacitása), ezért ahogy az agyban, a modellekben is több egységet kapcsolunk össze, így kapunk hálózatokat. Ennek egyik legegyszerűbb módja az előrecsatolás, ebben az esetben egy-egy neuron által kiszámolt eredményt továbbadunk egy másiknak, mint bemenet. Másik lehetőség a visszacsatolás valamely formája, ezeket rekurrens hálózatoknak nevezzük, de ez nem témája a dolgozatnak,

[4] http://neuralnetworksanddeeplearning.com

ugyan a javasolt modellek természetes kiterjesztése már a rekurrencia kihasználását is megkívánja.

Alapvetően a hálózatot szélességében is növelhetjük, ez azt jelenti, hogy párhuzamosan működtetünk egységeket, melyek így egy réteget alkotnak. Előre csatolt (Feed-forward) hálózatok esetén három típust különböztethetünk meg, bemeneti, rejtett és kimeneti réteg. A legelső során kapja meg a hálózat az adatokat, az utolsó a mesterséges neurális hálózat (artificial neural network, ANN) válaszát adja, míg a köztes rétegek végzik a számolást és tárolják el az információt. Multilayer Perceptron (MLP) hálózatot akkor kapunk, ha a rétegek között teljes összeköttetést alkalmazunk a fentiek szerint, azaz az i. réteg minden neuronjának kimenetét a i+1. réteg minden neuronjának bemenetére kötünk előrecsatoltan. A modell mélységét a rejtett rétegek száma jelzi [3]. Ismert, hogy a mélyebb architektúrák kevesebb neuronnal képesek közelíteni, mint a nem mély architektúrák [20,26,34].



2. ábra: Előre csatolt neurális hálózat[5]

Neurális hálózatok tanítása

Az ANN-ek tanítása tulajdonképpen visszavezethető az optimalizálási problémákra. A modellek által meghatározott többdimenziós súlytérben keressük egy szabadon választott hibafüggvény globális (vagy a gyakorlatban lokális) szélsőértékét. Erre a leggyakrabban gradiens alapú (gradient descent) módszereket használnak [17,28]. Három fő típusa létezik, az első a batch learning, amelyben a teljes adathalmazt a hálózat bemenetére tesszük és egyben számoljuk ki a változtatás mennyiségét. A másik véglet amikor az adatokat egyesével adjuk át a modellnek. A két típus pozitív tulajdonságait egyesíti a mini-batch learning, amely egy

[5] http://neuralnetworksanddeeplearning.com

hiperparamétere lesz a tanításnak, mi szabhatjuk meg, hány adat kerüljön egyszerre a bemenetre. Ezt nevezzük sztochasztikus gradiens módszernek (SGD, Stochastic Gradient Descent).

A súlyok frissítéséért a hibavisszaterjesztő algoritmus (backpropagation [28]) felelős, ezáltal lehetséges a többrétegű neurális hálózatok tanítása. A módszer azt használja ki, hogy ezek a modellek matematikailag egymásba ágyazott függvényként is leírhatók és a lánc szabály alkalmazásával kiszámolható, hogy a kimeneten érzékelt hiba javításához mely súlyokat kell változtatni, hogy a kívánt eredményt kapjuk.

Az optimalizáció gyorsítására többféle gyorsító eljárást dolgoztak ki a kutatók, napjaink egyik legjobb ilyen módszerét ADAM-nek nevezik [17]. A kutatásom során ezt használom a hálózatok tanítására.

Konvolúciós neurális hálózatok

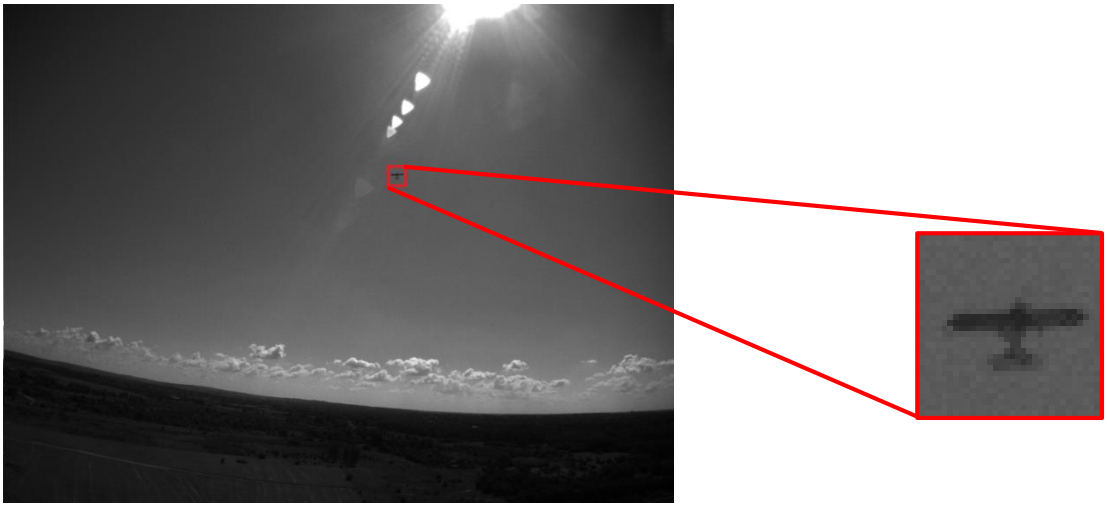
A fent említettekben eddig nem foglalkoztunk azzal, hogy milyen típusú a bemenet, az ANN-ek univerzális osztályozóknak tekinthetőek, azaz nem számít, hogy szöveget, fizikai jeleket, vagy képeket használunk. Viszont ha szigorítjuk ezt egy feltétellel akkor további optimalizációkat tudunk bevezetni.

A konvolúciós neurális hálózatok [19] pont azzal érnek el javulást például a képfelismerési feladatokban, hogy kifejezetten képeket várnak a bemeneten. A hagyományos teljesen csatolt rétegeket így kiegészíthetjük speciálisabb társaikkal, ezek lehetnek konvolúciós rétegek, pooling rétegek [11,12,15,19,29]. Előbbiek a jellemzőket nyerik ki a képekből és tárolják el a paramétereikben, utóbbiak pedig a dimenzió (paraméterszám) csökkentésében játszanak fontos szerepet, így növelve a hatékonyságot. A feldolgozó folyamatban a kezdeti képünkből, melynek a domináns kiterjedései a szélessége és a magassága voltak, egy olyan adatszerkezetet kapunk, ahol a mélység a domináns.

Ezek a hálózatok is a mélytanulás kategóriájába esnek, ugyanis az egyik nagy erejét az egyszerű rétegek egymás után csatolásával érhetjük el. Fontos újítás volt, mikor lecserélték a domináns szigmoid típusú aktivációs függvényeket és helyette ReLu-t (Rectified linear unit), illetve a módosításait használják a tanításhoz, ezt teszem én is a dolgozatomban. Ennek előnye, hogy gyorsabb a tanulás és kevesebb számolást igényel.

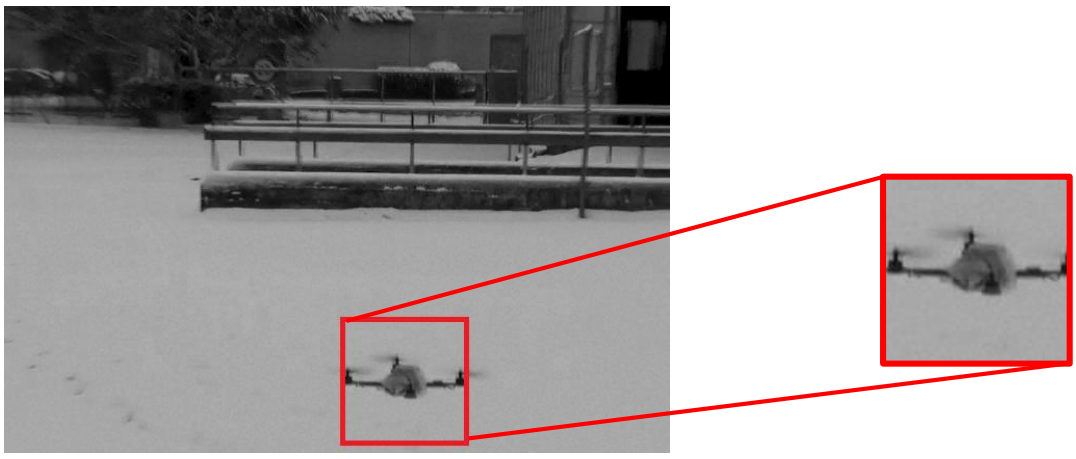
3. Adathalmaz

Alapvetően két nagyobb független halmazból áll a kialakult adatbázis. Az első nagyobb egységbe a SZTAKI csapata által egy RC ("remote control") repülőről készült felvételek tartoznak [35]. Ez összességében 8 különálló videót jelent, melyek a kutatás során fokozatosan váltak elérhetővé. A bináris fájlokhoz készültek leíró szövegfájlok, ezek tartalmazzák a címkéket, hogy melyik képkockán (frame) található a keresett objektum, illetve az azt befoglaló téglalap (bounding-box) koordinátái.



3. ábra: SZTAKI adatbázis

A második fele az adatoknak egy referencia halmazként tekinthető külső felvételsorozat, amelyet a lausanne-i műszaki egyetem (École polytechnique fédérale de Lausanne EPFL) gépi látással foglalkozó kutatólaboratóriuma tett közzé a hozzátartozó MATLAB projekttel az eredményeiket bemutató cikkük megjelenésével egyidőben [27]. A videófelvételek repülőkről és drónokról készültek. Előbbi formailag közel megegyező a fenti eszközzel, tehát várhatóan növeli a pontosságot, utóbbi viszont más felépítésű, ezért a pusztán



4.ábra: CVPR drón adatbázis

egy-egy képre hagyatkozva a képfelismerő modellek általánosítási képességét fejlesztheti, ha azonos pozitív címkével tanítjuk. Ezekhez is elérhetőek (sajnos teljesen más formátumban) a manuálisan létrehozott címkéket tartalmazó leírófájlok. Első lépés a két adathalmaz egységes formára alakítása. Mivel már létezett egy, a SZTAKI belső felvételeire épülő adatformátum, minden adathalmazt ennek megfelelően transzformáltam. A tanításhoz, szintén folytatólagos okokból, a képeket RGB színekből egycsatornás alakra hoztuk. Ennek eredménye, hogy minden jelölt (vagy véletlenszerűen választott negatív minta) terület leírását a következő formátumban tárolom:

- exp: az adott felvételsorozat azonosítója
- fid: a felvételen belüli frame sorszáma
- label: bináris érték, van a képen UAV: 1, nincs: 0
- cx, cy : a frame-ből kivágott bounding box közepének x, y koordinátája
- feat_{0,1023}: a kivágott bounding box 32x32-re méretezett kiterített pixelértékei

Azokon a frame-eken, amelyeken nincs annotáció repülő objektumról, a háttérből vágunk ki véletlenszerűen egy jelenetet negatív mintaként. Az 1. táblázat összefoglalja az így összeállt kb. 150.000 adat eloszlását (kerekített értékek).

Exp:	**Label 1 (db)**	**Label 0 (db)**
0	2.000	23.000
1	1.000	20.000
2	200	2.500
3	700	14.500
4	700	14.000
5	400	10.000
6	900	16.000
7	100	7.000
SZTAKI:	**6.000**	**107.000**
CVPR planes	4.500	3.000
CVPR drones	14.000	20.000
Teljes:	**24.500**	**130.000**

1. táblázat: Teljes tanító adathalmaz

4. Tanító környezet

Szoftver

A feladathoz adott volt, hogy a Caffe Deep Learning keretrendszert kell használni, ugyanis a teljes projekt szempontjából az is cél volt, hogy a korábban használt megoldásokhoz hozzá lehessen mérni a környezet teljesítményét, a legfontosabb szempontként a kész modell predikálási sebességét. Ezen felül kritikus korlát, hogy az előzetes tesztek alapján a SZTAKI UAV repülőjében már integrált Nvidia Jetson TX1-hez az Nvidia által fejlesztett TensorRT[6] a szignifikáns előnnyel a leghatékonyabb keretrendszer, mely általunk felhasznált verziójában még csak Caffe modelleket képes natívan kezelni.

A Berkerley amerikai egyetemen fejlesztett rendszer egyik legnagyobb előnye, hogy az alapvető funkcióit C++ nyelven írták, illetve remek CUDA (felhasznált verzió: 8) integrációt tartalmaz a GPU-n való tanításhoz, melyet tovább gyorsít a már említett cuDNN (felhasznált verzió: v7.0) modul. A feladat szempontjából kifejezetten előnyös, hogy a konvolúciós hálózatokra különösen nagy hangsúlyt fektettek az optimalizáció során. Összességében tehát az egyik leggyorsabb megoldást kínálja a feladat által igényelt szempontokban, továbbá command line és python interfész érhető el hozzá a könnyebb fejlesztéshez.

Hardver

A konkrét cél architektúrában, azaz a repülőn, amelyen fut a predikció, egy Nvidia Tegra X1 chip található a következő paraméterekkel:

- CPU: ARM Cortex-A57
- GPU: 256 CUDA számoló egység
- RAM: 4 GB

Caffe-val lehetőségünk van mind CPU-n, mind GPU-n tanítani, mi utóbbit választottuk. Kétféle hardveres környezetet használtunk, egyet a kutatásokhoz, modellek összerakásához és gyors teszteléshez, illetve egy másikat az érdembeli mély tanuláshoz.

[6] https://developer.nvidia.com/tensorrt

Fejlesztői architektúra

- CPU: Intel Core i7-6700HQ, 2,5/3,5 GHz
- GPU: NVIDIA GeForce 940MX, 2 GB VRAM
- RAM: 8 GB

Tanulási architektúra

- CPU: Intel Core i7-3770K, 3,5/3,9 GHz
- GPU: 2x NVIDIA GeForce GTX 980, 4 GB VRAM
- RAM: 16 GB

A tényleges tanításra célszerűbb a nagy számítási kapacitással rendelkező grafikus kártyákat alkalmazni, erre alkalmas a SZTAKI által nyújtott szerver.

Adatok feldolgozása, tisztítása.

Caffe-ban 3 fő lehetőség van fájlokból való tanulásra. Mivel a rendszer célja pusztán vizuális adatok alapján felismerni intrúdereket, lehetőség van a bemenet megadására közvetlen a képek fájlrendszerbeli útvonalainak felsorolásával is. Számunkra már van egy tömörítettebb csv fájl, ami leírja a képkockákat, de sajnos ezt közvetlenül nem lehet felhasználni. Helyette vagy LMDB adatbázisként, vagy HDF5 formátumban tudjuk megadni a bemenetet. Én utóbbit választottam, mert pythonban viszonylag egyszerűen lehet átalakítani ebbe az adatstruktúrába a CSV fájlokat.

A képek pixelértékein és címkéin túl más attribútumok is megtalálhatóak az adathalmazban. Ezek mind adminisztratív feladatot látnak el, azonosítják az egyes 32x32-es befoglaló négyzeteket. Ebből adódóan egyediek a képekre való tekintettel, tehát nem hordoznak a tanítás során olyan statisztikai értéket, amely alapján az ANN könnyebben megtanulhatná felismerni a repülő egységeket. Tehát a végső bemeneten csak a képeket adjuk be modelltől függően megfelelő formátumban, illetve a pixelértékeket már normalizálva tárolom a HDF5 struktúrában.

5. Alapvető neurális modellek összehasonlítása

Ez a kétosztályos osztályozási feladat egyszerűnek mondható a több kategóriás, nagyobb felbontású és színes képfelismerési társaihoz képest. Pont ez a cél, hogy a neurális hálózatok előnyeit effektíven átültessük beágyazott rendszerekbe. Tehát először azt kell meghatározni, hogy lehetséges-e az utóbbi problémákra szakosodott konvolúciós hálózatokat és mély tanulást eszközölni, vagy klasszikusabb modellekkel érdemes megoldani a feladatot.

Logisztikus regressziós alapú osztályozás

Két osztály esetén egy egyszerű logisztikus regresszió (logreg) is bevethető mint klasszifikáló modell. Tulajdonképpen egy neuronnak feleltethető meg. Hátránya, hogy csak lineárisan szeparálható feladatra működik helyesen. Caffe-ban egy 1 neuront tartalmazó teljesen csatolt rétegként implementálhatjuk.

Multilayer Perceptron

Ez a modell már alkalmazható nemlineáris problémák megoldására is. Hátránya, hogy kevés réteggel is nagy paraméterszám jár, ezért hamar túltanulhat, azaz a tanító adatokra pontosítja a súlyokat, ezzel veszítve az általánosítási képességéből. Az összehasonlítások során két rejtett réteggel rendelkező modellt használtunk, melyek 16, 32 vagy 64 neuront tartalmazhattak. Ez a két rendszer a képeket vektoros formában kapja a bemenetre.

Konvolúciós Neurális hálózat

Kifejezetten képek tanulására alkalmas neurális architektúra [19], ezentúl egyik nagy előnye a feladat szempontjából, hogy kisebb paramétertér szükséges a tanulás során, de jelen esetben az optimalizáció során itt is figyelembe kell venni, hogy a mélységet nem érdemes a végtelenségig növelni. A CNN modellek 32x32x1-es bemeneteket kapnak.

A tesztelések alatt két konvolúciós réteget, két pooling réteket és egy fully-connected réteget tanítottunk, ahol a két konvolúciós 16,32,64 míg az utolsó FC réteg 16,32,64,128 paramétert vehetett fel.

A pontosság (minél jobb predikciós eredmény) és a hatékonyság (minél kevesebb paraméter, minél gyorsabb működés) tekintetében a képen látható CNN modellt választottuk optimálisnak.

6. Adat augmentáció

Vegyünk két hipotetikus lehetőséget. Első esetben a tanító adataink “nem túl jók”, nehezen lehet belőlük kinyerni bármit is, ebben az esetben hiába van két (egyenként is jó!) tanuló modellünk, amelyből az egyik “sokkal jobb” a másiknál, nem fog annyival “jobb” eredményt adni, mint a másik. Ellenben a második esetben megfelelő adataink vannak, “jó” volt a gyűjtés és a feldolgozás is, ugyanazok a modelleket használva azt tapasztaljuk, hogy a “gyengébb” is eredményesebb volt, mint az előző esetben a “jobb”.

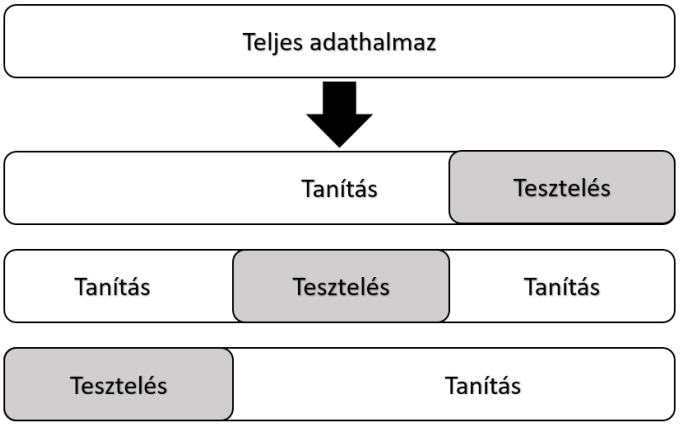
SZTAKI felvételek

Az 1. táblázatban látható, hogy kevés a pozitív felvétel, azaz olyanok, amelyeken található UAV egység. Mélytanulás során fontos, hogy kellő mennyiség álljon a rendelkezésünkre, de minden osztálynak megfelelő mennyiségű képviselője kell legyen az adathalmazban. Ezek hiánya két problémához vezethet.

Ha az egyes osztályokból nem hasonló arányú képviselő van, akkor jelen bináris klasszifikáció során fennáll az a veszély a modell egyszerűen az egyik osztályt “észre sem veszi” és mindenre a másikat tippeli, mert így is sok találata van. UAV környezetben sokkal fontosabb, hogy ne legyen olyan hiba, amely valójában veszélyt jelentett volna (false negative), még ha ennek árán többször végzünk kikerülési manővert olyan esetben is, amikor nem kellet volna (false positive). Ezt a költségfüggvény küszöbének (threshold) állításával is lehet szabályozni.

Ha kevés az adat, akkor nem tudjuk az adatokat a klasszikus tanító-validáló-teszt hármasokra bontani. Ezen javít, ha (k-fold) cross-validation technikát alkalmazunk. Ekkor az adathalmazt több (k db) hasonló felépítésű diszjunkt részhalmazra bontjuk, és úgy alakítjuk ki a tanulást, hogy mindig egy ilyen csoportot használunk validációra a többit tanításra. Az összesre (k lehetőség) elvégezzük a tanítást és a végén kombináljuk az így kapott modellek eredményét.

Jelen esetben a kézenfekvő felosztás a felvételenkénti csoportosítása az adatoknak. Mivel valamennyire különbözőek az egyes felvételekből származó képadatok ezáltal a modellek általánosítási képességére is tudunk következtetni. Sajnos nagy a különbség egy-egy videóból származó képek mennyiségei között, ezért a végül 3 körülbelül egyforma részre osztottam az adatokat.

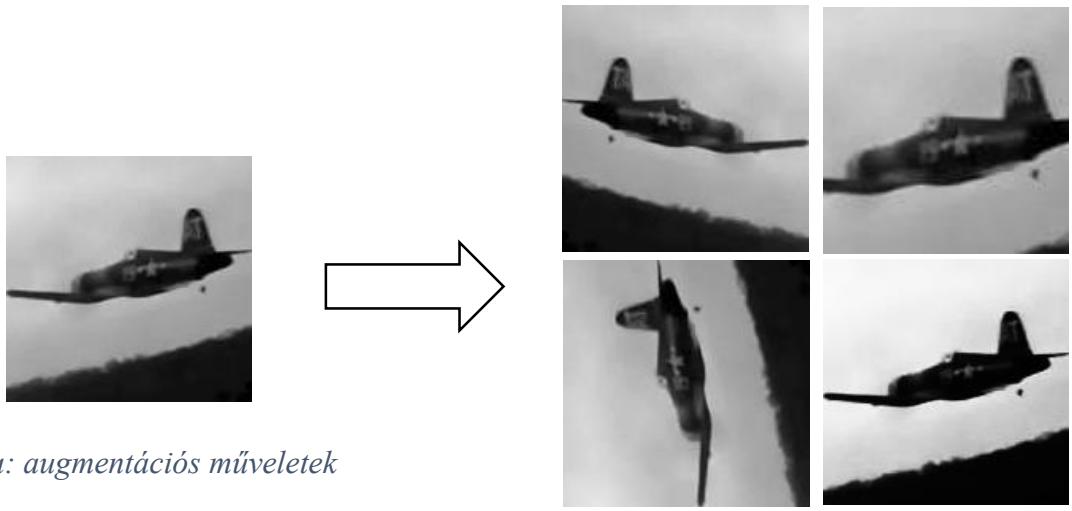


5. ábra: 3-fold cross validation

Az adathalmaz növelésére megoldást nyújt az adat augmentáció (data augmentation). Az emberi felismerés számára egy macskáról készült fotó és annak tükörképe nem jelent különbséget, ellenben a számítógépnek ezek különböző bemenetet jelentenek. Ilyen módon a tanítás előtt a képet egy feldolgozó pipeline-on vezetjük keresztül, amely plusz adatot generál.

Az alkalmazott műveletek:

- tengelyes tükrözés,
- forgatás, akár 1 fokonként is lehetséges, mi 45 fokot alkalmazunk
- kivágás és nagyítás,_a kép egy részletét vesszük és visszaméretezzük az eredeti dimenziók szerint. Jelenleg 24x24 és 16x16 méretű képrészleteket nagyítunk vissza 32x32-essé.
- kontraszt kiemelés,_a kép pixelértékeire alkalmazott filterrel kiemelhetünk vagy lecsökkenthetünk tulajdonságokat. A pixelértékeket egy szigmoid függvényen áteresztve sokkal inkább a szélsőértékek (fekete, fehér) felé tolódnak el a szürke árnyalatok, ezért a kontrasztosabb lesz a kép.



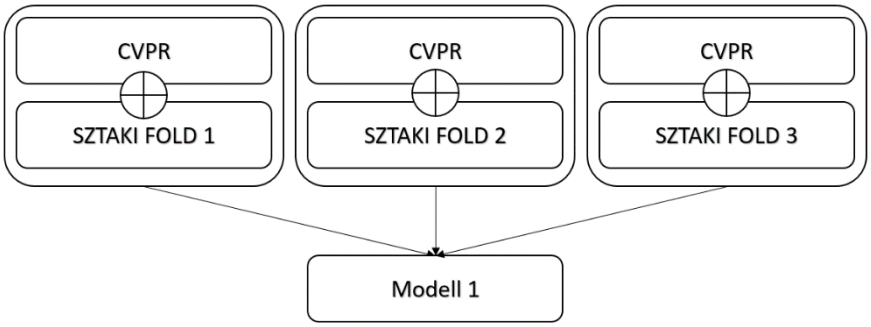
6.ábra: augmentációs műveletek

7. Saját modellek

Az eddigi két konstrukció (sima és augmentált adatok) tanítása során a modell véletlen állapotból indul. Ezen fejezetben a transfer learning lehetséges megvalósításait vizsgálom meg, a CVPR adathalmazt minden esetben előzetesen ismert halmaznak tekintem és a CVPR repülő és drón felvételeken betanított modelleket mint már stabil állapotú neurális hálózatokat tanítjuk tovább. A feladat kihívása, hogy a kapott modellek önmagukban is értékelhetőek (ld. Kiértékelés). A cél, hogy a SZTAKI saját adatait felhasználva növeljük a hálózat generalizálási képességét [1,30]. A tanító halmazokat kétféleképpen kombináljuk, illetve a következő fejezetben a tanulási algoritmusban is változtatást eszközölünk.

Együttes tanítás

Az architektúra lényege, hogy a CVPR és a saját adatainkat egybevéve végezzük a tanítást. Előbbit 70%-30%-os tanítási és validációs aránnyal, míg utóbbit a korábban alkalmazott 3-fold cross validation szerint osztjuk fel. Az így létrejött 3 konvolúciós hálózatot egyenként teszteljük és mérjük az eredményeit, majd több kiértékelési metrikával összevonjuk a teljesítményüket, amely tükrözi a kombinációból elérhető javulást.

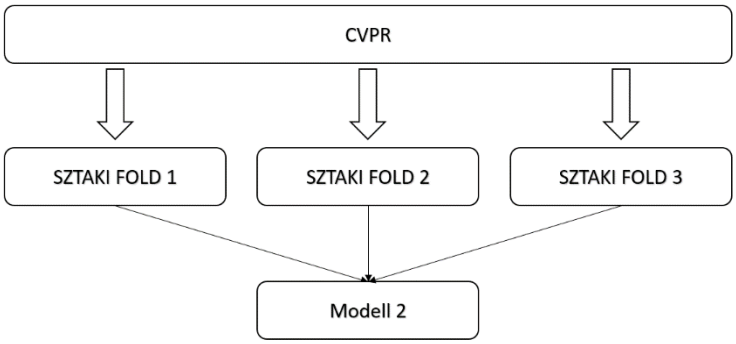


7. ábra: Együtt tanítás cross validation technikával

Előtanítás és finomhangolás

A CNN modellt először a CVPR adatokon tanítjuk be, 70%-30%-os tanítási és validációs arányt alkalmazva. Ezután a korábbi 3-fold cross validation szerint felosztott saját képeken finomhangoljuk a hálót. A betanult konvolúciós hálózat a súlyaiban őrzi a CVPR felvételekből kinyerhető feature-öket, ez segíti a továbbtanulást. A transfer learning módszerrel ellentétben most nem rögzítjük a konvolúciós rétegek súlyait, egyrészt a modell kis mérete miatt, másrészt nem az a cél, hogy egy hatalmas, többosztályon betanított modellt áthangoljunk

a saját többosztályos, hasonló domain-nel rendelkező feladatunkba felhasználva a kinyert jellemzőket, hanem ugyanabba a két csoportba tartozó adatokat finomítjuk és bővítjük.



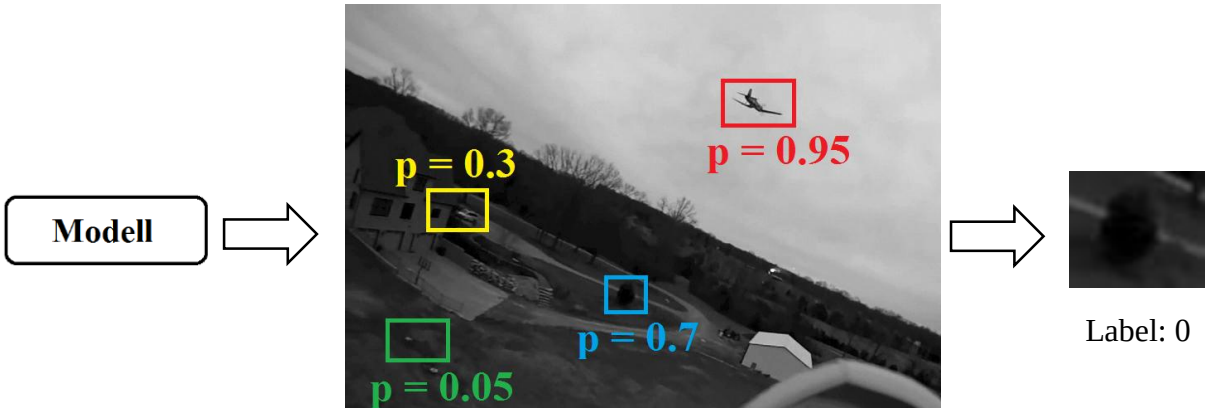
8. ábra: Finomhangolás cross validation technikával

Tanítási paraméterek

A fenti kutatás tapasztalataiból a következő paraméterekkel végezzük a tanítást. A mini-batch méretet 30-ra állítottuk, ADAM módszerrel optimalizált SGD tanítást végzünk az ajánlott alap paraméterekkel. Minden epoch után validációt végzünk és a túltanulás elkerülése végett korai megállítást alkalmazunk, ha a mért hibaérték elkezd rohamosan növekedni a tanítási hibához képest.

A tanítási módszer változtatása

A kutatás jelenleg itt tart, ebből a fázisból még nincsenek módszeresen végzett mérések, a fejlesztés és a várható javulás tesztelése zajlik. A konvolúciós hálózatot úgy tanítjuk, hogy a tanítás során olyan negatív mintákat képzünk, hogy a képről sok véletlenszerű mintát veszünk, amelyekre a súlyok aktuális állapota szerint a legnagyobb az esély, hogy UAV egység van rajta. Az így kapott legnagyobb értéket, azaz azt amelyikre leginkább azt mondta tévesen, hogy van rajta UAV, a hálózat bemenetére adjuk, mint 0-ás címkével rendelkező adat.



9. ábra Negatív minták gyűjtése

8. Kiértékelés

Osztályozási metrikák

Két osztály esetén a következő lehetőségeink vannak egy tesztképre adott válasz kiértékelésekor (1: positive, 0: negative):

- True positive (TP): a képet helyesen 1-es címkével láttuk el
- False positive (FP): a kép helytelenül kapott 1-es címkét, 0 a jó válasz
- False negative (FN): a kép helytelenül kapott 0-ás címkét, 1 a jó válasz
- True negative (TN): a képet helyesen 0-as címkével láttuk el

Ezeket felhasználhatjuk az úgynevezett precision és recall értékek kiszámításához.

$$Precision = \frac{TP}{TP + FP}$$

A jól eltalált 1-es címkéjű képek számát elosztjuk az összes pozitívnak jelölt képek számával.

$$Recall = \frac{TP}{TP + FN}$$

Ezt a mérőszámot úgy kapjuk meg, hogy most a jól eltalált 1-es címkéjű tesztképek számát az összes eredetileg 1-es címkével rendelkező kép számával osztjuk, ezt adja a nevező. Szokás még true positive rate-nek (TPR hívni, mert azt fejezi ki, hogy hányat találtunk el jól a pozitív adatokból.

Receiver operating characteristic görbe (ROC) és a görbe alatti terült (AUC)

Az osztályozás eredménye lehet, két diszkrét érték, 0 vagy 1, ha két osztályunk van illetve lehet a folytonos értek 0 és 1 között, ahogy a mi modelleink is működnek. Ekkor megállapíthatunk egy küszöbértéket (threshold), amely elválasztja a két osztályunkat. Egyik a küszöb feletti, másik a küszöb alatti tartományt jelenti. A ROC görbe azt mutatja meg, hogy ezt az értéket változtatva hogyan változik a TPR és az analóg módon definiált false positive rate (FPR) aránya.

A két tengelyen lévő mérték 0 és 1 között változhat, a (0,0) és (1,1) pontot összekötő egyenes jelenti a két érték egyenlő arányát, ez tulajdonképpen a véletlenszerűen tippelő modell mutatója, hiszen két osztályba 50-50% eséllyel lehet ily módon kategorizálni. A vonal alatti

értékkel rendelkező modellek rossznak tekinthetőek, a cél, hogy a vonal felett legyen legalább a görbe egy része, attól függően, hogy mennyire vagyunk megengedőek a FP értékekkel az alkalmazott környezetben. Például egy banki hitelrendszerben nem szeretnénk, ha tévesen fogadjuk el valaki hitelképességét és utána nem fizeti vissza a pénzünket.

A lehető legjobb érték a bal felső sarok, a (0,1) pont, ekkor nincs hibásan pozitívnak jelölt adat. Minél inkább közelíti ezt a pontot a modellünk ROC görbéje, annál inkább tekinthető "jónak".

A görbe alatti terület úgy lehet értelmezni, hogy azt a valószínűséget adja meg, hogy egy véletlenszerűen választott pozitív adat nagyobb értéket kap, mint egy ugyan így választott negatív. Tehát a (0,1) skálán nagyobb értéke lesz, ami jó, ha továbbra is azzal a feltételezéssel élünk, hogy a pozitív adatokat jelöljük a skála 1-es végén.

Normalized Discounted Cumulative Gain (NDCG)

Ez a metrika egy modell rangsorolási képességét értékeli. Általában keresőmotorok és ajánlórendszerek hatékonyságát szokták vele mérni. A mi esetünkben értelmezhetjük a következőképpen. A CNN által a tesztképekre adott értékeket sorrendbe állítjuk és az indexük alapján megnézzük a címkék tömbjében, hogy valóban pozitív, azaz releváns adat volt-e vagy sem.

A DCG érték egy p pozícióban:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

ahol rel_i a kiértékelt eredménye, relevanciája a tesztelt adatnak. Ezáltal azt érjük el, hogy amikor a modellünk azokat a képeket, amelyek nagy relevanciával rendelkeznek, azaz 1-es címkéjű kéne legyen, de mégis hátra került, azaz inkább 0-ás címkével láttuk el (threshold függő mit tekintünk jónak) hátra sorolja, büntessük, illetve a kis relevanciájúak ne kerüljenek előre.

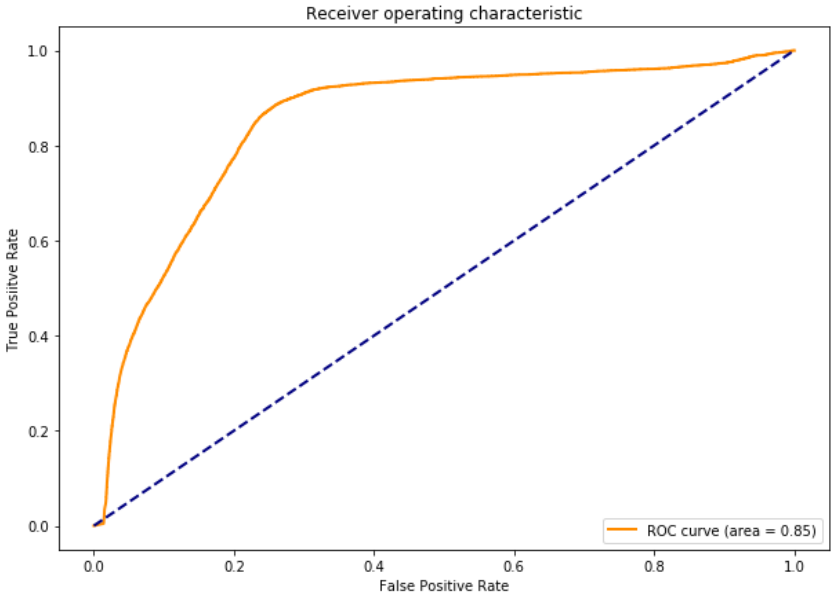
A normalizált DCG értéket a következőképpen lehet kiszámolni egy p pozicióban:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

Ahol az IDCG az ideális GCD értéket jelenti, azaz csak a pozitív értékeken kiszámolt DCG értéket. A különböző skálájú eredmények összehasonlítása végett kell normalizálni.

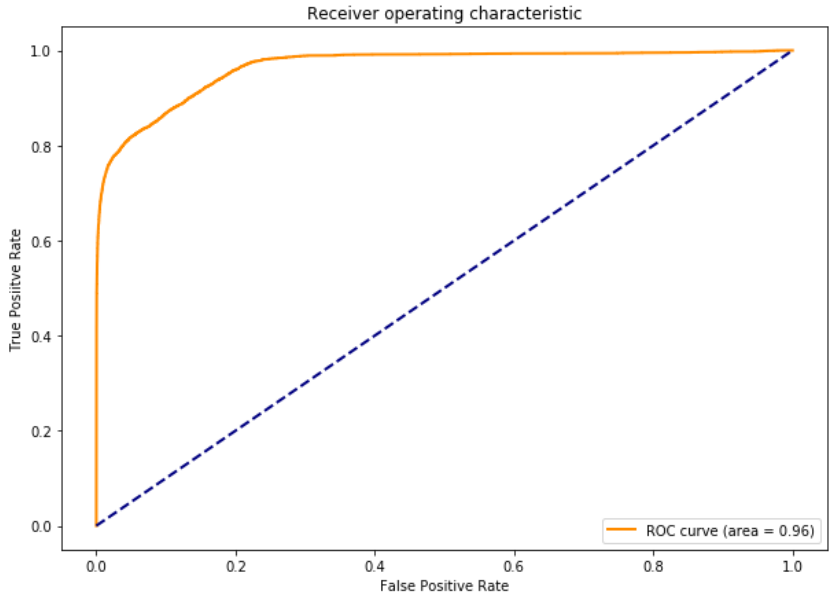
Eredmények

Az alap modell ROC-AUC értéke, az összes CVPR adaton tanítva és a SZTAKI saját felvételein tesztelve:



10. ábra: Alap modell a SZTAKI adaton tesztelve

Az alap modell ROC-AUC értéke, az összes CVPR adaton tanítva és a validációs képeken kiértékelve:

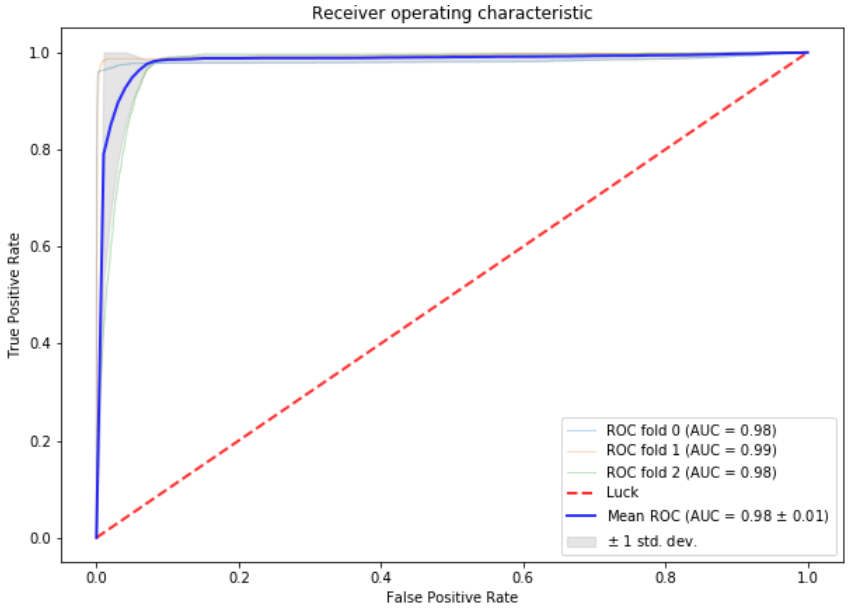


11. ábra: Alap modell a CVPR adaton tesztelve

Látható, hogy a CNN a tanult adatokon nagyon jó eredményt ért el (AUC = 0.96), nagyon meredeken megy fel a görbe az elején, ez azt jelenti, hogy a threshold változtatásával továbbra is sok TP értéket kapunk és kevés FN-at. Körülbelül 0,8-as TPR értéknél lépjük át azt a határértéket, amellyel elkezd nőni a FPR. Ahhoz, hogy megközelítsük az maximum TPR-t,

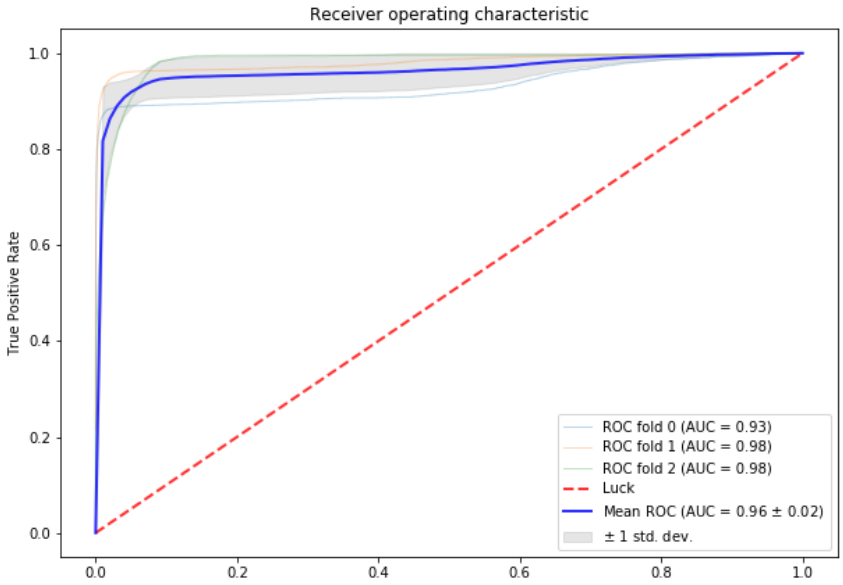
ez körülbelül 30%-os FPR értékkel jár. Ez remek validációs pontosság, de a modell valódi képességét a tiszta tesztadatokon lehet mérni, amely a fenti ábrán látható. Ott hasonló FPR mellett érjük a csúcsot, amely 0,9-es TPR értéket jelent, tehát minden 10. képet rosszul tippelünk 1-esnek. Ez után a pont után már sokkal jobban növekszik a FPR, így már nem éri meg a thresholdot változtatni. A másik két modell ezen az eredményen hivatott javítani.

CVPR és SZTAKI adatok együttes tanítása, utóbbi 3-fold cross-validation-nel, előbbi a 70-30%-os arányban a teljes cvpr adathalmazból:



12. ábra: Együtt tanítás cross-validation technikával

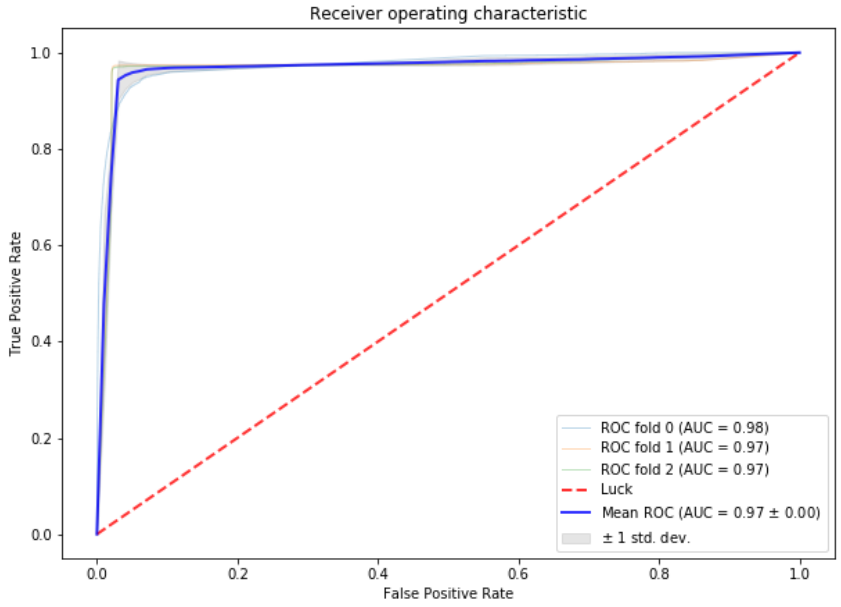
A 3 futás átlagos eredménye 0.98-as AUC érték (fenti kép) Ez jelentős javulás az alap modellekhez képest, miközben a validációs adatokon sikerült megtartani az átlagos AUC-ot (lenti kép). A modell körülbelül 0,1-es FPR rátánál éri el a legjobb TPR értékét, de az (1,1)



13. ábra: Együttes tanítás validálása

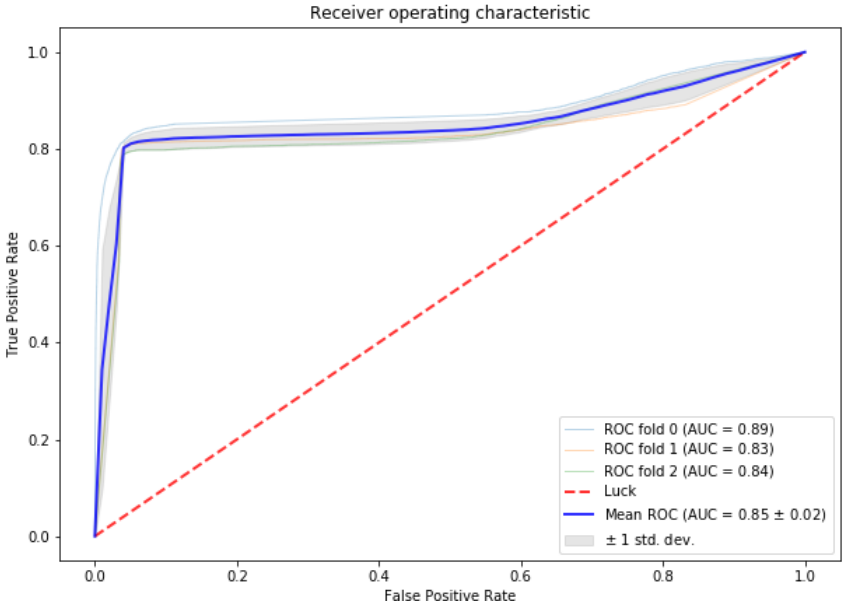
pontig nem éri el azt, így mindig lesznek FN predikciók. Kerekítve 1000 képből 9 hibás. A felvétel 24 fps (frame per second) sebességű tehát kb. 40 másodpercenként 9 képkockán vét FN hibát.

CVPR adatokon előtanítva, majd SZTAKI képeken tovább finomítva, előbbi 70-30%-os felosztásban, utóbbi 3-fold cross-validation-nel:



14. ábra: Finomhangolás cross-validation technikával

CVPR adatokon előtanítva, majd SZTAKI képeken tovább finomítva, előbbi 70-30%-os felosztásban, utóbbi 3-fold cross-validation-nel, CVPR validációs eredmény:



15. ábra: Finomhangolás validálása

Ez a modell az előbbihez hasonló 0,97-es átlagos AUC eredményt ért el, ellenben a validációs adatokon romlott a teljesítménye. Nem csak az AUC érték csökkent, hanem 0,8-as

TPR érték után drasztikusan nő az FPR és nem nő az előbbi, tehát sok a FN predikció. A konvolúciós hálózat bár még nem tanult túl, de mivel a finomhangolás során csak a SZTAKI saját adathalmazát kapta, amelyben nincs a CVPR-hoz hasnoló drón felvétel, csak az előbbieket tároló súlyok erősödtek, sajnos utóbbiak rovására. Ellenben sokkal gyorsabban konvergált a tanítás, átlagosan 30 epoch kellett ehhez, míg a másik modellnek 50. Tehát előnye, hogy gyorsabban tanul, illetve előrevetíti azt a lehetőséget, hogy a jövőben mindig frissítse a modellt a felvételek alapján a UAV egység. Ezt valamely korlátozással bevezetve, hogy ne ronsta a korábbi eredményeket, vagy ahogy fejlődnek a hardveres teljesítmények, lehet növelni a hálózat méreteit és paramétereit.

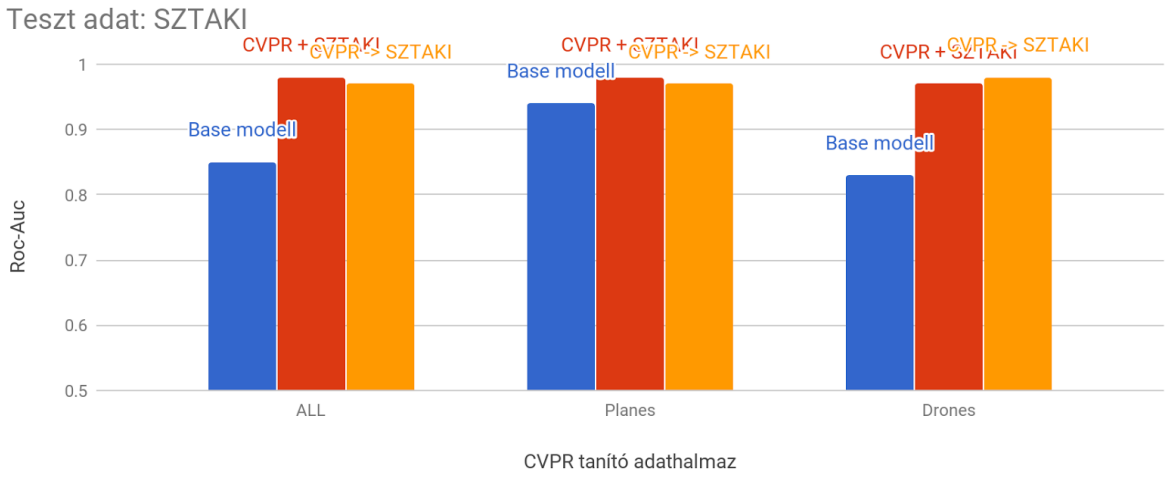
Modellek összehasonlítása

Mindegyik modellt háromféle módon tanítottunk. A fenti az elsőféle, amelyben az összes CVPR adatot használtuk alapnak, a második, harmadikban pedig vagy csak a repülőket, vagy csak a drónokat.

A jelölések:

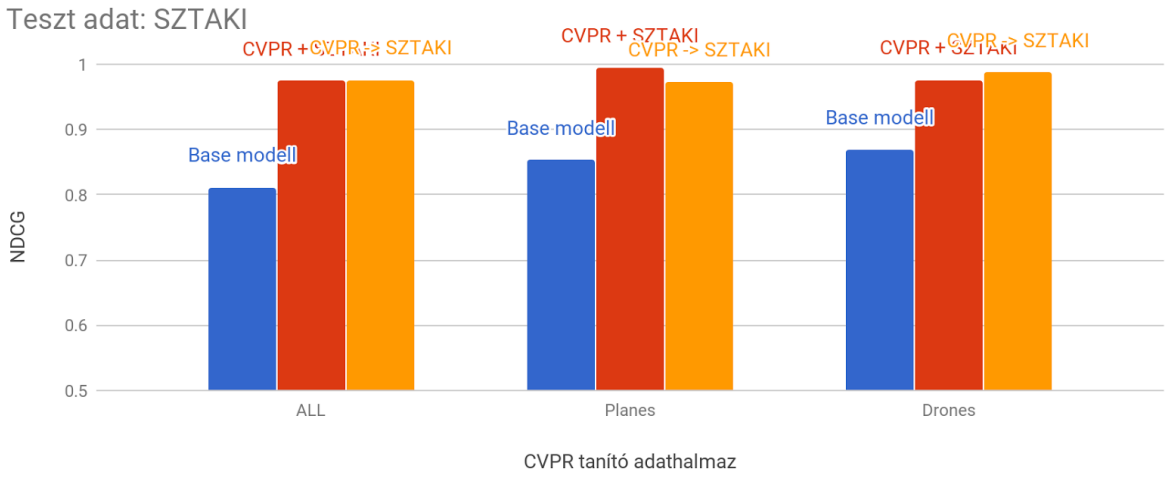
- Az alap modell: base modell, kék szín
- Együtt tanítás: + jellel, piros szín
- Finomhangolás: -> jellel, sárga szín

Az összes tanítóhalmazra külön megtalálható a ROC-AUC és a NDCG eredmény a következő táblázatokban. Az első kettő a SZTAKI adatokon tesztelve, utóbbi kettő a CVPR validációs képekkel.



16. ábra: A modellek ROC-AUC eredménye a SZTAKI adatokon

A legjobb eredményt a teljes tanítóhalmazon a fent kifejtett CVPR + SZTAKI együttes tanítás eredményezte a ROC-AUC értékekben, nem sokkal lemaradva a finomhangolt konvolúciós hálózattól. A NDCG mérésben ez a apró különbség is elvész. Pusztán a repülős adathalmazon tanítva születtek ebben a kategóriában a legjobb eredmények, mert ez a két felvételsorozat jobban hasonlít egymásra, mint a harmadikra, ezért eleve az alap modell egész jó teljesítményt nyújt. Az utolsó metodika szerint a CVPR -> SZTAKI hálózat lett a legjobb mind ROC-AUC, mind NDCG értékben.



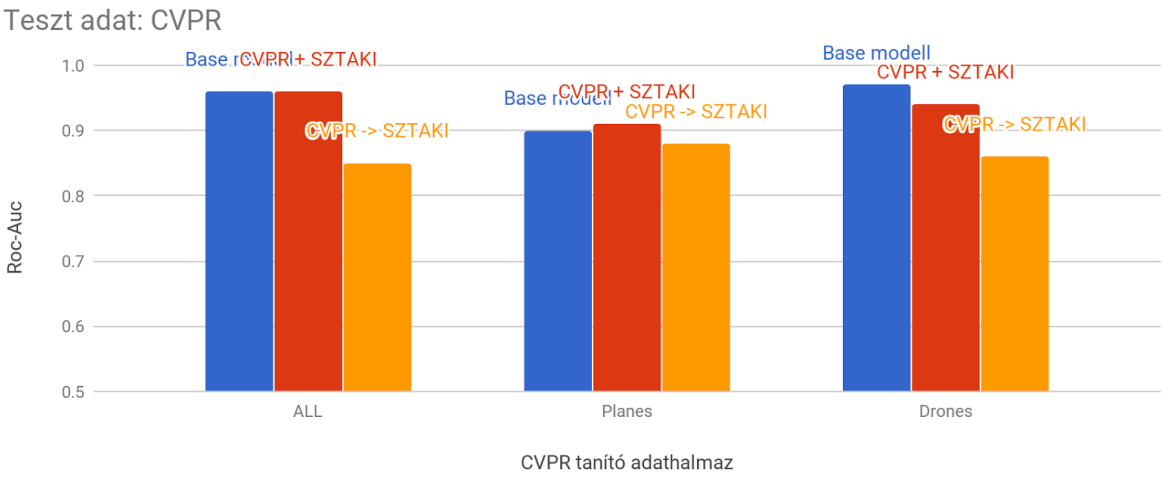
17. ábra: A modellek NDCG eredményei a SZTAKI adatokon

Látható, hogy összességében mindkét komplexebb tanítás jelentős javulást hozott a tesztelés során annak ellenére, hogy a képfelismerési problémák között ez a könnyebbek közé tartozik. A konvolúciós hálózat viszont továbbra is maradt egy viszonylag kicsi, kompakt modell, amely könnyen alkalmazható a mai UAV rendszerek által nyújtott hardveres környezetben.

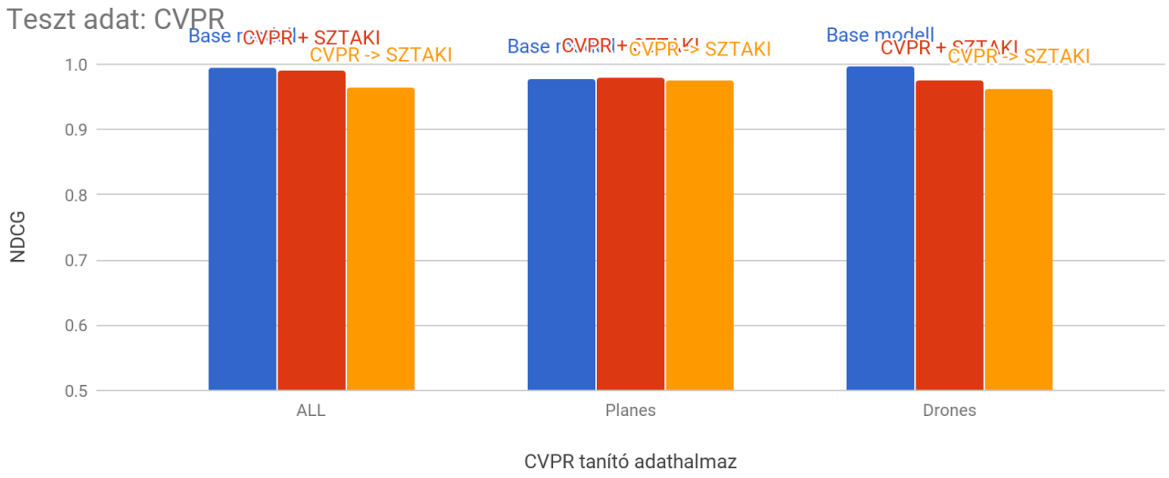
Validáció

A ROC-AUC érték a teljes halmazon a korábbi görbékről ismert, csak repülős és csak drónos CVPR felvételeken mindkét kombinált modell veszít az eredményéből az alaphoz képest, de míg az előbbinél körülbelül megtartják a kívánt szintet, utóbbinál főleg a finomhangolt verzió szerepel gyengébben.

A NDCG metrikát figyelve azt tapasztaltuk, hogy az összes modell közel azonos teljesítményt nyújt a validációs adatokon, jól megtanulva azokat.



18. ábra: A modellek ROC-AUC értéke a validációs képeken



19. ábra: A modellek NDCG értéke a validációs képeken

9. Összefoglalás

A kutatás célja az volt, hogy megmutassam, hogy a SZTAKI Avoid and Sense nevű projektjében alkamazhatunk konvolúciós neurális hálózatokat UAV egységek felismerésére a beágyazott rendszer által állított korlátozások figyelembevételével.

Ennek elérése érdekében először áttekintettem a szakirodalom erre vonatkozó részeit, és a talált lehetőségek közül válogattam a problémák megoldására. A kutatáshoz és fejlesztéshez a Caffe Deep Learning keretrendszert használtam, mely különös hangsúlyt fektet a képfelismerési feladatok optimalizációjára.

A kutatás elején megkerestem az ideális architektúrát, figyelembe véve, hogy a rendelkezésre álló adat ritka, illetve a célhardver, amelyen futnia kell a végső betanult modellnek, korlátozott számítási képességekkel rendelkezik. Az adatbázist növeltem augmentációs módszerekkel, és a tanításhoz cross-validationt alkalmaztam. Ezek után a rendelkezésünkre álló külső adatot felhasználva többféle, a tanulást javító architektúrát dolgoztam ki az osztályozás javítása céljából.

Az eredmények igazolták a modellek eredményességét, bár a konvolúció neurális hálózatoknak viszonylag egyszerű a probléma megoldása, kellő módszerekkel ez továbbfejleszthető. Kisebb javításokat eredményezhet még a tanítás hiperparamétereinek további finomítása, ám ezt majd a projekt végső modelljénél érdemes elkészíteni.

Az általam ajánlott két módszert összehasonlítottam két mérőszám segítségével is. Sikerült 0,98-as ROC-AUC pontszámot elérni. Miután a jelenlegi modell feltételez költséges előfeldolgozási lépéseket, a jövőben szeretnénk megvizsgálni, hogy az előfeldolgozási lépések kiválthatóak-e [10,25,31,33], milyen formában lehetséges kihasználni a felvételek belső dinamikáját [13,16] és milyen előnyökkel jár a dolgozatban említett környezet kalibrációs módszer.

10. Irodalomjegyzék

[1] Bartlett, P. L. and Maass, W. (2003). Vapnik-chervonenkis dimension of neural nets. The handbook of brain theory and neural networks, pages 1188–1192.

[2] Benenson, R., Mathias, M., Timofte, R., and Van Gool, L. (2012). Pedestrian detection at 100 frames per second. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2903–2910. IEEE.

[3] Bengio, Y. and Delalleau, O. (2011). On the expressive power of deep architectures. In International Conference on Algorithmic Learning Theory, pages 18–36. Springer.

[4] Bojarski, M., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U., and Zieba, K. (2016a). Visualbackprop: visualizing cnns for autonomous driving. arXiv preprint arXiv:1611.05418.

[5] Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision, pages 2722–2730.

[6] Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3):273–297

[7] Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In Workshop on statistical learning in computer vision, ECCV, volume 1, pages 1–2. Prague.

[8] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE.

[9] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255 IEEE.

[10] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE Interna- tional Conference on Computer Vision, pages 1440–1448.

[11] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into recti- fiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034.

[12] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778.

[13] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.

[14] Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayam- pallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al. (2015). An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716.

[15] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accel- erating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[16] Isele, D., Cosgun, A., and Fujimura, K. (2017). Analyzing knowledge transfer in deep q-networks for autonomously handling multiple intersections. arXiv preprint arXiv:1705.01197.

[17] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[18] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Ima- genet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

[19] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324.

[20] Lin, H. W. and Tegmark, M. (2016). Why does deep and cheap learning work so well? arXiv preprint arXiv:1608.08225.

[21] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359.

[22] Papageorgiou, C. P., Oren, M., and Poggio, T. (1998). A general framework for object detection. In Computer vision, 1998. sixth international conference on, pages 555–562. IEEE.

[23] Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. IEEE.

[24] Pomerleau, D. A. (1989). Alvinn, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Computer Science Department.

[25] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99.

[26] Rolnick, D. and Tegmark, M. (2017). The power of deeper networks for expressing natural functions. arXiv preprint arXiv:1705.05502.

[27] Rozantsev, A., Lepetit, V., & Fua, P. (2017). Detecting flying objects using a single moving camera. *IEEE transactions on pattern analysis and machine intelligence*, *39*(5), 879-892.

[28] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

[29] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958.

[30] Vapnik, V. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability & Its Applications, 16(2):264–280.

[31] Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignateva, A. (2015). Deep attention recurrent q-network. arXiv preprint arXiv:1512.01693, Deep Reinforcement Learning Workshop, NIPS 2015.

[32] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I–I. IEEE.

[33] You, Y., Pan, X., Wang, Z., and Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. arXiv preprint arXiv:1704.03952.

[34] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

[35] Péter Bauer, P. and Hiba, A. and Daróczy, B. and Melczer, M. and Vanek, B. Real Flight Demonstration of Monocular Image-Based Aircraft Sense and Avoid, 2017, ERCIM News 110

11. Köszönetnyilvánítás

Szeretném elsősorban konzulenseimnek, Ács Juditnak és Daróczy Bálintnak megköszönni az álhatatos munkájukat és a segítségnyújtásukat. Továbbá Aleksziev Rita munkatársamnak tartozok köszönettel, aki az utolsó utáni pillanatban is segített a célom elérésében. Végül, de nem utolsósorban hálás vagyok nővéremnek, Szabó Andreának a támogatásáért.