Detekcija i klasifikacija otpada na slici s kamere

Ogrizek, Tin

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1672

DETEKCIJA I KLASIFIKACIJA OTPADA NA SLICI S KAMERE

Tin Ogrizek

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1672

DETEKCIJA I KLASIFIKACIJA OTPADA NA SLICI S KAMERE

Tin Ogrizek

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1672

Pristupnik:	**Tin Ogrizek (0036543600)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Tamara Petrović

Zadatak: **Detekcija i klasifikacija otpada na slici s kamere**

Opis zadatka:

Cilj je ovog završnog rada implementacija algoritma za detekciju i klasifikaciju otpada na slici s kamere te analiza točnosti. Implementirani algoritam može se primijeniti u automatiziranim sortirnicama otpada. Algoritam koji je potrebno proučiti te koristiti je YOLOv8 koji se temeljeni na primjeni konvolucijskih neuronskih mreža. Potrebno je proučiti dostupne skupove podataka za problem detekcije i klasifikacije otpada, kao što je, primjerice TACO skup podataka. Analizirati točnost algoritma na odabranom skupu podataka te na vlastitom skupu podataka. Razmotriti primjenu drugih vrsta kamera, kao što su multispektralne kamere, za detekciju otpada.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem mentorici izv. prof. dr. sc. Tamari Petrović na savjetima i pomoći prilikom pisanja ovog rada.

Sadržaj

Uvod

Računalni vid je grana umjetne inteligencije s glavnim ciljem prepoznavanja predmeta na digitalnim slikama. Da bi se predmet s određenom točnošću mogao prepoznati, on se najprije mora ispravno detektirati te potom klasificirati. Metode detekcije i klasifikacije primjenjuju se u područjima poput autonomnih vozila, sigurnosnih sustava, medicinske dijagnostike, robotike i automatizacije [1] i sl.

Detekcija i klasifikacija predmeta na slikama bazira se na funkcionalnosti neuronskih mreža, posebno konvolucijskih neuronskih mreža (eng. *Convolutional Neural Network - CNN*), koje su dizajnirane za učinkovitu analizu vizualnih podataka. Danas je moguće putem interneta preuzeti već unaprijed naučene modele te ih dodatno trenirati prema specifičnim potrebama. Jednako tako postoje usluge za pronalazak velike količine podataka za treniranje modela te njihovo treniranje u oblaku kao što su Google Colab [2] i Kaggle [3]. Svi navedeni faktori omogućuju veliku dostupnost i olakšavaju rad sa strojnim učenjem, a samim time i računalnim vidom.

Klasifikacija otpada predstavlja jedan od ključnih izazova u upravljanju otpadom i očuvanju okoliša. Rastom ljudske populacije, povećava se i stvaranje otpada te njegovo odlaganje u prirodi. Tradicionalni postupci ručne klasifikacije otpada zahtijevaju značajan ljudski napor i vrijeme. Računalni vid, odnosno detekcija i klasifikacija otpada na digitalnim slikama mogu značajno poboljšati problem zbrinjavanja otpada u prirodi.

Ovaj rad se bavi pronalaskom zadovoljavajućeg skupa podataka kojim bi se mogao istrenirati model za detekciju i klasifikaciju otpada na slikama te analizom dobivenog modela. U budućnosti je moguća njegova moguća primjena na robotu zaduženom za razvrstavanje otpada.

Rad je strukturiran na sljedeći način. Prvo poglavlje ovoga rada objašnjava osnovne principe rada konvolucijskih neuronskih mreža, prikazuje najpopularnije algoritme za detekciju objekata te objašnjava postupak validacije istreniranih modela. Drugo poglavlje predstavlja skup podataka TACO [4] koji sadrži skup slika otpada u raznim okolnostima te će kao takav biti korišten za treniranje modela u ovome radu. Treće poglavlje implementaciju algoritma za detekciju i klasifikaciju otpada napravljenog u ovom radu, uz opisane postupke treniranja,

validacije te testiranja algoritma. U četvrtom poglavlju predstavljaju se eksperimentalni rezultati te se provodi detaljna statistička analiza rada modela.

1. Detekcija objekata na slici

Postupak detekcije objekata osjetilom vida evolucijski je uznapredovao do razine da čovjek to čini praktički automatski i bez razmišljanja. S druge strane, za računala ovo nije jednostavan proces. Da bi se računala približila ljudima u sposobnosti detekcije objekata, razvijeni su složeni algoritmi koji se posljednjih godina najviše temelje na primjeni strojnog učenja.

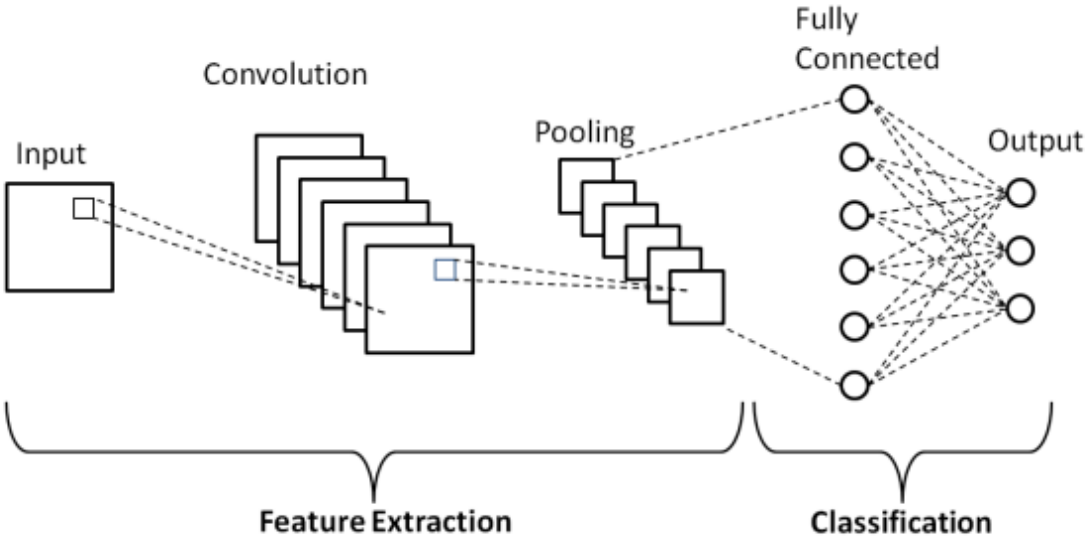
Cilj detekcije objekata jest prepoznavanje jednog ili više objekata koji pripadaju unaprijed određenim klasama, kao što su ljudi, auti, lica i sl. Kako bi se označilo da je neki objekt prepoznat, koriste se pravokutni granični okviri u kojima se nalazi prepoznati objekt. Osim oznake da je neki objekt prepoznat, granični okviri sadrže informacije o točnim položajima objekata unutar same slike. Te informacije su najčešće x i y koordinate koje označavaju središte graničnog okvira te širina i visina samoga okvira ili koordinate gornjeg lijevog i donjeg desnog vrha okvira [5]. Uz granični okvir, detekcija uključuje i klasifikaciju objekta te mjeru pouzdanosti.

1.1. Konvolucijske neuronske mreže

Jedna od najznačajnijih metoda za detekciju objekata su konvolucijske neuronske mreže (eng. *Convolutional Neural Network - CNN*). Potreba za konvolucijskim neuronskim mrežama proizlazi iz problema preopterećenosti parametrima prilikom rada s klasičnim neuronskim mrežama. Za primjer može se uzeti slika veličine 256×256 piksela s 3 kanala za boju (RGB). Tada u klasičnoj neuronskoj mreži postoji 256×256×3 = 196,608 parametara i to samo između ulaznog i prvog skrivenog sloja neurona. Konvolucijske neuronske mreže najčešće se sastoje od tri glavna dijela: **konvolucijski slojevi**, **slojevi sažimanja** te **potpuno povezani slojevi** [6]. Upravo takva struktura osigurava znatno manji broj parametara koje treba obrađivati. Konvolucijski sloj i sloj sažimanja glavni su faktori smanjenja broja argumenata prilikom uporabe konvolucijskih neuronskih mreža. Oni vrše ekstrakciju značajki iz ulaznog sloja te stvaraju mapu značajki. Potpuno povezani slojevi funkcioniraju jednako kao i standardne neuronske mreže. Oni rade s dobivenim mapama značajki te daju konačni rezultat, odnosno klasifikaciju (Slika 1.1).

Kroz godine razvile su se različite arhitekture konvolucijskih neuronskih mreža. Neke od njih su: LeNet, AlexNet, VGGNet te GoogLeNet. LeNet [7] je jedna od prvih i

najjednostavnijih konvolucijskih neuronskih mreža korištena za prepoznavanje slova i brojeva. Sastoji se od 3 konvolucijska sloja, 2 sloja sažimanja te 2 potpuno povezana sloja. Smatra se osnovom za sve buduće modele. AlexNet [8] se smatra jednom od najutjecajnijih u području računalnog vida. Sastoji se od 5 konvolucijskih i 3 potpuno povezana sloja. Revolucionarna stvar je bila upotreba ReLu aktivacijske funkcije umjesto dotadašnje sigmoida funkcije. Prednost ReLu aktivacijske funkcije jest što ona ublažava problem nestajanja gradijenta prilikom treniranja neuronskih mreža koji je prisutan kod ostalih često korištenih aktivacijskih funkcija. Gradijenti predstavljaju smjer i veličinu promjena koje je



Slika 1.1: Građa jednostavne konvolucijske neuronske mreže koja se sastoji od 5 slojeva. [11]

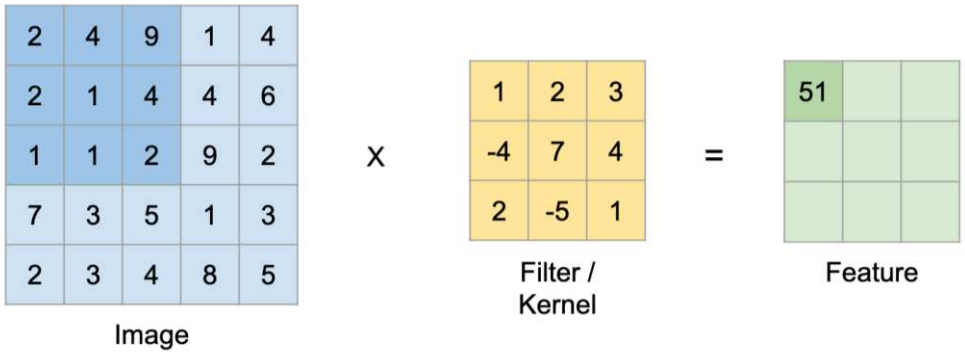
potrebno napraviti na mrežnim parametrima kako bi se smanjile pogreške prilikom učenja. Kada je gradijent jako malen, dolazi do otežavanja učenja neuronske mreže. Prestavljena je 2011. godine te je ubrzo postala najčešće korištena aktivacijska funkcija [9]. VGGNet [10] je povećao dubinu mreže dodavanjem više konvolucijskih slojeva s manjim filtrima. GoogLeNet [11] se sastoji od 22 sloja te je znatno smanjila broj parametara, postotak krive detekcije te vrijeme treniranja u usporedbi s AlexNet i VGGNet mrežama.

1.1.1. Konvolucijski sloj

Kao što mu ime sugerira, konvolucijski sloj ima glavnu ulogu unutar konvolucijske neuronske mreže. On je odgovoran za ekstrakciju parametara iz ulaza. Ta ekstrakcija se odvija kombinacijom operacije konvolucije te aktivacijske funkcije. Konvolucija je vrsta linearne operacije gdje se relativno malena matrica brojeva – filtar, primjenjuje na cijelu ulaznu matricu brojeva – tenzor [6]. Postupno se prolazi po cijelom ulaznom tenzoru na način da se uzimaju dijelovi veličine filtra te se tako dobiveni dijelovi skalarno množe sa

zadanim filtrom i dobivena vrijednost sprema u mapu značajki. Postupak se nastavlja sve dok cijeli ulazni tenzor nije obrađen, a mapa značajke nije ispunjena (Slika 1.2). Također ako postoji više filtara, postupak se provodi za svaki te se dobiva više mapa značajki, tj. dobiva se više različitih karakteristika ulaznih tenzora. Tijekom postupka radimo korake (pomake na ulaznom tenzoru) veličine 1. Ponovno gledajući primjer ulazne slike veličine 256×256 piksela s 3 kanala za boju, ako se koristi filter veličine 6×6, potrebno je 6×6×3 = 108 parametara za svaki neuron unutar konvolucijskog sloja. Ako se taj broj usporedi sa standardnim potpuno povezanim slojem, za koji je potrebno 196,608 parametara, uočava se značajno smanjenje broja parametara. Važna stavka ovog postupka je također proširivanje ulaznog tenzora s redcima i stupcima ispunjenih nulama. Razlog je taj što svaka dobivena mapa značajki ima dimenzije manje nego što je imao ulazni tenzor te se s vremenom može izgubiti točnost informacije o rubnim dijelovima slike. Proširivanje ulaznog tenzora nulama osigurava da se i rubove slike gleda jednako kao središnje dijelove slike [12]. Izlazi linearne operacije kao što je konvolucija šalju se kroz nelinearnu aktivacijsku funkciju kako bi se osigurala nelinearnost koja je ključna prilikom rada s kompleksnijim ulaznim podacima. Kada se to ne bi činilo, cijela neuronska mreža bi bila linearna, što bi drastično ograničilo njene mogućnosti modeliranja. Kroz povijest kao aktivacijska funkcija se najviše koristila sigmoidalna funkcija, jer ona matematički predstavlja biološko ponašanje neurona [6], no dolazilo je do grešaka u dubljim mrežama te je standard postala ReLU aktivacijska funkcija.

Slika 1.2: Dobivanje jedne mape značajki skalarnim umnoškom odabranog dijela ulaznog



tenzora i filtra. [13]

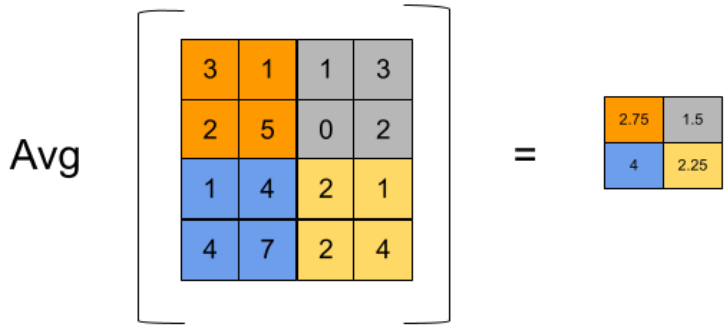
1.1.2. Sloj sažimanja (Pooling sloj)

Uloga sloja sažimanja je smanjivanje veličine mapa značajki dobivenih unutar konvolucijskog sloja, što pomaže u smanjenju broja parametara u mreži te ubrzava

procesuiranje. Sloj sažimanja uzima male lokalne značajke i čini ih invarijantnim na male translacijske pomake u ulaznom području. Najčešće korištene verzije sloja sažimanja su: sažimanje srednjom vrijednošću te sažimanje maksimalnom vrijednošću.

Sažimanje srednjom vrijednošću

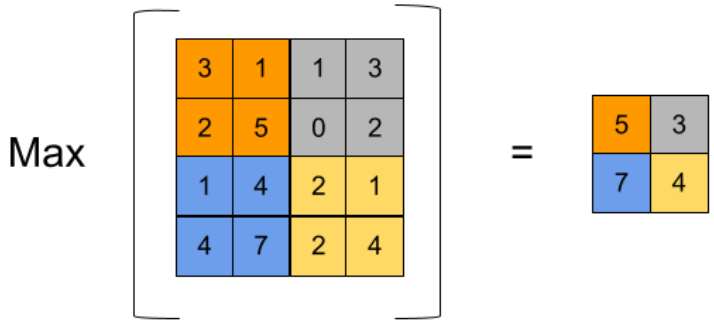
Prilikom sažimanje srednjom vrijednošću, uzimamo dio po dio matrice koje čine mapu značajki te za svaku odabranu matricu računamo srednju vrijednost. Svaku izračunata srednja vrijednost se sprema u sažetu mapu značajki (Slika 1.3)



Slika 1.3: Mapa značajki veličine 4×4 sažimanjem srednjom vrijednošću prelazi u mapu značajki 2×2. [33]

Sažimanje maksimalnom vrijednošću

Postupak je jednak kao i kod sažimanja srednjom vrijednošću, samo što umjesto srednje vrijednosti za svaku novu vrijednost reduciranje mape značajki uzimamo maksimalnu vrijednost iz svakoga dijela mape značajki (Slika 1.4).



Slika 1.4: Mapa značajki veličine 4×4 sažimanjem maksimalnom vrijednošću prelazi u mapu značajki 2×2. [33]

1.1.3. Potpuno povezani slojevi

Nakon što su prošle posljednji konvolucijski sloj ili sloj sažimanja, mape značajki se pretvaraju u jednodimenzionalnu (1D) listu brojeva, odnosno vektor. Takav vektor dolazi na

ulaz u potpuno povezani sloj. Ovaj sloj ima građu poput standardne neuronske mreže gdje je svaki neuron povezan sa svakim iz prijašnjeg sloja. Odvija se unaprijedni prolaz svih slojeva te se dolazi do zadnjeg sloja koji ima broj izlaza jednak broju klasa u pojedinom klasifikacijskom zadatku. Između svakog sloja unutar potpuno povezanog sloja, osim neposredno prije izlaznog sloja, primjenjuje se nelinearna aktivacijska funkcija.

1.2. Algoritmi za detekciju objekata

Razvojem konvolucijskih neuronskih mreža, razvili su se i razni algoritmi za detekciju i prepoznavanje objekata na slikama. Većina najpopularnijih algoritama bazira se upravo na osnovnim funkcionalnostima konvolucijskih neuronskih mreža. U nastavku teksta prikazane su tri inačice arhitekture konvolucijskih neuronskih mreža za primjenu na detekciji objekata.

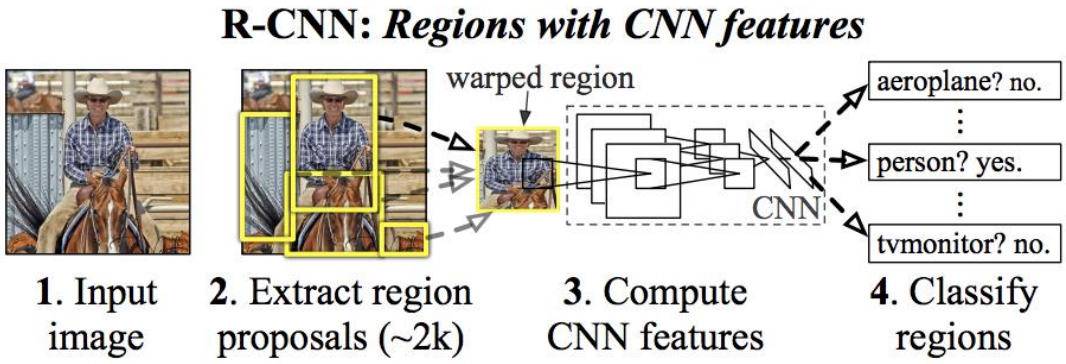
1.2.1. R-CNN

R-CNN (engl. Region-based Convolutional Neural Network) [14], konvolucijska je neuronska mreža koja se zasniva na izdvajanju regija slike te izvlačenju značajki iz izabranih regija (Slika 1.5). Algoritam koristi selektivno pretraživanje kako bi generirao oko 2000 predloženih regija na slici.

Selektivno pretraživanje se najčešće sastoji od 3 koraka [15]:

- Generiranje velikog broja potencijalnih regija slike
- Korištenje pohlepnog algoritma pretraživanja prilikom rekurzivnog spajanja sličnih regija u veće cjeline
- Uporaba generiranih cjelina prilikom stvaranja finalnih predloženih regija

Tako izabrane regije, smanjuju se na zadanu veličinu te se pomoću konvolucijske neuronske mreže za svaku zasebnu regiju izračunavaju značajke. U posljednjem sloju radi se klasifikacija značajki i predviđa koji se objekt nalazi na slici.

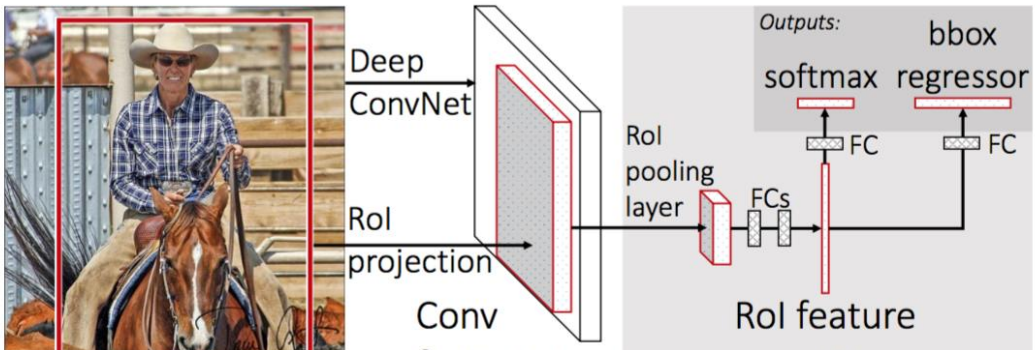


Slika 1.5: Prikaz arhitekture R-CNN. [13]

R-CNN, unatoč svojim dobrim rezultatima prilikom izdvajanja značajki, ima i značajne nedostatke [16]. Budući da svaka predložena regija mora zasebno proći kroz CNN, algoritam se izvršava veoma sporo. Također, dolazi do velike potrošnje memorije jer svaka predložena regija zahtjeva značajnu količinu memorije za pohranu značajki. Rješenje navedenih problema dovodi do razvitka poboljšanih inačica R-CNN-a.

Fast R-CNN

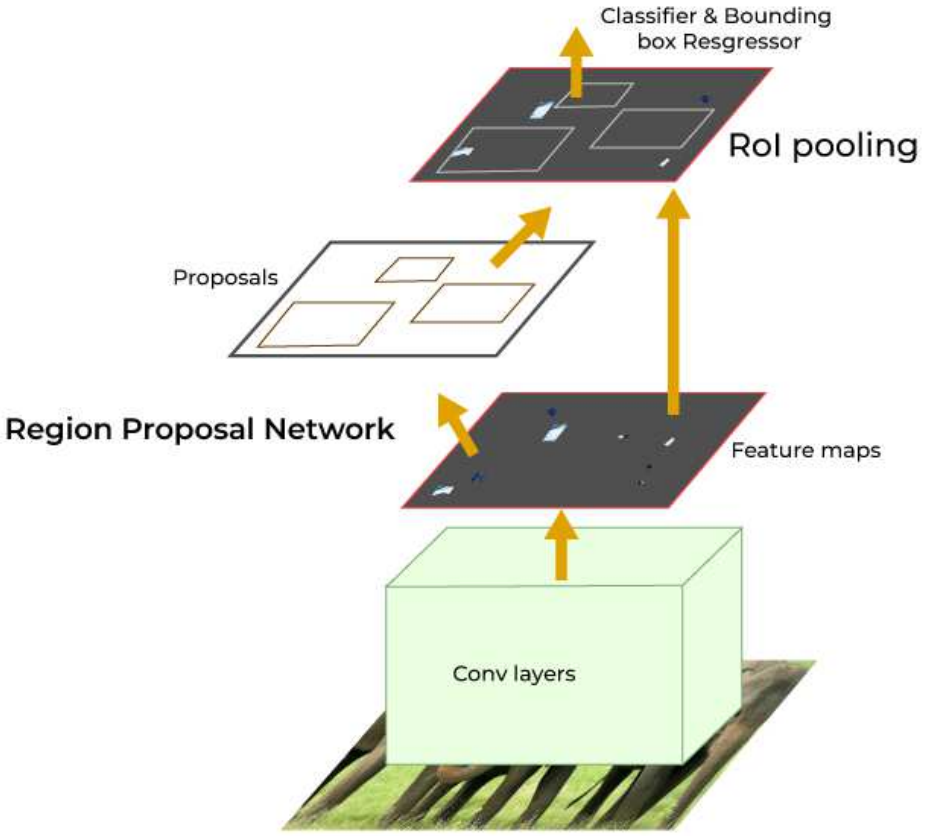
Za razliku od R-CNN-a, Fast R-CNN pomoću konvolucijske neuronske mreže dobiva mapu značajki cijele ulazne slike. Nakon toga, za svaki prijedlog mogućeg objekta, sloj udruživanja regija interesa (eng. *Region of Interest – RoI*) izvlači vektor značajke iz mape značajki. Svaki tako izvučeni vektor značajke daje se na ulaz potpuno povezanim slojevima koji se granaju u dva izlazna sloja. Jedan sloj daje četiri broja kojima je označen granični okvir za jednu klasu, a drugi daje procjenu vrijednosti vjerojatnosti nad klasama objekata [17]. Prikaz arhitekture Fast R-CNN-a možemo vidjeti na Slici 1.6. [17]



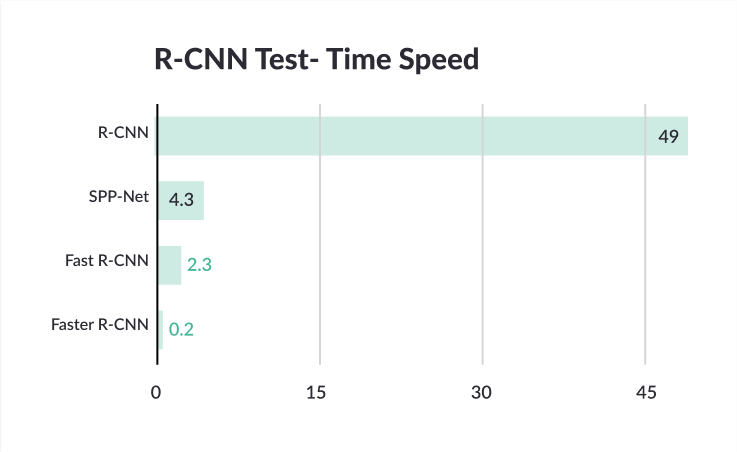
Slika 1.6: Arhitektura Fast R-CNN-a. [16]

Faster R-CNN

Mana koju dijele i Fast R-CNN i R-CNN je selektivno pretraživanje za generiranje predloženih regija slike. Kao i kod Fast R-CNN -a, dobiva se mapa značajki cijele ulazne slike. Glavna razlika, tj. unaprjeđenje jest izbacivanje selektivnog pretraživanja iz cijeloga procesa zbog njenog sporijeg vremena izvođenja. Faster R-CNN koristi zasebnu mrežu (eng. *Region Proposal Network – RPN)* kojom iz mape značajki cijele ulazne slike predviđa predložene regije. Predložene regije se preoblikuju pomoću sloja udruživanja regija interesa te dolazi do klasifikacije objekta u predloženoj regiji i predviđanja vrijednosti graničnog okvira objekta [18]. Prikaz arhitekture Faster R-CNN-a možemo vidjeti na Slici 1.7. Usporedba odnosa brzina između Faster R-CNN-a, Fast R-CNN-a te R-CNN-a vidljiva je na Slici 1.8



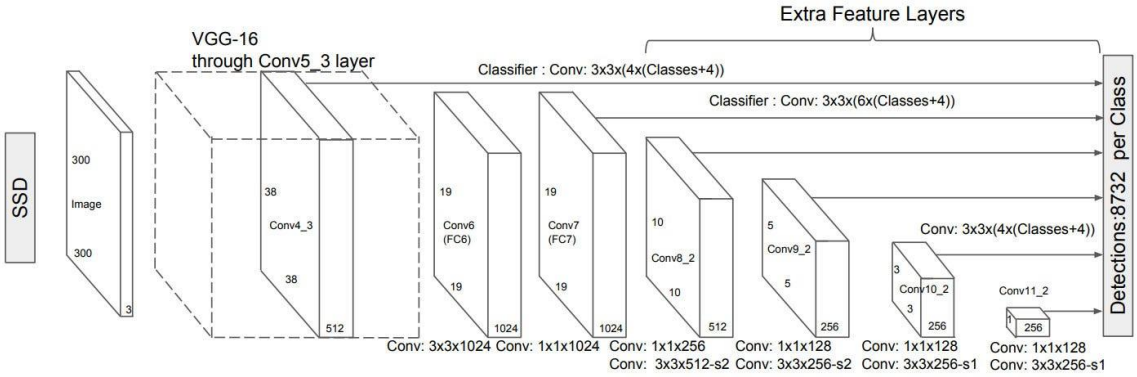
Slika 1.7: Arhitektura Faster R-CNN-a. [17]



Slika 1.8: Usporedba brzina provedbe R-CNN algoritama. [19]

1.2.2. Single Shot Detector (SSD)

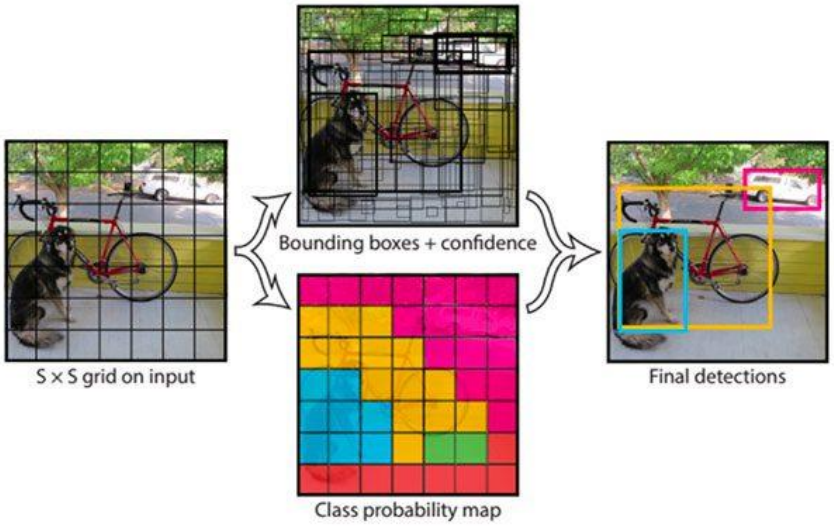
Za razliku od detekcije pomoću predloženih regija koja koristi dvije slike, SSD detektira objekte koristeći samo jednu, čineći ga znatno bržim od ranije navedene metode [20]. Arhitektura SSD-a sastoji se isključivo od potpuno konvolucijskih slojeva koji kao rezultat daju kolekciju graničnih okvira fiksne veličine. Uz kolekciju graničnih okvira, mreža boduje vjerojatnost prisutnosti određenih objekata unutar samih graničnih okvira te na posljetku provodi korak ne-maksimalne supresije za dobivanje konačnih detekcija. Slika prvo prolazi kroz konvolucijske slojeve dizajniranje za izvlačenje mapa značajki. Dobivene mape značajki provlače se kroz dodatne konvolucijske slojeve koji smanjuju njihove dimenzije kako bi se mogli detektirati objekti različitih veličina. Nakon toga se mapama značajki dodjeljuju fiksni granični okviri s različitim omjerima širine i visine. Zadane granične okvire mreža mijenja ovisno o stvarnim objektima na slici. Mreža koristi konvolucijske filtre veličine 3×3 kako bi napravila predikciju za svaki pojedini objekt. Ti filtri računaju vjerojatnost pojavljivanja objekata unutar zadanih graničnih okvira te vrijednosti prilagodbe potrebne za transformaciju graničnih okvira u konačne okvire detektiranih objekata. Dobiveni izlazni podaci prolaze kroz korak ne-maksimalne supresije koja eliminira suvišne i nepotrebne granične okvire, efektivno dajući najtočnije rješenje za granične okvire i klasne oznake [21].



Slika 1.9: Arhitektura SSD-a. [19]

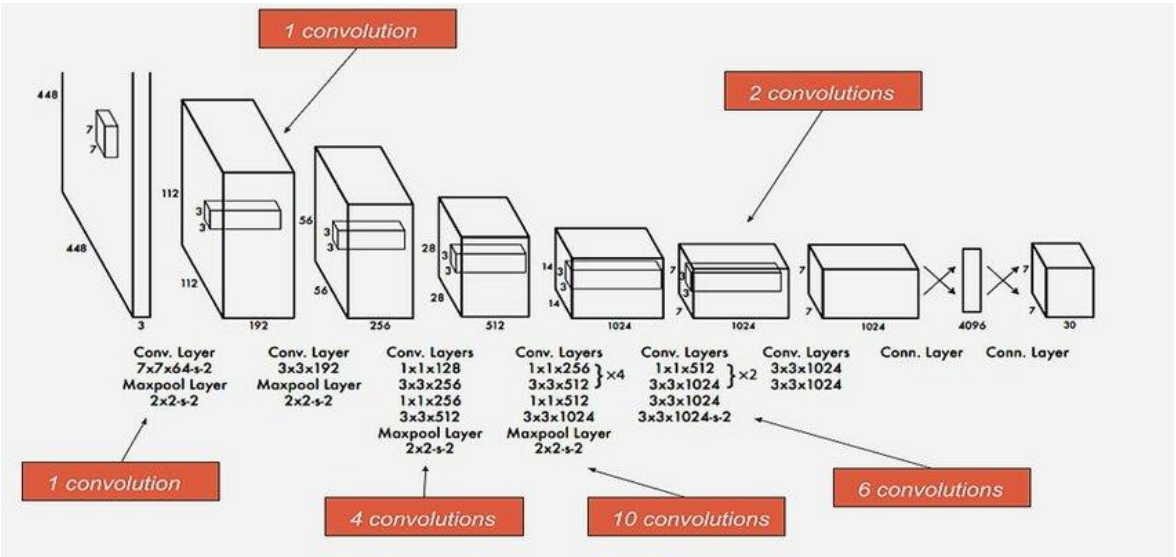
1.2.3. YOLO (You Only Look Once)

YOLO (eng. *You Only Look Once*) jedan je od najpopularnijih algoritama za detekciju objekata te će kao takav biti korišten u nastavku ovoga rada. Za razliku od R-CNN algoritama, YOLO ne gleda značajke prema regijama slike, već za predviđanje graničnih okvira koristi značajke iz cijele ulazne slike. Algoritam YOLO funkcionira na sljedeći način: ulazna slika se skalira na unaprijed zadanu veličinu (npr. 448×448 piksela u YOLOv1) te se nakon toga dijeli na $S \times S$ ćelija gdje je svaka ćelija zadužena za predikciju objekata čije središte spada unutar te ćelije. Izmijenjena slika se propušta kroz konvolucijsku neuronsku mrežu kako bi se izračunale značajke. Svaka ćelija podijeljene slike predviđa B graničnih okvira i računa njihove rezultate pouzdanosti. Granični okviri sastoje se od 5 predikcija: x, y, w, h i pouzdanosti predikcije. X i y koordinate predstavljaju središte graničnog okvira s obzirom na granice ćelije unutar koje pripada. W i h su mjere širine, odnosno visine koje se uzimaju s obzirom na veličinu cijele slike. Pouzdanost predikcije predstavlja mjeru između predviđenog i stvarnog graničnog okvira nekog objekta. Svaka ćelija također predviđa C kondicijskih vjerojatnosti klase objekta. Ove vjerojatnosti ovise o prisustvu samoga objekta u pojedinoj ćeliji, ako se u nekoj ćeliji ne nalazi niti jedan objekt, ova vjerojatnost jednaka je nuli. Konačni izračun dobiva se množenjem vjerojatnosti klase objekata s pouzdanošću predikcije kako bi se dobile vrijednosti pouzdanosti za svaki granični okvir. Svaka dobivena vrijednost predstavlja vjerojatnost pojavljivanja određene klase unutar graničnog okvira te koliko dobro sami okvir pokriva traženi objekt [22].



Slika 1.10: Princip rada YOLO algoritma. [22]

Jedna od najvećih prednosti YOLO-a nad ostalim algoritmima je njegova velika brzina procesuiranja te je iz tog razloga često upotrebljavan za detekciju objekata u stvarnom vremenu. Iako je jedan od najpopularnijih i najčešće korištenih algoritama, YOLO ima i svoje mane. Jedna od glavnih mana uzrokovana ograničenim brojem graničnih okvira po pojedinoj ćeliji jest problem detekcije malih objekata ili objekata koji se nalaze u neposrednoj blizini jedan drugome.



Slika 1.11: Arhitektura YOLOv1 algoritama. [23]

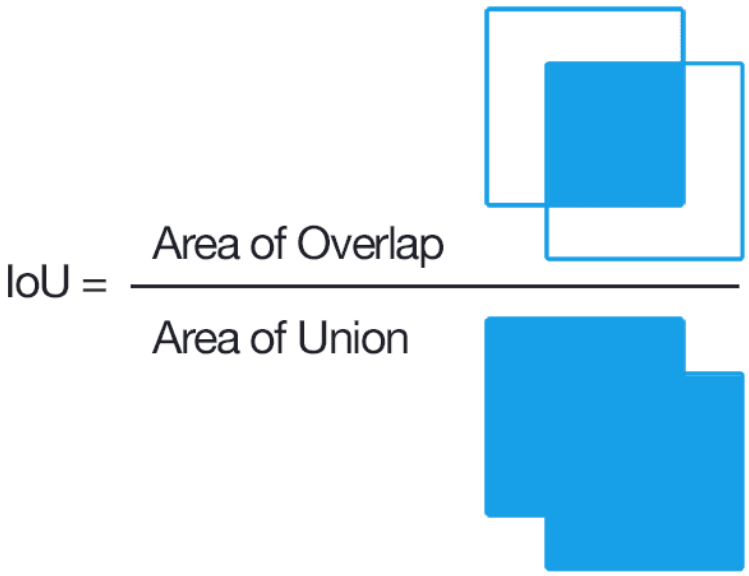
1.3. Evaluacija modela za detekciju objekata

Različiti algoritmi za detekciju objekata daju i različite rezultate prilikom treniranja modela. Neki algoritmi su brzi te ih je odlično koristiti za detekciju objekata u stvarnome vremenu, dok drugi zahtijevaju više vremena za detekciju, no pružaju veći postotak uspješno detektiranih objekata. Kako bi se proučili rezultati koje nam daju istrenirani modeli, razvijena su brojne metrike uspješnosti samih modela. Najčešće se gleda postotak točnosti klasifikacije objekata prema klasama te preciznost određenog graničnog okvira.

1.3.1. Osnovne metode evaluacije

Kao i ranije spomenuto, svaka detekcija se sastoji od graničnog okvira, klasifikacije samoga objekta te mjere pouzdanosti. Mjera pouzdanosti (eng. *confidence score*) predstavlja vjerojatnost da granični okvir sadrži neki objekt.

Osim mjere pouzdanosti, jedna od glavnih metoda za procjenu kvalitete i preciznosti modela jest Jaccardov indeks, popularnije nazvan IoU (eng. *Intersection over Union*). IoU predstavlja omjer površine presjeka dvaju graničnih okvira i ukupne površine tih graničnih okvira (Slika 1.12). Ovime dobivamo brojčanu vrijednost u rasponu [0, 1] koja nam govori koliko dobro model predviđa lokaciju objekta u usporedbi s njegovom stvarnom lokacijom na slici. Što je IoU veći, to je veće poklapanje između predviđenih i ispravnih (eng. *ground truth*) graničnih okvira [24]. IoU u praksi funkcionira odlično za relativno velike objekte, no ima problema prilikom detekcije malenih objekata.



Slika 1.12: Izračun IoU-a. [24]

IoU se koristi ukomponiran zajedno sa zadanim klasifikacijskim pragom. Zadani klasifikacijski prag koristimo kako bismo mogli definirati je li neka detekcija ispravna – TP (eng. *True Positive*). Detekcija je ispravna ako je predviđena klasa objekta jednaka ispravnoj klasi, IoU predviđenog graničnog okvira s ispravnim graničnim okvirom je veći od zadanog klasifikacijskog praga te ako je mjera pouzdanosti veća od zadanog praga. U slučaju krivo predviđene klase objekta ili ako je IoU manji od zadanog klasifikacijskog praga, detekcija je neispravna – FP (eng. *False Positive*). Ako model ne detektira objekte koji su prisutni onda to nazivamo FN (eng. *False Negative*).

Koristeći navedene parametre, određuju se dva vrlo važna podatka, a to su preciznost (eng. *precision*) i odziv (eng. *recall*) [25].

Preciznost je sposobnost modela da identificira samo relevantne objekte. Drugim riječima to je omjer ispravno detektiranih objekata i ukupnog broja detektiranih objekata.

$$P = \frac{TP}{TP + FP} = \frac{TP}{sve\ detekcije} \tag{1}$$

Odziv je sposobnost modela da pronađe sve ispravne granične okvire. To je omjer ispravno detektiranih objekata i ukupnog broja stvarnih objekata u skupu podataka.

$$P = \frac{TP}{TP + FN} = \frac{TP}{svi\ stvarni\ objekti} \tag{2}$$

1.3.2. Prosječna preciznost

Ako je mjera pouzdanosti modela takva da je broj FP detekcija nizak, preciznost će biti visoka. Međutim, u ovom slučaju može doći do propuštanja velikog broja pozitivnih rezultata, što uzrokuje velik broj FN, a samim time i niži odziv. Prihvaćanjem većeg broja pozitivnih rezultata povećat će se odziv, no može doći do povećanja broja FP, što smanjuje preciznost [25]. Dakle, pouzdanost i odziv imaju inverznu ovisnost. Potrebno je istaknuti da su obje mjere ovisne o zadanom klasifikacijskom pragu te s različitim pragovima imamo različite parove vrijednosti preciznosti i odziva. Kako bi se uskladile preciznost i odziv, odnosno kako bi probali učiniti da su obje vrijednosti što veće, crta se krivulja preciznosti i odziva (eng. *Precision-Recall (PR) curve*). Ona prikazuje kompromis između ovih mjera za različite vrijednosti klasifikacijskog praga te vizualizira koji klasifikacijski prag najbolje odgovara za trenutnu primjenu.

Kako bismo saželi informacije iz PR krivulje koristimo prosječnu preciznost (eng. *Average Precision – AP*). Prosječna preciznost je površina ispod PR krivulje te ima obilježje da je visoka upravo kada su i preciznost i odziv visoki [26]. Ona također ima vrijednosti u rasponu [0, 1] te se računa se prema izrazu (3).

$$Prosječna\ preciznost\ (AP) = \int_{r=0}^{1} p(r)dr \tag{3}$$

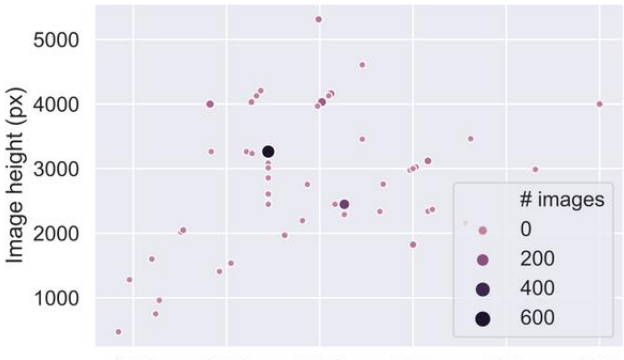
Najčešća mjera za validaciju modela u detekciji objekata je srednja prosječna preciznost (eng. *mean Average Precision – mAP*). Srednja prosječna preciznost se računa na način da se pronađe prosječna preciznost za svaku klasu te se potom uprosječi s ukupnim brojem klasa. Ona daje brojčanu vrijednost koja predstavlja uspješnost algoritma prilikom detekcije objekata i računa se prema izrazu (4).

$$Srednja\ prosječna\ preciznost\ (mAP) = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{4}$$

2. Skup podataka TACO

U ovom radu potrebno je detektirati i klasificirati otpad u slici s kamere. U tu svrhu koristi se TACO (eng. *Trash Annotations in Context*) [4] - skup podataka za detekciju i segmentaciju otpada. Sastoji se od 1500 slika koje se dijele na 60 kategorija i 28 nad-kategorija. Otpad se nalazi u različitim okolinama te su slike različitih veličina (Slike 2.1 i 2.2).

Slika 2.1: Primjer slika iz TACO-a. [4]

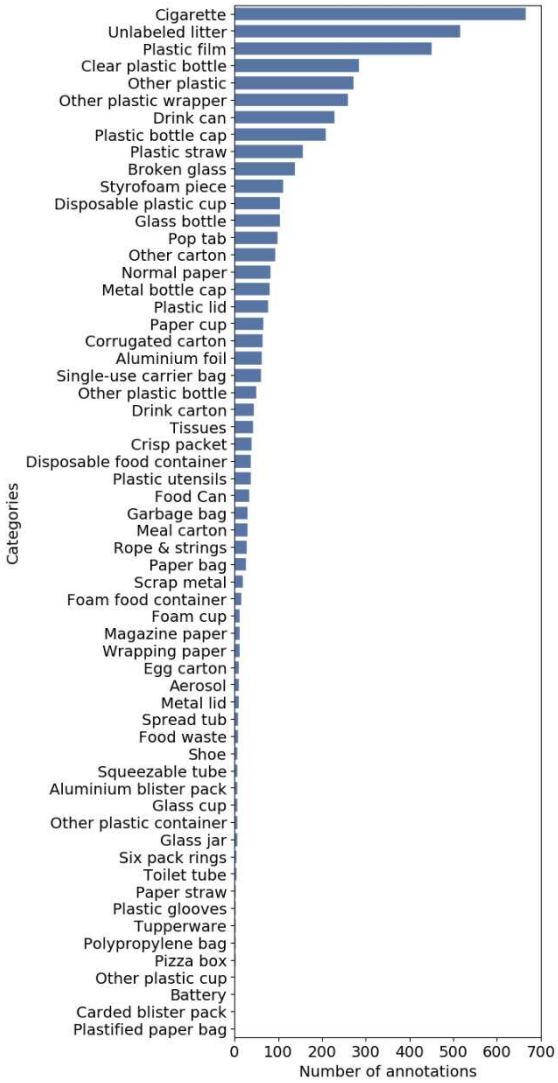


Slika 2.2: Raspon veličina slika u TACO-u. [4]

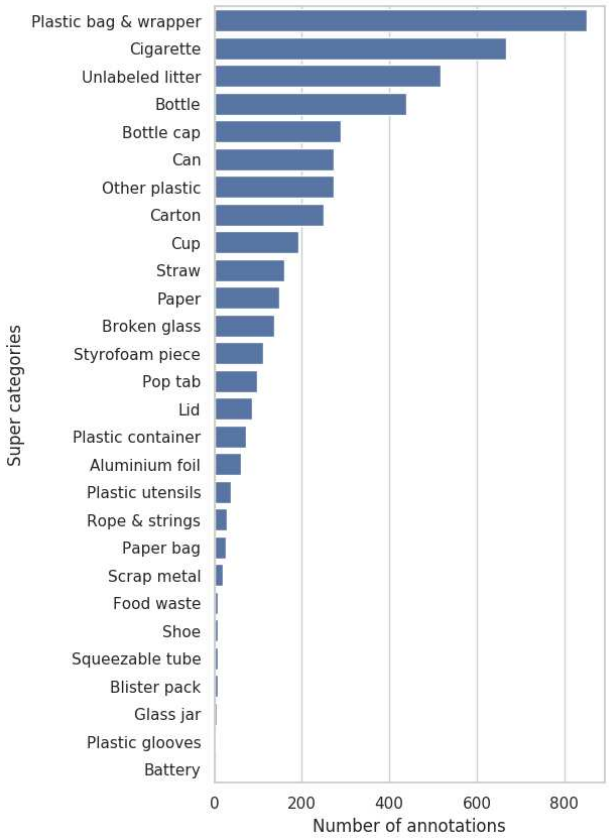
TACO je sakupljen s nabavom iz mnoštva (eng. *crowdsourcing*) te se skup prikupljenih slika konstantno povećava. Zbog malog broja slika te velikog broja kategorija, skup podataka je grupiran i u veće nad-kategorije. Sve kategorije i nad-kategorije su prikazane na slikama 2.3 i 2.4.

Sve anotacije vezane uz slike su u COCO [23] (eng. *Common Objects in Context*) formatu. U COCO formatu, anotacije su spremljene unutar JSON (eng. *JavaScript Object Notation*) datoteke koja sadrži informacije o putanji do slike ili videozapisa, njihovu veličinu te popis anotiranih objekata. Svaki objekt prikazan je graničnim okvirom, točnije, informacijama o položaju i veličini samog objekta unutar slike te oznakom kategorije unutar koje spada [27]. U slučaju TACO skupa podataka, JSON datoteka sadrži opće informacije kao što su širina, visina, licence, put do direktorija gdje se slika nalazi te poveznica na sliku postavljenu u oblaku na Flickr [28] web stranici. Također sadrži anotacijske informacije poput površine označenog objekta u pikselima, informacije o graničnim okvirima objekata u pikselima te

informacije o segmentaciji svakog objekta isto tako u pikselima. Na kraju stoji informacija vezana uz podjelu pozadina u kojima se pojedini objekt nalazi te informacija o kategoriji i nad-kategoriji u koju svaki objekt pripada. Ova vrsta anotacije ne odgovara prilikom korištenja YOLO algoritma, koji ima zaseban format anotacija, opisan nekoliko redaka ispod.



Slika 2.3: Broj anotacija po kategoriji. [4]



Slika 2.4: Broj anotacija po nad-kategoriji. [4]

U ovome radu se koristi skup podataka preuzet s Roboflow-a [29]. Roboflow je okvir za razvoj računalnog vida koji nudi razne tehnike treniranja modela te javne skupove podataka za treniranje. Preuzet je skup podataka koji se sastoji od 6004 slika [30], što je više nego orginalni TACO skup podataka, no to je tako jer sadrži ponovljene slika koje su drugačije orijentirane. Uz slike, u skupu podataka nalaze se i anotacije za svaku sliku. Anotacije su u ovom slučaju tekstualne datoteke u kojima su redom zapisani podaci o kategorijama objekata

na svakoj slici (broj kategorije), x-koordinatama središta graničnog okvira, y- koordinatama središta graničnog okvira objekata te normalizirana širina i normalizirana visina graničnih okvira. Npr.

```
2 0.41826923076923 0.52764423076923 0.0180288461538461 0.0168269230769230
12 0.94591346153846 0.88701923076923 0.034855769230769 0.0264423076923076
```

Ovakva anotacija odgovara algoritmu YOLO te su mjerne jedinice relativne dimenzijama same slike, odnosno normalizirane s visinom i širinom slike. Dodatno, ovaj skup podataka sadrži samo 18 kategorija objekata.

Podaci su podijeljeni u tri grupe, podatke za treniranje, podatke za provjeru te podatke za testiranje. Ovom se podjelom smanjuje mogućnost prenaučenosti. Modele različite složenosti treniramo na skupu za treniranje te svaki ispitujemo na podacima za provjeru. Izabiremo optimalni model na skupu za provjeru te njegovu točnost ispitujemo na skupu podataka za testiranje. Ukupni broj slika je podijeljen na opisana tri skupa u postotku: 70% za skup za treniranje, 28% za skup za provjeru te preostalih 2% za skup za testiranje.

3. Programska implementacija

3.1. Programska podrška

3.1.1. Python

Python [31] jedan je od najpopularnijih programskih jezika opće namjene. Spada u kategoriju programskih jezika visoke razine te podržava strukturno, objektno orijentirano i funkcionalno programiranje. On je interpreterski jezik te se ne prevodi u strojni kod, već se izvršni kod izvršava direktno pomoću interpretera. Karakteristika interpreterskih jezika je sporije izvršavanje od jezika čiji se izvršni kod prevodi u strojni. Python posjeduje veliku standardnu biblioteku što je za mnoge njegova najjača karakteristika te omogućuje korisnost u različitim područjima. U vrijeme pisanja ovoga rada, zadnja stabilna inačica Python-a jest Python 3.12.4 te je dostupan na mnoštvu operacijskih sustava, uključujući Windows, Linux/UNIX, macOs i dr.[28]

3.1.2. Programska biblioteka Ultralytics

Programska biblioteka Ultralytics [32] je alat koji se može koristiti unutar programskog jezika Python u svrhu detekcije, segmentiranja, klasifikacije, praćenja kretanja objekata te ostalih tema iz područja računalnog vida. Najpoznatija je po implementaciji YOLO algoritma te omogućavanju njegovog jednostavnog korištenja. Najnovije izdanje YOLO algoritma je YOLOv10, no u ovome se radu koristi YOLOv8.

3.1.3. Google Colab

Google Colab (Google Colaboratory) je platforma koju je razvila kompanije Google u svrhu pisanja i pokretanja vlastitog Python koda u web-pregledniku bez obzira na vlastito sklopovlje i programsku potporu. Ona omogućava pristup računalnim resursima i mnogim bibliotekama korištenim u području strojnog učenja. Njezina najveća prednost jest besplatan pristup grafičkim procesorskim jedinicama (eng. *GPU – Graphics Processing Units*) i tenzorskim procesorskim jedinicama (eng. *TPU – Tensor Processing Units*) koje su izuzetno korisne prilikom izvršavanja računalno intenzivnih zadataka, poput dubokog učenja [33].

3.2. Treniranje modela

3.2.1. Treniranje na skupu podataka preuzetih s Roboflow-a

Prvi korak je priprema skupa podataka na kojemu će se vršiti treniranje modela. Preuzeti skup podataka je već unaprijed podijeljen u skupove za treniranje, provjeru i testiranje pa je jedino preostalo urediti direktorij s podacima za treniranje i provjeru te njegovu komprimiranu verziju pohraniti na Google Drive. Sljedeći korak je stvaranje konfiguracijske datoteke u kojoj se navode podaci o putanjama do slika za treniranje i slika za provjeru te informacije o broju i nazivima kategorija objekata (Slika 3.1).

Slika 3.1: Prikaz sadržaja konfiguracijske yaml datoteke

Nakon pripremnih koraka, proces treniranja modela se nastavlja u Google Colab-u. Prvo se povezuje trenutna Colab bilježnica s Google Drive računom.

```
from google.colab import drive

drive.mount('/content/gdrive')
```

Nakon što je Google Drive račun gdje se nalaze skupovi podataka za treniranje i provjeru te konfiguracijska datoteka povezan, mora se raspakirati direktorij gdje se nalaze slike.

```
!scp '/content/gdrive/My
Drive/YOLO_object_detection/data.zip' '/content/data.zip'

!unzip '/content/data.zip' -d '/content'
```

Prva naredba kopira komprimirani direktorij `data.zip` u lokalnu okolinu Google Colab-a, dok druga raspakirava podatke i sprema ih unutar lokalnog direktorija `'/content'`. Sljedeći korak je preuzimanje biblioteke ultralytics unutar okoline Google Colaba naredbom:

```
!pip install ultralytics
```

Naredba `pip` služi za preuzimanje paketa koji sadrže biblioteke u Pythonu. Sada je sve spremno za treniranje modela, preostaje samo učitavanje YOLO klase iz biblioteke ultralytics i može se krenuti s treniranjem.

```
from ultralytics import YOLO

model = YOLO("yolov8n.yaml")

results = model.train(data='/content/gdrive/My Drive/
YOLO_object_detection/config.yaml', epochs=100, imgsz=640)
```

U varijablu `model` se učitava YOLOv8 te se nad njom zove funkcija `.train()` kako bi započeo proces treniranja modela. Prvi parametar koji prima funkcija .train() je put do konfiguracijske yaml datoteke, drugi je broj epoha za koji se model trenira te treći parametar koji predstavlja veličinu na koju se skaliraju sve slike unutar skupa podataka za treniranje. Epoha je jedan prolaz kroz ukupni skup podataka za treniranje. Dodatno postoje mnogi drugi parametri koji se mogu staviti u ovu funkciju, kao što su `save_period` kojim zadajemo frekvenciju spremanja modela zadanu u epohama, `patience` kojim označavamo broj epoha nakon kojeg dolazi do ranijeg zaustavljanja treniranja u slučaju kada nema poboljšanja u validacijskim metrikama modela i dr. [34]

Rezultat izvršavanja svih epoha je skup informacija vezanih uz performanse modela i težine modela nakon najbolje (best.pt), odnosno zadnje epohe (last.pt). Modeli na ovome skupu podataka su trenirani u periodima od 100, 200, 400 i 500 epoha.

3.2.2. Treniranje na originalnom TACO skupu podataka

Unatoč pronalaska odgovarajuće verzije TACO skupa podataka za treniranje YOLO algoritmom, u ovom dijelu slijedi opis rada s originalnim TACO skupom podataka. Kao što je ranije navedeno, TACO koristi COCO format za anotaciju objekata na slikama, no to nije odgovarajući format za rad s YOLO algoritmom. Kako bi se ovaj problem riješio, napisan je kod za konverziju iz COCO formata u YOLO format. U JSON datoteci u kojoj se nalaze relevantni podaci o anotacijama objekata na slikama moguće je pronaći informacije o širinama i visinama svake slike, položaju graničnog okvira svakog objekta te kategorije kojoj svaki objekt pripada. Sve mjere vezane za slike i objekte izražene su u pikselima. Dodatni problem je broj kategorija objekata. Zbog velikog broja različitih kategorija otpada, želja je smanjiti broj na najčešće kategorije koje se koriste prilikom razvrstavanja otpada. Kategorije bi se svele samo na: plastiku, papir, staklo, metal, biootpad te ostali otpad. U kodu koji vrši pretvorbu iz COCO u YOLO format dodatno se broje sve dostupne kategorije objekata te se spremaju u zasebnu JSON datoteku. Izvorni skup podataka ima slike raspoređene unutar više direktorija koji uz same slike sadrže i prikladne JSON datoteke gdje su sve informacije o anotacijama slika za svaki pojedini direktorij.

Programski kod ostvaren je u Pythonu te koristi os, json te shutil standardne biblioteke. Biblioteke služe za interakciju sa sustavom datoteka, čitanje i pisanje JSON datoteka te kopiranje dokumenata iz jednog direktorija u drugi.

Osnovna funkcionalnost programskog koda se sastoji od:

- Prolaska po svim direktorijima gdje su sadržane slike
- Čitanja prikladne anotacijske JSON datoteke
- Pronalaska svih dosad ne viđenih kategorija objekata i spremanje u rječnik
- Pronalaska svih relevantnih informacija za konverziju formata
- Pretvorba formata
- Spremanje informacija o slikama u tekstualne datoteke
- Kopiranje slika u zasebni direktorij zajedno sa svim tekstualnim datotekama
- Spremanje informacija o svim klasama

Pretvorba iz COCO u YOLO format ostvarena je funkcijom:

```
def convert_coco_to_yolo(img_width, img_height, bbox):
    x_min, y_min, width, height = bbox
    x_center = x_min + width / 2
    y_center = y_min + height / 2
    x_norm = x_center / img_width
    y_norm = y_center / img_height
    width_norm = width / img_width
    height_norm = height / img_height
    return x_norm, y_norm, width_norm, height_norm
```

Ulazni parametri funkcije su širina i visina slike te mjere graničnog okvira objekta. Svi parametri su izraženi u pikselima. Iz varijable `bbox` se čitaju koordinate gornjeg lijevog vrha graničnog okvira te njegova širina i visina. Računaju se koordinate središta graničnog okvira te se potom dijele sa širinom i visinom cijele slike kako bi se normalizirale. Također se računaju normalizirana visina i širina graničnog okvira te je konverzija gotova.

Nakon brojanja svi kategorija otpada, pronađeno je svih 60 kategorija kao što je i navedeno u dokumentaciji TACO skupa podataka.

Kategorije su svedene na 6 glavnih kategorija:

- Plastika – sadrži ukupno 22 kategorije iz originalne podijele
- Metal – sadrži 8 kategorija iz originalne podijele
- Papir – sadrži 14 kategorija iz originalne podijele
- Staklo – sadrži 4 kategorije iz originalne podijele
- Biootpad – sadrži 1 kategoriju iz originalne podijele
- Ostali otpad – sadrži 11 kategorija iz originalne podijele

Ukupan broj anotiranih objekata po kategoriji:

- Plastika – 2081 objekata
- Metal – 484 objekata
- Papir – 497 objekata
- Staklo – 254 objekata
- Biootpad – 8 objekata
- Ostali otpad – 1460 objekata

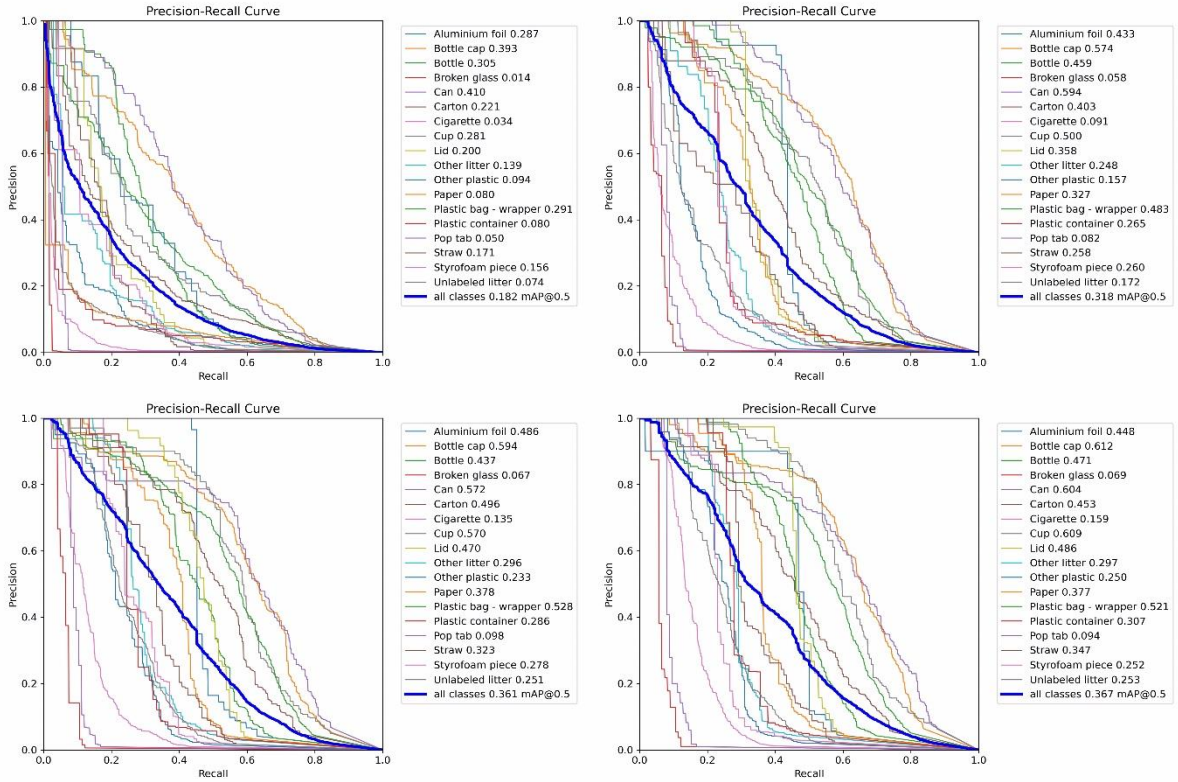
Ponovno postoji disbalans u broju anotiranih objekata, no to je odraz neproporcionalno prikupljenih slika unutar originalnoga skupa podataka. Postupak treniranja jednak je kao u poglavlju 3.2.1. te je istrenirano tri modela, s 250, 330 i 460 epoha.

4. Analiza istreniranih modela

U ovome se poglavlju gledaju rezultati treniranja modela na ranije spomenutim skupovima podataka te njihove performanse na dosad ne viđenim slikama. Za analizu modela koriste se metrike objašnjene u poglavlju 1.3. te matrica zabune. Matrica zabune je tablica kojom se prikazuje odnos između broja stvarnih kategorija objekta i broja predviđenih kategorija objekta. Stupci najčešće predstavljaju predviđene kategorije, dok je svaki redak reprezentacija stvarnih kategorija [35].

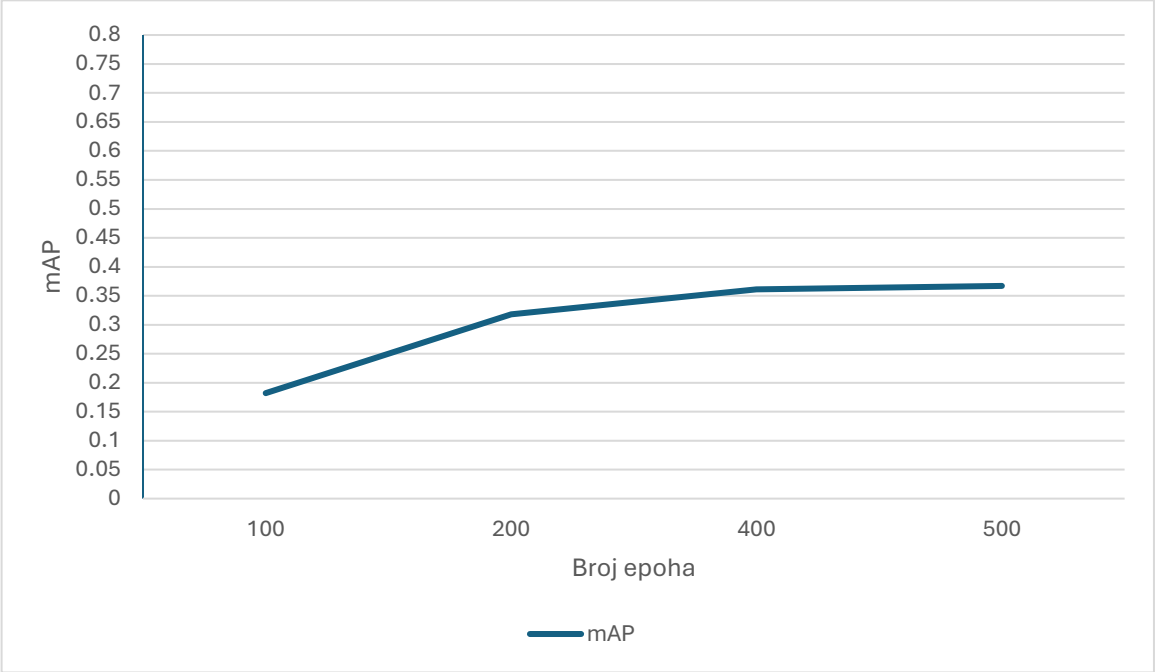
4.1. Rezultati treniranja na skupu podataka s Roboflow-a

Na skupu podataka preuzetih s Roboflow-a trenirana su 4 modela. Svaki je model treniran na skupu za treniranje koji se sastoji od 4200 slika te je provedena provjera sa skupom od 1704 slika. Modeli se razlikuju po broju epoha u trenažnom procesu. Jedna od najbitnijih metrika je srednja prosječna preciznost (mAP). U slučaju ove analize klasifikacijski prag za IoU jest 0.5 (50%), što znači da se predviđeni granični okvir smatra ispravnim ako se barem 50% njegove površine preklapa sa stvarnim graničnim okvirom nekog objekta.



Slika 4.1: PR krivulje za modele trenirane na 100 (gore lijevo), 200 (gore desno), 400 (dolje lijevo) i 500 (dolje desno) epoha

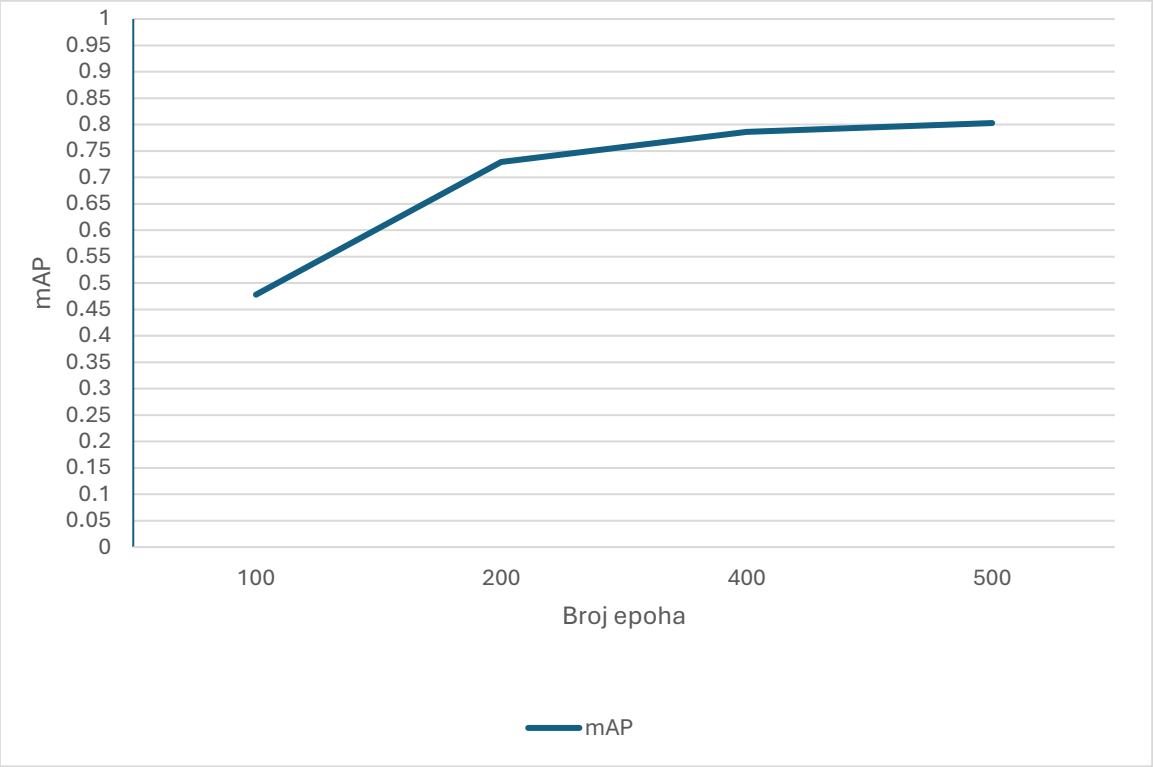
Iz PR krivulja pojedinih modela se vidi da postoji ovisnost između broja epoha i srednje prosječne preciznosti. Ovi podaci dobiveni su nakon treniranja na skupu podataka za provjeru.



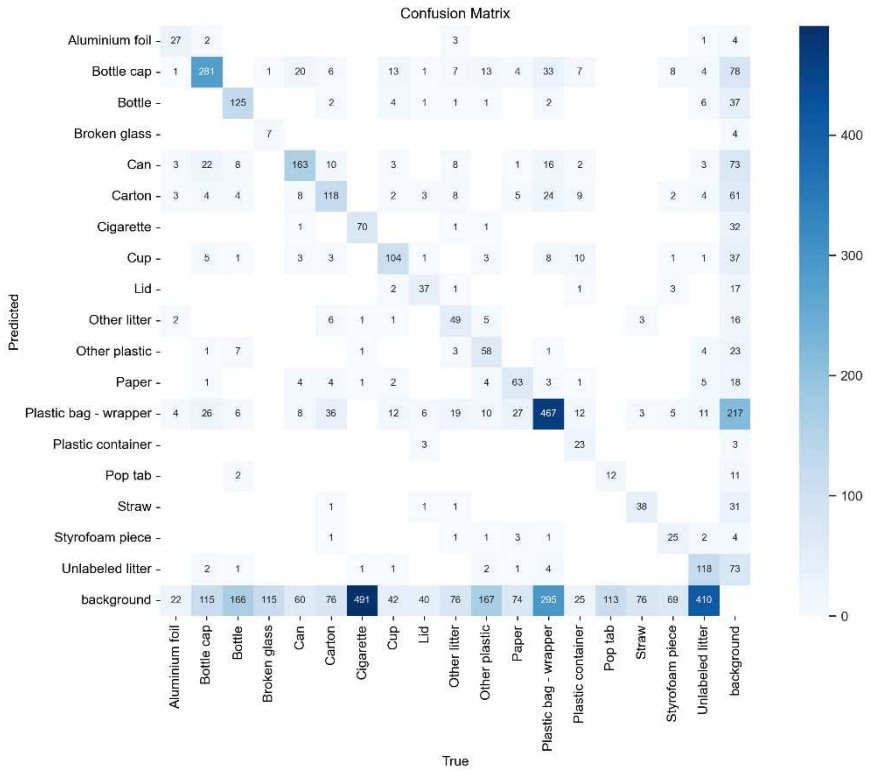
Slika 4.2: Grafički prikaz ovisnosti broja epoha i srednje prosječne preciznosti nad skupom podataka za provjeru

Jasno se vidi da je inicijalna faza treniranja najvažnija u razvoju modela tako što je između modela treniranog na 100 i modela treniranog na 200 epoha značajan skok u vrijednosti srednje prosječne preciznosti. Između modela treniranog na 200 epoha i modela treniranog na 400 epoha dolazi do malenog rasta srednje prosječne preciznosti te se nakon 400 epoha ulazi u laganu stagnaciju gdje nije nužno da će dodatnim treniranjem doći do boljih rezultata.

Rezultati dobiveni nad skupom za provjeru podosta se razlikuju od rezultata dobivenih na skupu za treniranje. Mogući razlog tomu jest što je skup podataka za treniranje relativno malen te postoji veliki nesrazmjer između broja anotacija unutar svake kategorije. Ovo uzrokuje veću šansu za prepoznavanjem kategorija koje imaju više anotacija za treniranje te krivu klasifikaciju objekata nakon što su detektirani. Unutar ovoga skupa podataka za treniranje upravo kategorije koje imaju najveću srednju prosječnu preciznost imaju i najveći broj anotacija. Još jedan mogući razlog je pretreniranost na skupu za treniranje što dovodi do loše generalizacije na neviđenom skupu podataka. Jedna od mogućih indikacija pretreniranosti jest i stagniranje srednje prosječne preciznosti što možemo vidjeti na slici 4.2 između 400 i 500 epoha.



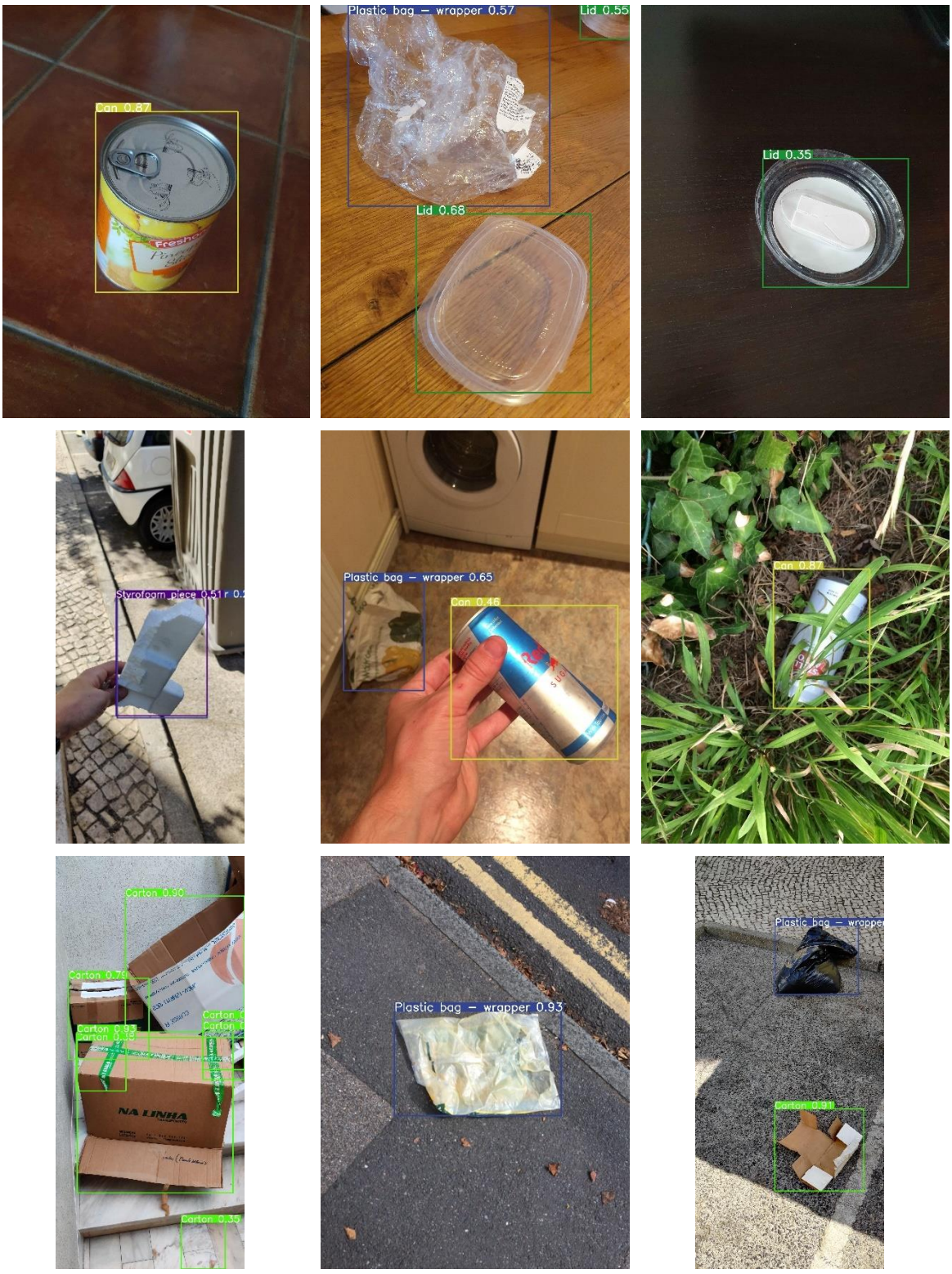
Slika 4.3: Grafički prikaz ovisnosti broja epoha i srednje prosječne preciznosti nad skupom podataka za treniranje



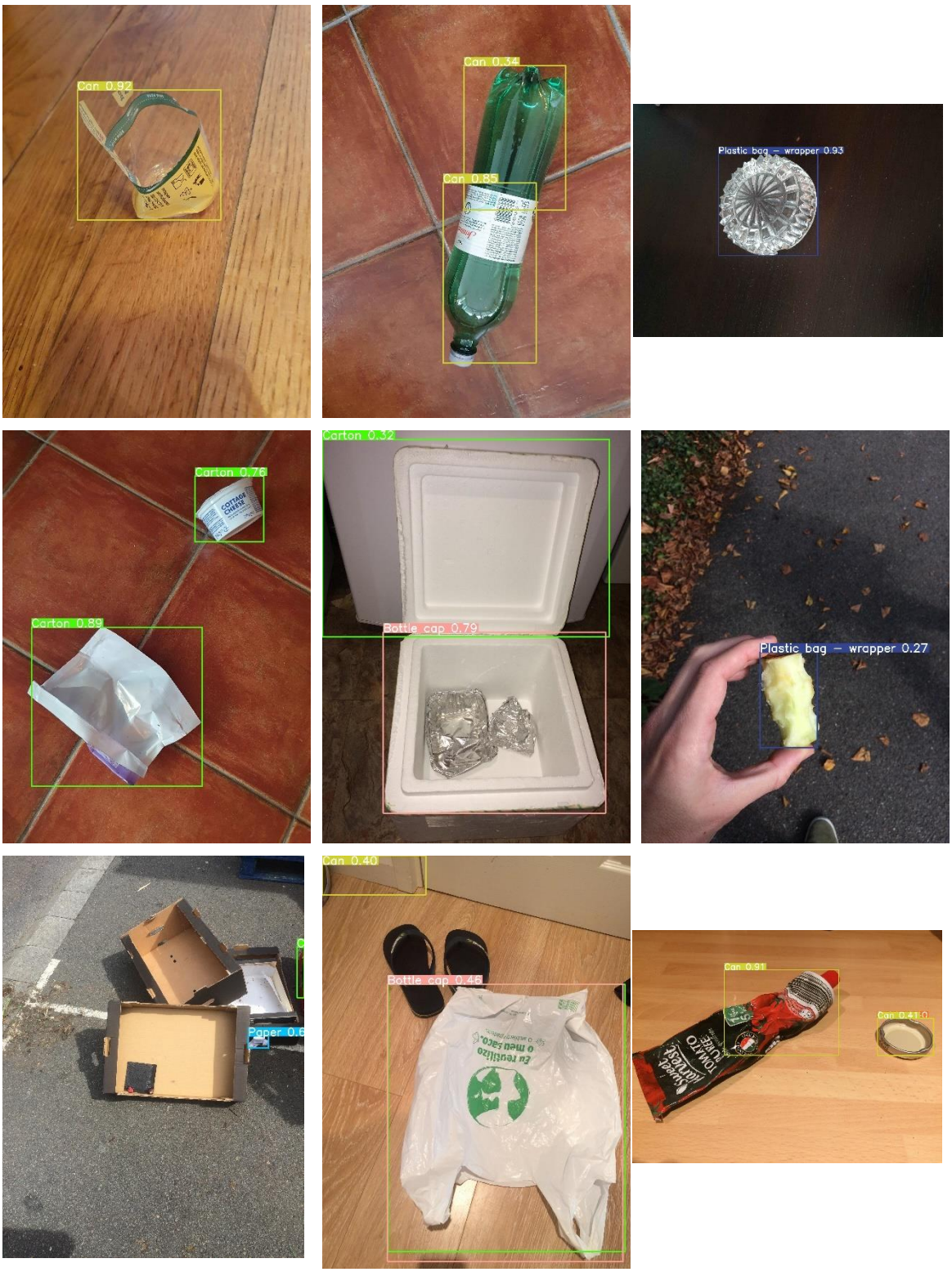
Slika 4.4: Matrica zabune modela treniranog na 500 epoha

Iz matrice zabune vidljivo je da ima raznih predikcija za različite kategorije, no ipak poslije krive detekcije okoline kao objekta, najviše podudaranja ima na dijagonali matrice, što bi i trebalo biti ako je model dobar u detekciji objekata. Uzevši sve u obzir s trenutno dostupnim modelima, zaključuje se da bi za zadani slučaj model s najboljim performansama trebao imati između 400 i 500 epoha za treniranje.

Slika 4.5: Detekcija otpada modela s 500 epoha treninga na vlastitim primjerima.



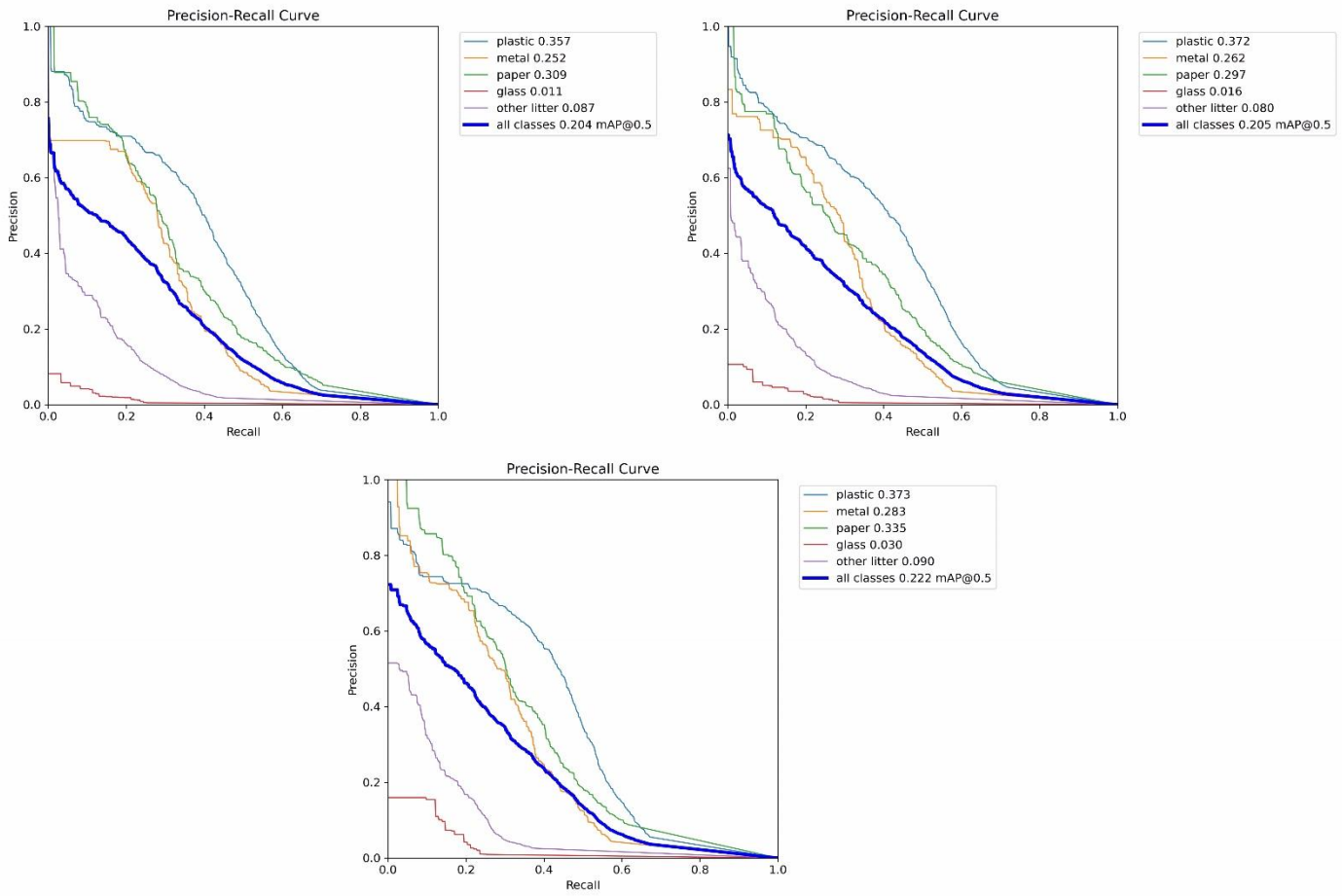
Slika 4.6: Primjer dobrih detekcija modela s 500 epoha treninga



Slika 4.7: Primjer loših detekcija modela s 500 epoha treninga

4.2. Rezultati treniranja na originalnom skupu podataka

Modeli trenirani na originalnom TACO skupu podataka također imaju jednak problem kao i modeli trenirani na skupu podataka preuzetih s Roboflow-a. Ovaj skup podataka ima još veći nesrazmjer u broju anotacija po svakoj kategoriji jer je on grupirana verzija originalnog TACO skupa podataka te ima manji broj slika za treniranje (1500). Ovdje ima samo šest kategorija objekata te je kategorija plastike najzastupljenija. Modeli su trenirani na 250, 330 i 460 epoha te koriste jednak skup podataka za provjeru kao i modeli u prošlom poglavlju.

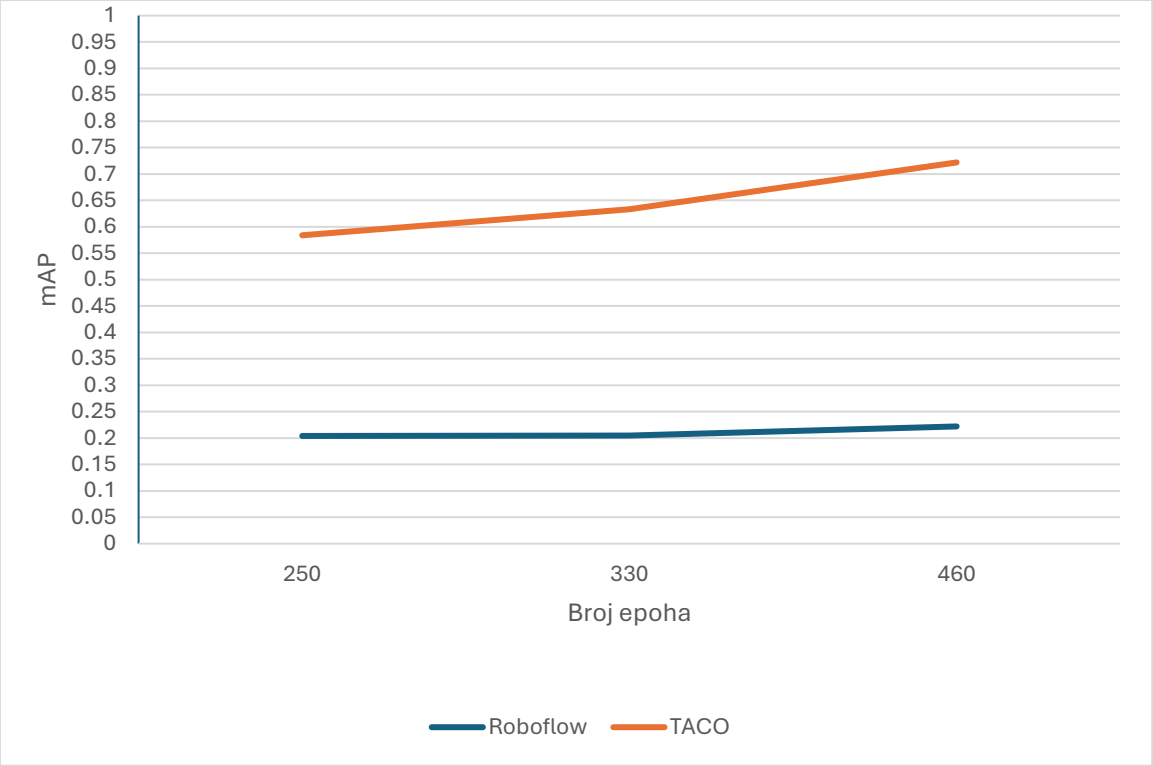


Slika 4.8: PR krivulje za modele trenirane na 250 (gore lijevo), 330 (gore desno), 460 (dolje) epoha

Primjetno je da kategorije koje imaju manju zastupljenost i/ili veliku raznolikost objekata (staklo i ostali otpad) također imaju znatno manju srednju prosječnu preciznost. Modeli trenirani ovim brojevima epoha imaju gotovo jednake srednje prosječne preciznosti, što znači da veći niz epoha nije rezultirao pa skoro nikakvim napretkom. Ovako slabi rezultati, između svega već navedenoga, mogu biti i uvjetovani slabijom kvalitetom slika u skupu za provjeru. Modeli su trenirani na originalnom TACO skupu podataka gdje su slike velikom

većinom visoke kvalitete što bi potencijalno moglo uzrokovati slabije prepoznavanje na slikama lošije kvalitete. Rješenje bi bilo ukomponirati u originalni TACO skup podataka i slike otpada koje su slabije kvalitete kako bi modeli mogli biti spremni i na takve prilike.

Provedena je i provjera na skupu slika koje su mješavina originalnih slika iz TACO skupa podataka, zajedno s verzijama slika zaokrenutih oko vertikalne, odnosno horizontalne osi te su rezultati drastično bolji.



Slika 4.9: Usporedba srednjih prosječnih preciznosti za Roboflow i TACO skupove podataka za provjeru

Rezultati dobiveni nad skupom podataka za provjeru gdje se nalaze originalne TACO slike s povremenim promjenama orijentacije približni su rezultatima dobivenim tijekom treniranja, stoga je vjerojatno premalena razlika u skupu za treniranje i ovome skupu za provjeru te su rezultati zanemarivi.

Ovaj skup podataka je najvjerojatnije premalen da bi se dobio solidan model za prepoznavanje otpada te ga je potrebno proširiti raznim augmentacijama slika te dodatnim slikama za manje zastupljene kategorije.

Slika 4.10: Detekcija otpada modela s 460 epoha treninga na vlastitim primjerima

Slika 4.11: Primjer dobrih detekcija modela s 460 epoha treninga

Slika 4.12: Primjer loših detekcija modela s 460 epoha treninga

Zaključak

Ovaj rad se bavi proučavanjem i primjenom detekcijskih i klasifikacijskih algoritama kako bi se mogao detektirati otpad na slikama. Kroz uvod se predstavlja problem i sama tematika rada. Objašnjava se princip rada konvolucijskih neuronskih mreža te se prolazi kroz najpoznatije algoritme za detekciju objekata na slikama. Prikazuju se najvažnije evaluacijske metrike za procjenu rada modela. Prikazuje se skup podataka TACO koji se koristi za detekciju otpada. Treniraju se modeli za detekciju otpada pomoću Python biblioteke ultralytics i njene implementacije algoritma YOLO te se rezultati detaljno opisuju i proučavaju.

Modeli dobiveni treniranjem nisu savršeni te nisu spremni za upotrebu u stvarnome svijetu. Razlozi variraju, no definitivno jedan od važnijih faktora mora biti veličina skupa podataka te velika raznolikost u kategorijama otpada. Kao rješenja na ovaj problem nameće se spajanje kategorija u veće grupe te moguća augmentacija postojećih slika kako bi se proširio skup podataka i spriječila pretreniranost. Također se nameće problem resursa za treniranje modela, naime kako bi se dobili što detaljniji rezultati, potrebno je imati što više modela s različitim parametrima treniranja poput broja epoha na kojima su trenirani. Problem je što je to skup proces te je potrebna dobra grafička kartica kako bi se treniranje maksimalno ubrzalo.

Iako rezultati rada nisu previše zadovoljavajući, ostaje otvorena mogućnost za nadogradnju skupa podataka, a i samih modela te moguća primjena nekog drugog algoritma za detekciju objekata kako bi se vidjele razlike u primjeni svakog algoritma. [36]

Literatura

[1] „What is Object Detection? | IBM“. Pristupljeno: 02. lipanj 2024. [Na internetu]. Dostupno na: https://www.ibm.com/topics/object-detection

[2] „Google Colab“. Pristupljeno: 02. lipanj 2024. [Na internetu]. Dostupno na: https://colab.research.google.com/

[3] „Kaggle: Your Home for Data Science“. Pristupljeno: 02. lipanj 2024. [Na internetu]. Dostupno na: https://www.kaggle.com/

[4] P. Proença i P. Simões, „TACO Dataset“. Pristupljeno: 02. lipanj 2024. [Na internetu]. Dostupno na: http://tacodataset.org/

[5] „14.3. Object Detection and Bounding Boxes — Dive into Deep Learning 1.0.3 documentation“. Pristupljeno: 02. lipanj 2024. [Na internetu]. Dostupno na: https://d2l.ai/chapter_computer-vision/bounding-box.html

[6] R. Yamashita, M. Nishio, R. K. G. Do, i K. Togashi, „Convolutional neural networks: an overview and application in radiology“, *Insights Imaging*, sv. 9, izd. 4, str. 611–629, kol. 2018, doi: 10.1007/s13244-018-0639-9.

[7] Y. LeCun, L. Bottou, Y. Bengio, i P. Ha, „Gradient-Based Learning Applied to Document Recognition“, 1998.

[8] A. Krizhevsky, I. Sutskever, i G. E. Hinton, „ImageNet classification with deep convolutional neural networks“, *Commun. ACM*, sv. 60, izd. 6, str. 84–90, svi. 2017, doi: 10.1145/3065386.

[9] E. Academy, „Kako aktivacijska funkcija ‚relu‘ filtrira vrijednosti u neuronskoj mreži? - EITCA akademija“, EITCA Academy. Pristupljeno: 09. srpanj 2024. [Na internetu]. Dostupno na: https://hr.eitca.org/umjetna-inteligencija/eitc-ai-tff-tensorflow-osnove/uvod-u-tensorflow/osnovni-ra%C4%8Dunalni-vid-s-ml/ispit-pregledni-osnovni-ra%C4%8Dunalni-vid-s-ml/kako-aktivacijska-funkcija-relu-filtrira-vrijednosti-u-neuronskoj-mre%C5%BEi/

[10] K. Simonyan i A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition“. arXiv, 10. travanj 2015. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/1409.1556

[11] C. Szegedy *i ostali*, „Going Deeper with Convolutions“. arXiv, 16. rujan 2014. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/1409.4842

[12] L. Alzubaidi *i ostali*, „Review of deep learning: concepts, CNN architectures, challenges, applications, future directions“, *J Big Data*, sv. 8, izd. 1, str. 53, ožu. 2021, doi: 10.1186/s40537-021-00444-8.

[13] P. Ratan, „What is the Convolutional Neural Network Architecture?“, Analytics Vidhya. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/

[14] R. Girshick, J. Donahue, T. Darrell, i J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation“. arXiv, 22. listopad 2014. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/1311.2524

[15] R. Gandhi, „What is R-CNN?“, Roboflow Blog. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: https://blog.roboflow.com/what-is-r-cnn/

[16] B. K, „Object Detection Algorithms and Libraries“, neptune.ai. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: https://neptune.ai/blog/object-detection-algorithms-and-libraries

[17] R. Girshick, „Fast R-CNN“. arXiv, 27. rujan 2015. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/1504.08083

[18] S. Ren, K. He, R. Girshick, i J. Sun, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“. arXiv, 06. siječanj 2016. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/1506.01497

[19] „Faster R-CNN: Objects Detection Without Slowness“, IndianTechWarrior. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: https://indiantechwarrior.com/faster-r-cnn-objects-detection-without-slowness/

[20] S.-H. Tsang, „Review: SSD — Single Shot Detector (Object Detection)“, Medium. Pristupljeno: 27. svibanj 2024. [Na internetu]. Dostupno na: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

[21] W. Liu *i ostali*, „SSD: Single Shot MultiBox Detector“, sv. 9905, 2016, str. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[22] J. Redmon, S. Divvala, R. Girshick, i A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection“. arXiv, 09. svibanj 2016. Pristupljeno: 27. svibanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/1506.02640

[23] M. Shenoda, „Real-time Object Detection: YOLOv1 Re-Implementation in PyTorch“. arXiv, 28. svibanj 2023. Pristupljeno: 14. lipanj 2024. [Na internetu]. Dostupno na: http://arxiv.org/abs/2305.17786

[24] H. Vedoveli, „Metrics Matter: A Deep Dive into Object Detection Evaluation“, Medium. Pristupljeno: 03. lipanj 2024. [Na internetu]. Dostupno na: https://medium.com/@henriquevedoveli/metrics-matter-a-deep-dive-into-object-detection-evaluation-ef01385ec62

[25] R. Padilla, S. Netto, i E. da Silva, *A Survey on Performance Metrics for Object-Detection Algorithms*. 2020. doi: 10.1109/IWSSIP48289.2020.

[26] A. Anwar, „What is Average Precision in Object Detection & Localization Algorithms and how to calculate it?“, Medium. Pristupljeno: 03. lipanj 2024. [Na internetu]. Dostupno na: https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b

[27] Manu, „COCO format , what and how“, Medium. Pristupljeno: 09. lipanj 2024. [Na internetu]. Dostupno na: https://medium.com/@manuktiwary/coco-format-what-and-how-5c7d22cf5301

[28] „About Flickr“. Pristupljeno: 09. lipanj 2024. [Na internetu]. Dostupno na: https://www.flickr.com/about

[29] „Roboflow: Computer vision tools for developers and enterprises“. Pristupljeno: 03. lipanj 2024. [Na internetu]. Dostupno na: https://roboflow.com/

[30] Divya, „TACO Dataset“. 2022. [Na internetu]. Dostupno na: https://universe.roboflow.com/divya-lzcld/taco-mqclx

[31] „Welcome to Python.org“, Python.org. Pristupljeno: 09. lipanj 2024. [Na internetu]. Dostupno na: https://www.python.org/

[32] Ultralytics, „Ultralytics“. Pristupljeno: 09. lipanj 2024. [Na internetu]. Dostupno na: https://docs.ultralytics.com/

[33] Melanie, „Google Colab: the power of the cloud for machine learning“, Data Science Courses | DataScientest. Pristupljeno: 09. lipanj 2024. [Na internetu]. Dostupno na: https://datascientest.com/en/google-colab-the-power-of-the-cloud-for-machine-learning

[34] Ultralytics, „Train“. Pristupljeno: 10. lipanj 2024. [Na internetu]. Dostupno na: https://docs.ultralytics.com/modes/train

[35] „What is a confusion matrix? | IBM“. Pristupljeno: 11. lipanj 2024. [Na internetu]. Dostupno na: https://www.ibm.com/topics/confusion-matrix

[36] „F_02 Understanding of Pooling Layers - EN“. Pristupljeno: 25. svibanj 2024. [Na internetu]. Dostupno na: https://wikidocs.net/165406

Sažetak

Detekcija i klasifikacija otpada važan je korak u rješavanju mnogih ekoloških problema. Ovaj rad proučava taj problem s ciljem da se istrenira vlastiti model koji će biti sposoban detektirati, a potom i klasificirati određeni otpad. Kroz rad objašnjen je pojam konvolucijskih neuronskih mreža, njihova struktura te princip rada. Prikazuju se najpoznatiji algoritmi za detekciju objekata te se pojašnjavaju metrike korištene za evaluaciju istreniranih modela. Objašnjava se TACO skup podataka i programska potpora potrebna za treniranje modela za detekciju objekata. Koristi se algoritam YOLO kako bi se modeli trenirali te se provodi trening na dva različita skupa podataka s istom namjerom. Na kraju rada, prikazuju se i analiziraju ostvareni rezultati.

Ključne riječi: Detekcija objekata, klasifikacija objekata, računalni vid, konvolucijske neuronske mreže, YOLO, TACO skup podataka

Summary

Detection and classification of waste is a very important step in addressing many environmental issues. This paper studies this problem with the aim of training a custom image detection and classification model. The concept of convolutional neural networks, their structure and how they work are explained in this thesis. Also, the thesis reviews the most well-known algorithms for object detection, as well as the metrics used for evaluating trained models. It described the TACO dataset and describes the software requirements used for training an object detection model. The YOLO algorithm is used to train the models, and training is performed on two different datasets with the same intent. At the end of the thesis, the achieved results are described and analyzed.

Keywords: Object detection, object classification, computer vision, convolutional neural networks, YOLO, TACO dataset