

Debreceni Egyetem
Műszaki Kar
Mechatronika tanszék



Autóvezetést támogató rendszer fejlesztése számítógépes látás és mélytanulás alkalmazásával

Szerző: Ari László MSc hallgató

Témavezető: Sarvajcz Kornél
Tanszékvezető: Dr. Husi Géza

Debrecen
2021.

TARTALOMJEGYZÉK

Tartalmi összefoglaló	1
Bevezetés.....	2
1. Autóvezetést támogató rendszerek	3
1.1. Miért van szükség rájuk?	3
1.2. Példák vezetéstámogató rendszerekre	3
1.2.1. Adaptív sebességtartó automatika.....	3
1.2.2. Holttér felderítés.....	3
1.2.3. Parkolási asszisztens	4
1.2.4. Sávelhagyásra figyelmeztető rendszer	4
1.2.5. Vészfék asszisztens	4
1.2.6. Beépített táblafelismerő rendszer	4
1.3. Szenzorok	4
1.3.1. Ultrahang szenzor.....	5
1.3.2. Radarérzékelők.....	5
1.3.3. Lézeres távolságmérők.....	5
1.3.4. Kamera alapú érzékelők	6
1.4. Összetett adatforrásból való jelfeldolgozás	6
1.5. Vezetéstámogató rendszerek fejlesztésével kapcsolatos kutatások.....	8
2. Számítógépes látás mélytanulással	11
2.1. Mesterséges intelligencia.....	11
2.2. Gépi tanulás	12
2.3. Mélytanulás	13
2.4. Hogyan dolgozza fel az információt az emberi agy?	14
2.5. Mesterséges neurális hálózatok	14
2.6. Konvolúciós neurális hálózatok.....	16
3. Kamera alapú autóvezetést támogató rendszer tervezése	18
3.1. Kutatómunkám során elért eredmények	18
3.2. Felmerült problémák.....	18
3.3. Célkitűzések.....	19
3.4. Felhasznált eszközök	19
4. Feladat kivitelezése	21
4.1. Jetson Nano konfigurálása.....	21

4.2.	Adathalmaz előkészítése.....	22
4.3.	Neurális háló mélytanítása.....	25
4.4.	Modell pontosságának meghatározása	29
5.	Tesztelés valós körülmények között	32
5.1.	Tesztelés nappal.....	32
5.2.	Éjszakai üzemmód tesztelése.....	34
6.	Továbbfejlesztési lehetőségek	37
7.	Összefoglalás	38
	Köszönetnyilvánítás	39
	Hivatkozások	40

TARTALMI ÖSSZEFOGLALÓ

Az alábbi TDK dolgozat egy kamera alapú autóvezetést támogató rendszer továbbfejlesztését mutatja be, mely előző kutató munkám tárgyát képezte. A rendszer egy Raspberry PI 4B egykártyás számítógépen futott, a program számítógépes látás által képes érzékelni a gyalogosokat, a gyalogátkelőhelyeket, illetve az elsőbbségadásra felszólító KRESZ táblákat, majd a felismerést követően figyelmeztető jelzést küld a gépjárművezető számára, értesítve őt elsőbbségadási kötelezettségéről.

A dolgozatom első felében bemutatom napjaink legnépszerűbb autóvezetést támogató rendszereit, ezek fejlődését, működési elvét, majd a későbbiekben ismertetem a számítógépes látás alapjait, illetve, hogy milyen folyamatok szükségesek ahhoz, hogy a számítógépek értelmezzék a képek kontextusait és felismerjék a tárgyakat.

Dolgozatom második felében szemléltetem a rendszerem működési elvét, illetve bemutatom kutatási témám során elért eredményeket, rávilágítok a tesztelések során felmerülő problémákra és kísérleteket teszek ezeknek a problémáknak a minimalizálására, optimalizálom a rendszerem működését.

BEVEZETÉS

Napjainkban az autó, mint közlekedési eszköz egyre nagyobb teret hódít, egy emberi alapszükségletté vált. Évről évre egyre több autó kerül forgalomba, mely a közutak zsúfoltságához és a közúti balesetek számának megnövekedéséhez vezet. Továbbá a vezetés, mint mindennapi tevékenységünk, egy elég összetett feladat, nagy mennyiségű interakcióval az autó és a gépjárművezető között. Figyelmünket az állandóan változó forgalom mellett, a tükrök, a műszerfal, illetve a perifériáson történő eseményekre párhuzamosan kell összpontosítsuk, emellett előre kell gondolkozzunk, kiszámolva partnereink viselkedését, majd döntéseket hozni, mindezt nagyon gyorsan, kevés idő alatt. Emellett a vezetési környezet bizonytalansága (rossz látási viszonyok, csúszós útfelület) tovább növeli a vezetés veszélyességi faktorát. Ráadásul, ha a gépjárművezető a figyelmét még megosztja olyan másodlagos feladatokkal, mint például a beszélgetés vagy a kényelmi beállítások elvégzése, jelentősen megnő a baleset bekövetkeztének kockázata.

TDK dolgozatom célja egy olyan számítógépes látáson alapuló autóvezetést támogató rendszer fejlesztése, mely valós időben, városi tempóban haladva, egy fedélzeti kamera segítségével, legalább 90%-os pontossággal, az emberi reakcióidőt felülmúló sebességgel, képes érzékelni a gyalogosokat, a gyalogátkelőhelyeket, illetve az elsőbbségadásra felszólító KRESZ táblákat. A rendszer a felismerés után vizuális alapú figyelmeztetést jelenít a gépjárművezető számára, értesítve őt elsőbbségadási kötelezettségéről. Egy ilyen biztonsági funkcióval minimalizálható lenne az emberi figyelmetlenségből adódó gázolások és elsőbbségadás elmulasztásából származó balesetek megtörténi valószínűsége, továbbá óriási segítséget nyújtana a sofőrnek nehezebb látási viszonyok között a tájékozódásban.

Dolgozatom további részében megismerkedhetünk napjaink legmodernebb vezetéstámogató rendszereivel, továbbá megtapasztalhatjuk, hogy milyen folyamatok szükségesek ahhoz, hogy a gépek felismerjék az embereket és a közúti jelzéseket. Szemrevételezhetjük az emberi agy és a számítógépek agya közötti hasonlóságokat és ismertetem a mesterséges neurális hálózatok működési elvét. Megvilágítom a feladatom megvalósítása közben felmerülő akadályokat és ezeknek a problémáknak a minimalizálására szolgáló módszereket.

1. AUTÓVEZETÉST TÁMOGATÓ RENDSZEREK

1.1. MIÉRT VAN SZÜKSÉG RÁJUK?

Az Egészségügyi Világszervezet (WHO) 2018-as jelentése szerint évente körülbelül 1,35 millió ember hal meg közlekedési balesetek következtében, ez naponta közel 3700 halálesetet jelent. Továbbá évente több tízmillió ember sérül meg, melyek közül rengetegen életet megváltoztató, hosszan tartó károsodást szenvednek. A jelentés egyik legszomorúbb statisztikája, hogy a balesetek leggyakoribb áldozatai az 5-29 év közötti emberek [1].

Ebből kifolyólag egyre több kutatás zajlik a vezetést támogató rendszerek és önvezető járművek fejlesztésével kapcsolatban. Az autógyártó cégek mostmár inkább az aktív biztonsági rendszerek fejlesztésére összpontosítanak a passzívak helyett. A passzív biztonsági rendszerek, mint például a biztonsági öv, légzsák, párnázott műszerfal vagy a gépjármű szerkezete, védelmet nyújtanak ütközés esetén. Az aktív biztonsági rendszerek célja, hogy a gépjárművezető eleve elkerülje a baleseteket. Mivel a legtöbb közúti baleset emberi hiba miatt következik be, így a nehéz vagy ismétlődő műveletek automatizálásával csökkenthető az emberi figyelmetlenségből eredő balesetek száma. Ide sorolhatók a különféle autóvezetést támogató rendszerek (ADAS – Advanced Driving Assistance System), melyek célja, mint ahogy a neve is mutatja, a gépjárművezető segítése. [2] [3].

1.2. PÉLDÁK VEZETÉSTÁMOGATÓ RENDSZEREKRE

1.2.1. ADAPTÍV SEBESSÉGTARTÓ AUTOMATIKA

Az ACC (Adaptive Cruise Control) képes automatikusan beállítani a jármű sebességét az aktuális forgalmi helyzetekhez, korlátozásokhoz viszonyítva. Továbbá biztonságos követési távolságot nyújt az előttünk haladó gépjárműtől. Figyelmeztető jelzést szolgáltat, ha fennáll az ütközés veszélye, amennyiben nem reagál időben a gépjárművezető, a rendszer közbeavatkozik. [4]

1.2.2. HOLTTÉR FELDERÍTÉS

A BSD (Blind Spot Detection) érzékeli a jármű mellett, illetve mögött, a holttérben levő járműveket a visszapillantótükrökbe beépített szenzorok segítségével. [4]

1.2.3. PARKOLÁSI ASSZISZTENS

Az autó köré felszerelt ultrahang szenzorok által egy olyan széles detektálási zóna van kialakítva, mint maga a jármű. Az érzékelők által szolgáltatott jelek függvényében hangjelzéseket („beep-beep”) szolgáltat a rendszer a gépjárművezető számára, mely frekvenciája folyamatosan növekszik, ha az autó egyre közelebb kerül az akadályhoz. Amennyiben a gépjármű rendelkezik „önparkoló” asszisztenssel, az autó magától is képes beparkolni a sofőr beavatkozása nélkül. [4]

1.2.4. SÁVELHAGYÁSRA FIGYELMEZTETŐ RENDSZER

Az LWDS (Lane Departure Warning System) kamera segítségével figyeli a forgalmi sáv két szélét határoló szaggatott vagy folytonos fehér felfestéseket, miközben folyamatosan ellenőrzi, hogy az autó hossz tengelye párhuzamos-e velük. Amennyiben azt érzékeli, hogy a gépjármű elhagyni készül a sávot, anélkül, hogy a sofőr jelezte volna erre való szándékát (irányjelző használatával), hang alapú jelzéssel, illetve a kormány vagy az ülés megrezegtetésével figyelmezteti a gépjárművezetőt. [4]

1.2.5. VÉSZFÉK ASSZISZTENS

Az EBA (Emergency Break Assist) az autó elülső részén található érzékelőkkel (radar, LIDAR) kommunikál. Ha az ütközés veszélye fennáll, először figyelmezteti a gépjárművezetőt, amennyiben az nem reagál időben, automatikusan bekapcsol a fékrendszer és megpróbálja megállítani az autót még az ütközés előtt. [4]

1.2.6. BEÉPÍTETT TÁBLAFELISMERŐ RENDSZER

Az RSA (Road Sign Assist) kamera segítségével felismeri a KRESZ táblák jelzéseit és megjeleníti ezeket a kijelzőn vagy a szélvédőre vetítve segít minket abban, hogy egyetlen táblát se hagyjunk figyelmen kívül. A rendszer hang- és fényjelzéssel figyelmezteti a sofőrt, ha nem veszi figyelembe a korlátozásokat. [4]

1.3. SZENZOROK

Az ADAS több forrásból származó bemenő jelekre támaszkodik, mint például ultrahangos érzékelők, kamerák, radarok és lézeres távolságmérők, melyek adatokat gyűjtenek a közúti környezetről, majd egyetlen vagy több információforrásként eljuttatja őket a feldolgozó egységhez. A biztonsági funkciókat úgy tervezték, hogy figyelmeztesse a sofőrt a potenciális veszélyekre. Bizonyos esetekben, ha a gépjárművezető nem reagál időben a

figyelmeztetőjelzésre, a rendszer átveszi az autó irányítását, ezzel elkerülve az ütközést. Az ADAS-ban leggyakrabban használt szenzorok az alábbiak [4] [5]:

1.3.1. ULTRAHANG SENZOR

A gépkocsi lökhárítójába vannak beépítve, melyek működésük az ultrahang-frekvencia tartományba eső hullámokat bocsátanak ki, majd az akadályokról visszaverődő hullámokból állapítják meg annak autóunktól való távolságát. Segítségével érzékelhetjük a gépkocsink mögött levő, esetleg nem is látható akadályokat (pl. magas járdaszegély, kivágott fatörzs, kisgyerek). Parkolást segítő rendszerekben, illetve automataparkoló asszisztensekben használják őket [2] [4].

1.3.2. RADARÉRZÉKELŐK

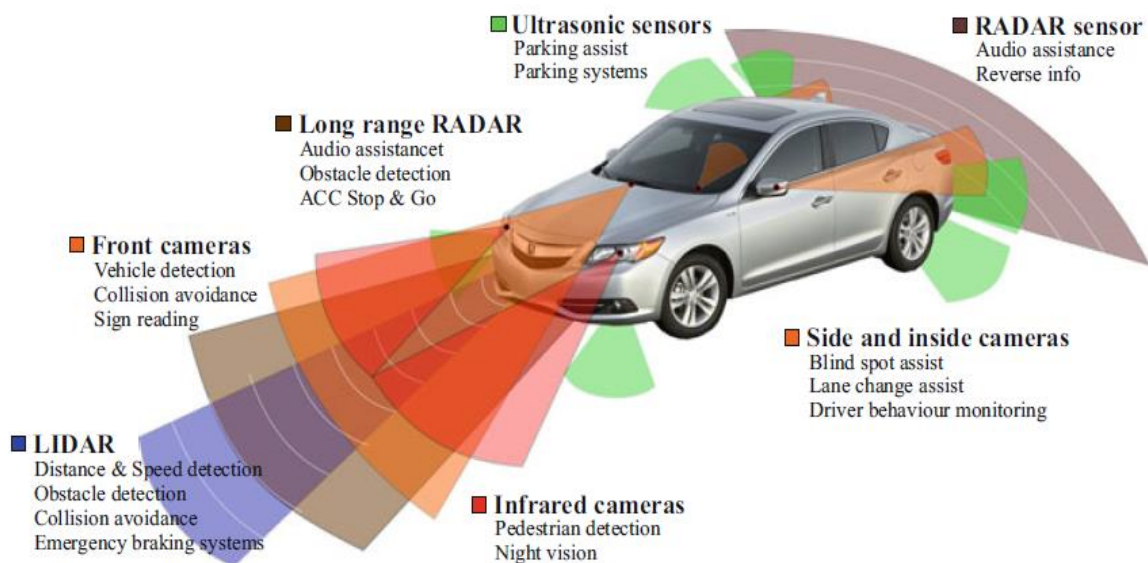
A radarszenzor mikrohullámokat bocsát ki (80 GHz frekvenciáig), melyek visszaverődési ideje adja meg a távolságot köztünk és az előttünk levő akadálytól. Ezek az érzékelők képesek érzékelni a távolságot és a mozgó tárgyak relatív sebességét a beérkező hullámok frekvenciájának változásának mérése által (Doppler effektus). Sebességtartó automatikában (ACC), ütközés figyelmeztető rendszerekben, illetve vészfékasszisztens (EBA) berendezésekben előszeretettel használják őket. A radar alapú rendszer működését nem befolyásolja az eső és a köd, viszont megzavarhatják a hidak, fák, oszlopok, korlátok felületéről visszaverődő jelek, továbbá jelvesztést is szenvedhet a többszörös visszaverődések miatt. [2] [4].

1.3.3. LÉZERES TÁVOLSÁGMÉRŐK

Működési elvük legjobban a radarok működési elvével hasonlítható össze, viszont ez esetben a kibocsátott és fogadott elektromágneses sugárzásnak (infravörös) magasabb a frekvenciája. A LIDAR (Light Detection And Ranging) érzékelő egy lézerdióda által infravörös jeleket állít elő, melynek visszaverődését egy fotodióda érzékeli. A visszatükröződő sugarak intenzitásából, a fáziseltolásból, az impulzus szélességéből és az oda-vissza út megtételéhez szükséges időtartamából nyerjük ki a tárgyak észleléséhez szükséges információkat. Előnye, hogy rendkívül pontos térbeli felbontás (0.1°) érhető el vele, ideális látási viszonyok között a LIDAR rendszerek a legjobbak, viszont az időjárás megváltozásával már megbízhatatlanná válnak [2] [4].

1.3.4. KAMERA ALAPÚ ÉRZÉKELŐK

A kamerák a tárgyfelismerés elengedhetetlen eszközei, ellátják a rendszert bemeneti adatokkal, melyek az utómunkálatok (post processing) során értelmes információkká alakulnak át, lehetővé téve, hogy a mesterséges intelligencia segítségével olyan objektumokat észlelhessünk, mint például egyéb járművek, gyalogosok, illetve KRESZ táblák. A kamerák pontosan meg tudják mérni a szögeket, így például meghatározható, ha egy közeledő jármű valószínűleg kanyarodni fog. Ugyanakkor ahol láthatóak a sávjelzések, egy sávtartó segédberendezés megvalósítása is lehetséges [2] [4].



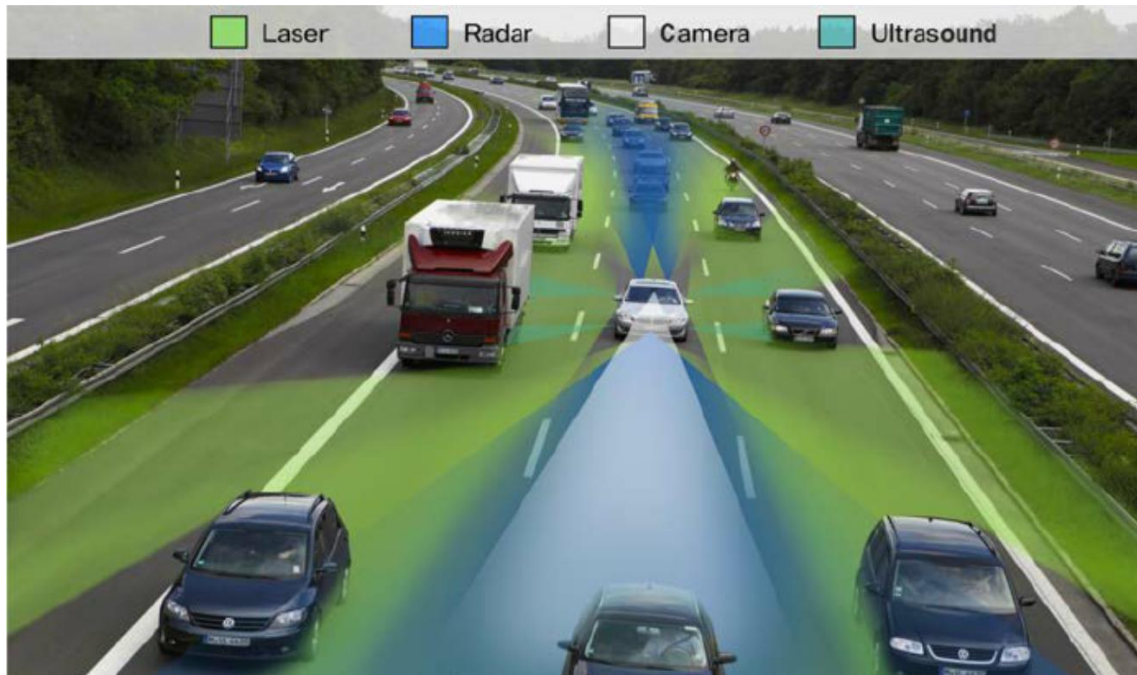
1. ábra: Vezetéstámogató rendszerekben használt szenzorok típusai [2]

A fentebb említett érzékelők mindegyikének megvan az erőssége és a korlátja. Egy összetettebb feladathoz nem elegendő egyetlen szenzor használata, ezek különböző mértékű pontatlansága és korlátozott működési tartománya miatt, ezért több érzékelő szinergikus használatával (MSDF - Multi Sensor Data Fusion) az egyes szenzorok hátrányaiból (pontatlanság, korlátozott működési tartomány, zavaró tényezők) adódó hibák kiküszöbölhetők [6].

1.4. ÖSSZETETT ADATFORRÁSBÓL VALÓ JELFELDOLGOZÁS

Az összetett adatforrásból való jelfeldolgozás (MSDF – Multi Sensor Data Fusion) komplex ismereteket igényel az irányítás- és szabályozástechnikában, a jelfeldolgozásban, a mesterséges intelligenciában, a valószínűségi számításban és a statisztikában. Az MSDF előnyei közé sorolható a redundancia, az információk kiegészítő jellege, illetve ezek időszerűsége. A redundáns információk csökkentik a rendszer bizonytalanságát, ezzel egyidőben megnövelve

annak pontosságát. A több érzékelőből származó kiegészítő információk együttes feldolgozása lehetővé teszi a környezet olyan paramétereinek vizsgálatát, mely lehetetlen lenne az érzékelők külön-külön való használatával. Továbbá több szenzor használata megnöveli a rendszer biztonságát egy érzékelő meghibásodása esetén. Az MSDF végső célja biztonsági zónák kialakítása a jármű körül a különböző szenzorok egy rendszerbe való beágyazásával (2. ábra). [3] [6]

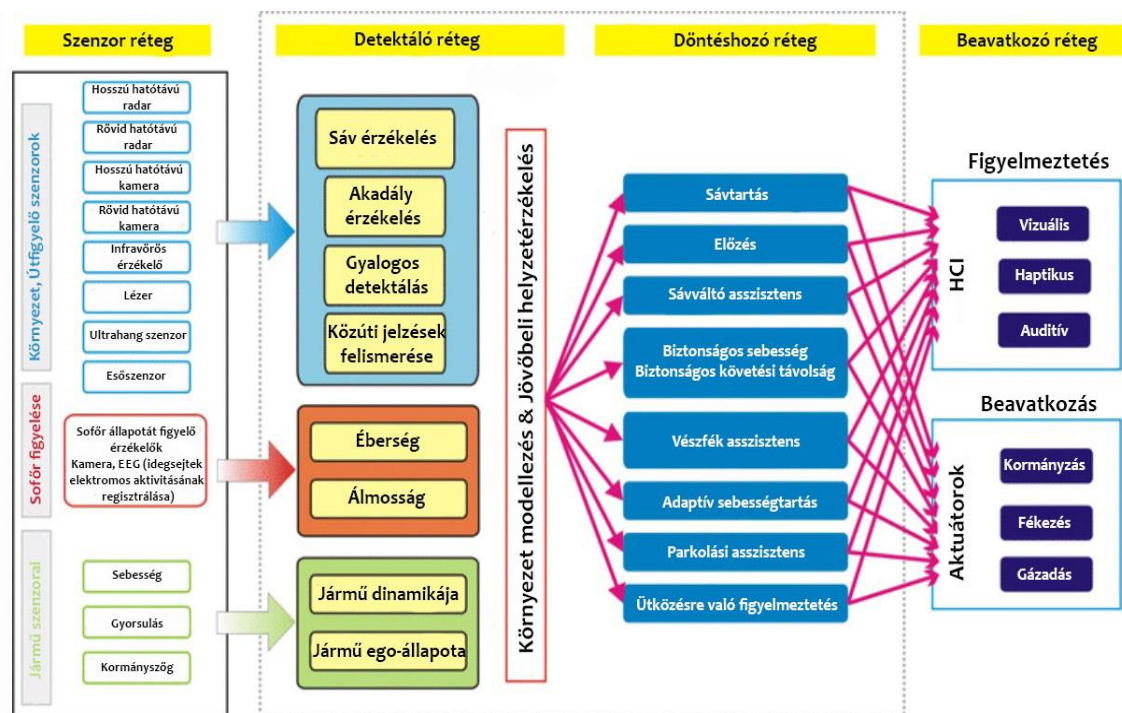


2. ábra: Az érzékelők elhelyezése teljes lefedettséggel a jármű körül [2]

A veszély típusától és jellegétől függően fogja a biztonsági rendszer informálni, figyelmeztetni a gépjárművezetőt, vagy aktívan közbeavatkozni a balesetek elkerülése vagy annak lehetséges következményeinek enyhítése érdekében. Az ADAS vezetéstámogató rendszerek négy fő rétegből tevődnek össze (3. ábra):

- Szenzor réteg (Sensor Array): beépített környezetfigyelő szenzorokból (radar, LIDAR, ultrahang, kamera), sofőr állapotát figyelő érzékelőkből, illetve a jármű dinamikáját monitorizáló szenzorokból (sebesség, gyorsulás, kormányaszög) áll.
- Detektáló réteg (Perception Layer): Célja a gépjármű körüli környezet reális ábrázolása, az akadályok és gyalogosok érzékelése, a sáv és útjelző táblák észlelése, a sofőr éberségének és álmoságának monitorizálása, továbbá a jármű dinamikus viselkedésének vizsgálata az előző rétegben található szenzorok együttes használatával.

- **Döntéshozó réteg (Decision Making Layer):** A közlekedési szituációk alapján és az előző réteg kimenetétől függően döntést hoz, hogy milyen rendszer lépjen érvénybe, ha a baleset kockázatának esélye megnő.
- **Beavatkozó réteg (Action Layer):** Két fő részből áll: HMI (Human Machine Interface), illetve aktuátorok. Ez a réteg felelős a megfelelő figyelmeztető vagy beavatkozó műveletek kivitelezéséért. Az információközlés, a javaslatok, illetve a riasztások egy HCI-n (Human-Computer Interaction) keresztül valósulnak meg. Amennyiben a gépjárművezető nem reagál időben a veszélyre, megtörténik a beavatkozás (gázadás/elvétel, vészfékezés, kormányzás). [3]



3. ábra: Az ADAS rendszerek felépítése összetett adatforrások alapján (saját fordítás) [3]

1.5. VEZETÉSTÁMOGATÓ RENDSZEREK FEJLESZTÉSÉVEL KAPCSOLATOS KUTATÁSOK

A közutak megnövekedett zsúfoltságának és a mai rohanó világnak köszönhetően egyre gyakoribbá váltak a közúti balesetek. Viszont, a folyamatosan fejlődő technológia jobbnál jobb lehetőségeket nyújt a kutatók számára. Ezáltal egyre több tanulmány zajlik a vezetéstámogató rendszerek fejlesztésével kapcsolatban. Az alábbiakban szeretnék bemutatni pár ilyen jellegű kutatást.

Amit Choksi fejlesztése olyan rendszert mutat be, mely képfeldolgozó algoritmusok segítségével segíti a gépjárművezetőt a megfelelően felfestett és kitáblázott útvonalakon. A

kamera érzékeli a sávokat, továbbá van egy szervomotor, amely a kormánykerék vezérlésére szolgál annak érdekében, hogy az autót a saját sávjában tartsa. A képfeldolgozó kód OpenCV-ben íródott és egy Raspberry PI egykártyás számítógépen fut, mely a kormánykerék vezérléséért felelős, a kamera által szolgáltatott bemenőjelek alapján. [7]

Luis Barba-Guaman és társai tanulmányukban egy gyalogos- és gépjárműfelismerő alkalmazást fejlesztenek ki, mely egy dedikált GPU-val rendelkező beágyazott rendszeren, egy NVIDIA Jetson Nano-n fut. Az objektumfelismerés mély neurális hálók által valósul meg. A kutatásban öt különböző előtanított hálót használnak: PedNet, MultiPed, SSD-MobileNet v1 és v2, illetve SSD-Inception v2. A felsorolt neurális hálókat összesen 7150 db képen tesztelték, gyalogosfelismerésre a PedNet és MultiPed modellek nyújtottak a legjobb pontosságot, míg gépjárműdetektálásra az SSD-MobileNet v2 és az SSD-Inception-v2 háló. [8]

A számítógépes látáson alapuló vezetéstámogató rendszerek egyik legnagyobb kihívása a dinamikusán változó látási körülmények kezelése. Koray Ozcan és társai olyan algoritmuson dolgoznak, mely képes érzékelni kamerakép alapján az útburkolat állapotát és az éppen aktuális időjárást. A látási viszony érzékelése egyedi betanított neurális háló által valósul meg. Ehhez, több ezer képet készítettek az utakon, melyek a neurális háló (VGG16) tanító adathalmazát képezik. A betanítás után a modell három féle látási körülményt képes érzékeli: száraz, nedves és havas. [9]

Egyre több tanulmány zajlik az intelligens felügyeleti rendszerek fejlesztésében is. Ilyen például a fáradtság-, éberségfigyelő rendszer. A mai világban sokkal több álmos sofőr vezet, mint ahogy gondolnánk. Ezen kívül a mai társadalomban a szállítás, a fuvarozás, mind nélkülözhetetlen logisztikai folyamat, melyek minden napi életünk részét képezik. Azok a személyek, akiknek ez a megélhetésük, naponta több órát vezetnek és meg kell küzdeniük a fáradtsággal. Sajnos rengeteg baleset tulajdonítható a sofőrök figyelmetlenségének, melyet a kialvatlanság eredményez. V. Sanjay Kumar és társai kutatásukban olyan rendszer fejlesztésével foglalkoznak, mely egyszerre három, egymással összekapcsolt modulból áll: gépjárművezető fáradtságészlelése, alkoholtartalom érzékelés és baleset detektálás. A rendszerben a fáradtságérzékelés számítógépes látás által van detektálva, mely az álmoság jeleire irányul, mint például az ásítás, gyakori pislogás vagy a fejrázás. A második modul, az alkoholtartalom-érzékelő mechanizmus, az elektrokémia fogalmain alapul. A gázérzékelő (MQ3) belsejében lévő elektródán, alkoholgázmolekulák hatására, elektrokémiai reakció megy végbe, mely eredményeként feszültség keletkezik és információként kerül kiküldésre a

rendszerben használt Raspberry PI számítógépnek. Az utolsó modul a piezoelektromos koncepción alapszik. Amikor egy autó balesetet szenved, azt hatalmas mechanikai energia éri, melyet a prototípus piezoelektromos érzékelője elektromos energiává alakít át. Előállítja a szükséges feszültséget, majd továbbítja a Raspberry PI miniszámítógépnek. Legvégül, ez a rendszer IoT alapú, ezáltal képes értesíteni a kijelölt személyeket GSM modulon keresztül internetes hívás és üzenet által. [10]

Ahmad Aljaafreh kutatómunkájában egy olyan rendszeren dolgozik, mely a gépjárművezető fiziológiai és pszichológiai állapotát monitorizálja számítógépes látás és neurális háló segítségével. Az algoritmus a járműben lévő kamerán keresztül folyamatosan figyelemmel kíséri a sofőr tevékenységét, hogy észlelje a rendellenes vezetési magatartást, illetve kiszűrje azokat a pillanatokat, amikor a gépjárművezető figyelmét eltereli valami az úttestről. Ehhez a sofőr fejpózát és a szemének forgási vektorát vizsgálja az alkalmazott ESN (Echo State Network) neurális háló, mely egy NVIDIA Jetson Nano-n fut. A rendszer 98%-os pontossággal képes kiszűrni a gépjárművezető gyanús, szokatlan viselkedéseit. [11]

A fentebb említett kutatómunkák mindegyike a számítógépes látáson alapszik. Dolgozatom további részében szeretném röviden ismertetni ezt a technikát, mely lehetővé tette a mai világban használt intelligens rendszerek kifejlesztését.

2. SZÁMÍTÓGÉPES LÁTÁS MÉLYTANULÁSSAL

A számítógépes látás az emberi látás azon szerepeit valósítja meg, melyek a retinára vetített kép elemzését végzik. Ezek a funkciók elsősorban a képi tartalom értelmezésére irányulnak, mint például: a látott kép alapján az objektumok 3D alakjára való következtetés (felület rekonstrukció), a tárgyak térbeli elhelyezkedése, egymáshoz való viszonya (mélységi információ kinyerése), illetve a mozgás érzékelése és a mozgó objektumok követése [12].

Ez a technológia fontos szerepet játszik az arcfelismerő alkalmazásokban, a számítógépek felismerik a képeken látható arcok vonásait, egy előzetesen definiált arc profil adatbázis alapján. Ugyanakkor a számítógépes látásnak köszönhetően értik meg az önvezető autók a környezetüket. Az autókba felszerelt kamerák különböző szögekből készítenek videofelvételeket az autó körül, majd továbbítják a feldolgozó egységbe, mely valós időben feldolgozza a képeket a KRESZ táblák felismeréséhez, illetve más forgalmi résztvevők (autók, biciklisek, gyalogosok) azonosításához. Továbbá a gép látás nagyon használatos az egészségügy területén is, segítségével észlelhetők a daganatos elváltozások a bőrképeken, vagy felfedezhetők egy páciens tünetei a röntgenfelvétel, vagy MRI vizsgálat alapján. A fentebb megemlített alkalmazási területek mindegyike mélyreható tanulást igényel. [13]

A digitális berendezések segítségével, olyan részletességgel rögzíthetünk képeket, melyek messze meghaladják a saját látásrendszerünket, ugyanakkor a számítógépek nagyon nagy pontossággal felismerik a színek közti különbséget. Viszont a képek tartalmainak értelmezése és a rajtuk található tárgyak közötti kapcsolat megértése már egy olyan probléma, mellyel évtizedek óta küszködnek a számítógépek. [13]

A mesterséges intelligencia megjelenésének és folyamatos fejlődésének köszönhetően, mély tanulási folyamatok és neurális hálózatok által, a számítógépes látás az utóbbi években nagy ugrást ért el. Jelenleg a mély tanulás által a számítógépes látás meghaladja az emberi teljesítményt a képfalkotó osztályozásban. [13]

2.1. MESTERSÉGES INTELLIGENCIA

Ahhoz, hogy megértsük a mélytanulást, szükségünk van arra, hogy megértsük a mesterséges intelligencia alapjait. Az MI az egyik legmodernebb tudományterület. Az elnevezés 1956-ban született meg. Az MI napjainkban az olyan általános rendeltetésű területektől, mint a látás, beszéd és a tanulás, egészen olyan specifikus feladatokig, mint például a matematikai tételbizonyítás, vagy az orvosi diagnózis, a legkülönbözőbb részterületek óriási

választékát foglalja magába. A mesterséges intelligencia releváns az értelmes emberi cselekvés minden területén, mivel az intellektuális képességeket igénylő feladatokat rendszerezi, illetve automatizálja. [14]

Alan Turing szerint a számítógépnek az alábbi képességekkel kell rendelkeznie, hogy az intelligensnek mondható legyen:

- természetes nyelvfeldolgozás: a sikeres angol, vagy bármilyen más emberi nyelvű párbeszéd teljesítéséhez
- tudásreprezentáció: az ismert vagy hallott információk tárolása
- automatizált következtetés: a tárolt információkat kérdések megválaszolására, illetve új következtetések levonására lehessen használni
- gépi tanulásra: az új körülményekhez való adaptáláshoz, továbbá a különböző mintázatok detektálására
- gépi látásra: a tárgyak érzékeléséhez
- robotikára: a tárgyak mozgatásához [14]

2.2. GÉPI TANULÁS

A mesterséges intelligenciával rendelkező rendszerek képesek kell legyenek önállóan a tudás megszerzésére nyers adatok alapján, továbbá tudniuk kell általánosítani, ismeretlen adatokról következtetéseket levonni, osztályozni vagy akár előre jelezni különböző algoritmusok segítségével. Ezt a képességet nevezzük gépi tanulásnak (machine learning), mely a mesterséges intelligencia egy részét képezi. A mindennapi életünk részét képezi, jelen van az automatikus ajánlásokban (pl. Google keresőmotor, Facebook képeken arcfelismerés, Netflix-es filmajánló). A gépi tanulás tette lehetővé a számítógépek számára a valós problémák megoldását és a szubjektívnek tűnő döntések meghozatalát. Egy szimpla gépi tanuló algoritmus, pl. a logisztikus regresszió képes eldönteni, hogy a szülés során ajánlott-e a császármetszés, a “naive Bayes” pedig képes elszeparálni a spam-eket az emailoktól. [15]

Ezeknek az algoritmusoknak a teljesítménye nagymértékben függ az adatok ábrázolásától. Például a logisztikus regresszió során a mesterséges intelligencia nem vizsgálja meg közvetlen módon a beteget, hanem az orvos által megadott releváns információk alapján, mely belekerül a beteg ábrázolásába, az algoritmus megtanulja, hogy az egyes információk milyen összefüggésben vannak a lehetséges kimenetekkel. Viszont, ha a logisztikus regresszió egy MRI vizsgálat eredményeit kapja meg, mely számos mellékes információt is

tartalmaz, akkor a rendszer nem tud hasznos előrejelzést tenni a szülés közbeni komplikációkkal kapcsolatban. [15]

A gépi tanulásnál fontos, hogy kategorizáljuk az adatokat annak érdekében, hogy egy strukturált adatbázist hozzunk létre. Ugyanúgy, mint ahogy a könyvtárban hamarabb megtalálunk egy adott könyvet, ha azok rendszerezve vannak, ugyanúgy a számítástechnikában is az olyan műveletek, mint például egy adathalmazban való keresés, exponenciálisan gyorsabban végrehajtható, ha a gyűjteményünk megfelelően strukturált és indexelt. Arab számokkal sokkal könnyebben tudnak az emberek műveleteket végezni, viszont római számokkal ez már sokkal időigényesebb lenne. Az alábbiakból arra következtethetünk, hogy a gépi tanulások eredményeire óriási hatással van az ábrázolási mód megválasztása továbbá, ha a kimenet nem megfelelő számunkra, akkor emberi beavatkozás szükséges. [15]

2.3. MÉLYTANULÁS

Mind a gépi, mind a mélytanulás az MI alcsoportja, viszont a mélytanulás a gépi tanulás egy speciális része, melyet az emberi agy struktúrája inspirált. A számítógépek neurális hálózatokat használnak a tanuláshoz, akárcsak az emberi agy. A folyamat lényege, hogy ezeket a mesterséges neurális hálózatokat állítsuk be úgy, hogy hasznos reprezentációkat állíthassunk elő velük. [16]

A mély tanulási hálózatok már nem igényelnek emberi beavatkozást, nincs szükség egy programozóra, aki megmondja, mit tegyen a rendszer az adatokkal, mivel az ideghálózatok egymásba ágyazott rétegjein keresztül történik az adatvitel, a számítógép úgy elemezi az adatokat struktúrában, mint az emberek, illetve a saját hibáiból tanul. A mélytanulás egy olyan számítógépes tanulási technika, amely megtanítja a gépeknek azt, ami nekünk ösztönös, azaz példákból tanulni. Ez a technológia áll az önvezető járművek mögött, mely lehetővé teszi számukra, hogy felismerjék a KRESZ táblákat és megkülönböztessék a gyalogosokat a többi tárgytól. Továbbá kulcsfontosságú szerepet játszik még a hangvezérlésre képes eszközökben, mint például a telefon. A mélytanulás segítségével olyan eredményeket érhetünk el, amelyek ezelőtt lehetetlenek voltak, viszont mégis csak napjainkban lett elterjedt. Ennek két fő oka van. Az első ok az, hogy nagy mennyiségű adatot igényel. Például az önvezető autók fejlesztéséhez több millió képre és több ezer órányi felvételre van szükség. Továbbá jelentős számítási teljesítményt igényel. A nagy teljesítményű GPU-k párhuzamos architektúrával rendelkeznek, amely hatékony a mélytanuláshoz. [15] [16]

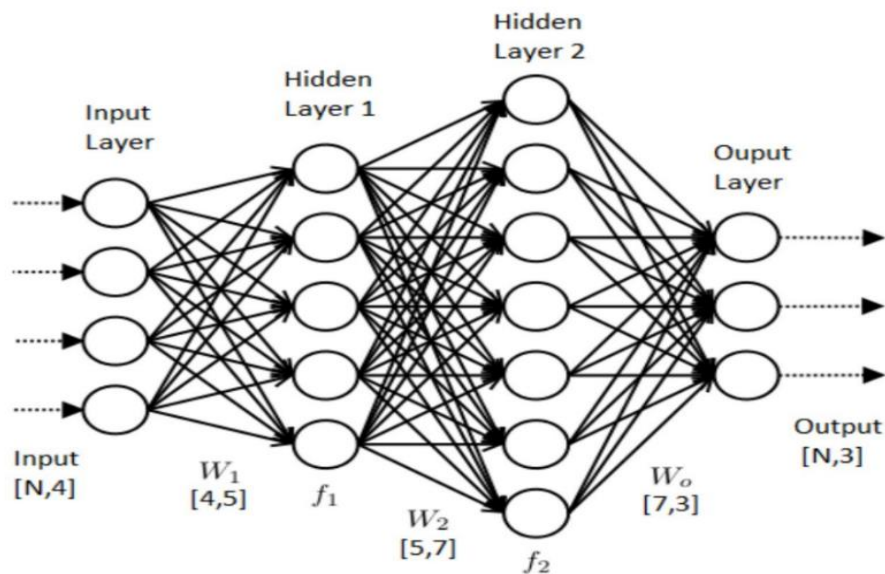
2.4. HOGYAN DOLGOZZA FEL AZ INFORMÁCIÓT AZ EMBERI AGY?

Az emberi agy idegsejtekből épül fel, melyek összekapcsolódva hálózatokat alkotnak, amelyek felelősek az információ továbbításáért, továbbá annak feldolgozásáért. Minden idegsejt rendelkezik egy sejttesttel, amely egy sejtmagot tartalmaz. A sejttestből ágazik el néhány dendritnek nevezett és egy hosszú axonnak nevezett szál (idegrostok). A dendritek veszik fel az információt és el irányítják a sejttestbe, ahonnan pedig az axonok továbbítják az ingerületeket. Egy neuron 10-10 000 más neuronnal van kapcsolatban a szinapszisokon keresztül. Azt a helyet ahol két idegsejt szorosan kapcsolódik egymáshoz, szinapszisknak nevezzük. Egy összetett elektrokémiai reakció által a jelek neuronról neuronra továbbítódnak. A jelek rövid távon az agy aktivitását szabályozzák, hosszú távon pedig befolyásolják az idegsejtek helyzetét és kapcsolási rendszerüket. Úgy tartották a kutatók, hogy ez képezi az agyi tanulás alapját, ezért a mesterséges intelligencia korai szakaszában ilyen mesterséges neurális hálózatok kialakítását célozták meg. [14]

2.5. MESTERSÉGES NEURÁLIS HÁLÓZATOK

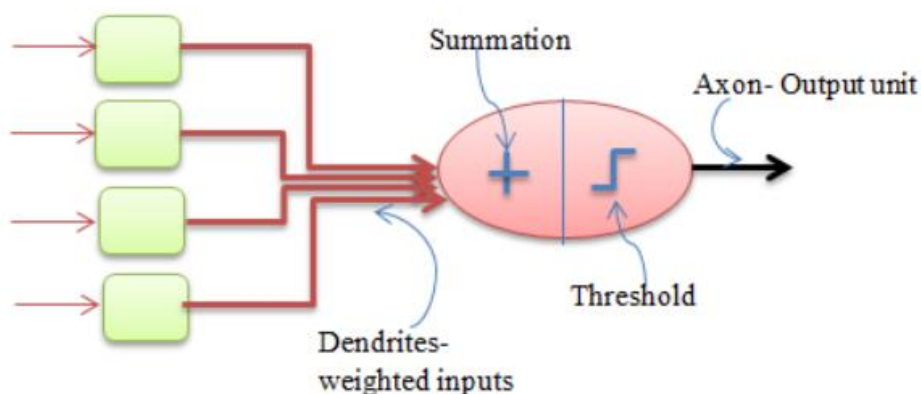
A számítógépes programok általában meghatározott utasítások sorozata, melyek a programozó parancsainak megfelelően hajtódnak végre. Viszont, ha a programozó nem tudja, hogyan oldjon meg egy adott problémát, akkor a program megoldóképesége korlátozott lesz. A mesterséges neurális hálózat az emberi agyhoz hasonlít, példákön és tapasztalatokon keresztül tanul, nem előre meghatározott parancsokból. Ezáltal a számítógépek képesek megoldani az olyan problémákat melyeket nagyon nehéz lenne leprogramozni, mint például minta-, tárgyfelismerés, vagy adatok osztályozása. [17] [18]

A mélytanulási modell lényeges példája az előrecsatolt mély hálózat, vagy másnéven többrétegű hálózat (MLP – multilayer perceptron), mely tulajdonképpen a legismertebb mesterséges neurális hálózat. Ezt több rétegbe szerveződött neuronok építik fel, melyek a rétegen belül nincsenek összekötve, viszont két egymás melletti réteg minden neuronja kapcsolatban áll egymással (4. ábra). Minimum három rétegből épül fel ez a hálózat: egy bemeneti-, egy vagy több rejtett- és egy kimeneti réteg. Az MLP egy matematikai függvény, melyben a neuronok (csomópontok) a beérkező jelek súlyozott összegét alakítják át egy átviteli függvényen keresztül. [15] [18]



4. ábra: Mesterséges neurális hálózat felépítése [18]

Minden neuronnak (5. ábra) saját súlyi és aktivációs függvénye van. A súly és küszöbértékek meghatározása a tanulási folyamat során történik. A neuron az előző kapcsolat alapján történő adat bevitelét egyesíti egy olyan súllyal, amely vagy erősíti, vagy csillapítja az adott bemenetet. Segítségével meghatározható egyes paraméterek fontossága, vagyis mely bemenetek a leghasznosabbak a hibamentes kimenet becsléséhez. A csomópontba érkező bemeneteket az összegzés után egy aktivációs függvény segítségével dolgozzuk fel, mely segít meghatározni, hogy az adott jel továbbhaladjon vagy sem, illetve, hogy ez hogyan fogja befolyásolni a kimenetet. Ha a jel keresztülhalad az adott neuronon, akkor az aktívvá válik és a jel tovább léphet a következő rétegbe, melyben ismét bizonyos neuronokat aktivál, majd továbbhalad, mindez addig ismétlődik, amíg el nem érünk a kimenethez. [19]



5. ábra: Mesterséges neuron felépítése [17]

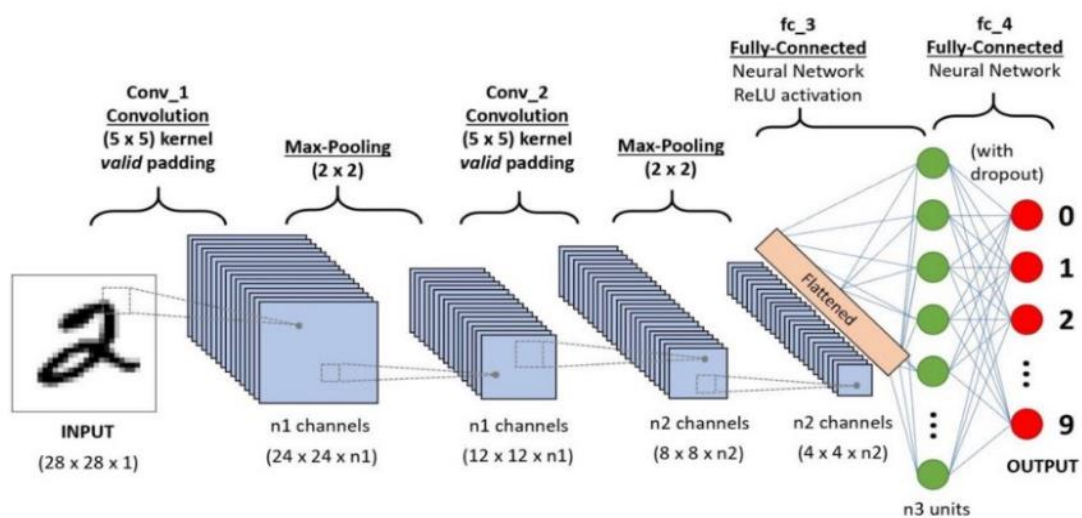
A mélytanulás során a tanító adatok felhasználásával, megadott bemenetek segítségével beállítjuk a hálózatban levő súlyokat és aktivációkat úgy, hogy a várt és a valódi kimenet között minél kisebb legyen a differencia, azaz a hiba. A továbbiakban a neurális hálózat már készen

áll, hogy újabb bemeneti adatok alapján meghatározzuk a nem ismert kimeneteket. Egy gyakori alkalmazási területe az osztályozás (pl. tárgyfelismerés). Ehhez szükséges egy címkézett adathalmaz, mivel az emberi értelem alapján kell strukturálni az adatokat annak érdekében, hogy a neurális hálózat megismerje a címkék és az adatállomány közötti kapcsolatot. [19]

2.6. KONVOLÚCIÓS NEURÁLIS HÁLÓZATOK

Manapság az autók képesek felismerni a közúti jelzéseket, a Facebook azonosítja a feltöltött képeken található személyeket, a modernebb telefonok pedig Face ID technológiával rendelkeznek már. Mindez a konvolúciós neurális hálózatnak (CNN – Convolutional Neural Network) köszönhető, melyet képi objektumazonosításra fejlesztettek ki, ugyanis a CNN képes kivonni egy kép jellemzőit, melyre egy olyan algoritmus, mint pl. az MLP nem képes. [20]

A konvolúciós neurális hálózat (6. ábra) egy olyan mélytanulási algoritmus, mely bemeneti képek alapján, felderíti azok jellemzőit (pl. jellemző formák, szélek), ezek összességét megvizsgálja, és legvégül képes a teljes képet értelmezni és osztályozni. Az aspektusok felderítése során, a kép egységein összevonások (konvolválások) történnek, mivel a hálózat a bemenetként megadott adatokat nem teljes egészében elemzi, hanem részenként szkenneli le. [20]



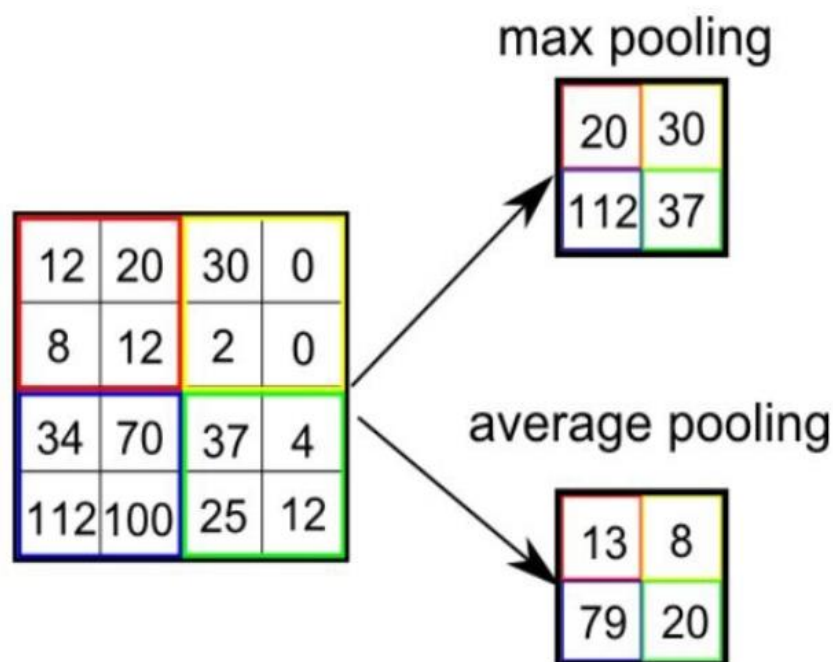
6. ábra: A CNN felépítése és működési elve [20]

A Konvolúciós réteg (pl. Kernel szűrő) egy általában $5 \times 5 \times n$ méretű szűrő (használatos a $3 \times 3 \times n$ méret is), melyet végighúzva a kép minden részén, az adott részekkel kiszámolja a konvolúciót, azaz a skalárszorzatot. Az n jelöli a csatornák számát (pl. RGB kép: 3 csatorna). A kép összes részén kiszámolva a skalárt, az eredmény egy aktivációs térkép lesz. Ha a

bemeneti kép például $28 \times 28 \times 3$ méretű volt, akkor $n1$ darab $5 \times 5 \times 3$ kernel szűrőt végighúzva rajta mindenhol, az aktivációs térkép nagysága $24 \times 24 \times (3 \times n1)$ lesz. Ennek a konvolúciós eljárásnak a lényege a jellemzők kivonása a képekből (pl. körvonalak, szélek, orientációk, színek). [20]

Az összevonó réteg (Pooling layer) hasonlít a konvolúciós réteghez, viszont ennek célja a konvolúciós jellemző méretének csökkentése. Segítségével minimalizálható az adatok feldolgozásához szükséges számítási teljesítmény. Két fajta eljárás létezik: max pooling és average pooling (7. ábra). A max pooling eljárás során a kép több mátrixra lesz felbontva, majd egy kisebb mátrixban maximalizálva az értékek. Hasonlóképpen működik az average pooling eljárás is, annyi különbséggel, hogy itt az értékek átlaga kerül bele az újabb mátrixba. [20]

Az összevonási eljárások során kapott mátrixok fogják képezni a további rétegek bemeneteit. A CNN hálózatok utolsó rétege egy (vagy több) teljesen összekötött (FC - Fully Connected) réteg, melyben a bemenet (a kép előzetesen oszlopvektorra lett konvertálva) minden eleme össze van kötve a réteg összes neuronjával. Az FC feladata az osztályozás, a felderített jellemzők alapján. [20]



7. ábra: 2x2-es tömörítő eljárások [20]

3. KAMERA ALAPÚ AUTÓVEZETÉST TÁMOGATÓ RENDSZER TERVEZÉSE

3.1. KUTATÓMUNKÁM SORÁN ELÉRT EREDMÉNYEK

TDK dolgozatomban témája a kutatómunka előző változata során megvalósított vezetéstámogató rendszer továbbfejlesztése. A rendszer egy Raspberry PI 4 Model B [21] egykártyás számítógépen fut, mely számára az érzékeléshez szükséges bemenő jeleket egy PI V2.1 kameramodul [22] szolgáltatja. A rendszer számítógépes látással képes érzékelni a gyalogátkelőhelyeken áthaladó gyalogosokat, illetve az elsőbbségadásra felszólító táblákat, majd ezek után fény és hang alapú figyelmeztetőjelzést küld a sofőr számára.

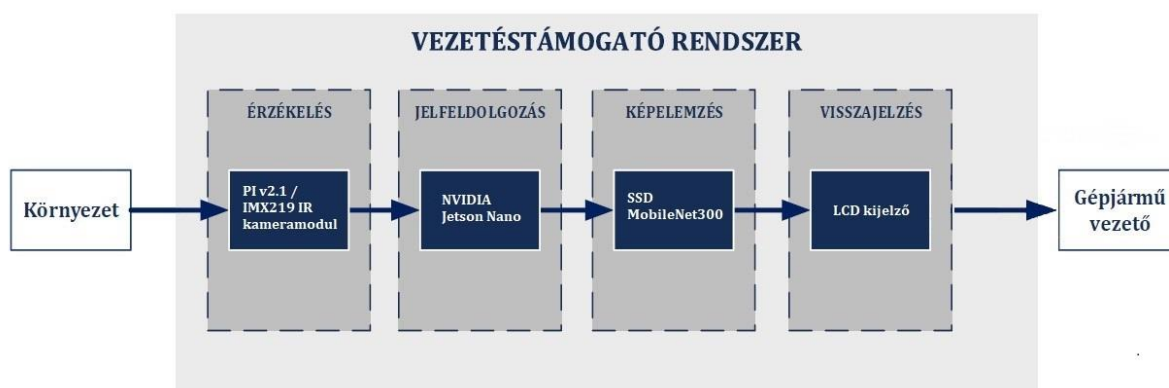
A felismeréshez alkalmazott neurális háló az SSD MobileNet 300x300 [23], melyet a Tensorflow [24] mélytanulási könyvtár alkalmazásával tanítottam be. Az objektumfelismerés lassú reakcióideje miatt, a rendszer számos optimalizációs eljárás ment keresztül, melyek során sikerült elérni az 5,6 FPS-t, majd a processzor órajelének növelése után (overclock: 1500 → 2000 MHz) a 7,2-7,3 FPS-t. A rendszer pontossága 80,5% volt.

3.2. FELMERÜLT PROBLÉMÁK

Elsősorban a legnagyobb probléma az alacsony képfrissítési sebesség. A rendszer reakcióidejét tekintve ($1/7,2 \text{ FPS} \sim 0,138 \text{ másodperc}$), két képkocka eltelése között, 50 km/h sebességgel haladva körülbelül 1,9 métert halad a gépjármű. Egy, az utolsó pillanatban az úttestre lelépő gyalogos esetében ez a reakcióidő túl sok, nem megengedhető. Továbbá, a rendszer pontossága 80,5%, mely egy vezetéstámogató rendszerben sajnos nem elegendő és elég nagy pontatlanságra és megbízhatatlanságra utal. Ez a probléma az egyes objektumok téves azonosításában jelentkezett. Ezen kívül, az éjszakai tesztelések során azt tapasztaltam, hogy a megvilágítás mértékének függvényében változik a rendszer stabilitása. Kivilágított helyeken megbízhatóan működött, viszont gyenge fényviszonyok között már jól érezhetően romlott a rendszer pontossága. TDK munkám során elsősorban a fentebb felsorolt problémák kiküszöbölésére fókuszáltam.

3.3. CÉLKITŰZÉSEK

A számítógépes látással kapcsolatos feladatok gyorsabban futnak egy grafikus feldolgozóegységen (GPU), mint a központi feldolgozóegységen (CPU). Ebből kifolyólag a rendszert szerettem volna átimplementálni egy célspecifikusabb eszközre. Továbbá szerettem volna elkerülni az overclock-ot, mivel az ilyen módszerek az eszköz élettartamát rövidítik. Választásom az NVIDIA Jetson Nano-ra [25] esett, mivel ez egy dedikált 128 magos videokártyával rendelkező fejlesztőpanel, MI alkalmazásokhoz lett kifejlesztve, illetve ez egy kompakt egykártyás számítógép lévén kevés helyet foglal, ezáltal könnyen beszerelhető az autóba, emellett elérhető áron beszerelhető. A rendszer megbízhatóságának növelése érdekében még több képet szerettem volna implementálni a tanító adatbázisba, melyek különböző látási viszonyokat lefednek. Az éjszakai pontosság növelése érdekében az éjszakai tárgyfelismerést egy másik, éjszakai képekkel betanított mély neurális hálóval szerettem volna megvalósítani, mely számára a bemenőjelet az IR-kamera biztosítja. Mivel a Jetson Nano B01 verziója két külön CSI bemenettel rendelkezik, ezáltal két kamera is csatlakoztatható hozzá egyidőben. Legvégül, szeretném a rendszert esztétikusabb kinézetre hozni, azaz külön kameratartót használni a kamerákkal, illetve egy LCD kijelzőt vizuális visszajelzés céljából.



8. ábra: A tervezett vezetéstámogató rendszer blokk diagrammja [saját ábra]

3.4. FELHASZNÁLT ESZKÖZÖK

Az NVIDIA Jetson Nano B01 fejlesztői készlet egy kis méretű, de nagy teljesítményű számítógép, mely lehetővé teszi több neurális háló párhuzamos futását, olyan alkalmazásokhoz, mint például a képosztályozás, tárgyfelismerés, szegmentálás és beszédfeldolgozás. A fejlesztőpanel egy 4-magos ARM A57 @ 1.43 GHz processzorral, 128-magos Maxwell videokártyával és 4 GB 64-bites LPDDR4 RAM-mal rendelkezik. A fejlesztőpanel tápellátását egy 5V 4A-es, barrel jack kimenettel rendelkező adapter biztosítja. [26]

A feladathoz két kameramodult alkalmaztam. Nappali üzemmódhoz a kutatómunkám során használt PI V2.1 kameramodult (3280 x 2464 pixel, 62,2°-os látómező), míg az éjszakai használathoz egy IMX219-160IR kamerát (3280 x 2464 pixel, 160°-os látómező, infravörös kamera) [27]. Mindkettő dedikált CSI szalagkábelen csatlakozik a számítógéphez, melynek köszönhetően gyorsabban kommunikál velük, mint az USB kamerákkal. Az autóban a fejlesztőpanel tápellátását egy szivargyújtó adapterrel (5V4A DC-DC kapcsolóüzemű feszültségátalakító) biztosítottam [28].

Továbbá szerettem volna, hogy a rendszer esztétikusan nézzen ki az autóban, ezért 3D nyomtatóval műszerdobozt készítettem a fejlesztőpanelnek, továbbá a kameráknak tokozást terveztem és nyomtattam, mely felszerelhető a visszapillantótükörrre, illetve csuklós mechanizmus segítségével állíthatóak, függetlenül a visszapillantótükör állapotától. Ez azért nagyon fontos, hogy a gépjárművezetőnek jól legyen pozícionálva a tükör vezetés közben, viszont a kamerák is az optimális működéshez megfelelő irányban legyenek konfigurálva. Mivel nyáron rendkívül nagy forróságok uralkodnak a napon hagyott autókban, ezért PETG anyagból nyomtattam őket, mely 70 fokig nem deformálódik.

A rendszer vezérléséhez, illetve a visszajelzéshez egy 7 inches érintő képernyős LCD kijelzőt [29] választottam, mely HDMI és USB kábelén keresztül kommunikál a fejlesztőpanellal. Ez a kijelző a szélvédőre van felszerelve, olyan helyre, ahol nem zavarja a gépjárművezetőt a kilátásban, illetve nem esik bele a felszerelt kamerák látószögébe. Az alábbi ábrán (9. ábra) látható a rendszer felépítése az autóban.



9. ábra: A rendszerben felhasznált eszközök beszerelve az autóba [saját ábra]

4. FELADAT KIVITELEZÉSE

4.1. JETSON NANO KONFIGURÁLÁSA

A feladat első lépése az NVIDIA JetPack [30] telepítése volt a fejlesztőpanelra, mely tartalmazza magát az operációs rendszert (Ubuntu), a CUDA könyvtárakat, valamint a számítógépes látás, mélytanulás és gyorsított számítástechnikához szükséges API-kat (Application Programming Interface). A telepítést a balenaEtcher ingyenes és nyílt forráskódú segédprogrammal hajtottam végre egy 64 GB-os SD kártyára, mely képes .iso kiterjesztésű fájlok írására.

Ezt követően telepítettem az objektumfelismeréshez szükséges szoftverkönyvtárat [31], mely a TensorRT keretrendszert használja a mély neurális hálók hatékony futtatására a Jetson platformra, optimalizálva a teljesítményt, átviteli sebességet, memórafogyasztást és az energiafelhasználást.

A neurális háló betanítását a PyTorch gépi tanulás keretrendszerrel, a transzfer tanulás [32] módszerével szerettem volna megvalósítani. Ez egy olyan módszer a DNN (Deep Neural Network) újratanítására saját adatkészlet alapján, mely sokkal kevesebb időt vesz igénybe, mint a hálózat nulláról történő oktatása. Előző kutatómunkám során a betanítást dedikált GPU-val rendelkező PC-n hajtottam végre a Tensorflow szoftverkönyvtárral, majd a kapott modellt optimalizálva és konvertálva Tensorflow Lite-ra futtattam azt a Raspberry-n. Ez a módszer rendkívül idő- és energiaigényes folyamat volt, mivel a szoftver telepítése (mind Windows-ra, mind Raspbian-ra) a szükséges modulokkal, API-val együtt meglehetősen bonyolult volt, továbbá a modell konvertálása is problémás volt, mivel a Tensorflow folyamatosan megújuló szoftverkönyvtár, számos modul szükséges a tárgyfelismeréshez, nagyon érzékeny a verziókülönbségekre és az újabb változatoknál már más módszereket kell alkalmazni az optimalizálásnál. Viszont a transzfer tanulás által a háló betanítható közvetlenül a Jetson Nano-n és az exportált modell már optimalizálva van a TensorRT keretrendszerre, mely gyorsabb futást eredményez.

Legvégül 4 GB memóriát swapoltam az SD kártyára, mely célja a memória oly módon való kiegészítése, mely az épp nem használt programokat a RAM memóriából a swap területre helyezi az operációs rendszer, így a többi programnak több memóriát tud juttatni. Erre azért van szükség, mert a DNN tanítása során előfordulhat, hogy az eszköz kifogy a RAM memóriából.

4.2. ADATHALMAZ ELŐKÉSZÍTÉSE

A tárgyfelismeréshez használt neurális háló létrehozásához szükséges egy adatbázis, mely a betanításhoz szükséges képeket tartalmazza, ezért a következő lépés a képek begyűjtése volt. Az adathalmaz megfelelő előkészítése kulcsfontosságú lépés, úgy kell elképzelni, mint egy tankönyv felépítését. Amennyiben a tankönyv megfelelően van felépítve, a diákok könnyebben tanulnak belőle, mint egy rosszul strukturált tankönyvből. Ugyanez igaz a mélytanulásnál is. Mivel a tervezett rendszerben valós forgalmi körülmények kerülnek feldolgozásra, így a képeket szintén valós közlekedési helyzetekből szerettem volna kiemelni. Ehhez, a rendszert eleinte fedélzeti kameraként használtam, majd a videókból kivágtam azokat a képkockákat melyeken a betanítandó objektumok jelen vannak. A képkockákat teljes méretben emeltem ki a videókból azért, hogy ne csak a betanítandó objektum, hanem annak a környezete is látszódjon.

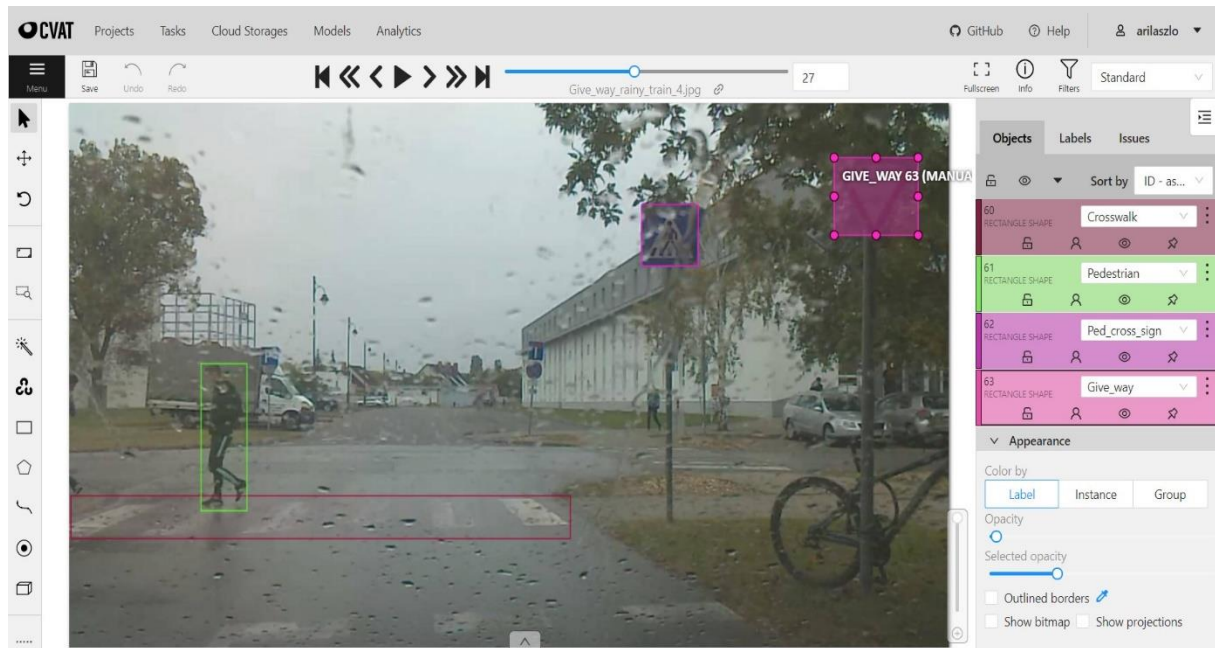
Az osztályozási feladatok kivitelezésében kulcsfontosságú szerepet játszik a nagy mennyiségű felcímkézett adathalmaz. Annak érdekében, hogy a rendszer rossz látási viszonyok között is megbízhatóan működjön, az adathalmazba bevontam rosszabb látási viszonyok között készült képeket egyaránt (pl. esős/felhős idő, naplemente, erős napsütés szemből), illetve olyan képeket, melyeken az objektumok részben ki vannak takarva, vagy a tábla van elfordítva a gépjárművezetőtől (10. ábra). Ezáltal 740 képet gyűjtöttem össze a két kamerával: 400 nappali (abból 90 felhős időben, 60 esőben készült) és 340 éjszakai (melyből 60 esőben készült).



10. ábra: Részben elfordított/kitakart táblák [saját ábra]

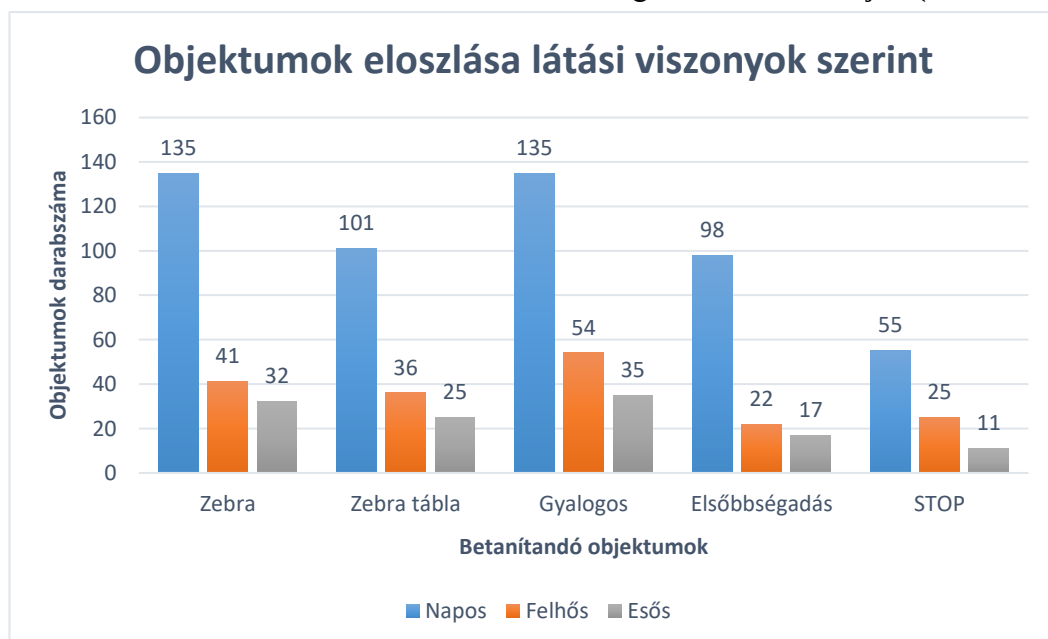
A képeket a begyűjtés után egységes méretűre alakítottam (1280x720), majd a CVAT (Computer Vision Annotation Tool) [33] segítségével felcímkéztem a rajtuk látható objektumokat. Az annotációk alapján a tárgyak nevei és koordinátái XML fájlokba lesznek

lementve. Legvégül az adathalmazt, az annotációkkal együtt, Pascal VOC formátumban exportáltam.

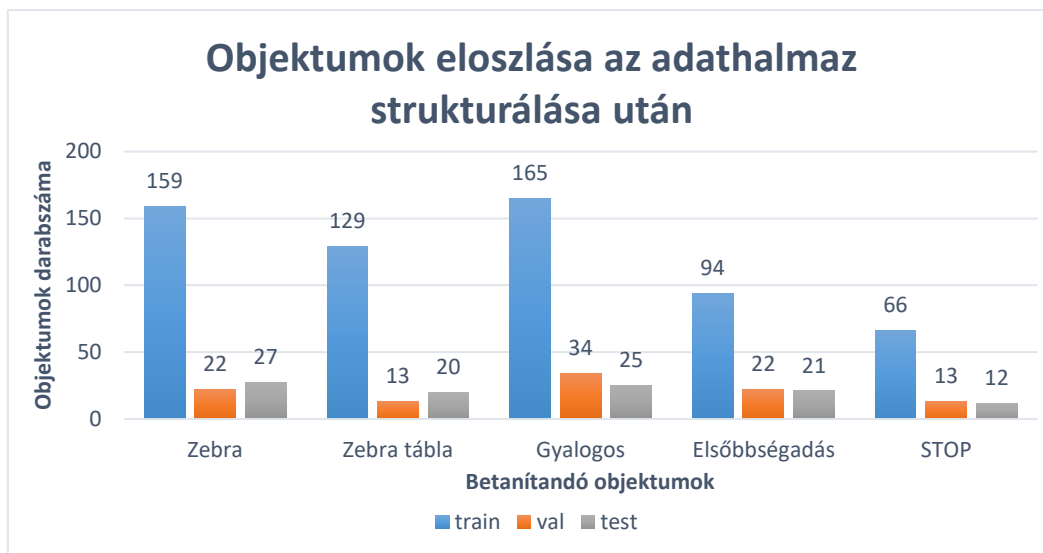


11. ábra: Képek annotálása CVAT segítségével [saját ábra]

Ezt követően az adathalmazomat három részre strukturáltam: a képek 70%-át betanításra (ezek képezik a tanulási adatbázist), 15%-át validálásra (minden tanulási iteráció után ezek alapján értékelődik ki a modell), 15%-át tesztelésre (olyan képek melyekkel soha nem találkozott még a tanító algoritmus). A képeken látható objektumok eloszlását látási viszonyok, illetve az adathalmaz strukturálása szerint, az alábbi diagrammokon láthatjuk (12-15. ábrák).



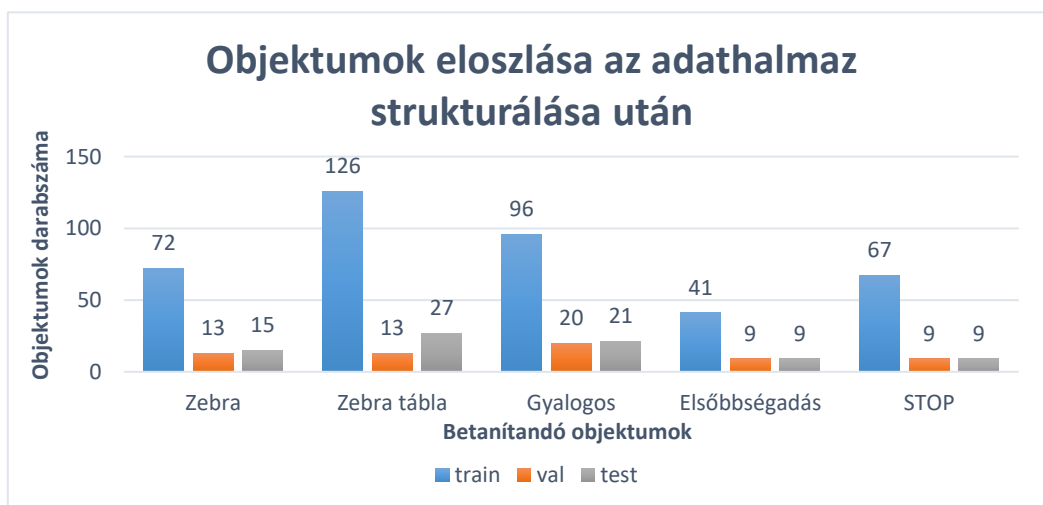
12. ábra: Nappali képeken látható objektumok megoszlása látási viszonyok szerint [saját ábra]



13. ábra: Nappali képeken látható objektumok eloszlása az adathalmaz strukturálása alapján [saját ábra]



14. ábra: Éjszakai képeken látható objektumok megoszlása látási viszonyok szerint [saját ábra]



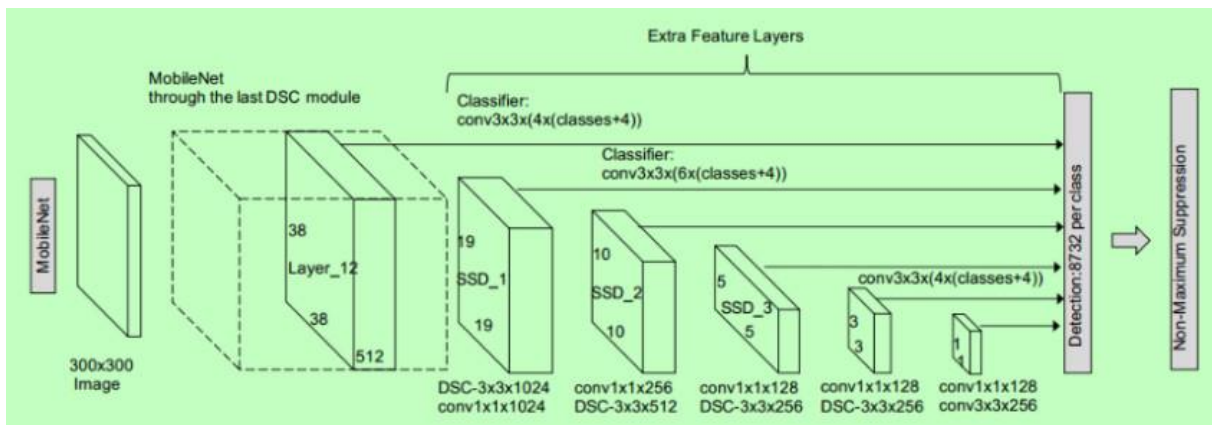
15. ábra: Éjszakai képeken látható objektumok eloszlása az adathalmaz strukturálása alapján [saját ábra]

4.3. NEURÁLIS HÁLÓ MÉLYTANÍTÁSA

A neurális háló betanítását, majd ennek kiértékelését a PyTorch nyílt forráskódú gépi tanulási könyvtár segítségével valósítottam meg. Az alkalmazott szoftverkönyvtár lehetővé teszi számos más népszerű konvolúciós neurális hálózatokkal való munkát, mint például az SSD MobileNet, Inception, Faster R-CNN. Lehetőségünk van előtanított modelleket letölteni, majd ezeket újra felhasználni, mely sokkal hatékonyabb megoldásnak bizonyul, mind idő, mind erőforrás takarékoság szempontjából. [32]

Az általam kiválasztott háló az SSD MobileNet, melyet kisteljesítményű beágyazott rendszerekre fejlesztettek ki. Ez az architektúra (Single Shot Detection), mint ahogy a neve is mutatja, egy olyan CNN, mellyel csak egy képet kell készíteni ahhoz, hogy a képen belül több tárgyat észleljünk. Egyszerűbb számítások hajtnak végre benne, ami előnyös a számítógép erőforrásfelhasználása szempontjából. Ebből kifolyólag ez az architektúra nagyobb sebességet nyújt, mint pl. egy Faster R-CNN modell, melynek két képre van szüksége a tárgyfelismeréshez. [23] [34]

Mivel a kutatómunkám továbbfejlesztése során az egyik kitűzött célom a képfrissítési sebesség növelése, ezért az SSD MobileNet 300 neurális hálót választottam, mely a leggyorsabb futási sebességet biztosítja. Ez a modell 300x300 méretű 3 csatornás (RGB) képeket fogad el bementként. Az alábbi ábrán (16. ábra) látható a háló réteges architektúrája.



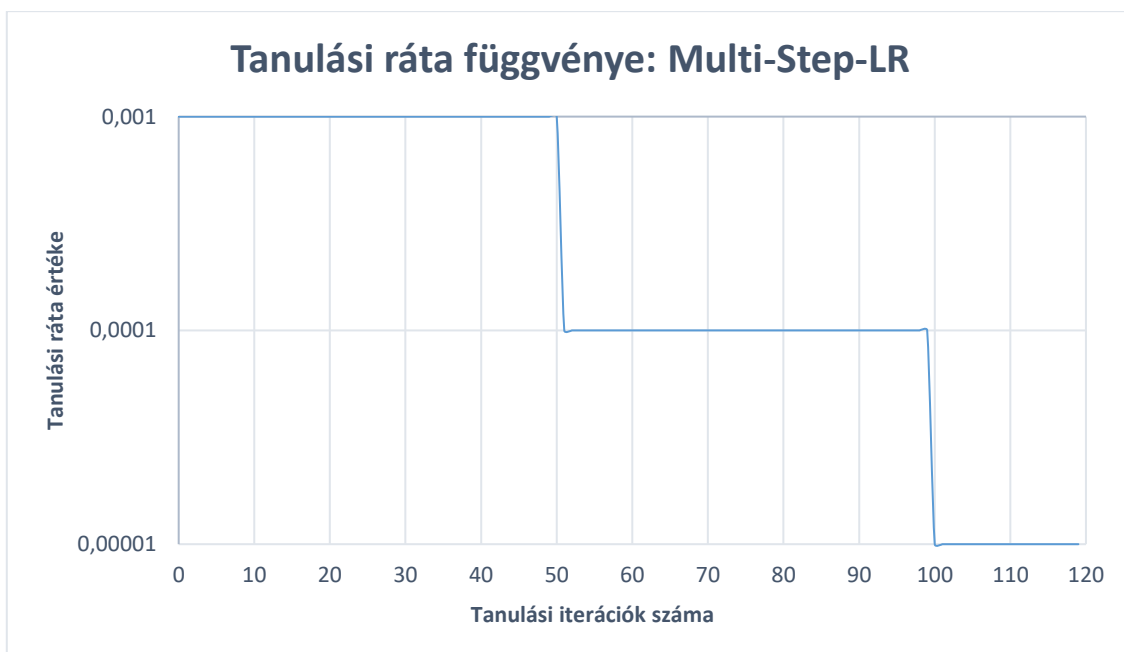
16. ábra: SSD MobileNet 300x300 neurális háló felépítése [23]

A mélytanulási folyamat több iterációból tevődik össze, mely az alábbi lépéseket foglalja magába: adatok kiértékelése, neurális háló hibájának meghatározása a veszteségfüggvény segítségével, továbbá a modell frissítése. A modell frissítése akkor számít sikeresnek, amikor a veszteségfüggvény által számolt veszteség értékek az iterációk során csökkennek.

A mélytanulás optimalizálási algoritmusai során a tanító adathalmaz képeinek felhasználásával a számítógép megbecsüli a neurális háló aktuális állapotának a hibagradiensét, majd frissíti a modell súlyát. A veszteségfüggvény azt mutatja meg, hogy mennyire térnek a számítógép által jósolt értékek a valódi értékektől, azaz megmutatja, hogy mennyire pontatlan a modellünk. A neurális háló tanítása során a cél a veszteségfüggvény értékeinek minimalizálása. Ez órák hosszát tart, mely függ a képek mennyiségétől, a felhasznált modelltől, a keresett objektumok komplexitásától, az alkalmazott számítógép hardwares specifikációjától, továbbá a beállított tanulási rátától, mely a modell tanulási sebességét szabályozza és azt mutatja meg, hogy hány százalékban legyenek frissítve a modell súlyai. [35]

Számos mélytanulási kísérlet során vizsgáltam a tanulási ráta és a veszteségfüggvény változásának kapcsolatát. Azt tapasztaltam, hogy minél kisebb nagyobb a validációs hiba, annál nagyobb tanulási ráta szükséges annak a korrigáláshoz, viszont a hiba csökkenésével egyidejűleg szükséges a tanulási ráta értékét is csökkenteni. Túl magas tanulási ráta nemkívánatos divergáló viselkedést okoz a veszteségfüggvényében, viszont amennyiben az túl alacsony, feleslegesen elnyújtja a tanulási folyamatot.

A tanító algoritmust két különböző tanulási ütemtervvel futtattam végig. Elsőként manuálisan, lépcsőzetesen csökkentettem a tanulási rátát (Step Decay/Multi Step Learning Rate). A kezdeti tanulási ráta értékét 10^{-3} -ra állítottam be, melyet 50 epoch után 10^{-4} -re, majd ismét 50 epoch után 10^{-5} -re csökkentettem (17. ábra).



17. ábra: Tanulási ráta változása lépcsőzetesen a tanulási iterációk függvényében [saját ábra]

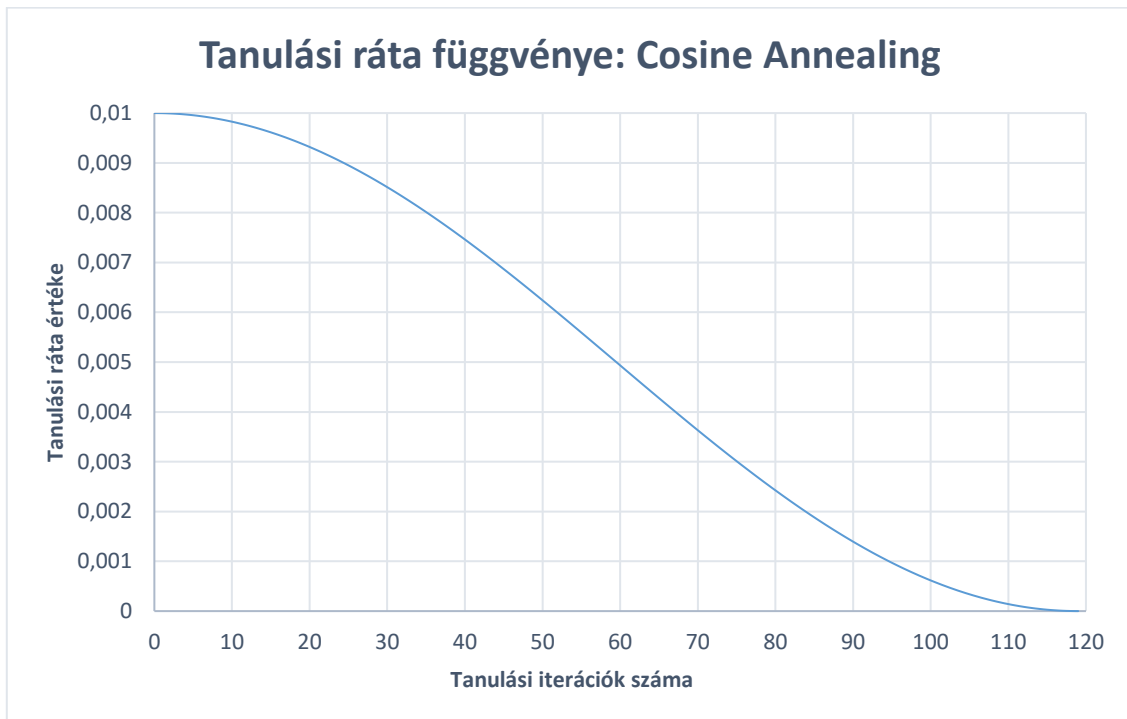
Másodjára egy olyan módszert vizsgáltam meg, mellyel a tanulási ráta folyamatosan csökken, a koszinusz függvénynek megfelelően (Cosine Annealing Schedule). Az alábbi képlet (1. képlet) illusztrálja a koszinuszosan változó tanulási ráta változását az iterációk függvényében. [36]

$$LR_i = \frac{1}{2} * LR_{max} * (1 + (\cos \frac{T_i}{T_n} * \pi)) \quad (1.)$$

A fenti képlet a koszinusz függvényből származik. LR_i reprezentálja az aktuális tanulási rátát, T_i az aktuális epochot, míg T_n az összes epochot, ezáltal a kettő aránya megszorozva π -vel, a $[0-180^\circ]$ félperiódust adja meg, mely 1 és -1 között vesz fel értékeket. Mivel nem szeretnénk negatív értékeket kapni, ezért 1-et hozzáadva a függvény 2 és 0 között vehet fel értékeket, $\frac{1}{2}$ -vel beszorozva már csak 1 és 0 között. Amennyiben LR_{max} reprezentálja a kezdeti tanulási ráta értékét (koszinusz amplitúdója), a függvény értékkészlete $[LR_{max}; 0]$ lesz, azaz a tanulási ráta koszinuszosan csökken LR_{max} -tól 0-ig.

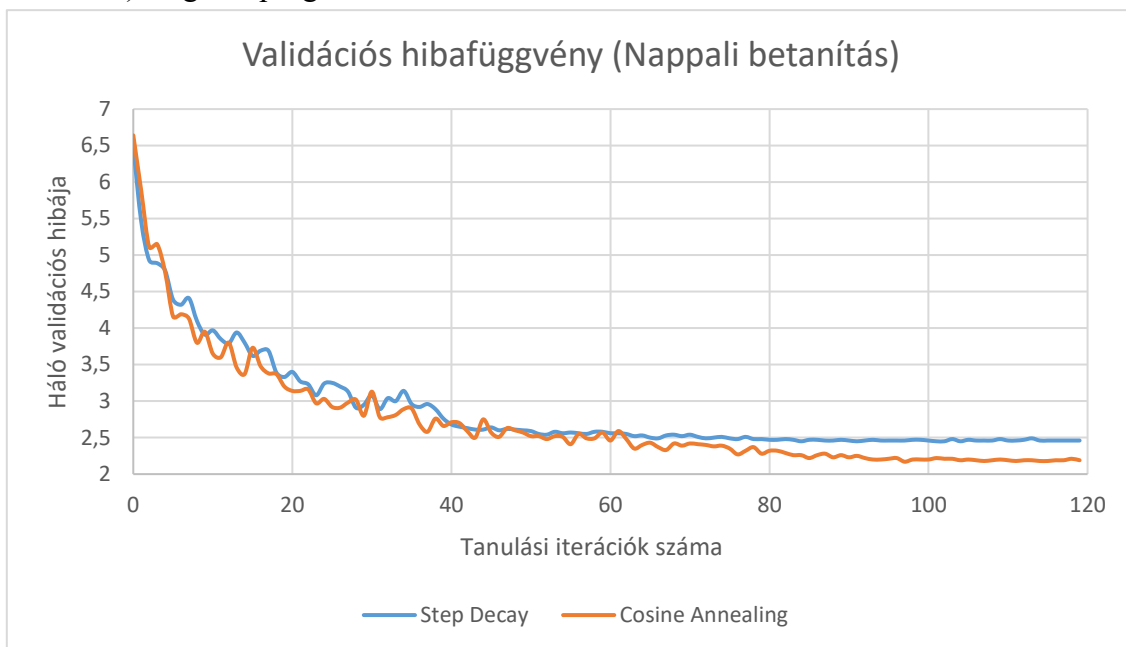
Ezt már egy magasabb értékről, 0,01-ről indítottam, mely 0-ig csökken (18. ábra). Ebben az esetben is a tanítási folyamat 120 epoch-ot tartott. Az általam használt tanulási ráta függvényt az alábbi képlet írja le:

$$LR_i = \frac{1}{2} * 0,01 * (1 + (\cos \frac{T_i}{120} * \pi))$$

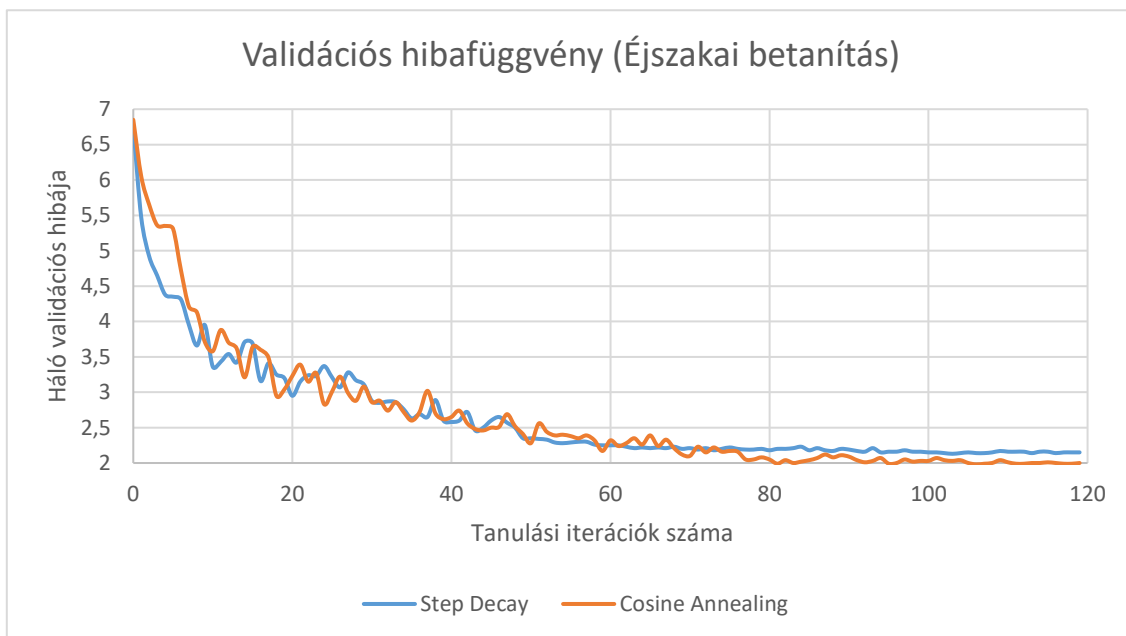


18. ábra: Koszinuszosan változó tanulási a tanulási iterációk függvényében
[saját ábra]

Az alábbi két diagramm ábrázolja (19-20. ábrák) a tanulási folyamat során a hibafüggvény változását, a két különböző tanulási ütemterv alkalmazásával. A validációs hiba a validálásra szánt képek felhasználásával lett meghatározva minden egyes tanulási iteráció (lásd később), végén a program által.



19. ábra: Nappali betanítás validációs hibafüggvényének változása az iterációk függvényében [saját ábra]



20. ábra: Éjszakai betanítás validációs hibafüggvényének változása az iterációk függvényében [saját ábra]

A fenti ábrákon jól látható, hogy a koszinuszosan csökkenő tanulási rátával jobban sikerült minimalizálni a validációs hibafüggvény értékeit. Véleményem szerint ez annak

tulajdonítható, hogy egyrészt a folyamat nagyságrendileg magasabb tanulási rátákat használ, továbbá a Cosine Annealing módszerrel a tanulási ráta folyamatosan csökken az iterációk során, nem lépcsőzetesen és ez azért jelentős, mert minél kisebb a validációs hiba, annál kisebb tanulási ráta szükséges annak a korrigálásához.

Tanulási iterációnak vagy epochnak nevezzük azt amikor a tanító adatbázisunk összes példáján egyszer végigfutott az algoritmus. A túltanulás (overfitting) elkerülése végett, minden epoch után a validációs halmazon kiértékelődik a modell. Amennyiben az eredmények romlani kezdenek a tanulási iterációk során, az az overfitting jelenségre utal, a modell túl lett tanítva („be lett magolva”) és újabb példákra már nem teljesít jól. [37]

4.4. MODELL PONTOSSÁGÁNAK MEGHATÁROZÁSA

A számítógépes látásban a precizitás egy olyan értékelési mutató, mely az előrejelzések pontosságát reprezentálja, azaz, hogy hány százalékban helyesek a predikciók (2. képlet). [38]

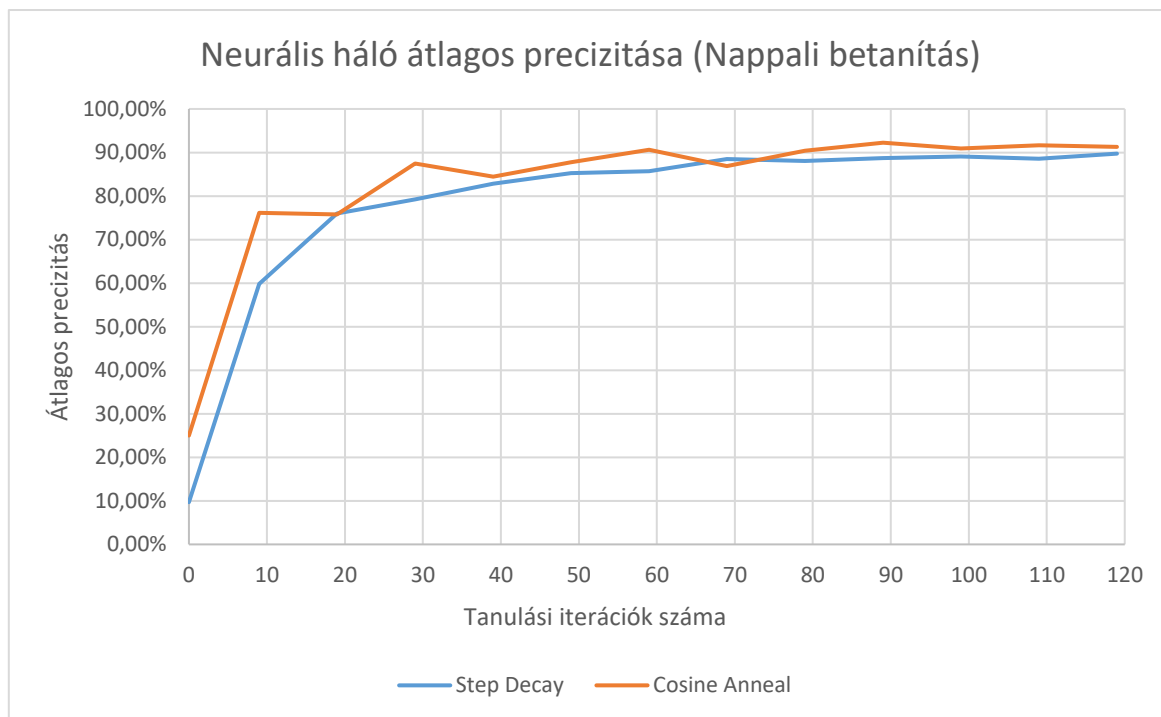
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2.)$$

A képlet alapján:

- TP (True Positive) – Helyes előrejelzések: a valós és a prediktált kimenet is 1
- FP (False Positive) – Hibás előrejelzések: a valós érték 0, de modell 1-et prediktált (pl. STOP tábla helyett előzni tilos táblát érzékelt a modell) [38]

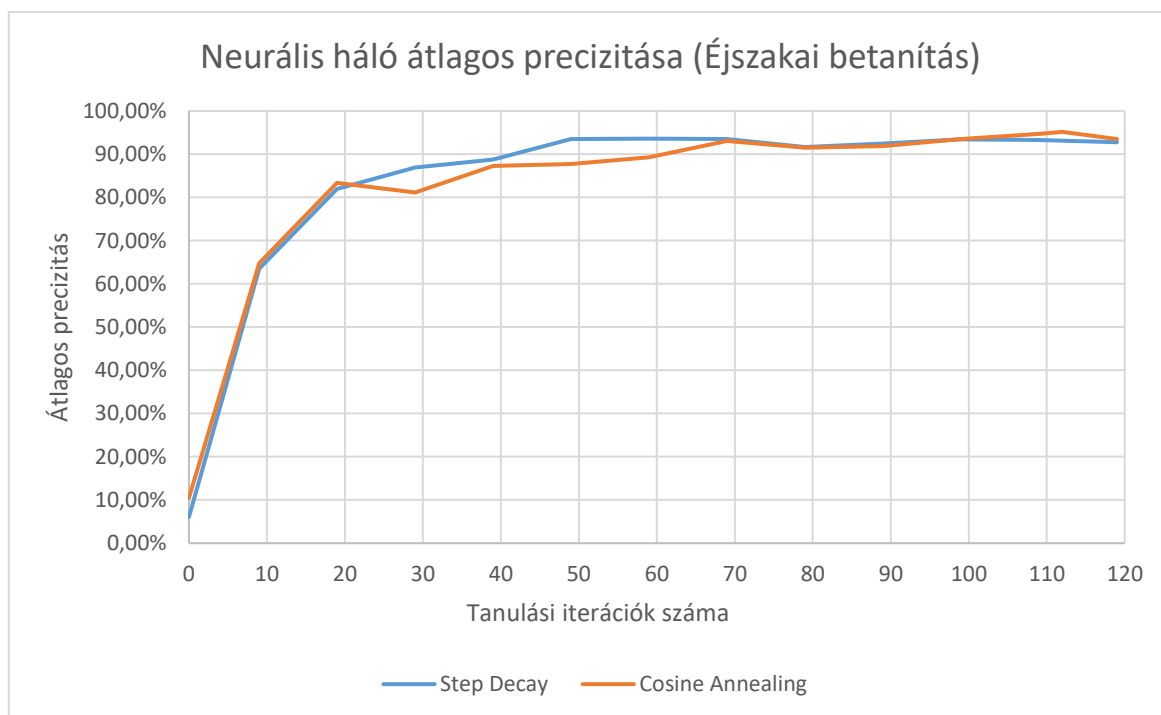
A modell kiértékelése során a pontosság értékek az IoU (Intersect of Union) küszöbértékek alapján vannak számolva, mely a prediktált határolódoboz és a valós határolódoboz közötti lefedést reprezentálja. Például 0,5 IoU küszöbérték esetében, csak azok a predikciók számítanak igaznak (TP), melyek határolómezői legalább félig lefedik az objektum valós határolódobozát, különben hamisnak veszi őket a program. [39]

A háló kiértékelését egy Python kód által valósítottam meg, mely során a program kiszámolja a tesztelésre szánt képekre a pontosság értékeket, 0,5 IoU esetében, majd leátlagolja őket. A kódot végigfuttattam minden tizedik checkpoint fájlra, melyek a mélytanulás során generálódnak az epoch-ok végén. Ezek az állományok felelnek meg a tanulási folyamat kimeneteinek, melyek a neurális háló állapotáról tárolnak információt. A kapott precizitás értékeket Excel táblázatban rögzítettem, majd megjelenítettem diagrammon őket (21-22. ábra). Ezzel az volt a célom, hogy megtaláljam a neurális hálónak azt az állapotát, mely a legnagyobb precizitással rendelkezik.



21. ábra: Nappali képek által betanított neurális háló átlagos precizitásának változása a tanulási iterációk függvényében [saját ábra]

A nappali betanítás során a Step Decay módszerrel elért legjobb pontosság 89,74% a 119. epoch-ban, míg a Cosine Annealing Schedule módszerrel sikerült elérni a 92,25% pontosságot (23. ábra) a 89. epoch-ban.



22. ábra: Éjszakai képek által betanított neurális háló átlagos precizitásának változása a tanulási iterációk függvényében [saját ábra]

Az éjszakai betanítás során a Step Decay módszerrel elért legjobb pontosság 93,59% az 59. epoch-ban, míg a Cosine Annealing módszerrel sikerült elérni a 95,15% (24. ábra) pontosságot a 112. epoch-ban.

```
root@arilaszo-desktop: /jetson-inference/python/training/detection/ssd
Prediction: 0.242240 seconds.
process image 57
Load Image: 0.019518 seconds.
Inference time: 0.038442134857177734
Prediction: 0.407422 seconds.
process image 58
Load Image: 0.021394 seconds.
Inference time: 0.03655815124511719
Prediction: 0.239667 seconds.
process image 59
Load Image: 0.021521 seconds.
Inference time: 0.04012250900268555
Prediction: 0.254693 seconds.

Average Precision Per-class:
Crosswalk: 0.8634773046153036
Give_way: 0.9216047110783954
Ped_cross_sign: 0.9826839826839826
Pedestrian: 0.9584980237154153
STOP: 0.8863636363636365

Average Precision Across All Classes: 0.9225255316913467
root@arilaszo-desktop: /jetson-inference/python/training/detection/ssd#
```

23. ábra: Nappali képek által betanított neurális háló átlagos precizitása minden egyes objektumra [saját ábra]

```
root@arilaszo-desktop: /jetson-inference/python/training/detection/ssd
Prediction: 0.149658 seconds.
process image 48
Load Image: 0.016714 seconds.
Inference time: 0.037225961685180664
Prediction: 0.177169 seconds.
process image 49
Load Image: 0.017000 seconds.
Inference time: 0.03634357452392578
Prediction: 0.152775 seconds.
process image 50
Load Image: 0.017208 seconds.
Inference time: 0.036036014556884766
Prediction: 0.153450 seconds.

Average Precision Per-class:
Crosswalk: 1.0000000000000002
Give_way: 1.0000000000000002
Ped_cross_sign: 0.901931283510231
Pedestrian: 0.9090909090909093
STOP: 0.9466089466089468

Average Precision Across All Classes: 0.9515262278420176
root@arilaszo-desktop: /jetson-inference/python/training/detection/ssd#
```

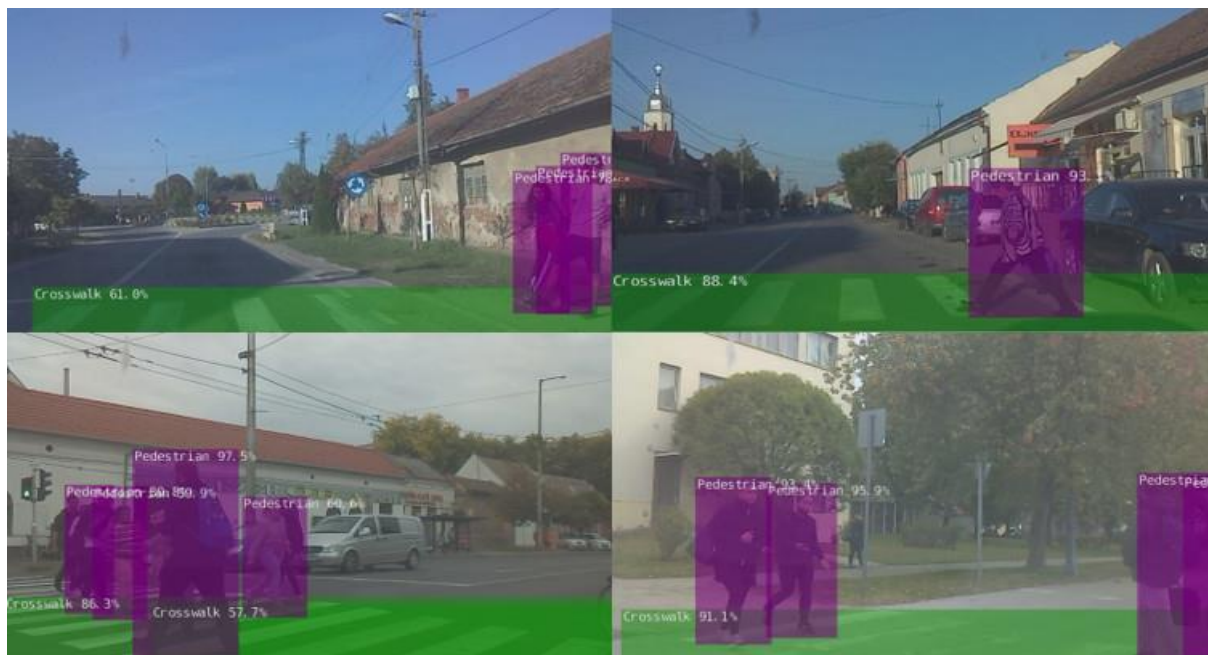
24. ábra: Éjszakai képek által betanított neurális háló átlagos precizitása minden egyes objektumra [saját ábra]

5. TESZTELÉS VALÓS KÖRÜLMÉNYEK KÖZÖTT

5.1. TESZTELÉS NAPPAL

A modell átlagos pontossága meghatározása után exportáltam a modellt .ONNX kiterjesztésben, mely futtatható a Jetson Nano-ra optimalizált TensorRT keretrendszeren, majd leteszteltem a működését valós körülmények között. A program futása során a modell predikciókat tesz a képkockákon látható objektumok valószínűségére és amelyekre ez az érték legalább 50%, azokra visszajelzést küld a felhasználó számára (objektum neve és a határoló téglalap).

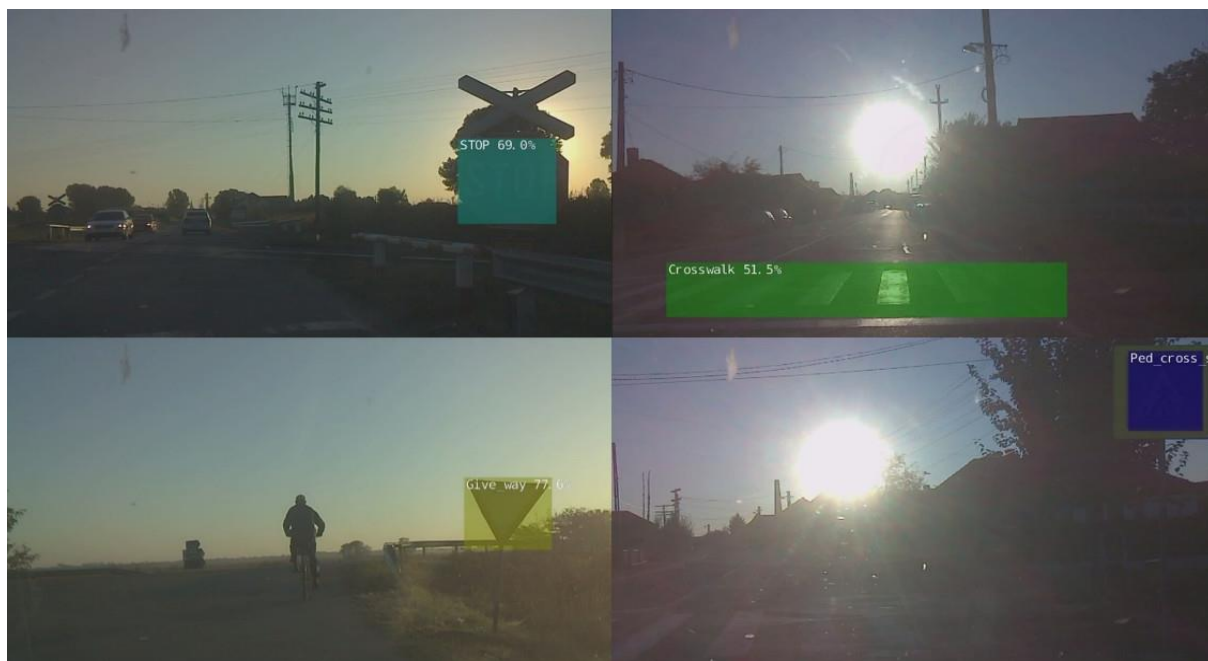
A több órás tesztelesek során azt tapasztaltam, hogy a rendszer kiváló pontossággal és megbízhatósággal érzékeli a gyalogosokat, különböző szögekben és helyzetekben is, mint ahogy az alábbi ábra (25. ábra) is reprezentálja.



25. ábra: Gyalogosfelismerés tesztelése nappal [saját ábra]

A betanított KRESZ táblákat is kiválóan érzékeli a rendszer, a 90% feletti átlagos precizitásnak köszönhetően sokkal kisebb volt a tévesen beazonosított objektumok előfordulási valószínűsége, mint előző kutatómunkámban. Negatívumként azt tapasztaltam, hogy mivel a táblák kisebb méretűek, ebből kifolyólag inkább közelebbről érzékeli őket a rendszer. Véleményem szerint ez annak tulajdonítható, hogy a neurális háló, a kamera által rögzített képkockákat 300x300 méretűekre konvertálja, ezáltal a kisebb tárgyak úgymond „elvesznek” rajta. A rendszert teszteltem hajnalban (26. ábra), ugyanakkor naplementében is (27. ábra). A

betanított objektumokat ekkor is kiválóan érzékeli és nagy mértékben segíti a gépjárművezetőt a vezetésben, amennyiben őt teljesen elvakítja a nap. A gyalogátkelőhely felfestésekkel adódott probléma, melyen a visszaverődő fény olykor megzavarta a rendszert.

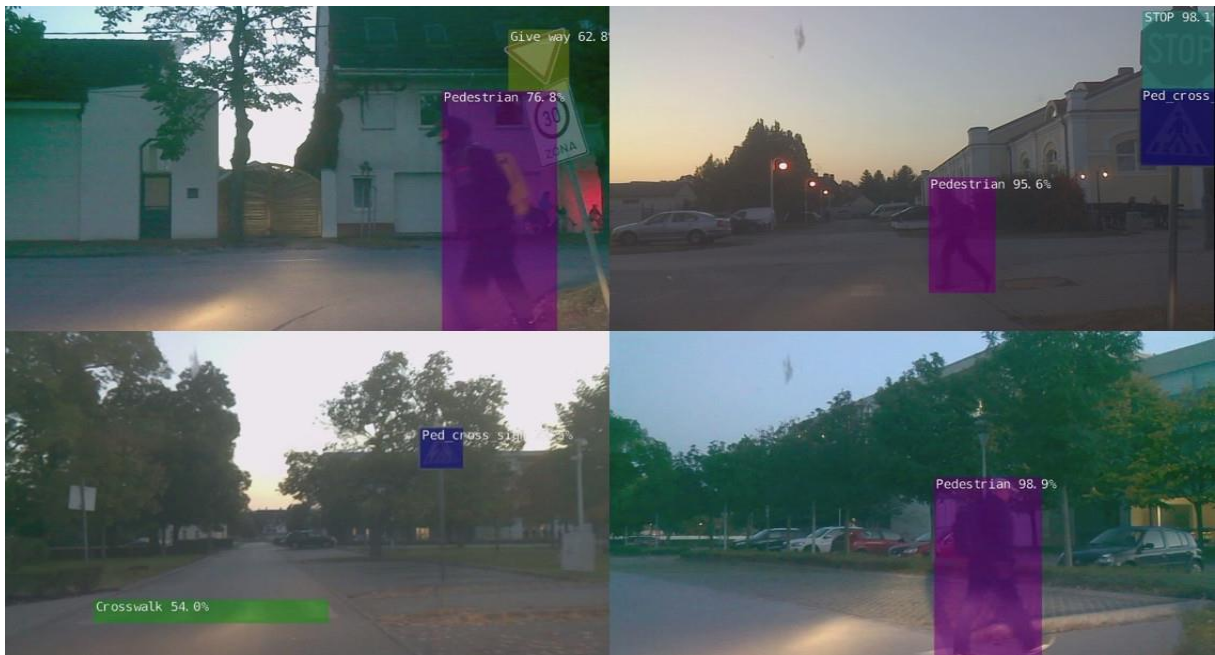


26. ábra: Rendszer tesztelése napfelkeltekor [saját ábra]



27. ábra: Rendszer tesztelése délután, naplementében [saját ábra]

Legvégül a rendszert teszteltem alkonyatban is (28. ábra). A rendszer hasonló megbízhatósággal működött, mint az előbbi látási viszonyokban, viszont már jól észrevehetően csökkent az érzékenysége a gyalogátkelőhelyek felismerésében (a lekopott felfestésűekre volt jellemző) a gyengébb fényviszonyoknak köszönhetően.



28. ábra: Rendszer tesztelése alkonyatban [saját ábra]

5.2. ÉJSZAKAI ÜZEMMÓD TESZTELÉSE

Szürkületben, a fényviszonyok jelentős romlásával már az vehető észre, hogy jelentősen csökken a rendszer megbízhatósága. Tapasztalataim alapján az alkonyat sötétebb stádiumában már jobb megbízhatóságot nyújt az IR-kamera és az éjszakai képek által betanított modell használata (29-30. ábrák). A betanított objektumokat nagyon kiváló pontossággal és érzékenységgel érzékeli, nem tapasztaltam romlást a nappali üzemmódhoz képest. Egyetlen gyenge pontja a rendszernek, ismételten a gyalogátkelőhely felfestések felismerése volt.



29. ábra: Rendszer tesztelése szürkületben, este [saját ábra]



30. Gyalogosfelismerés tesztelése szürkületben, este [saját ábra]

Legvégül leteszteltem a rendszer viselkedését egy éjszakai esős időben is. A rendszer ekkor sem okozott csalódást, a rossz körülmények ellenére sem. Kivilágítatlan utcában, tompított fényszóró használatával, esős szélvédőn keresztül is sikerült érzékelni a gyalogosokat (31. ábra), illetve a betanított közúti jelzéseket (32-33. ábrák), melyet az alábbi ábrák is igazolnak.



31. Gyalogosfelismerés tesztelése éjszaka, esőben [saját ábra]



32. ábra: Rendszer tesztelése éjszaka, esőben [saját ábra]



33. ábra: Rendszer tesztelése éjszaka, esőben [saját ábra]

6. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A rendszer vezérléséhez szükséges egy billentyűzet, mivel a gépjárművezető manuálisan kell beállítsa, hogy melyik üzemmódot szeretné használni, a saját megítélése alapján, viszont a további fejlesztésekben ezt automatizálnám. Egy fotoellenállás által szeretném érzékelni, hogy a nap milyen stádiumában vagyunk és ennek függvényében futna az épp releváns modell.

Az infravörös kamera használata jelentősen javított a rendszer megbízhatóságán éjszaka, gyenge megvilágítási viszonyok és rossz látási körülmények között használva. A továbbiakban, az őszi és téli közeledtével tesztelném a rendszer viselkedését ködben és havas időben is, illetve bővíteném a tanító adathalmazt ennek megfelelő képekkel. A rendszer pontossága a tesztképek alapján, 90% feletti, mindkét neurális háló esetében, viszont véleményem szerint egy olyan rendszerben, melytől emberi életek függenek, a maradék kevés hibaszázalék is sok tud lenni bizonyos szituációkban. TDK munkám során az egyik legegyszerűbb komplexitással rendelkező neurális hálót alkalmaztam a magasabb képfrissítési sebesség elérése érdekében, viszont ez a pontosság rovására ment. Ezt az adathalmaz további bővítésével szerettem volna kompenzálni, változatos látási viszonyokat és helyszíneket implementálva a tanító adathalmazba. Egy komolyabb erőforrással rendelkező rendszert használva már komplexebb neurális hálók betanítását és futtatását tenné lehetővé, mellyel sokkal jobb eredményeket lehetne produkálni.

Legvégül egy, a modernebb autókban található visszajelzésekhez hasonló, visszacsatolást szeretnék megvalósítani. Ezáltal a felismerést követően, az LCD kijelzőn az objektumok bekeretezésén kívül egy külön figyelmeztetőjelzés jelenne meg, melyet kísérhet fény és hang alapú visszajelzés is, mely testreszabható lenne a gépjárművezető által. Ezek a visszajelzések a felismerések függvényében változnának (pl. figyelmeztetés gyalogosra, figyelemfelhívás közeledő gyalogátkelőhelyre vagy értesítés elsőbbségadás kötelezettségéről). Jelzőlámpák közbeiktatásával kiküszöbölhető lenne a lámpás kereszteződésekben való félrevezetések, viszont ez további megoldandó problémákat vonzana maga után. Ez esetben meg kéne oldani, hogy a rendszer csak a saját sávunkhoz tartozó jelzőlámpa állapotát figyelje, viszont ehhez szükséges egy sávfelismerő algoritmus is. Fejlesztések szempontjából a határ a csillagos ég, viszont ehhez már komolyabb eszköz, tudás és még több idő igényeltetik.

7. ÖSSZEFOGLALÁS

A TDK munka elején kitűzött célokat sikerült elérjem. A célspecifikusabb eszköz használatával sikerült a képfrissítési sebességet megnövelni 37 FPS-re, mely közel 500% javulást jelent a kutatómunkám első változatához képest. Ezáltal a rendszer nagyobb sebességek mellett is használható, illetve annak reakcióideje sokat javult (~27 ms), jelentősen felülmúlva az emberi reakcióidőt (~200 ms).

A több órás tesztelések során azt tapasztaltam, hogy a rendszer kiváló pontossággal, és kitűnő érzékenységgel érzékeli a gyalogosokat, az úttesten/gyalogátkelőhelyen való elhelyezkedésüktől függetlenül, mind nappal, mind éjszaka. Véleményem szerint a megvalósított rendszer őket érzékeli a legjobban. Továbbá, a közlekedési táblákat is megfelelő érzékenységgel és pontossággal érzékeli, a magasabb pontossági mutatónak köszönhetően ritkábban fordulnak elő hibás detektálások. A rendszer gyengepontja a gyalogátkelőhelyek, melyeket gyakrabban figyelmen kívül hagy. Ez annak tulajdonítható, hogy ezek felismerését nagyban befolyásolja az úttest és a felfestés állapota és a rajtuk visszaverődő fény.

A rendszer összességében jól működik, kevés helyet igényel, könnyen beszerezhető, a szükséges eszközök elérhető áron beszerezhetők. Az érintőképernyős LCD kijelzőnek köszönhetően a visszajelzés vizuális módon érzékelhető a gépjárművezető által. Úgy gondolom, hogy ez a rendszer további fejlesztések után, használható lesz a régebbi gépjárművekben, melyek nem rendelkeznek vezetéstámogató rendszerekkel.

Munkám során hasznos információkat szereztem az autóvezetést támogató rendszerekről, azok múltjáról és működési elvükről. A feladat megvalósítása során megtapasztaltam, hogy milyen komplex folyamatok szükségesek ahhoz, hogy a számítógépek felismerék a különböző tárgyakat, betekintést nyertem a „computer vision” és a „deep learning” világába, továbbá megismerkedtem a mélytanulási folyamatokat segítő mély neurális hálókkal. Újabb eszköz használatát és programozását ismerhettem meg. Úgy gondolok mindezek megalapozó mérföldkövek lehetnek számomra a továbbiakban.

A feladat megvalósítása sok kihívással teli percet nyújtott, megtapasztaltam, hogy kellő kitartással és türelemmel bármit sikerül elérni, akármennyire lehetetlennek tűnjön az adott probléma. Biztos vagyok benne, hogy ezen a területen elsajátított ismeretek a jövőben is hasznomra válhatnak, hiszen a gépjárművezetést támogató rendszerek folyamatosan fejlődő innovációk, az autók pedig nélkülözhetetlen részét képezik az életünknek.

KÖSZÖNETNYILVÁNÍTÁS

Köszönetemet fejezem ki konzulensemnek, Sarvajcz Kornélnak, aki észrevételeivel, értékes tanácsival, valamint biztató szavaival járult hozzá a dolgozatom elkészüléséhez.

HIVATKOZÁSOK

- [1] WHO, „Global status report on road safety 2018,” [Online]. Available: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en. [Hozzáférés dátuma: 27 10 2020]
- [2] M. Rezaei és R. Klette, *Computer Vision for Driver Assistance Simultaneous Traffic and Driver Monitoring*, Switzerland: Springer, 2017.
- [3] R. Sabzevari és M. Rezaei, „Multisensor Data Fusion Strategies for Advanced Driver Assistance Systems,” February 2009. [Online]. Available: https://www.researchgate.net/publication/221787813_Multisensor_Data_Fusion_Strategies_for_Advanced_Driver_Assistance_Systems. [Hozzáférés dátuma: 28 10 2020].
- [4] T. Denton, *Automated Driving and Driver Assistance Systems*, New York: Routledge, 2020.
- [5] M. Csaba, „Computerworld,” 22 Január 2015. [Online]. Available: <https://computerworld.hu/uzlet/az-autovezetest-tamogato-rendszerek-technologia-trendjei-158014.html>. [Hozzáférés dátuma: 28 10 2020].
- [6] G. Foresti és C. S. Regazzoni, „Multisensor data fusion for autonomous vehicle navigation in risky environments,” *IEEE Transactions on Vehicular Technology*, 2002.
- [7] P. K. P. D. P. Y. D. Prof. Amit Choksi, „Smart Driver Assisting System using Raspberry PI,” *Researchgate*, 2018.
- [8] J. E. N. A. O. Luis Barba-Guaman, „Deep Learning Framework for Vehicle and Pedestrian Detection in Rural Roads on an Embedded GPU,” *Researchgate*, 2020.
- [9] A. S. S. K. J. M. N. H. M. R. Koray Ozcan, „Road Weather Condition Estimation Using Fixed and Mobile Based Cameras,” *Researchgate*, 2020.
- [10] S. N. A. I. G. S. A. B. E. P. V. Sanjay Kumar, „Smart driver assistance system using raspberry pi and sensor networks,” *Sciencedirect*, 2020.
- [11] A. Aljaafreh, „Camera-Based Driver Monitoring System for Abnormal Behavior Detection,” *Researchgate*, 2020.
- [12] K. Zoltán és C. László, *Számítógépes látás*, Typotex, 2011.

- [13] B. Dickson, „TechTalks,” 14 01 2019. [Online]. Available: <https://bdtectalks.com/2019/01/14/what-is-computer-vision/>. [Hozzáférés dátuma: 06 11 2020].
- [14] R. Stuart és P. Norvig, Mesterséges Intelligencia Modern megközelítésben Második, átdolgozott bővített kiadás, Budapest: Hungarian Translation Panem Könyvkiadó, 2005.
- [15] I. Goodfellow, Y. Bengio és A. Courville, „Deep Learning Book,” [Online]. Available: <https://www.deeplearningbook.org/contents/intro.html>. [Hozzáférés dátuma: 02 11 2020].
- [16] A. Rosebrock, Deep Learning for Computer Vision with Python, PyImageSearch, 2017.
- [17] T. Gnanasegaram, „Medium,” 27 02 2018. [Online]. Available: <https://medium.com/@tharanignanasegaram/artificial-neural-network-a-brief-introduction-572d462666f1>. [Hozzáférés dátuma: 06 11 2020].
- [18] J. B. Ahire, „Medium,” 24 08 2018. [Online]. Available: <https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4>. [Hozzáférés dátuma: 06 11 2020].
- [19] K. Róbert, „Mesterin,” 03 03 2019. [Online]. Available: <https://mesterin.hu/neuralis-halozatok-a-mesterseges-intelligencia-szent-gralja/>. [Hozzáférés dátuma: 06 11 2020].
- [20] S. Saha, „Towards data science,” 15 12 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. [Hozzáférés dátuma: 05 11 2020].
- [21] „Raspberrypi,” [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/?resellerType=home>. [Hozzáférés dátuma: 19 10 2021].
- [22] „Raspberrypi,” [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/camera/>. [Hozzáférés dátuma: 19 10 2021].
- [23] M. Singhal, „Medium,” [Online]. Available: <https://medium.com/@techmayank2000/object-detection-using-ssd-mobilenetv2-using-tensorflow-api-can-detect-any-single-class-from-31a31bbd0691>. [Hozzáférés dátuma: 19 10 2021].
- [24] A. Unruh, „Opensource,” 09 11 2017. [Online]. Available: <https://opensource.com/article/17/11/intro-tensorflow>. [Hozzáférés dátuma: 19 10 2021].

- [25] „NVIDIA Developer,” [Online]. Available: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>. [Hozzáférés dátuma: 19 10 2021].
- [26] „Seedstudio,” [Online]. Available: <https://www.seedstudio.com/NVIDIA-Jetson-Nano-Development-Kit-B01-p-4437.html>. [Hozzáférés dátuma: 19 10 2021].
- [27] „Waveshare,” [Online]. Available: <https://www.waveshare.com/imx219-160ir-camera.htm>. [Hozzáférés dátuma: 19 10 2021].
- [28] „Aliexpress,” [Online]. Available: <https://www.aliexpress.com/item/32278662678.html?spm=a2g0s.9042311.0.0.27424c4dstk5uT>. [Hozzáférés dátuma: 19 10 2021].
- [29] „Aliexpress,” [Online]. Available: <https://www.aliexpress.com/item/1005002139289868.html?spm=a2g0s.9042311.0.0.27424c4d3EoP0E>. [Hozzáférés dátuma: 19 10 2021].
- [30] „NVIDIA Developer,” [Online]. Available: <https://developer.nvidia.com/jetson-nano-sd-card-image>. [Hozzáférés dátuma: 19 10 2021].
- [31] D. Franklin, „Github,” [Online]. Available: <https://github.com/dusty-nv/jetson-inference>. [Hozzáférés dátuma: 19 10 2021].
- [32] D. Franklin, „Github,” [Online]. Available: <https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-ssd.md>. [Hozzáférés dátuma: 19 10 2021].
- [33] „Computer Vision Annotation Tool,” [Online]. Available: <https://cvat.org>. [Hozzáférés dátuma: 19 10 2021].
- [34] S.-H. Tsang, „Towards Data Science,” 03 11 2018. [Online]. Available: <https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11>. [Hozzáférés dátuma: 19 10 2021].
- [35] J. Brownlee, „Machine Learning Mastery,” 12 09 2020. [Online]. Available: <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>. [Hozzáférés dátuma: 19 10 2021].
- [36] J. Jordan, 01 03 2018. [Online]. Available: <https://www.jeremyjordan.me/nn-learning-rate/>. [Hozzáférés dátuma: 19 10 2021].
- [37] G. A. Dr. Tóth Pál, *Gépi látás és gépi tanulás alkalmazása ipari méretű tüzelési folyamatok ellenőrzésére*, Miskolc: Miskolci Egyetem, 2019.

- [38] R. J. Tan, „Towards Data Science,” 24 02 2019. [Online]. Available: <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>. [Hozzáférés dátuma: 19 10 2021].
- [39] J. Hui, „Medium,” 07 03 2018. [Online]. Available: <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>. [Hozzáférés dátuma: 19 10 2021].