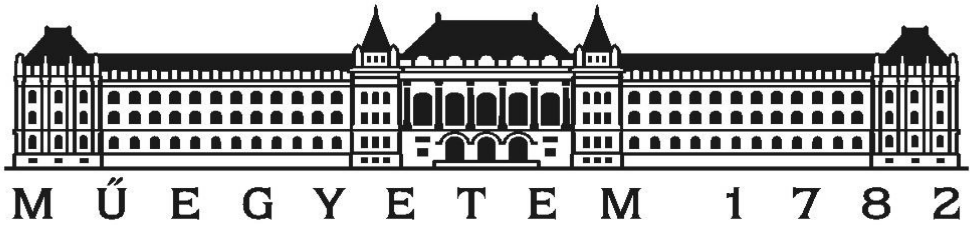


Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Kubinyi Bálint Miklós**
Neptun-kód:	OMJWZ6
Ágazat:	**Infokommunikáció**
E-mail cím:	**balint.kubinyi@gmail.com**
Konzulens(ek):	**Dr. Toka László**
E-mail címe(ik):	**toka.laszlo@vik.bme.hu**

Téma címe: Konvolúciós gépi tanulás az önvezető autók adatfeldolgozásában

Feladat

Az félévben a feladatom egy képfelismerő alkalmazás fejlesztése volt, konvolúciós neurális hálózatok (CNN)[i] segítségével. Ezen belül, specifikusan egy olyan CNN modell kialakítása volt a cél, mely képes a kiválasztott objektum pontos felismerésére és lokalizálására a képeken. A kiválasztott objektum egy általam tetszőlegesen választható objektum jelen esetben a tanító adatokban megtalálható oszlopokra esett a választás. A modell lehetett saját architektúrájú és lehetett egy előre tanított modell, amely modell tovább tanításával érjük el a kívánt eredményt.

2022/2023. 2. félév

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1 Bevezető

A projekt, amelynek keretében dolgoztunk, a Federated Learning technológiát fejlesztett önvezető autók működéséhez. A Federated Learning alapelve, hogy a tanulási folyamat elosztottan, az összes részt vevő eszközön párhuzamosan zajlik. Minden autóban egy különálló modell tanul, amely csak az általa megtanult összefoglaló információkat - azaz a súlyokat - továbbítja a központi modellnek. Ez a módszer kiemelkedően magas szintű adatvédelmet biztosít, valamint jól skálázható, így nagyszámú eszközt lehet bevonni a tanulási és modellképzési folyamatokba. A projekt legfőbb célja a Federated Learning elveinek alkalmazása az önvezető autók objektum felismerési képességének fejlesztésére.

A félév során a projekten belüli feladatom az volt, hogy objektum felismerést valósítsak meg konvolúciós neurális hálózat (CNN) segítségével. Csapatban dolgoztunk: míg én és egy társam: Ferenczi Gusztáv a CNN modellek kifejlesztésén és tanításán dolgoztunk, addig két másik csapattag az adatok előállításáért felelt. Célunk az volt, hogy mindegyikünk külön-külön válasszon ki egy-egy objektumot, amelyekhez jól működő modelleket fejlesztünk ki, amelyek képesek az adott objektumok pontos felismerésére és lokalizálására a képeken. Ehhez először saját modelleket fejlesztettünk, majd később előre tanított modellek tovább tanítását alkalmazva jutottunk el a megoldáshoz. Mindenki önállóan különböző modelleket fejlesztett az általa választott objektumok felismerésére. A tapasztalatokat egymással megosztottuk, így kialakítva egy gördülékeny csapat munkát.

.

1.2 Elméleti összefoglaló

A projektünkben alkalmazott technológiák közül talán a legfontosabb a konvolúciós neurális hálózat (CNN), melyet az objektum felismerés feladatára használtunk. A konvolúciós neurális hálózatok a gépi tanulásnak egy speciális, képfelismerésre optimalizált ága. Képesek a képen található összetett mintázatok felismerésére, lokalizálására és osztályozására, így kiválóan alkalmazhatóak önvezető autók számára, ahol a környezetben lévő objektumok azonosítása és helyzetük meghatározása elengedhetetlen.

A CNN működése[2] a következőképpen írható le: a hálózat a képet, mint egy nagy mátrixot kezeli, ahol minden egyes pixel egy elemet jelent ebben a mátrixban. A hálózat konvolúciós rétegei ezután egy kisebb, a kép egy kis részletét leíró, ún. kernelt "végigcsúsztatnak" a képen. Ez a kernel egy kisebb mátrix, amely egy speciális mintázatot határoz meg. A hálózat a konvolúció műveletével számítja ki, hogy a kép egyes részei mennyire hasonlítanak a kernelben meghatározott mintázatra. Ezzel a módszerrel a hálózat képes azonosítani a kép jellemzőit.

A konvolúciós rétegeket általában úgynevezett "pooling" rétegek követik, amelyek a kép méretét csökkentik, és kiemelik a legfontosabb, a konvolúciós rétegek által megtalált jellemzőket. Ez a folyamat segít csökkenteni a túltanulás (overfitting) lehetőségét, ami akkor következik be, amikor a hálózat túlságosan specifikus jellemzőket tanul meg és elveszti általánosítási képességét.

Ezt követően a hálózatban általában "fully connected" rétegek találhatóak, amelyek végül a hálózat kimenetét adó osztályozó rétegeket alkotják. Ezek a rétegek minden neuronját összekötik minden előző réteg neuronjával, így az összes információ összevonásra kerül, és a végén a hálózat képes döntést hozni a bemenetek alapján.

Ezeknek az elméleti ismereteknek mélyebben is utána lehet olvasni a szakirodalomban[3], habár a projekt megértéséhez véleményem szerint elegendő ennyi háttér tudás. Ez annak köszönhető, hogy a projekt során előre tanított modelleket alkalmaztam, amelyek magukban

foglalják ezeket az alapelveket. Általánosságban egy előre tanított modell egy adott objektum felismerési feladatra optimalizált architektúrát alkalmaz, amely nagy adatokon tanult. Ezek a modellek egyenként más-más adatokat várnak a bemenetre és adnak kimenetül. A projekt során használt előre tanított modelljeim a "resnet"[4] és a "YOLO"[5] nevű modellek melyekről részletesebben a dokumentációkban [3] lehet olvasni. A választás azért esett ezekre a modellekre, mert ezek egyszerűen teljesítik az objektum felismerés elvárásait. Ahhoz, hogy képesek legyünk egy objektum felismerési feladatot megoldani, valójában egy összetett problémával nézünk szembe, mivel két különböző képfelismerési problémát is egyszerre kell megoldanunk. Nem elég megmondani, hogy egy adott képen rajta van-e a keresett objektum, de meg kell tudni határozni, hogy hol helyezkedik el. Ez a két feladat az osztályozás és a regresszió. Az osztályozás egy előre meghatározott diszkrét számú kategóriákhoz tartozó címkéket rendel az objektumokhoz. Ezek az objektumok jelen esetben lehetnek autók, oszlopok vagy közlekedési lámpák stb. Általában a modell jósol egy valószínűségi eloszlást az osztályok között, és megad egy bizonyossági mértéket, hogy mennyire biztos, hogy az objektum egy adott osztályhoz tartozik. A regressziós feladat folyamatos értékek „jóslására” alkalmas és az objektum felismerés esetében az objektumot körbe foglaló doboz (bounding box) koordinátáit adja meg. Következésképp ezekhez a „multi-task” feladatokhoz olyan veszteség függvényt kell alkalmazni, amely alkalmas egyszerre a két probléma kezelésére.

1.3 A munka állapota, készültségi foka a félév elején

Ez a félév az első, hogy ezzel a projekttel foglalkozom, a félév elején megkaptam a feladatot, majd a félév során a generált adatokat, amellyel a modelleket kellett tanítani.

2. Az elvégzett munka és eredmények ismertetése

Szoftver	GitHub Csillagok	GitHub Commitok
CARLA	9,237	5,833
BeamNG.py	187	640

1. ábra A szoftverek népszerűségének felmérése github aktivitások alapján

1.1 Szimulációs környezet kiválasztása

Annak érdekében, hogy elkezdhessem egy modell fejlesztését és betanitását az elvárásoknak megfelelően először számításba kellett venni néhány szempontot, melyek alapján eldönthető, hogy mely technológiákat érdemes választani a probléma megoldásához. Az első és talán az egyik legfontosabb szempont a tanításhoz felhasználható adatok mennyisége és minősége. Jelen esetben két különböző szimulációs környezet által generált képekből álló adathalmaz állt rendelkezésre és állított döntés elé. A két szimulátor a BeamNG és a Carla nevű programok voltak, ezek közül kellett kiválasztani azt, amelyik által generált képek a legmegfelelőbbek objektumfelismerési feladatok megoldására. A választásnál figyelembe kellett venni az annotációs fájlok minőségét és komplexitását, a képek méretét és a generálható objektum típusok számát. Kezdetben a Carla által generált annotációs fájlokat vizsgálva arra a következtetésre jutottunk, hogy rendkívül bonyolultak, felhasználásuk körülményes és sok esetben hibás adatokat szolgáltat. Így annak ellenére, hogy a github aktivitások vizsgálatát követően kiderült, hogy a Carla szimulátort jelentősen többen használják, végül a BeamNG -re esett a választás. Ez a döntás a későbbiekben jelentősen megkönnyítette az adatok előfeldolgozását az egyes modellek elvárásainak megfelelően

1.2 Felhasznált fejlesztői eszközök

Mivel a neurális hálózatok tanítása rendkívül hardver igényes feladat, ezért nem volt elhanyagolható szerepe a megfelelő fejlesztői eszközök kiválasztásának sem. Legfőképpen erős grafikus számítási egységre van szükség a tanításhoz. Ez magában foglalja a megfelelő méretű memóriát, a félkész eredmények, súlyok tárolására, minél gyorsabb működést. Habár a számítások nagy részét az erős grafikus egység végzi az ilyen feladatok esetében, nem elhanyagolható a CPU teljesítménye sem. Egy erős processzor képes felgyorsítani a tanítás nem párhuzamosítható részeit. Nem elhanyagolható a rendelkezésre álló tárhely sem, mivel a tanuláshoz szükséges minden adatot el kell tudnunk tárolni. Ezen szempontok figyelembevételével minden kétséget kizáróan könnyen belátható volt, hogy valamilyen fajta felhő tárhelyet kell használnunk, valamint a számításokat is a felhőben kell elvégezni mivel lokálisan nem áll rendelkezésre nagy teljesítményű hardver. Ahhoz, hogy el lehessen dönteni mely eszköz a leg alkalmasabb erre a feladatra számos szempontot kellett számításba venni. Például a felhőben rendelkezésre álló tárhely méretét és árát, a rendszer által rendelkezésre bocsátott grafikus kártyát és a grafikus számításhoz kötött különböző számlázási módokat. A felhő által felkínált fejlesztői környezet komplexitását és a felhasználók körében mért népszerűségét, hogy a támogatottság is megfelelő legyen. Így a legnépszerűbb felhő platformokat megvizsgálva arra a megállapításra jutottunk, hogy a leg egyszerűbben használható platform a Google Colab[6] és a Google Cloud Platform, ami ráadásul a kezdetben ingyenesen használható és könnyedén menedzselhető erőforrásokat valamint egy Jupyter notebookot is biztosít.

1.3 Felhasznált Python könyvtárak:

	Keras	PyTorch	TensorFlow
API Level	High	Low	High and Low
Architecture	Simple, concise, readable	Complex, less readable	Not easy to use
Datasets	Smaller datasets	Large datasets, high performance	Large datasets, high performance
Debugging	Simple network, so debugging is not often needed	Good debugging capabilities	Difficult to conduct debugging
Does It Have Trained Models?	Yes	Yes	Yes
Popularity	Most popular	Third most popular	Second most popular
Speed	Slow, low performance	Fast, high-performance	Fast, high-performance
Written In	Python	Lua	C++, CUDA, Python

[7]

2. ábra A felhasznált python könyvtárak összehasonlítása

A modellek tanításához először ki kellett választani a feladatra legalkalmasabb python könyvtárat, amelynek segítségével a legegyszerűbb módon lehetséges modelleket tanítani és széles körben támogatottak. Három népszerű könyvtárra esett a választás, melyekkel minden tervezett funkció, mint például a saját modell építése és a tovább tanítás is megvalósítható és megfelelő mennyiségű és minőségű dokumentáció áll rendelkezésre hozzájuk. Ezek a könyvtárak[8] a Pytorch, Tensorflow és Keras. Ezek közül a Pytorch egy egyszerűen használható facebook által fejlesztett aránylag újnak mondható könyvtár. A Tensorflow a legszéleskörűbben elterjedt több absztrakciós szinten működő könyvtár. A Keras mostanra már a Tensorflow része, de könnyebben használható, felhasználó barát és a gyors fejlesztésre fókuszáló. Ahogy a 2. ábra is mutatja ezek a könyvtárak néhány szempont szerint eltérőek, de ennek köszönhetően felváltva alkalmazva a gépi tanulási feladatok egy nagyon széles skálájának implementálására alkalmasak. Erre azért van szükség a projekten belül, mert így számos objektum felismerő modell implementálása és tesztelése válik elérhetővé, mert az egyes könyvtárak más és más modelleket implementálnak.

1.4 Saját modell építése:

Annak érdekében, hogy egy saját modell működését is vizsgálni tudjam, készítettem több leegyszerűsített modellt, ami nem tartalmazta a regressziós részt. Ezek a csak klasszifikációra alkalmas modellek képesek voltak a kiválasztott osztályok azonosítására a képeken, de beláthatóvá vált, hogy a regressziós rész megvalósítása sokkal bonyolultabb probléma és egy előre tanított modell tovább tanításával jelentősen jobb eredményeket leszünk képesek elérni.

1.5 Adatok előkészítése:

```
dataset = {
    "image_name": [],
    "top_x": [],
    "top_y": [],
    "bottom_x": [],
    "bottom_y": [],
    "class_name": [],
}

# Iterate over all the annotation files
for label_file in os.listdir(labeldir):
    if label_file.endswith('.xml'):
        # Parse the XML
        tree = ET.parse(os.path.join(labeldir, label_file))
        root = tree.getroot()

        # Find corresponding image file
        num = re.search(r'\d+', label_file).group()  # extract number from annotation file
        image_file = f'image_{num.zfill(4)}.jpg'

        if image_file in os.listdir(imagedir):
            for obj in root.iter('object'):  # iterate over all objects in the xml file
                dataset["image_name"].append(imagedir + "/" + image_file)
                dataset["top_x"].append(int(obj.find('bndbox/xmin').text))
                dataset["top_y"].append(int(obj.find('bndbox/ymin').text))
                dataset["bottom_x"].append(int(obj.find('bndbox/xmax').text))
                dataset["bottom_y"].append(int(obj.find('bndbox/ymax').text))
                dataset["class_name"].append(obj.find('name').text)
```

3. ábra Az xml beolvasása a resnet50 által elvárt formátumban.

Egy modell tanításánál az egyik legfontosabb feladat az adatok előkészítése a tanításra. Ez azért jelenthet kihívást, mert a kezdeti adatformátum, amelyben a bemeneti adatok érkeznek sosem egyezik meg a modell által előírt formátummal. A modell csak akkor képes tanulni, ha a bemeneti adatok minden előírásnak megfelelnek. Az elvárások közé tartozhat az adatok struktúrája és a képek felbontása. A BeamNG által generált adathalmaz a következőképp állt rendelkezésre. 1024*1024 -es felbontású „webp” -formátumú képek és a képekhez tartozó annotációk „xml” -formátumban. Ahhoz, hogy az adatok bármilyen formában használhatók legyenek először be kellett olvasni őket. Ehhez a 3. ábrán látható függvényt alkalmaztam, így külön sorban tárolódik minden adat az objektumokat körbe foglaló dobozonként. Egyetlen képhez több bejegyzést tárolva. Erre a tárolási módra azért van szükség mert az első általam tanított modell a resnet50 ezt az adatformátumot várta el.

1.6 Előre tanított modellek tovább tanítása

4. ábra resnet50 által prediktált dobozok.

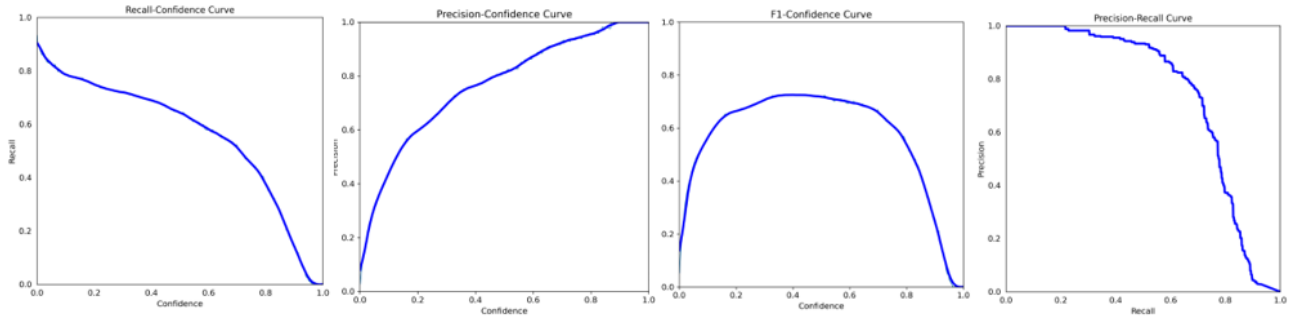
Az előre tanított modellek kiválasztásánál a legfőbb szempontok azok, hogy legyen a modell egy lépésben végezze el a klasszifikációt és a regressziót ez által hatékonyan működjön és jelentős mennyiségű munkát spóroljon. Ezért esett az első választás a resnet50 re. A resnet 50 tovább tanítása során értékelhető eredményt ért el 10 epoch[9] esetén. Ennek az eredményét a 4. ábra

szemlélteti. Megfigyelhető, hogy habár megtalálja az oszlopok nagy részét, nem működik mindig tökéletesen, mert vannak esetek amikor egy objektumot többször is megtalál és vannak esetek amikor nem találja meg a szükséges objektumot. Ezen egy keveset lehet még kissé javítani az epoch szám növelésével, de ez jelentős hardver igény növekedést is jelent. Lehetséges magasabbra vagy alacsonyabbra állítani azt a határértéket, mely meghatározza, hogy mennyire „biztos” dobozok jelenhetnek meg végeredményként. Ezen paraméterek finomhangolásával sikerült valamivel jobb eredményt elérni ezzel a modellel. Ennek ellenére arra a döntésre jutottam, hogy egy másik előre tanított modell talán még jobb eredményre vezethet. A kettőt összehasonlítva kiválaszthatom azt amelyik a jobb eredményeket tudja szolgáltatni. Ezért találtam rá az egyik legnépszerűbbnek mondható modellre a YOLO -ra. A YOLO frissebb verzióit a Keras nevű könyvtár már nem támogatta, ezért kellett váltani pytorch-ra, annak érdekében, hogy a lehető legegyszerűbb módon lehessen tanítani a yolo modellt. A yolo bemenetül nem ugyan olyan adatokat vár el mint a resnet50, ezért módosítani kellett az adat tárolásért felelős kódot. A YOLO egy sorban várja a kép elérési útját és az összes hozzá tartozó doboz [10]koordinátát és osztálycímkéket. Továbbá hatalmas különbség még a resnet50 -hez képest, hogy a YOLO a doboz koordinátáit úgy várja el, hogy a doboz középpontját és a szélességet illetve magasságot kell megadni neki, ahogy az 5. ábrán is látható.

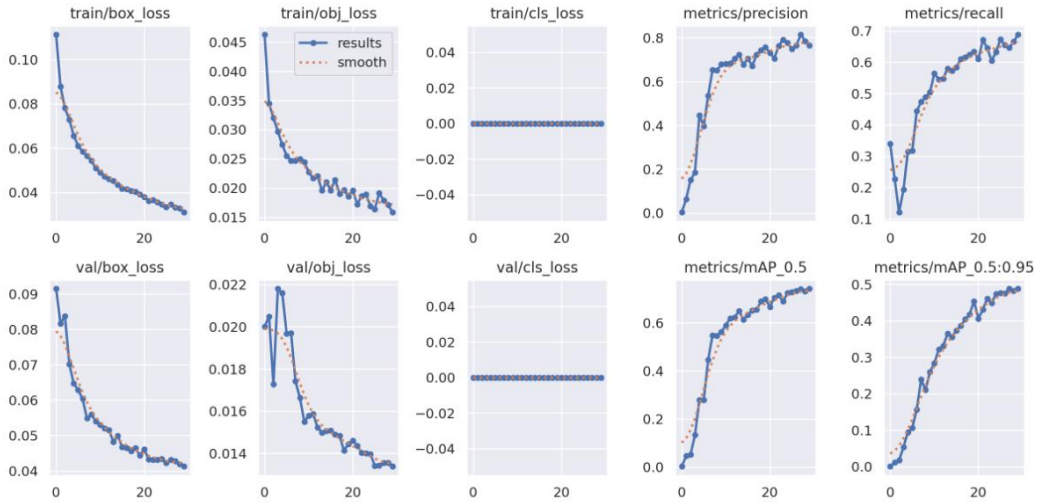
5. ábra YOLO által elvárt adatformátum

Mivel a YOLO jelentősen jobb eredményeket ért el a resnet50 -nél, ezért részletesebben a YOLO által elért eredményeket elemzem. Az objektum detektálás eredményessének vizsgálatára több mutató áll rendelkezésre, melyeknek elemzésével pontosabb képet kaphatunk a modell működéséről. Egy modellt sok féle képpen tesztelhetünk. Az egyik legfontosabb lépés amit még a modell tanítása előtt el kell végeznünk az a rendelkezésre álló adatok két részre bontása. Az egyik rész a tanításra lesz fenntartva általában ez a nagyobb rész nagyjából 80-20 arányban. A másik a validációs adathalmaz. Ez arra szolgál, hogy a tanítási folyamat közben ellenőrizhető legyen a működés egy olyan adathalmazon amelyet a modell még nem látott a tanulás közben. Így megfigyelhetővé válik, hogy a modell mennyire teljesít jól valójában. Ezt a módszer arra is alkalmazható, hogy észrevehető legyen a túltanulás. A túltanulást úgy lehet észrevenni, hogy elemezve a validációs és tanító adatokon elért eredményeket, láthatóvá válik, ha a modell nagyszerűen teljesít a tanító adatokon, de jelentősen rosszabbul a validációs adatokon. Ez azt jelenti, hogy a modell "túltanulta" a tanító adatokat, vagyis túlságosan specifikusan igazodott hozzájuk, ami csökkenti a modell általánosító képességét, és nem képes jól teljesíteni olyan adatokon, amelyeket korábban nem látott.

1.7 Modell kiértékelése



6. ábra sorrendben: recall confidence curve, precision confidence curve, F1 confidence curve, precision recall curve



7. ábra modell kiértékelése

8. ábra validációs és tanító halmaz veszteség függvényei

Az F1[1112] Score, a Pontossági Görbe (Precision Curve) és a Visszahívási Görbe (Recall Curve) mind olyan eszközök, amelyek segítségével értékelni tudjuk a modellünk teljesítményét.

Az F1 Score a precision és a recall harmonikus középértéke, ami azt jelenti, hogy a pontosság és a recall közötti összefüggést vizsgálja. Minél magasabb a confidence küszöbérték, annál magabiztosabb a modell a predikcióiban. Azonban túl magas confidence küszöbérték esetén a modell sok igaz pozitív esetet (True Positive) hagyhat ki, így csökkenhet a Recall értéke. Ez a két mutatószám külön-külön is fontos információt hordoz a modell teljesítményéről, de az F1 Score összegzi őket egyetlen számban, ami segít abban, hogy könnyebben összehasonlíthassuk a különböző modelleket. A legjobb F1 Score érték 1, ami tökéletes precision és recall értéket jelent, míg a legrosszabb 0, ami azt jelenti, hogy a modell sem a precision, sem a recall tekintetében nem

teljesít jól. Ez az eset a valóságban nagyon ritkán fordul elő, így ez a két érték általában kompromisszumot jelent az aktuális modell céljának igénye szerint. Az általam készített modell esetében az F1 görbe a 6. ábra harmadik grafikonján látható és azt tudjuk meg róla, hogy a modell elég jól teljesít mind a pontosság mind a recall esetében. Mivel itt egy kompromisszumot kell kötni az ábrán egy még elég jó pontosságnak mondható a 0,6, amelyhez kb 0,75 -ös recall érték tartozik.

A "precision-confidence" görbe, a 6. ábra 2. grafikonja, az objektum detektálás pontosságát (precision) ábrázolja a modell bizonyossága (confidence) függvényében. A görbe minden pontja azt mutatja meg, hogy a modell milyen pontossággal képes detektálni az objektumokat egy adott bizonyossági küszöbérték mellett. Például, ha egy adott bizonyossági küszöbérték mellett a modell 90%-os pontossággal képes detektálni az objektumokat, akkor a "precision-confidence" görbe ezen a ponton magas értéket fog mutatni. Ha viszont a bizonyossági küszöbérték növekszik, és a modell pontossága csökken, akkor a görbe értéke csökken.

A Visszahívási Görbe, a 6. ábrán az első grafikon, más néven Recall Curve, a recall értékeit ábrázolja különböző döntési küszöbértékek mellett. A recall azt mutatja meg, hogy az összes ténylegesen pozitív esetből hányat ismert fel helyesen a modell

Ezek a mutatószámok és görbék tehát segítenek minket abban, hogy mélyebben megértsük a modellünk teljesítményét, és hogy lássuk, hogyan változik a modell teljesítménye a döntési küszöbérték változásával. Ezenkívül segítenek felismerni a trade-offot vagy kompromisszumot a precision és recall között: a tökéletes pontosság gyakran a recall rovására megy, és fordítva.

Az objektum felismerések kiértékelése:

A 7. ábrán látható box loss[13] és obj loss függvények arra szolgálnak, hogy megmutassák, mennyire jól képes a modellünk megjósolni az objektumok pozícióját és annak létezését. A box loss az objektum pozíciójának pontosságát jellemzi - minél alacsonyabb az értéke, annál jobban sikerült a modellnek meghatározni a megfelelő "keretet" (bounding box) az objektum körül. A train box loss görbe lefelé ível, ami azt jelenti, hogy a modell egyre jobban tanul a tanítási adatokon, és végül eléri a 0.04-es minimumot, ami azt jelenti, hogy a modellünk jelentős mértékben javult az idő előrehaladtával.

Az obj loss az objektum létezésének bizonyosságát jellemzi - minél alacsonyabb az értéke, annál biztosabban detektálja a modell az objektumokat. A train obj loss görbe szintén lefelé ível, végül 0.015-ös minimumot ér el, ami azt jelzi, hogy a modellünk egyre jobban képes az objektumok detektálására a tanítási adatokon.

A val box loss és val obj loss grafikonok a validációs adatsoron mért értékeket jellemzik. Itt is látható a lefelé ívelő trend, a val box loss 0.04-es, a val obj loss pedig 0.014-es minimumot ér el. Ez azt jelenti, hogy a modellünk jól generalizál, nemcsak a tanítási adatokon teljesít jól, hanem az olyan új adatokon is, amelyeket eddig nem látott a tanítás során. Ez fontos, mert ez azt mutatja, hogy a modellünk valószínűleg jól fog teljesíteni a valós világban, nemcsak a tanítási adatokra "emlékezik".

A 7. ábra class -ra vonatkozó része jelen esetben nem túl beszédes, mert csak egyetlen osztályt a pole-t használtam a betanításhoz ezért látható csak egy egyenes vonal.

A metrics/precision és metrics/recall grafikonok az osztályozási teljesítményt mutatják a modell számára.

A metrics/precision grafikon arról tájékoztat minket, hogy milyen arányban detektálta helyesen a modellünk az objektumokat. Amikor a precision értéke magas, az azt jelenti, hogy a modell nagyon pontosan képes felismerni az objektumokat, és kevés a téves pozitív találat. A 27. epochnál elért 0.8 feletti érték azt jelzi, hogy a modell nagyon pontosan képes felismerni az objektumokat. Azonban az utolsó epochnál valamivel visszaesik 0.8 alá, ami azt jelzi, hogy a modell pontossága valamelyest csökkent ezen a ponton. Ez utalhatna arra, hogy a modell esetleg

túltanult, de a visszaesés mértéke elhanyagolható és így is felfele mutató trendet mutat a grafikon így több epoch használatával valószínűleg javulna az eredmény ebből kifolyólag ez még nem utal túltanulásra

A metrics/recall grafikon az érzékenységet, vagy más néven a visszahívást jellemzi. A recall azt mutatja, hogy a modell mennyire képes felismerni az összes létező objektumot. Az első epochnál 0.35-ös érték azt jelenti, hogy a modell képes volt felismerni a létező objektumok 35%-át. A 3. epochig tartó zuhanás, ami 0.15-ös értéknél áll meg, azt mutatja, hogy a modell kevésbé képes az objektumok felismerésére ebben az időszakban. Azonban a folyamatos növekedés, amely a tetőpontját az utolsó epochnál éri el 0.7-et, azt jelzi, hogy a modell fokozatosan javul, és végül képes felismerni a létező objektumok 70%-át. Ez azt mutatja, hogy a modell folyamatosan tanul és javul az idő előrehaladtával.

A bounding box predikciók kiértékelése:

A metrics/mAP_0.5 és metrics/mAP[14]_0.5:0.95 grafikonok a Mean Average Precision (átlagos pontosság) mutatószámokat jelzik, amelyek egy fontos mércéi a detekciós modellek teljesítményének.

A metrics/mAP_0.5 azt mutatja meg, hogy a modell mennyire pontosan képes detektálni az objektumokat, amikor az IoU (Intersection over Union) küszöbértékét 0.5-nek vesszük. Ez azt jelenti, hogy az általunk elfogadott overlap a prediktált és a valós bounding box között legalább 50%. A grafikonon látható folyamatos növekedés azt mutatja, hogy a modellünk egyre jobban képes detektálni az objektumokat 0.5-ös IoU küszöbérték mellett, és a végén eléri a 0.7 körüli értéket. Ez azt jelenti, hogy a modell a detektált objektumok nagy részét helyesen ismerte fel ebben az IoU tartományban.

A metrics/mAP_0.5:0.95 azt jelenti, hogy a modellünk átlagos pontosságát több különböző IoU küszöbérték (0.5-től 0.95-ig, 0.05-ös lépésekkel) mellett vizsgáljuk. Ez a mutatószám ad egy átfogóbb képet a modell teljesítményéről, mivel vizsgálja a modell pontosságát különböző overlap értékeknél. A folyamatos növekedés itt is azt mutatja, hogy a modellünk egyre jobban képes detektálni az objektumokat a különböző IoU küszöbértékek mellett, és a végén eléri a 0.5 körüli értéket. Ez azt jelenti, hogy a modell a detektált objektumok jelentős részét helyesen ismerte fel ebben a széles IoU tartományban.

Utoljára a 8. ábrán látható grafikont készítettem el, melynek célja szemléltetni, a validációs és tanuló adatokhoz tartozó veszteség függvény grafikonjait. Ez azért lehet hasznos, mert ennek segítségével könnyedén felismerhető a túltanulás. Ha a validációs veszteség értéke emelkedni kezd, miközben a tanuló adatok vesztesége továbbra is csökken vagy alacsony szinten marad, akkor túltanulásról (overfitting) beszélhetünk. A 7. ábrán látszik, hogy a validációs és tanuló veszteség görbe is lefelé ívelő trenden vannak. Ebből arra lehet következtetni, hogy nem történt overfitting. Ha több epochot használva tovább tanítottuk volna a modellt és észleljük az overfitting jeleit akkor célszerű lenne azt az utolsó epoch által előállított modellt alkalmazni, amelynél a validációs veszteség a minimális és még nem látszanak a túltanulás jelei. Ezt egy úgynevezett early stopping azaz korai befejezés beiktatásával lehet elérni és így automatikusan a legjobb epoch által készített modell kerül felhasználásra. Ezt a funkciót a YOLO magától is támogatja, de jelen esetben nem nincs jelentősége, mert nem használok nagyon magas epoch számot.

2.2 Összefoglalás

A félév során Convolutional Neural Networks (CNN) gépi tanulási modellekkel foglalkoztam. A feladat objektum felismerés volt, ez az objektumok klasszifikációját és a felismert objektumok helyzetének bounding boxokkal történő megjelölését foglalja magában. A munkám célja egy ilyen modell létrehozása volt, melyet a federated learning technológia alkalmazásával önvezető autókhoz szeretnénk használni a tágabb projekt részeként.

A cél eléréséhez elsősorban a YOLO és ResNet50 előre tanított modelleket használtam, amelyeket tovább tanítottam a kiválasztott objektum felismerésére, mert ketten dolgoztunk a feladaton, Ferenczi Gusztáv kollégámmal és mindenkinek egy objektumot kellett választani amelyre modellt tanít be. Ez a munkafolyamat több lépésben valósult meg: először saját modellekkel kísérleteztem, melyeket saját magam szerettem volna megtervezni, de ez a módszer nem volt sikeres. Majd kiválasztottam a fejlesztői környezetet, ami a Colab és a Google Cloud lett. Python könyvtárként a TensorFlow, Keras és PyTorch könyvtárakat használtam.

Az adatok beolvasása és előkészítése után megkezdődött a modell tanítása és kiértékelése. A legfontosabb eredményem egy jól működő, általam tovább tanított YOLO modell sikeres betanítása volt.

Az elemzésből levont következtetések között szerepel, hogy kis mennyiségű adat esetén célszerű előre tanított modelleket használni. Azt is megállapítottam, hogy a használt modellek közül a YOLO a legmegbízhatóbb, és hogy fontos a megfelelő modell kiválasztása a probléma megoldásához.

A munkám további fejlesztési lehetőségei közé tartozik a YOLO modell további epochokkal történő tanítása, valamint több objektum hozzáadása a modellhez, hogy az önvezető autók számára szükséges objektumokat képes legyen felismerni.

Összefoglalva, a félév során elért eredményem egy jól működő objektum detektáló modell létrehozása volt, ami képes a kiválasztott objektumokat felismerni és helyzetüket bounding boxok-kal megjelölni. Ez a modell elősegítheti az önvezető autók fejlesztését, ahol az objektumok felismerése és helyzetük pontos meghatározása elengedhetetlen.

3. Irodalom, és csatlakozó dokumentumok jegyzéke

A tanulmányozott irodalom jegyzéke:

[i] https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network : convolutional neural network (CNN) utolsó letöltés ideje: 2023.05.30

[2] https://www.kaggle.com/learn/computer-vision Kaggle tutorial: Computer Vision utolsó letöltés ideje: 2023.05.29

Build convolutional neural networks with TensorFlow and Keras.

[3] Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions: https://www.mdpi.com/2079-3197/11/3/52

[4] Deep Residual Learning for Image Recognition: https://arxiv.org/pdf/1512.03385.pdf utolsó letöltés ideje: 2023.05.29

[5] Object Detection with YOLOv5 and PyTorch: https://www.section.io/engineering-education/object-detection-with-yolov5-and-pytorch/ utolsó letöltés ideje: 2023.05.29

[6] https://colab.research.google.com/

[7] https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article : Keras vs Tensorflow vs Pytorch: Key Differences Among Deep Learning utolsó letöltés ideje: 2023.05.30

[8] https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article

[9] egy epoch azt jelenti hogy az összes tanító adatot egyszer körbe járjuk a modell súlyainak frissítése érdekében. Minél több epochot futtatunk annál eredményesebben tudja megtanulni a modell a mintázatokat. Túl sok epoch esetén a modell túl tanulhat, ami szintén nem előnyös a feladat megoldása szempontjából.

[10] https://docs.cogniflow.ai/en/article/how-to-create-a-dataset-for-object-detection-using-the-yolo-labeling-format-1tahk19/: How to Create a Dataset for Object Detection using the YOLO Labeling Format utolsó letöltés ideje: 2023.05.29

[11] https://deepai.org/machine-learning-glossary-and-terms/f-score what is the F-score utolsó letöltés ideje: 2023.05.30

[12] https://www.v7labs.com/blog/f1-score-guide What is F1 score? utolsó letöltés ideje: 2023.05.30

[13] https://www.baeldung.com/cs/training-validation-loss-deep-learning: Training and Validation Loss in Deep Learning utolsó letöltés ideje: 2023.05.30

[14] https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52: Breaking Down Mean Average Precision (mAP) utolsó letöltés ideje: 2023.05.30