

Yolov8-as szegmentációs háló és annak magyarázhatósága EigenCAM típusú modellfüggő magyarázó rendszerrel

Nyilas Péter, Konzulensek: Dr Hullám Gábor

2024. május 14.

Tartalomjegyzék

1. Bevezetés	3
2. Mély tanulás és interpretálhatóság	4
2.1. Magyarázhatóság	4
3. YOLOv8: Szemantikus szegmentációs háló és működése	5
3.1. Yolov8 architektúra	5

1. Bevezetés

Közlekedési objektumok detektálásával foglalkoztam a tavalyi témalaboratóriumomban, gépi látás (**CV! (CV!)**) témakörében, mindezt Yolo architektúrájú neurális hálóval. Akkor a YOLOv5-tel foglalkoztam hagyományos keretes adatrepräsentáció mellett és a dolgozatom végén haladási tervet fogalmaztam meg azért, hogy jobban elmerüljek ebben a területben. Ezek akkor az alábbiak voltak:

- a. újabb hálóarchitektúra vagy nagyobb háló alkalmazása
- b. áttérni a szemantikus szegmentációs objektumrepräsentációra a bounding box alapúról
- c. új (hardveres) erőforrás és jobb szoftveres kiszolgálás beszerzése és felépítése

A két félév között viszont megismerkedtem a modellmagyarázási módszerekkel így hozzáadódott az alábbi pont:

- d. a háló döntéseit milyen magyarázó módszer mellett tudnám lehető legérhetőbben megjeleníteni és ezáltal megérthetővé tenni.

Előző féléves szakmai gyakorlatomban, képfeldolgozó neurális hálókkal dolgoztam. Itt feladatom volt megismerkedni az adatfeldolgozással, MLOps-al is ami, a mesterséges intelligenciával segített szoftverek kiszolgálását és fejlesztését könnyíti. Ekkor szembesültem azzal, hogy a neurális hálók azonban gyakran fekete dobozként (black-box) működnek, ami komplikálttá teszi azok megértését. Azért, hogy biztonságosabbnak tudhassuk ezeket és megértésükkel későbbi hibáikat kijavíthatunk, fontos interpretálhatóság. Az interpretálhatóság nem más mint a modell döntéseinek megértése és magyarázata, ami kulcsfontosságú ezen modellek elfogadhatóságában és alkalmazhatóságában.

A YOLOv8 (You Only Look Once version 8) egy népszerű és hatékony konvolúciós neurális háló. Ennek a szegmentációs változatát használom (Yolov8m_seg), ez például objektumfelismerésre és objektum-klasszifikációra is tökéletesen használható. A modell architektúrája ismeretében betekinthetünk majd valamilyen külső eszközzel a rétegei közé, és megpróbálhatjuk megérteni a rétegek (layerek) közötti aktivációs függvények kimeneti alapján, a háló viselkedését.

Egy ilyen eszköz az EigenCAM (Eigen Class Activation Mapping) ami egy modell-függő és gradiensmentes magyarázó rendszer, amely képes vizualizálni, hogy a mély tanuló hálózatok mely részei járultak hozzá a döntéshozatalhoz. Ehhez pedig az aktivációs függvények értékeit kiemeli a háló rétegeiből és a bemeneti képre vetíti ezeket, majd kombinálja ezt a mátrixot és az eredeti képet eggyé, megmutatva azt, hogy a kép mely részei milyen fontosak a döntések meghozásában.

2. Mély tanulás és interpretálhatóság

A mesterséges intelligenciát tartalmazó szoftvereket azért használjuk gyakran, hogy rugalmasabb megoldást adjon akár nehezen algoritmizálható problémáinkra is. Azonban ezek működése nehezen nevezhető zártnak és kauzálisnak. Ezért elengedhetetlen az interpretálhatóság, vagyis annak képessége, hogy a modellek döntéseit érthető és ésszerű módon magyarázza meg. Különböző magyarázó módszerek léteznek a mély tanulási modellek interpretálhatóságának növelésére.

2.1. Magyarázhatóság

A gépi tanulásban használt magyarázó módszerek két fő kategóriára oszthatók: modellfüggő és modellfüggetlen. A modellfüggő magyarázó módszerek közvetlenül figyelembe veszik a modellek belső szerkezetét és működését a magyarázatok létrehozásában. Ezek a módszerek arra törekednek, hogy feltárrák a modell döntéshozatali folyamatainak mechanizmusait és az egyes predikciók alapját. Például a gradiens visszaszámítás, osztály-aktivációs térképek(??) és a LIME modellfüggő magyarázó módszerek.

Azonban a modellfüggetlen magyarázó módszerek általánosabb megközelítést alkalmaznak az interpretálhatóságra. Ezek a módszerek nem igénylik a modell belső szerkezetének ismeretét a magyarázat létrehozásához, és általában a bemeneti adatok és a modell predikciói közötti kapcsolatokat vizsgálják. Például a perturbációs alapú módszerek, mint például a SHAP és a LIME együttes használata.

KÉRDÉS: Miért is ilyen fontos a modell interpretálhatósága és a magyarázhatósága?

Válasz:

. Különösen fontos a jogi szabályozásban, hogy bármilyen szoftver-termék használatakor a döntések átláthatóak és érthetők legyenek. Például az EU! (EU!) által 2016-ban életbe léptetett Általános Adatvédelmi Rendelet (**GDPR!** (**GDPR!**)) előírja, hogy az automatizált döntéshozatalnak átláthatónak kell lennie, és az érintetteknek joguk van tudni, hogy egy algoritmus milyen döntéseket hoz róluk.

3. YOLOv8: Szemantikus szegmentációs háló és működése



1. ábra. Ultralytics logo

A YOLOv8 ([6]) egy hatékony és népszerű osztályozó és detektáló neurális hálózatcsalád legújabb példánya, amelyet számos számítógépes látás (**CV!**) feladatban használnak. A ”You Only Look Once” (YOLO) megközelítést alkalmazza, amely gyors és pontos objektumdetektálást tesz lehetővé egyetlen neurális háló segítségével.

A YOLOv8 működése során a bemeneti képet egyszer veszi figyelembe, és az objektumok pozícióját és osztályát egyetlen predikcióval határozza meg. Ez a modell különösen alkalmas valós idejű alkalmazásokhoz, mint például az önvezető járművek vagy a valós idejű videoelemzés.

Ennek a hálónak én a szegmentációs változatát használtam, amely manapság egy elég új és népszerű irány a közlekedési objektumok detektálásában. Eddig ugyanis inkább 2–3 dimenziós ún. ”bounding boxokat” azaz kereteket használtak a detektálni kívánt objektumok reprezentációjára, azonban a szegmentációs hálók képesek ezeknek pontosabb azonosítására és lokalizációra. Az információ-gazdagabb adatreprézentáció miatt, az objektumok pontos körvonalainak meghatározása lehetővé válik. Ezáltal a szegmentációs hálók pontosabb és részletesebb információkat nyújtanak az objektumokról, mint a keretes objektum-interpretáció.

3.1. Yolov8 architektúra

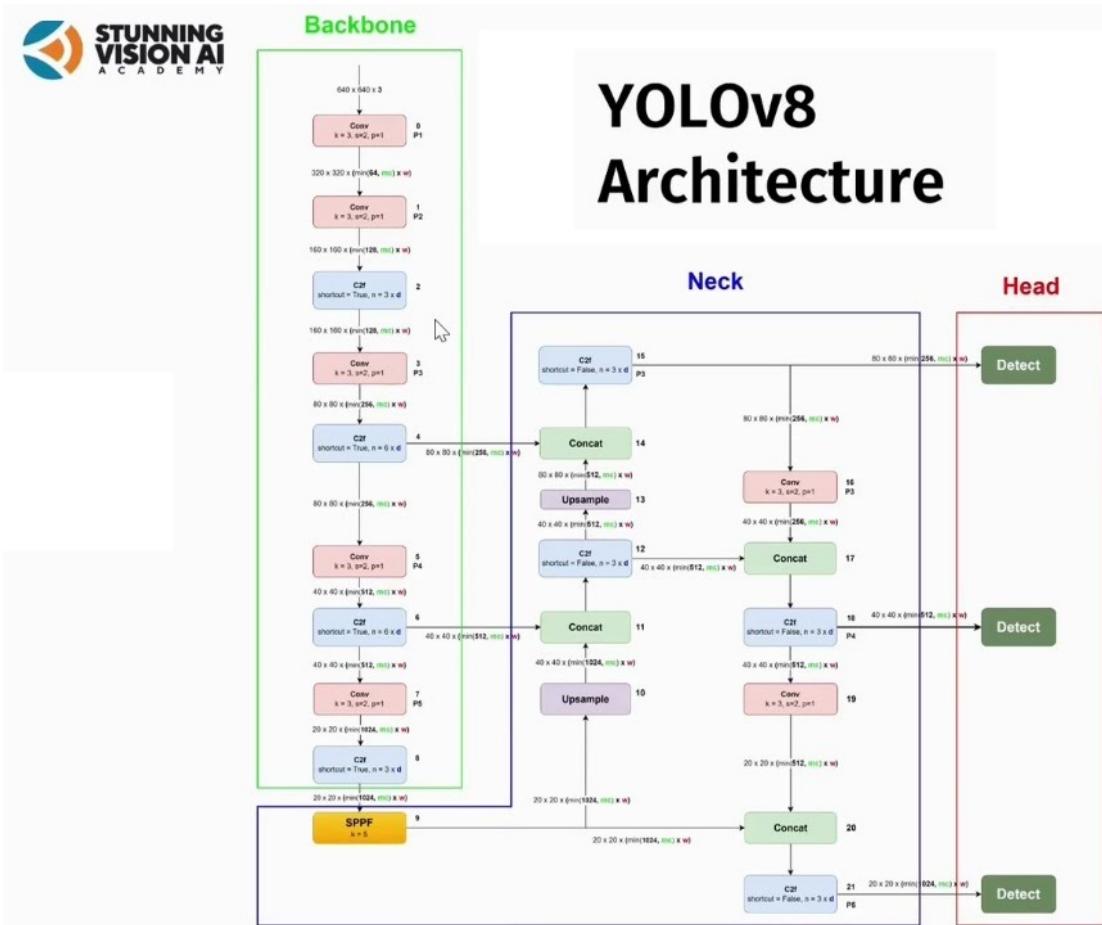
A YOLOv8 egy képfeldolgozó (konvolúciós) mély neurális hálózat. Ami három különböző réteget használ a hálóarchitektúrában a képek feldolgozásához:

- konvolúciós rétegeket
- teljesen összekapcsolt rétegeket
- ”MaxPooling” rétegeket,

A háló bemenetién a képet konvolúciós és ”MaxPooling” rétegeken keresztül feldolgozza, majd a kimeneti rétegekben meghatározza az objektumok pozícióját és osztályát, mindezt egy iteráció alatt, innen is jön a ”You Only Look Once” elnevezés. A YOLOv8 architektúra ?? általában három részre osztható:

1. Az előfeldolgozó rétegek, amelyek a bemeneti képet előkészítik a további feldolgozásra.

2. A konvolúciós rétegek, amelyek a képet feldolgozzák és az objektumokat azonosítják.
3. A kimeneti rétegek, amelyek meghatározzák az objektumok pozícióját és osztályát.



A YOLOv8 architektúra eltérő méretű és kapacitású változatokban érhető el, amelyek különböző feladatokhoz és alkalmazásokhoz használhatóak: (Attribútumok COCO adathalmaz alapján) Ezenfelül beszélhetünk szegmentációs hálókról (bővebben: a ?? alfejezetben) és hagyományosan a bounding boxokat detektáló hálókról, amik az objektumok köré illesztenek egy befogó minimális területű téglalapot/téglatestet.(Yolov8_seg vs. Yolov8)

Témalaboratórium alatt bounding box adatrepräsentációjú neurális hálót használtam, most viszont, áttértem a szemantikus szegmentációra. Ennek okait a következő alfejezetben felsorolom.

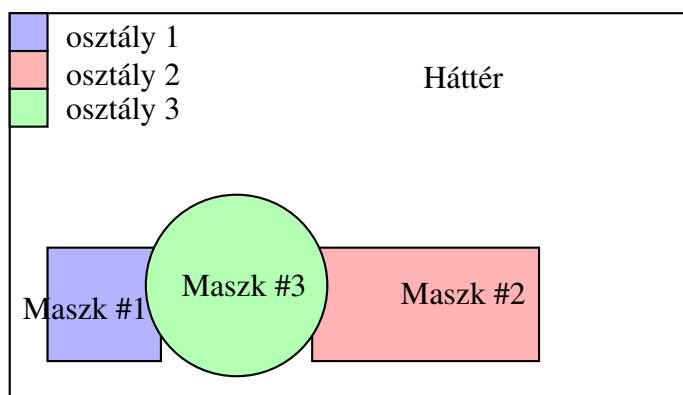
YOLOv8 verzió	Méret	Max. Pontosság (mAP!)	Sebesség (ms!)
YOLOv8s	Kis	37.3	0.99
YOLOv8m	Közepes	44.9	1.2
YOLOv8l	Nagy	50.2	2.39
YOLOv8X	Hatalmas	53.9	3.53

1. táblázat. Yolov8 modell méretei

3.2. Szemantikus szegmentáció

A szemantikus szegmentáció egy olyan adatreprézentációs forma, amelyben a cél az objektumokat tartalmazó kép egyes részeinek (például pixeljeinek) címkézése az objektumokhoz tartozó osztályok szerint. Van emellett egy másik szegmentációs adatreprézentációs forma, az ún. egyed szegmentációs szemantikus szegmentáció] Egyedszegmentáció, amelyben minden objektumot külön, azaz egyedenként kell detektálni és akár számon tartani (indexelni), míg a szemantikus szegmentációban csak az objektumok osztályait kell meghatározni az egyed megkülönböztetése nélkül. Más szavakkal, minden képpontot hozzá kell rendelni egy osztályhoz vagy kategóriához, például autó, biciklis, gyalogos stb. Szakmai gyakorlatom során eddig csak szemantikus szegmentációval foglalkoztam. Amely módszer lehetővé teszi a rendszereknek, hogy pontosan azonosításuk és lokalizálják az objektumokat egy adott képen.

Tehát a szemantikus szegmentációt leírhatjuk úgy, ha egy képet jelölünk I -vel, akkor a szemantikus szegmentáció pedig egy olyan függvény, $F : I \rightarrow L$, ahol L a lehetséges osztályok halmaza, és F minden képpontot hozzárendel egy osztályhoz, ez az osztály-hozzárendelés lesz a maszk. A szemantikus szegmentáció kiemelt fontosságú az önvezető autók, a videoelemzés, a térképek építése és sok más alkalmazásban, ahol pontos és részletes objektum-felismerésre van szükség. Tehát az adatreprézentációs



2. ábra. szegmentáció példa

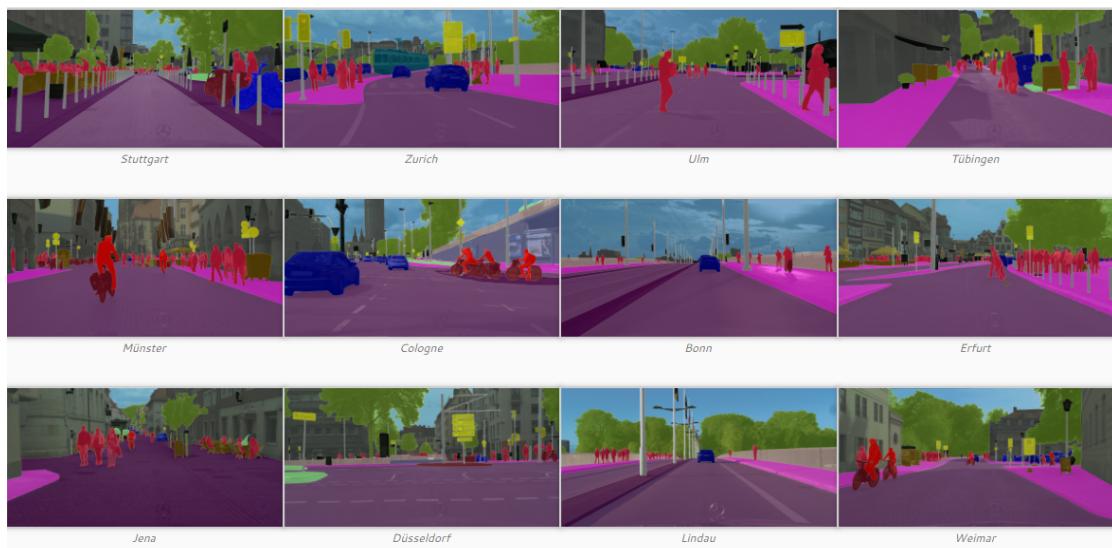
módszerváltást azért tartottam fontosnak, mert:

1. pontossága jóval felülmúlja az előző módszerét.
2. az adathalmaz alapvetően szemantikus szegmentálási módszerrel van annotálva, kevés adat-előkészítési munkálat szükséges.
3. könnyebben érthető a feldolgozott kép, nincs olyan hogy az objektumok túlzott sűrűsége miatt túl sok doboz keletkezik ami megnehezíti vagy akár ellehetetlenítheti a kiolvasást.

3.3. Adathalmazok és kísérletek

Adathalmazként a témalaboratórium alatt végzett munkámhoz hasonlóan a CityScapes (<https://www.cityscapes-dataset.com>) szemantikusan szegmentált adathalmazt, annak is a pontosan annotált (Fine Annotated), a ??-képen bemutatott, adathalmazát használtam, ami osztályszegmentációs maszkokat biztosít az elérhető képekhez. Ezek a képek kicsit több mint 5000 városi közlekedési szituációt ábrázolnak németországi városokból.

Ez az adathalmaz elég nagy varianciával rendelkezik számunkra ahhoz, hogy az adathalmazból egy robusztus szegmentációs hálót tudjunk tanítani közlekedési objektumok detektálására és osztályozására.



3. ábra. CityScapes Examples

A tanításhoz és validációhoz a maszkokat poligonok formájában kaptam meg szöveges (.yaml) formában, ezeket a poligonokat minimális átalakítás után be is tudtam

adni a háló bemenetére a megfelelő képekhez rendelve (szegmentációs maszkokról: ??-oldalon értekezek.) A kísérletek: ??-táblázatban találhatóak.

A magyarázat a kísérletekhez a ??-oldalon található.

Ezeket az adatokat használom a későbbiekben arra, hogy megpróbáljuk megérteni a hálónkat az aktivációs függvényeinek kimenetei alapján. Ezeket a teszteket amiket az EigenCAM-mal végeztem, a németországi Bonn városában vették fel és mivel a teszt-halmaz része, így a háló tanításban és a validációban eddig nem szerepelt.

3.4. MLOps

A háló tanításához és validálásához használtam egy online kiértékelő, és eredmény-összesítő felületet, ez pedig a Comet.ml online platform ("alternatívája a Weights & Biases"-nek), amely egy Github fiókhoz rendelve, segít a **ML! (ML!)** projektek adminisztrációjában. A Comet.ml (<https://www.comet.ml>) egy kollaboratív **MLOps!** (**MLOps!**) platform, amely lehetővé teszi, hogy könnyen és hatékonyan figyelhessük a futó tanításainkat, validációinkat és Inferenceinket. Számos funkciót kínál:

- Rögzíti és követi a kísérletek metrikáit, hiperparamétereit, modellváltozatokat és naplófájljait.
- Különböző diagramok és grafikonok segítségével jeleníti meg és elemzi az adatokat és az eredményeket.
- Lehetőséget ad a különböző kísérletek összehasonlítására és megosztására, akár csapatmunkára is használható.
- Automatikusan rögzíti és követi a modell változásait és fejlődését a tanítás folyamán az idő műlásával.
- Integrálható más keretrendszerrel és eszközökkel, és rendelkezik egy API-val is, amely lehetővé teszi a platform saját igényeiknek megfelelő testreszabását.

Összességében a Comet.ml egy teljes körű platform, amely segíti a **ML!** hatékony kezelését és nyomon követését, valamint lehetővé teszi a felhasználók számára, hogy gyorsabban és hatékonyabban dolgozzanak valamint jobban menedzseljék kísérleteiket.

4. EigenCAM: Modellfüggő magyarázó módszer

Az EigenCAM (Eigen Class Activation Mapping) egy modellfüggő magyarázó módszer (forrás [1]):, amelyet a mély tanulási modellek interpretálhatóságának növelésére fejlesztettek ki. Célja, hogy vizualizálja és magyarázza meg a modellek döntéseit a bemeneti adatok alapján, osztály-diszkrimináció nélkül, tehát ebben az esetben nem lesznek külön osztályokra szedve az aktivációk, hanem az összes osztály átlagos aktivációját kapjuk meg.

Az EigenCAM működése során az algoritmus végiglépked a háló kiválasztott rétegein és kiértékeli az aktivációs függvények mátrixát. Ezt a mátrixot kiterjeszti (ha kell) annak a képnek méretére amire éppen az Inferenceet futtatjuk. (forrás [3]:) Az ez által generált "hőterkép" (Heatmap) már jó esetben lehetővé teszi azt, hogy közelebb kerüljünk annak megértéséhez, hogy a kép mely tartományai játszanak nagyobb szerepet a detekcióban és klasszifikációban.

Azonban fontos tudni, hogy az EigenCAM csak egy interpretálhatósági eszköz, és nem biztosít teljes képet a modell működéséről. Ugyanis ezek az eszközök csak a hálónak egy szeletébe engednek betekintést nyerni, amik önmagukban nehezen értelmezhetőek. Sokszor ez nem is enged logikus következtetést levonni.

Az aktivációs csoportok képenként újrarendeződnek így metszeteket is nehéz automatizáltan készíteni. Ez pedig elengedhetetlen annak érdekében hogy biztosra mondjuk egy következtetés magyarázását.

4.1. Aktivációk

Az aktivációs függvények a modell rétegeinek kimeneti függvényei ahol "x" az input, "W" a súlymátrix és "b" a "bias" vagyis eltolási vektor. Az utolsó rétegekben egy lineáris aktivációs(**ReLU!** (**ReLU!**)) (??) függvényt használunk, míg az összes többi rétegen egy szivárgó rektifikált lineáris aktivációs függvényt (**LReLU!** (**LReLU!**)) (??) használunk:

$$\text{LReLU:} \varphi(x) = \begin{cases} x, & \text{ha } x > 0 \\ 0.1x, & \text{különben} \end{cases} \quad (1)$$

$$\text{ReLU:} \varphi(x) = \begin{cases} x, & \text{ha } x > 0 \\ 0, & \text{különben} \end{cases} \quad (2)$$

$$\text{Sigmoid:} \varPhi(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

5. Az EigenCAM alkalmazása a YOLOv8-ra

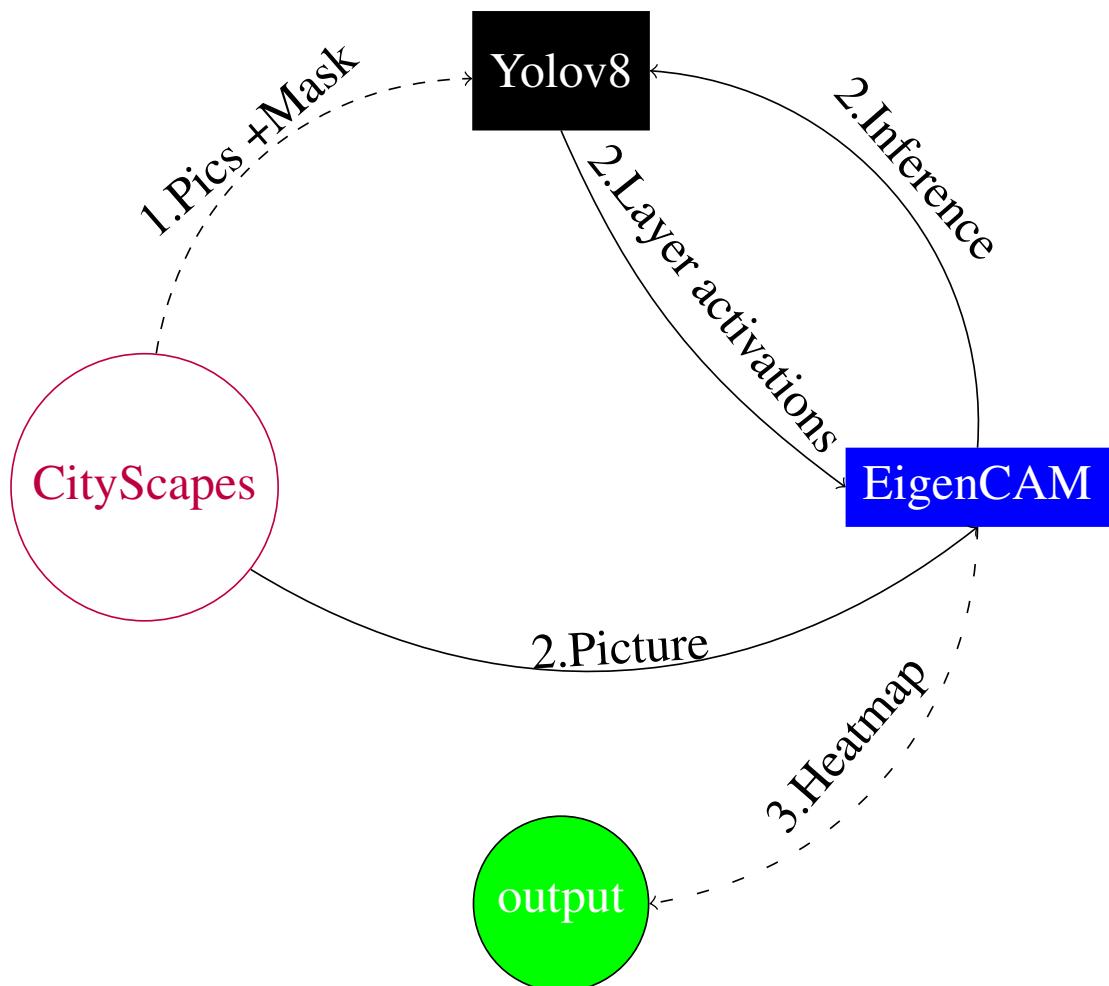
Az EigenCAM alkalmazása a YOLOv8 szemantikus szegmentációs hálóra lehetővé teszi számunkra, hogy megértsük miképpen azonosít és lokalizál objektumokat képeken és videókon a modell. ([5] alapján).

Az EigenCAM konkrét működési folyamata a ?? ábrán látható. A CityScapes adathalmazon tanítottuk a Yolov8m-seg hálót (ami egy közepes méretű Yolov8 szegmentáló háló), majd az EigenCAM segítségével vizualizáltuk az aktivációkat, amelyeket a háló a képek feldolgozása közben produkál (ezt az EigenCAM adja a hálónak egy olyan folyamatban keresztül amit Inferencenek nevezünk). Ezeket az EigenCAM kivezeti és rávetíti a bemenő képre (ezt a folyamatot lásd a ?? képen ésa ?? pontban), a különböző rétegek aktivációit külön-külön. Amikor végez a kép előállításával azt hőterkép formájában az output mappájába helyezi. Ezeket a hőterképeket majd ?? oldalon látunk.

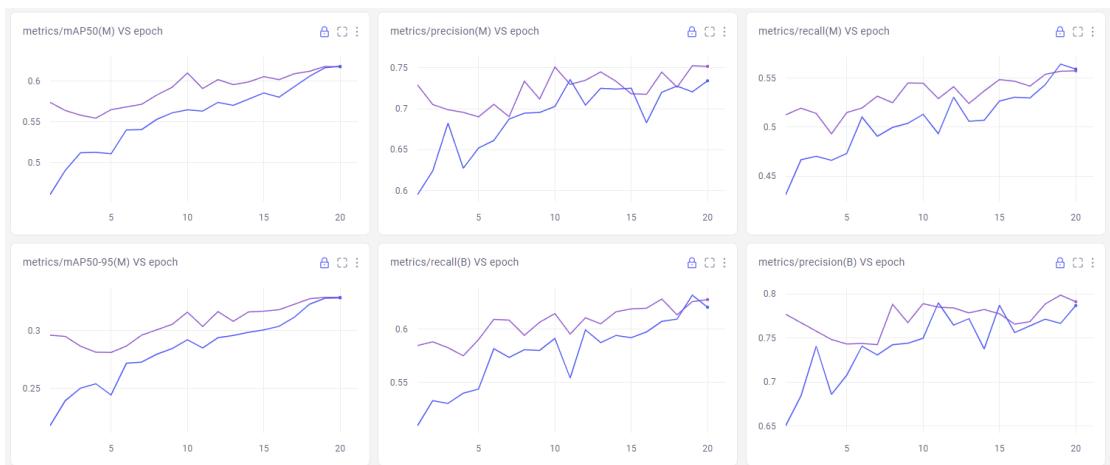
- Normál Szaggatott: Tanítási időben.
- Teli nyíl: Inference (EigenCAM használata) közben.
- Laza szaggatott nyíl: Inference idő után.

5.1. Implementáció lépései

1. Github és Comet repository létrehozása és felkonfigurálása
2. Tanító, validációs és detektáló állományok megírása
3. Adatfeldolgozás, és adatbázis-menedzsment
4. Tanítás és validáció
5. Modellanalízis
6. Modellmagyarázó módszer bevetése
7. Modellmagyarázó módszer kimenetének értékelés és magyarázata



4. ábra. Munkafolyamat

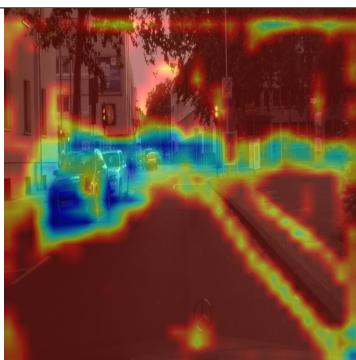
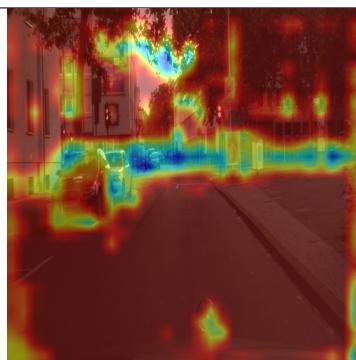
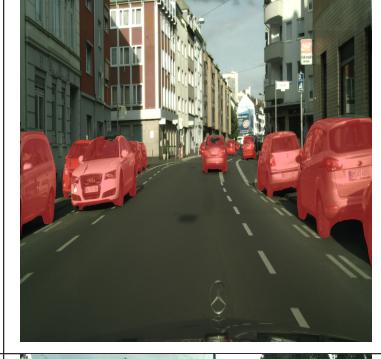
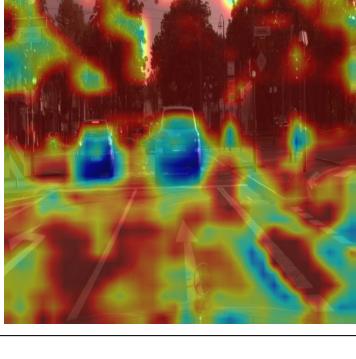
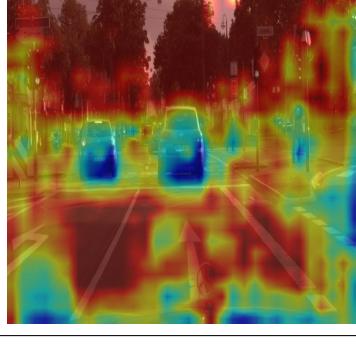


5. ábra. A tanítás folyamata

6. Magyarázhatosági eredmények és értékelés

A két hálót, legyen az egyik ”old” a másik ”new”, ”old” egy yolov8m-seg háló 20 epoch keresztől tanult a ugyanazon teljes adathalmazon, a másik pedig 40 epoch keresztül tanítottam.

Az EigenCAM által nyújtott kimenet amit ábrázoltam a ?? táblázatban. A két háló detekciója alig különbözik. Kizárolag abban különböznek, hogy milyen bizonyosságban képesek megmondani átlagosan az objektumok osztályát. Ezért elegendőnek találtam a következőkben csak az ”new” háló eredményeit tárgyalni.

Layer	”old”	”new”	detection
-4.			
-3.			
-2.			

2. táblázat. Réteg aktivációk összehasonlítása a két háló között.

6.1. Magyarázat

Nem megszokott módon, a hőterképeken az alacsony aktivációs értékeket vöröses árnyalatokkal míg a magas aktivációs értékeket kékes árnyalatokkal jelölik.

Layer	Magyarázat
-4	Ez a réteg egyetlen konvolúciós réteget tartalmaz. Ami nagyrészt eldetektícióval foglalkozik, így az aktiváció nem konzisztenčen jelzi a detektálni kívánt osztályokat. Inkább elekre aktiválódik amelyek az osztálydetekcióban hasznosak lehetnek ezeket featureökké fogja össze és ezeket a featureöket továbbítja a következő hálónak. Az ennél előrébb lévő rétegek, túl kis méretű featureökkel foglalkoznak és ezek közül sokkal kevesebb lesz használva, viszont a sok aktiváció miatt emberi szemmel keveset érhetünk belőle. Túl keveset ahhoz, hogy a magyarázhatóság vizsgálatába belevegyem, hiszem emberi szem által nehezen olvasható és érthető.
-3	Ez egy konkatenációs réteg itt a rétegek előzőleg különböző méretű bemeneteket dolgoznak fel. Ez a réteg valószínűleg az összes feldolgozott adatot egyetlen tensorba egyesíti, hogy aztán további feldolgozásra kerülhessen. Persze az, hogy hol történik feature detekció (hol nagyobb az aktiváció) az jól látszik a képeken. Ezek a fontos összefűzött feature-k melyek a következő rétegek is fontosak lehetnek.
-2	Ez a réteg konvolúciós rétegeket (cv1 és cv2) tartalmaz, amelyeket egy ModuleList (m) követ, amely két Bottleneck blokkot tartalmaz. Ezek a blokkok az adatok dimenzióinak csökkentésére és a reprezentációk összeállítására szolgálnak, lehetőséget adva az egyszerűbb és hatékonyabb feldolgozásra az utolsó réteg számára. Az utolsó réteg kimenetei maguk az osztályaktivációk, ezek egyszer sem produkáltak olyan hőterképeket amit érdemes lett volna vizsgálni, a kijövő kép "horizontális vonalkód" szerű, így azt nem vettettem figyelembe a magyarázhatósági vizsgálatba.

3. táblázat. Magyarázat

Ezen felül az EigenCAM által nyújtott magyarázatokat összehasonlítottam a két hálóra vonatkozóan. Az "old" háló kimenetén jól láthatjuk a gyengébb aktivációkat, míg "new" háló centralizált aktivációkat mutat ott, ahol ténylegesen fel kell ismernie az objektumokat. Ezen felül még egyértelműen látható, hogy a tanulással arányosan csökken az aktivációk értéke olyan helyeken, ahol csak valamiféle tetszőleges él található mint például a -4. Layer estében. Tehát egyértelműen kijelenthetjük hogy a háló tanulását kitudom mutatni kizárálag csak az aktivációk alapján, az EigenCAM segítségével.

7. Az EigenCAM alkalmazásának gyakorlati haszna

Az EigenCAM és hasonló magyarázó rendszerek alkalmazása esélyt adhatnak nekünk arra, hogy esetekre , tehát egyéni bemenetekre (a képfelismerés esetében ezek a képek), vonatkozó döntéseihez a kép minden részei milyen fontossággal asszisztáltak. Lehet értelmezni a háló architektúrájának tudatában, hogy éppen melyik réteg minden információt dolgoz fel és ezekből milyen hiedelmeket (értelmezett információkat) alkot. Érdekes az is, hogy mint ahogy ???. oldalon láthattuk a háló tanulását is meg tudjuk figyelni az aktivációk alapján és ebből következtetéseket tudunk levonni arra vonatkozóan, hogy mik alapján észlelik az objektumokat, kiküszöbölv olyan hibákat, amelyeket a korai mesterséges intelligencia alapú CV! szoftverek tartalmazhattak, például a háttér túlzott befolyása az objektum osztályának meghatározásában.

7.1. Eredmények összegzése

Féléves munkámban tehát a EigenCAM modellmagyarázó módszert és a YOLOv8 szemantikus szegmentációs háló interpretálhatóságát vizsgáltam valamint feladatom volt az adat előkészítése és elemzése is.

Az EigenCAM segítségével vizualizáltam az aktivációkat, és kísérletet tettem a hálók működésének megértésére. Ezáltal figyelhettem meg a háló tanulását: mit jelenthet adott esetben a túltanulás fogalma és azt, miben is különbözik egy háló aminek jobb az "Accuracy" és a "Recall" mutatója is, egy másik ugyanazon az adaton és osztályokra tanított, ugyanolyan architektúrájú hálótól.

Ezen felül foglalkoztam az új architektúrával, Yolov5-ről Yolov8-ra tértem át, és először használtam felhő alapú MLOps megoldásokat munkám segítésére, mint például a Comet (<https://www.comet.ml>) amelyben a ?? fejezetben részleteztem.

8. Jövőbeli irányok és kutatási lehetőségek

A YOLOv8 neurális háló most is vezető szerepet élvez az autóipari szolgáltatásokban képfelismerés téma körében. Azon belül a szemantikus szegmentációban, ami az elmúlt években előnyt élvez a hagyományos két- és háromdimenziós keretekkel szemben, a kereteknél jóval nagyobb pontossága és a maszkok által hordozott temérdek információ gazdagsága miatt.

A következőkben tehát a jelenlegi munkámban még a kezdetlegesebb, kiforratalabb módszert, a modellmagyarázást (vagyis Explainable AI-t **XAI!** (**XAI!**)) szeretném továbbfejleszteni. Mivel a jelenlegi megoldásom modellfüggő és Rétegaktiváció-alapú, nem vesz figyelembe gradienseket, valamint érzékeny a modell belső felépítésére is. Célom a jövőben a modellek tanulásának folyamatát figyelni más módszerekkel is, amelyek például a rétegek gradienseit figyelembe veszik, azért hogy a magyarázataik pontosabbak, rendszerszerűbbek és megbízhatóak legyenek. A következő konkrét tervet határoztam meg további munkámhoz:

1. Lime vagy Shap módszerek behozása és összehasonlítása az EigenCAM-mal. [4, 2]
2. EigenGradCAM bevezetése a modell tanulásának folyamatának figyelembevétele.

Tárgymutató

szegmentáció

szemantikus szegmentáció, 27

CityScapes, 8, 19, 22

Comet, 20

EigenCAM, 19, 21, 22, 24

szegmentáció

egyedszegmentáció, 18

szemantikus szegmentáció, 18

Yolo, 11

Yolov5

Yolo, 7

Yolov8

Yolo, 13

9. Szó- és rövidítés jegyzék

Szójegyzék

Eigen Class Activation Mapping Egy modellfüggő magyarázó rendszer, amely képes vizualizálni, hogy a mély tanuló hálózatok mely részei járultak hozzá a döntéshozatalhoz. 13

epoch Az Epochok száma adja meg hányszor megy végig a tanulás során a hűlő a teljes adathalmazon. 24

feature A feature egy adott kép vagy adat egyik jellemzője, amelyet a modell felhasznál a döntéshozatalhoz. 25

Inference Az inferencia a gépi tanulásban a modell által megtanult minták alkalmazását jelenti új adatokon. 20–22

You Only Look Once You Only Look Once, egy gyors és hatékony objektumdetektáló neurális hálózat. 15

You Only Look Once version 8 A YOLOv8 egy hatékony és népszerű szemantikus szegmentációs hálózatcsalád legújjabb példánya. 13

9.1. Rövidítésjegyzék

EU Európai Unió	4
GDPR Általános Adatvédelmi Rendelet	4
CV Computer Vision	3

Hivatkozások

- [1] jacobgil and Pytorch Contributors. GRADCAM implementation. <https://github.com/jacobgil/pytorch-grad-cam>, 2024.
- [2] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.
- [3] M. Bilal Muhammad and Mohammed Yeasin. Eigen-cam: Class activation map using principal components. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, July 2020.
- [4] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. „Why Should I Trust You?“: Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California, June 2016. Association for Computational Linguistics.
- [5] rigvedrs. Yolov8-CAM implementation. <https://github.com/rigvedrs/YOLO-V8-CAM>, 2024.
- [6] Ultralytics. You Only See Once Project at Ultralytics. <https://github.com/ultralytics/yolov5>, 2024.

Formai elem	Megvalósítás
irodalomjegyzék	itt
tartalomjegyzék	itt
rövidítésjegyzék	itt
indexjegyzék	itt
táblázat	itt és itt
hivatkozás táblázatra	itt
vektor-grafikus kép	itt
raszter-grafikus kép	itt
hivatkozás képre	itt
tikz ábra	itt
hivatkozás ábrára	itt
képlet	itt
hivatkozás képletre	itt
képletcsoport	itt
hivatkozás képletcsoport egy képletére	itt
fejezet	itt
hivatkozás fejezetre	itt
lista	itt
hivatkozás lista elemre	itt
hivatkozás oldalszámra	itt
hivatkozás irodalomra	itt
saját makró használata	itt