# Računalni vid u upravljanju rampom i identifikaciji vozila

**Majnarić, Luka**



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# ZAVRŠNI RAD

**Luka Majnarić**

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

# Računalni vid u upravljanju rampom i identifikaciji vozila

Mentori:

Prof. dr. sc. Dubravko Majetić, dipl. ing.

Student:

Luka Majnarić

Zagreb, 2023.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru, Dubravku Majetiću, na strpljenju, razumijevanju i potpori. Posebna zahvala gospođi Izidori Herold, bez nje ne bih ni imao priliku braniti ovaj rad.

Za kraj, hvala obitelji i prijateljima, osobito mami Vesni i teti Đurđici što su uvijek bile tu.

Luka Majnarić

SVEUČILIŠTE U ZAGREBU
**FAKULTET STROJARSTVA I BRODOGRADNJE**
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

| Sveučilište u Zagrebu<br>Fakultet strojarstva i brodogradnje | |
|---|---|
| Datum | Prilog |
| Klasa: **602 – 04 / 23 – 6 / 1** | |
| Ur.broj: **15 - 1703 - 23 -** | |

# ZAVRŠNI ZADATAK

Student: **Luka Majnarić** JMBAG: **0035208900**

Naslov rada na hrvatskom jeziku: **Računalni vid u upravljanju rampom i identifikaciji vozila**

Naslov rada na engleskom jeziku: **Computer vision in ramp control and vehicle identification**

Opis zadatka:

Obrada slike i prepoznavanje oblika sve je češći inženjerski zadatak. Jedna od primjena je i upravljanje rampom za autorizirani ulazak na parkiralište ili garažni prostor. Kada sustav prepozna registarsku pločicu ili neku drugu predviđenu oznaku na vozilu, rampa na ulazu se automatski otvara i propušta vozilo. Ovakvi su komercijalni sustavi poprilično skupi, pa se radom predviđa rješenje temeljeno na umjetnoj inteligenciji, a koje je dovoljno učinkovito i bitno jeftinije.

U radu treba načiniti sljedeće:

1. Opisati problem prepoznavanja registarske pločice vozila.
2. Kamerom identificirati vozilo uz procjenu pouzdanosti prepoznavanja.
3. Koristiti besplatne softverske biblioteke poput biblioteke OpenCV.
4. Predložiti hardversku realizaciju u povezivanju kamere, računala i rampe.
5. Razmotriti zaštitu sustava od zlouporabe.
6. Izvesti zaključke rada.
7. Navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

30. 11. 2022.

Zadatak zadao:

Prof. dr. sc. Dubravko Majetić

Datum predaje rada:

**1. rok:** 20. 2. 2023.
**2. rok (izvanredni):** 10. 7. 2023.
**3. rok:** 18. 9. 2023.

Predvideni datumi obrane:

**1. rok:** 27. 2. – 3. 3. 2023.
**2. rok (izvanredni):** 14. 7. 2023.
**3. rok:** 25. 9. – 29. 9. 2023.

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

# SADRŽAJ

# POPIS SLIKA

# POPIS TABLICA

# POPIS OZNAKA

| | |
|---|---|
| $W$ | Visina ili širina slike |
| $K$ | Veličina filtera/kernela |
| $S$ | Iskorak |
| $P$ | Ispunjavanje |
| $\omega_{m,n}$ | Težine konvolucijskog kernela |
| $x_{i,j}$ | Elementi 2D ulaza |
| $h_{i,j}$ | Jedinice dobivene konvolucijom |
| $a(x)$ | Aktivacijska funkcija |
| $\beta$ | Pristranost |
| $C_i$ | Broj ulaznih kanala |
| $C_o$ | Broj izlaznih kanala |

# SAŽETAK

U ovom radu se razmatra uporaba računalnog vida za očitavanje registarskih pločica vozila s ciljem primjene u sustavima koji koriste rampe. Trenira se model za prepoznavanje registarskih pločica, a tekst se očitava pomoću odvojenog modela. Koristi se prijenosno učenje na najnovijim modelima za detekciju objekata. Opisane su njihove arhitekture i načini na koje razni hiperparametri utječu na točnost prepoznavanja. Nakon izolacije registarske pločice, primjenjuje se niz metoda predobrade slike radi povećanja preciznosti prepoznavanja teksta. Odabir modela uzima u obzir rad u industriji, stoga je prioritet ravnoteža između brzine, točnosti i potrebne računalne snage. Predložena je hardverska implementacija cijelog procesa.

Ključne riječi: Računalni vid, Prijenosno učenje, Detekcija objekata, Optičko prepoznavanje znakova, Predobrada slika

## SUMMARY

This paper considers the use of computer vision for reading vehicle license plates with the aim of application in systems that use ramps. A license plate recognition model is trained, and the text is read using a separate model. It uses transfer learning on state-of-the-art object detection models. Their architectures and the ways in which various hyperparameters affect recognition accuracy are described. After isolating the license plate, several image preprocessing methods are applied to increase the accuracy of text recognition. As the selection of the model considers the industry applications, the priority is always the balance between speed, accuracy, and desired computing power. A hardware implementation of the entire process is also proposed.

Key words: Computer vision, Transfer learning, Object detection, Optical character recognition, Image preprocessing

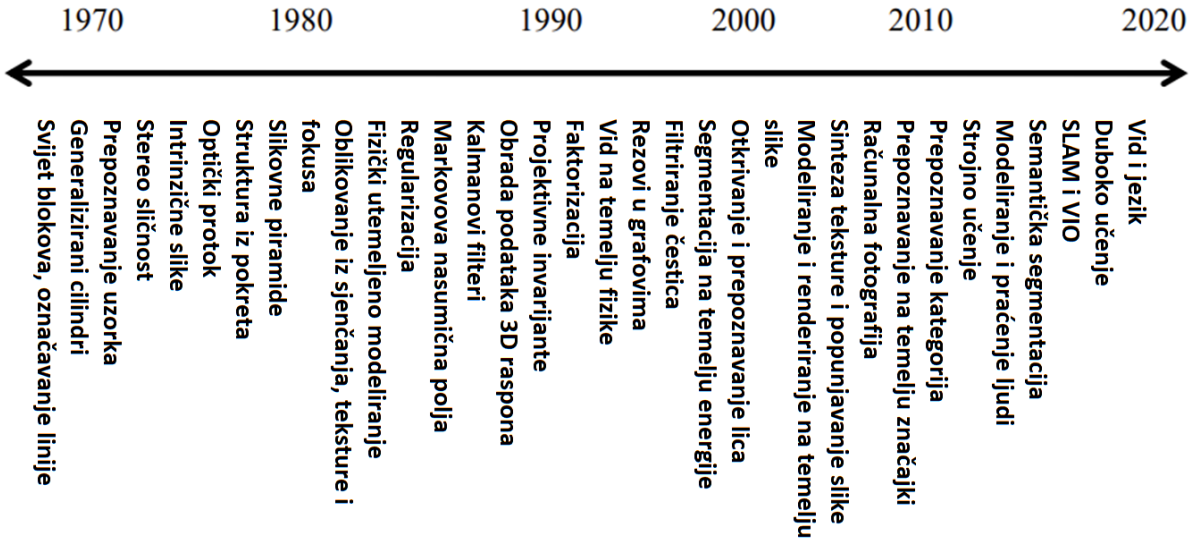
# 1. UVOD

Ono što potiče inženjere na pothvate poput ovakvog sustava je zamjena ili smanjenje monotonog rada, odnosno automatizacija. Skupa i nerijetko neupotrebljiva infrastruktura rampi u garažama i parkinzima ljudima gubi vrijeme, novac i živce. S obzirom na eksponencijalan napredak računalnih sustava u zadnjih nekoliko desetljeća otvara se mogućnost automatskih sustava koji zahtijevaju optičko prepoznavanje nekih segmenata slike.

Koliko god se prepoznavanje dijelova objekta normiranog poput registarske pločice vozila činilo jednostavnim, objašnjavanje tog procesa računalu i učenje tako zadanog zadatka je sa stanovišta pouzdanosti iznimno zahtjevno.

## 1.1. Računalni vid

### *1.1.1. Povijest*




**Slika 1. Najaktivnije teme računalnog vida kroz vrijeme**

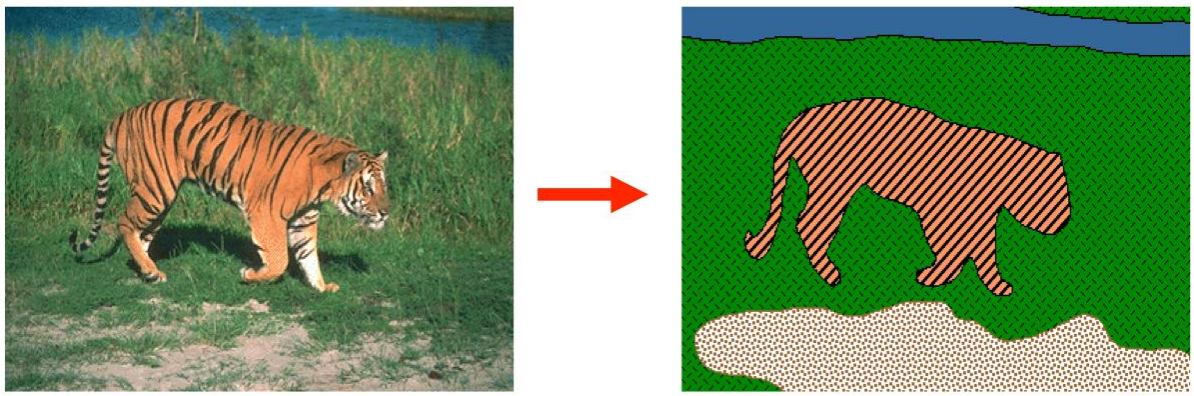
U počecima računalnog vida tijekom ranih 1970-ih, smatralo se vizualnom komponentom ambicioznoga cilja oponašanja ljudske inteligencije i podupiranje robota inteligentnim ponašanjem [1]. Također se smatralo da će to biti jednostavan korak na putu do rješavanja kompleksnih problema poput planiranja i razuma.

Prema priči iz 1966., Marvin Minsky (suosnivač laboratorija za umjetnu inteligenciju na MIT-u) je dao studentu preddiplomskog studija zadatak da " provede ljeto povezujući kameru na računalo te da omogući računalu da opiše što vidi" [2].

David Marr je u svojoj knjizi sažeo kako se tada smatralo da je vid funkcionirao [3]. Uveo je ideju koja sadrži 3 razine koje opisuju vizualni sustav za obradu informacija (okvirno opisano):

- Teorija računanja: Što je cilj izračuna i koja su poznata ili postavljena ograničenja kako bi se nosili s problemom?
- Prikazi i algoritmi: Kako su ulazne, izlazne i međuinformacije predstavljene te koji algoritmi se koriste za izračun željenog rezultata?
- Hardverska implementacija: Kako se prikazi i algoritmi implementiraju na stvarnom hardveru, npr. na specijaliziranom komadu silicija? S druge strane, kako hardverska ograničenja mogu utjecati na odabir prikaza i algoritama? Današnje grafičke kartice i višejezgrene arhitekture čine ovo pitanje izuzetno relevantnim.

Jedna od metoda upotrijebljenih u ovom radu je otkrivanje rubova i kontura [4] (engl. *edge and contour detection*). Mogla bi se također koristiti segmentacija slike (engl. *image segmentation*) [5].



**Slika 2. Segmentacija slike [6]**

### *1.1.2. Usporedba s biologijom*

Neki od najranijih algoritama učenja bili su računski modeli biološkog učenja (inspirirani procesima učenja u ljudskom mozgu). To je dovelo do naziva „umjetna neuronska mreža“ (engl. *Artificial Neural Networks*). Takvi inženjerski sustavi inspirirani su mozgom, bilo ljudskim ili životinjskim.

Najvažnija arhitektura umjetnih neuronskih mreža koja se koristi u ovom radu je konvolucijska neuronska mreža (engl. *convolutional neural network*) [7], koja će kasnije biti detaljnije opisana. Računalni vid se u hijerarhiji umjetne inteligencije smješta uz, ili ispod razine dubokog učenja.

Da bismo duboko shvatili algoritme koje mozak koristi, trebali bismo moći nadgledati aktivnost najmanje nekoliko tisuća međusobno povezanih neurona u isto vrijeme. Budući da to trenutno ne možemo, još smo daleko od razumijevanja čak i najjednostavnijih i dobro proučavanih dijelova mozga [8].

Neuroznanost nam je dala nadu da jedan algoritam za duboko učenje može riješiti mnoge različite zadatke. Neuroznanstvenici su otkrili da tvorovi mogu naučiti "vidjeti" pomoću slušnog dijela mozga ako im se mozak preusmjeri tako da šalje vizualne signale u to područje [9]. To sugerira da većina mozga sisavaca koristi sličan algoritam za rješavanje različitih problema. Prije ovog otkrića, istraživači dubokog učenja smatrali su svoje discipline kao odvojene probleme. Iako se i danas često tretiraju odvojeno, postoji sve veća povezanost između različitih disciplina istraživača. Područje dubokog učenja se prvenstveno bavi izgradnjom računalnih sustava koji uspješno rješavaju zadatke koji zahtijevaju inteligenciju, dok se područje računalne neuroznanosti usredotočuje na modeliranje načina na koji mozak stvarno funkcionira.

### *1.1.3. Napredak*

Jedna od temeljnih ideja razvijena tijekom 1980-ih, kada se era dubokog učenja nazivala „konekcionizam“ (engl. *connectionism*) [10], jest da velik broj jednostavnih računalnih jedinica može postići inteligentno ponašanje kada se međusobno povežu. Ovaj princip vrijedi kako za neurone u biološkim sustavima, tako i za skrivene slojeve u računalnim modelima.

Tijekom tog perioda razvijeno je nekoliko ključnih koncepata. Jedan od njih je i „podijeljeno prikazivanje“ (engl. *distributed representation*) [11], teza prema kojoj svaki ulaz u sustav treba biti prikazan s mnogo značajki, dok svaka značajka treba predstavljati različite ulaze. Na primjer, razmotrimo vizijski sustav koji prepoznaje automobile, autobuse i pse, gdje ti objekti mogu biti smeđi, bijeli ili crni. Jedan pristup prikazivanju ovih ulaza je da se koristi zaseban neuron, za svaku od 9 mogućih kombinacija, poput smeđeg automobila, smeđeg autobusa, smeđeg psa, bijelog automobila i tako dalje. To bi zahtijevalo 9 neurona gdje svaki od njih mora zasebno naučiti koncept objekta i boje. Efikasniji pristup je koristiti podijeljeno

prikazivanje: upotrebom 3 neurona za boje i 3 za objekte, ukupno 6 umjesto 9, svaki neuron za boju može učiti svoju boju iz slika automobila, autobusa i pasa, a ne samo iz slika jedne specifične kategorije objekata [12].

Uz eksponencijalni napredak računalnih sustava, kako hardverskih tako i softverskih, važno je naglasiti i povećanje dostupnih podataka. S povećanjem količine podataka, manja je potreba za vrhunskim sposobnostima kako bi algoritmi dubokog učenja postigli dobre performanse. Algoritmi učenja koji danas postižu ljudske razine na izazovnim zadacima su gotovo identični onima koji su se borili s rješavanjem osnovnih problema 1980-ih, iako su modeli koje treniramo s tim algoritmima prošli kroz promjene koje pojednostavljuju treniranje vrlo dubokih arhitektura.

Digitalizacija društva je dovela do porasta aktivnosti koje se odvijaju na računalima, rezultirajući time da je sve više naših radnji digitalno sačuvano. Kako su naša računala međusobno povezana, takve zapise je lako centralizirati u skupove podataka pogodne za strojno učenje. Na slici **Error! Reference source not found.** su navedeni neki od najpoznatijih i najčešće korištenih skupova podataka za treniranje modela računalnog vida.

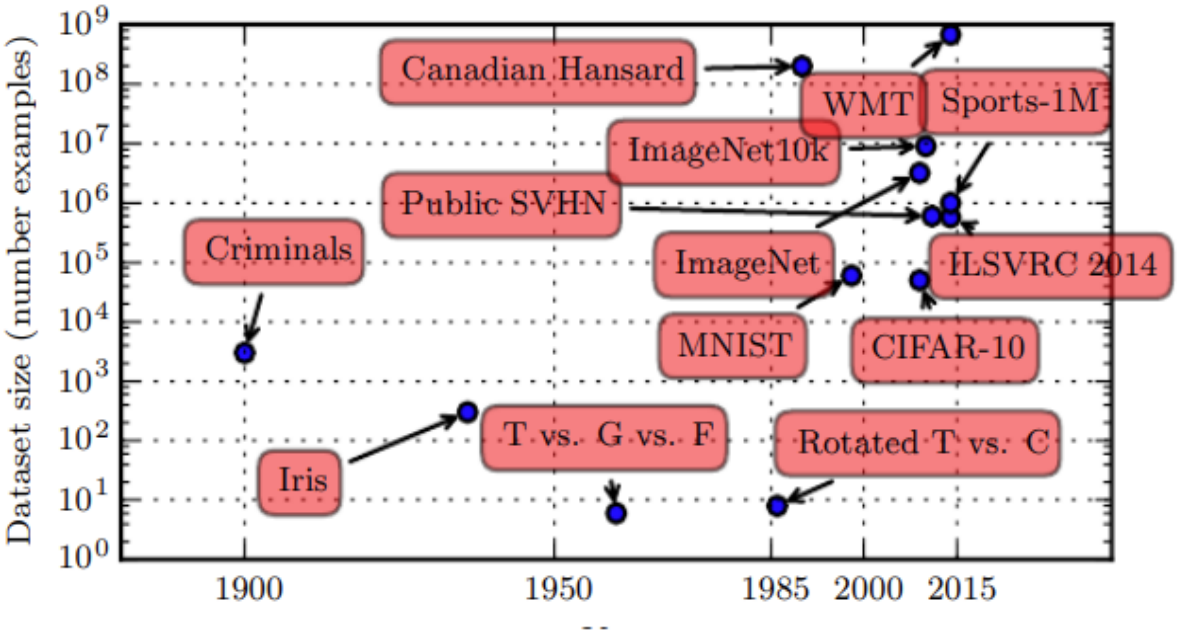
U kontekstu ukupne količine neurona, do nedavno su neuronske mreže bile iznenađujuće male. S uvođenjem skrivenih slojeva, kapacitet umjetnih neuronskih mreža udvostručivao se otprilike svake 2,4 godine. Veće mreže imaju potencijal postići veću točnost na kompleksnijim zadacima. Osim toga, moguće je da biološki neuroni obavljaju složenije funkcije od trenutnih umjetnih neurona. U retrospektivi, ne čudi što neuronske mreže s brojem neurona manjim od broja neurona pijavice nisu mogle rješavati sofisticirane probleme umjetne inteligencije [12].

**Tablica 1. Najveći skupovi podataka do 2022. godine za računalni vid**

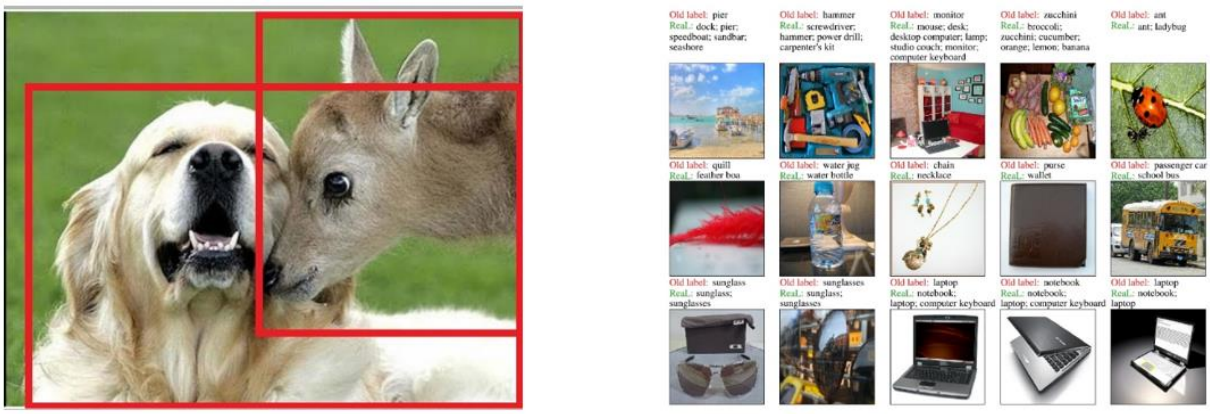
| *Ime skupa podataka* | *Broj slika (u milijunima)* | *Broj klasa* |
|---|---|---|
| *Tencent ML-Images* | 17,5 | 11 166 |
| *Open Images Dataset V7* | 9 | 20 638 |
| *ImageNet* | 14 | 21 841 |
| *IG-3.5B-17k* | 3500 | 17 000 |
| *JFT-3B* | 3000 | 30 000 |

Za razliku od prethodno navedenih skupova podataka (**Error! Reference source not found.**), koji su označavani pomoću okvira, skupovi navedeni u Tablica 1 označeni su tako da

se sadržaj slike opiše pomoću *synset*-a [13] ili na neki drugi sličan način opiše riječima. Drugi način označavanja bio bi praktičniji za klasifikaciju slika, slično MNIST skupu podataka. Na Slika 4 ilustrirani su navedeni načini označavanja podataka na slikama.




**Slika 3. Skupovi podataka kroz vrijeme [12]**




**Slika 4. Primjeri označavanja slika u navedenim bazama podataka za računalni vid: (a) pomoću okvira (na taj način su registarske pločice označavane u ovom radu) [14], (b) način označavanja koristeći jedan ili više sinonima (*synset*) [15]**

## 1.2. Računalni vid oko nas

Veliki skupovi podataka kombinirani s grafičkim karticama, kao i brza razmjena ideja putem arXiv-a (repozitorij za objavu znanstvenih radova) i otvorenog dijeljenja modela neuronskih mreža, doprinijeli su eksplozivnom napretku ovog područja. Specijalizirani senzori i hardver, poput Microsoft Kinect dubinske kamere (lansirane 2010.), ubrzo su postali ključne komponente raznih sustava za 3D modeliranje i praćenje ljudskih pokreta [16].

Iako senzori dubine još uvijek nisu u širokoj uporabi (osim za sigurnosne aplikacije na vrhunskim telefonima), algoritmi računalne fotografije postoje i djeluju u svim modernim pametnim telefonima. Panoramsko spajanje fotografija i uklanjanje šuma na slabom osvjetljenju samo su neki od primjera. Važno je istaknuti i mobilne aplikacije proširene stvarnosti (engl. *augmented reality*) koje izvode procjenu položaja u stvarnom vremenu i povećanje okoline koristeći kombinaciju praćenja značajki i inercijskih mjerenja. Inteligentna HDR (engl. *high dynamic range*) tehnologija automatski spaja više fotografija, pružajući slike s većim dinamičkim rasponom bez potrebe za dodatnom obradom.

Na sofisticiranijim platformama poput autonomnih vozila i bespilotnih letjelica, moćni algoritmi poput SLAM-a (**engl. *Simultaneous localization and mapping*** - istodobna lokalizacija i mapiranje) i VIO-a (engl. *Visual Inertial Odometry* - vizualna inercijalna odometrija) omogućuju izradu preciznih 3D mapa u stvarnom vremenu, primjerice, za autonomno letenje kroz izazovne okoliše poput šuma. Čak su i komercijalno dostupne bespilotne letjelice opremljene algoritmima za napredne sustave pomoći pilotu (*APAS - Advanced Pilot Assistance Systems*).

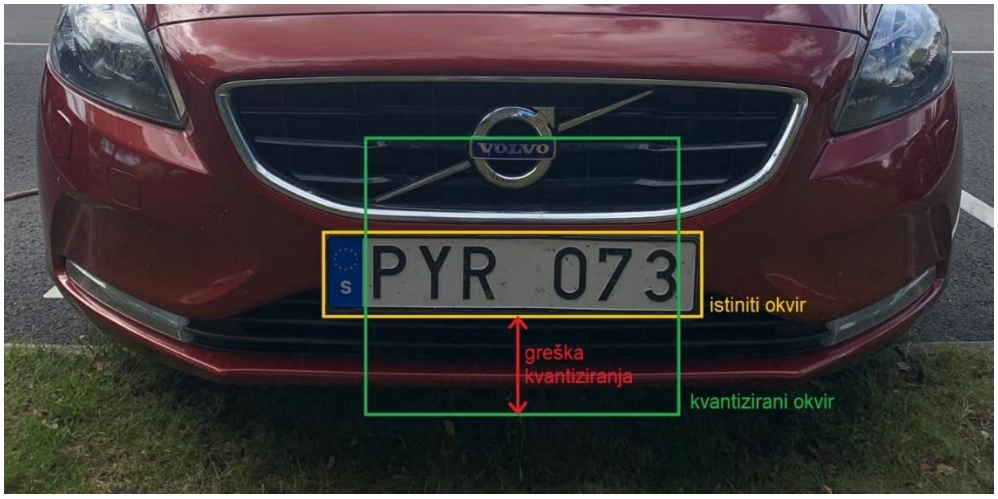
Međutim, sve je veća zabrinutost zbog etičkih posljedica primjene računalnog vida. Primjeri uključuju autonomne automobile koji su uzrokovali smrtne slučajeve zbog grešaka u percepciji, nadzorne tehnologije koje mogu postati sredstvo za gušenje slobode, te eksploataciju radnika s niskim primanjima za označavanje algoritama prepoznavanja lica koji su pokazali rasnu pristranost. Utvrđeno je da glavni skupovi podataka koji su desetljećima bili temelj istraživanja računalnog vida nenamjerno uključuju pristrane, rasističke i mizogine slike i oznake, što je dovelo do stotina tisuća uklonjenih slika sa prethodno navedenog ImageNet-a [17].

# 2. Prepoznavanje registracija

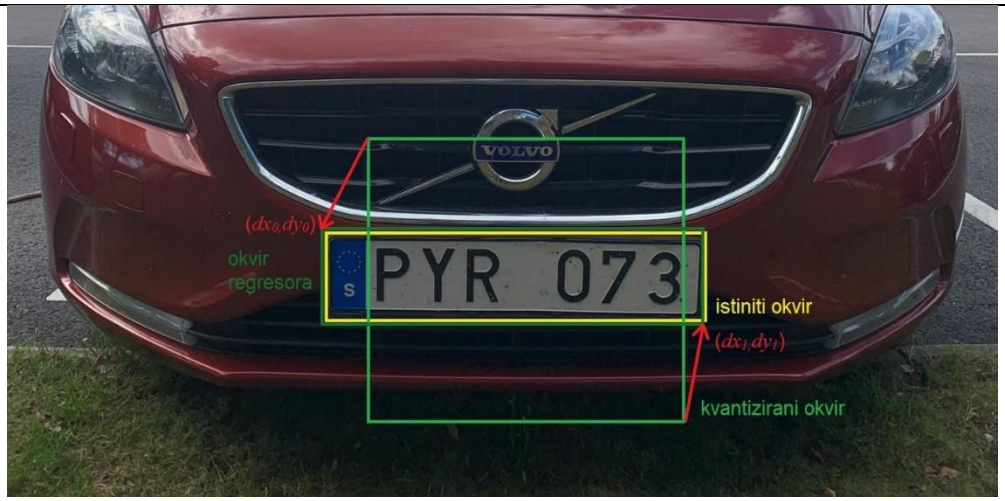
## 2.1. Prepoznavanje objekata kao problem strojnog učenja

Ideja je da razvijemo model s parametrima koji će se hraniti parovima ulazno-izlaznih podataka. Treba imati na umu da je to izbor i da je u suprotnosti s korištenjem tvrdo kodiranih (engl. *hard-coded*) pravila bez učenja. Ulaz je jednostavno slika, a izlaz je bilo koji konačni podskup mogućih okvira u slici. Treba imati na umu da je ovaj podskup tehnički beskonačan. Za svaki okvir u ovom podskupu detektor mora ispisati oznaku klase i razinu uvjerenosti da je to ta klasa.

Prvi problem je što imamo previše okvira za klasificiranje. Umjesto da radimo s ovim beskonačnim prostorom, aproksimirat ćemo ga s konačnim skupom okvira. Ovo možemo zamisliti kao kvantizaciju izvornog kontinuiranog prostora u skup reprezentativnih okvira. Posrednički problem (engl. *proxy problem*) sada uključuje klasificiranje svakog od kvantiziranih okvira umjesto izvornih okvira. Većina okvira u izvornom prostoru ne odgovara točno niti jednom kvantiziranom okviru. Dakle, čak i ako možemo uvježbati model da ih klasificira, odgovarajući na „što“ dio pitanja detekcije, lokalizacija ili „gdje“ dio bi mogao biti netočan. Kako bismo riješili ovaj gubitak točnosti lokalizacije (vidjeti Slika 5), možemo uvesti dodatni zadatak. Modelu možemo dodati regresor (uobičajeno skup značajki), kao što je prikazano na Slika 6, koji će biti osposobljen za predviđanje pogreške kvantizacije. Ako se može dobro istrenirati, regresor će transformirati netočno lokalizirani kvantizirani okvir u točno lokalizirani okvir.

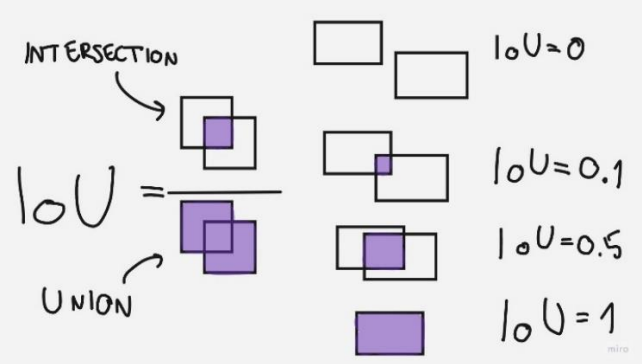



**Slika 5. Greška kvantiziranja na primjeru jednog okvira**

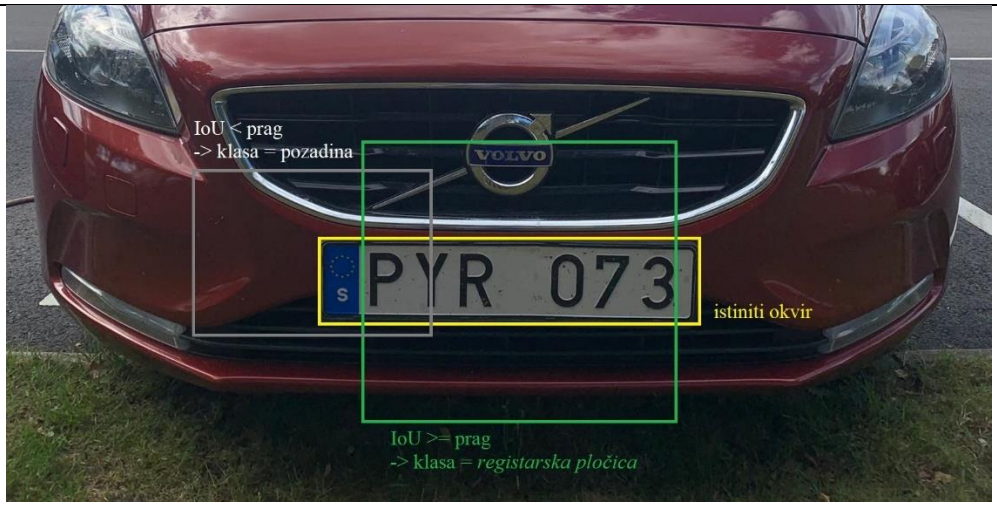



**Slika 6. Vizualizacija treniranja regresora**

Drugi problem je taj što je naše pravilo klasifikacije nedefinirano, pretpostavljamo da naš model može ispisati istiniti okvir (engl. *ground truth box* – temeljni okvir istine) - što je, prema prethodnom problemu, tehnički nemoguće. Moramo klasificirati svaki od kvantiziranih okvira, što zapravo nema smisla jer bi svi kvantizirani okviri trebali biti označeni kao pozadina (engl. *background*), osim ako jedan od njih ne odgovara točno osnovnom istinitom objektu. Stoga nikakvi objekti ne bi bili otkriveni. Moramo definirati pravilo klasifikacije na nešto što je razumnije. Navest ćemo pravilo dodjele klasa koje će dodijeliti oznaku prednjeg plana (engl. *foreground*) nekim od kvantiziranih okvira. Standardni pristup je definiranje heuristike označavanja na temelju različitih kriterija kao što su pragovi presjeka preko unije (IoU – engl. *Intersection over Union,* Slika 7) ili korištenje mjera centriranosti i uokviravanja (npr. kada je predviđeni okvir u potpunosti unutar istinitog okvira). Uobičajen primjer je, ako IoU između kvantiziranog i okvira istine premašuje prag, kvantizirani okvir može biti označen objektom u prvom planu, a ako je preniska, oznaka će biti postavljena na pozadinu, odnosno biti zanemarena.




**Slika 7. Presjek preko unije**

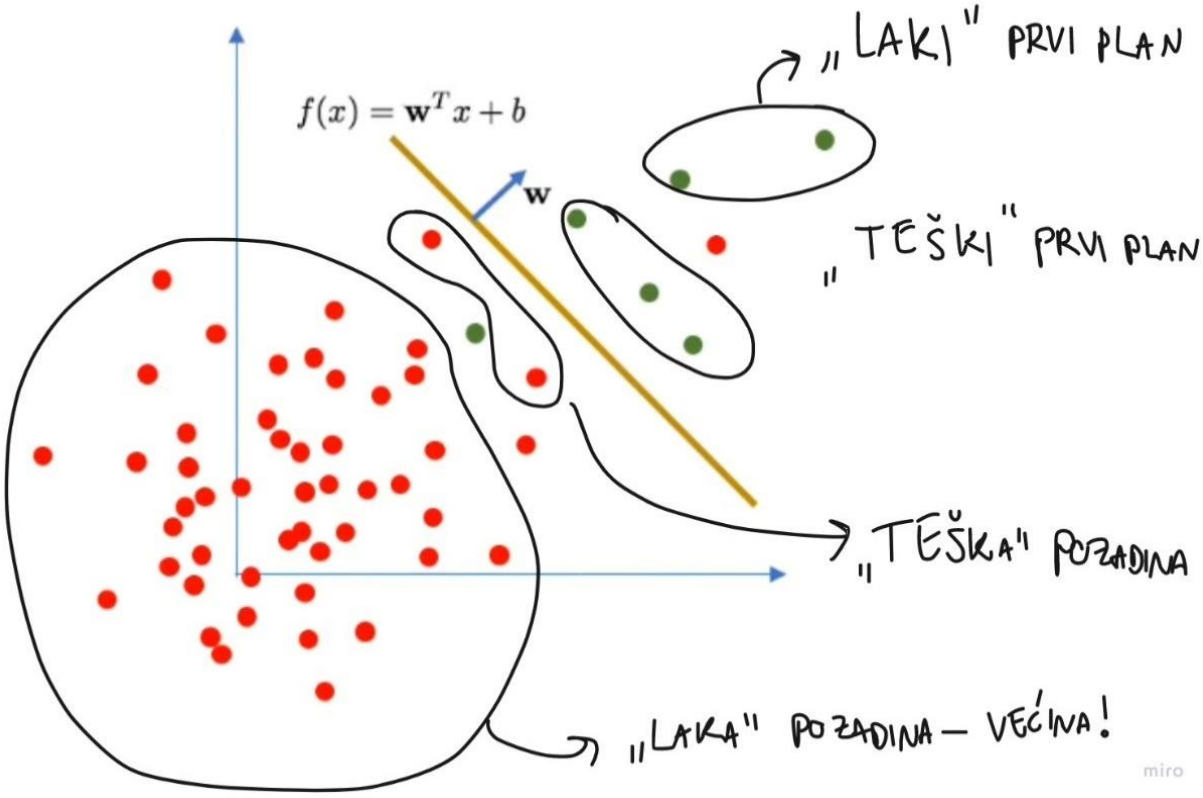



**Slika 8. Primjer presjeka preko unije gdje zadovoljava i ne zadovoljava prag**

Pretpostavimo da imamo okvir istine i dva kvantizirana okvira kojima je dodijeljena ista klasa kao zlatnom okviru prema pravilu dodjele klasa (vidjeti Slika 8). Oba predviđanja su „dobra“ i oba su slična jedno drugome. Ova vrsta redundantnog izlaza je upravo ono što smo tražili od modela. Za ovaj problem postoji standardno rješenje desetljećima staro. Okviri koji se preklapaju zamjenjuju se jednim predstavnikom njihovim grupiranjem. Postoje mnoge implementacije ovog algoritma grupiranja, a često uključuje pohlepno odabiranje okvira s najvećom pouzdanošću, dok se potiskuju otkrivanja s nižom pouzdanošću koja imaju visoko preklapanje. U današnje vrijeme, najčešće je to algoritam zvan ne-maksimalno potiskivanje (NMS – engl. *Non-Maximum Suppression*). U većini situacija, koristi se samo za evaluaciju (engl. *validation*) i predviđanje (odnosno zaključivanje – engl. *inference*), dok se tijekom treninga koristi ublažena verzija istog. Za razliku od praga prikazanog na slici 8, uobičajeni IoU prag je 0,7. Ovaj prag je posebno koristan kada imamo mnoštvo sličnih okvira te želimo izdvojiti najbolji među njima.

Naš posljednji problem nije izravno povezan s ostalima kao što su međusobno, a radi se o neravnoteži prednjeg i pozadinskog plana. Prije kvantizacije, u biti smo imali beskonačnu neravnotežu između oznaka (imali smo N broj istinitih okvira i beskonačno mnogo pozadinskih okvira). Nakon kvantizacije obično imamo oko 100 tisuća odluka (ovisno o rezoluciji ulazne slike) o klasifikaciji po slici (što je bolje nego beskonačno!), ali samo 0,01-0,1% su klase prvog plana. Znamo da, izvan konteksta prepoznavanja objekata, obuka klasifikatora na neuravnoteženim podacima može biti teška (čak i kada imamo samo jednu klasu, zapravo klasificiramo i pozadinu). Ukoliko se ne poduzmu odgovarajuće mjere, klasifikatori mogu jednostavno zanemariti manjinsku klasu, što može dovesti do gotovo 100%

točnosti klasifikacije, a istovremeno proizvesti užasan detektor objekata. Zatim, problem je brzina obrade. Sustav koji troši mnogo računalne snage na klasificiranje pozadinskih okvira može biti nepotrebno neučinkovit. Postoji nekoliko klasičnih rješenja, prvo se odnosi na neravnotežu putem funkcije gubitka koja se koristi za trening klasifikatora. Neslužbeno ćemo podatke podijeliti u skupine kao što je prikazano na Slika 9. Neslužbena podjela klasa Prvi su laki primjeri u prvom planu, slijede teški primjeri u prvom planu koji su blizu granice odluke ili čak na njezinoj pogrešnoj strani. Isto tako, imat ćemo teške i lake pozadinske primjere. Obično postoji ogroman broj lakih pozadina, odgovaraju dosadnim pozadinskim mrljama koje se nalaze posvuda po slikama, koje očito ne pripadaju niti jednoj kategoriji prednjeg plana.




**Slika 9. Neslužbena podjela klasa**

Često korištene funkcije gubitka, kao što je unakrsna entropija (engl. *cross-entropy*), osjetljive su na ovaj veliki oblak lake pozadine. To je zato što se gubitak povezan s tim velikim točkama margine ne približava nuli dovoljno brzo u odnosu na broj takvih podatkovnih točaka. Rezultat je da mogu imati neželjeni utjecaj na naučeni klasifikator što dovodi do loše izvedbe prepoznavanja. Na bolju funkciju gubitka neće utjecati ovaj oblak lakih primjera, već će umjesto toga prvenstveno utjecati teški primjeri. Općenito, ovo svojstvo

se naziva rudarenje teških primjera (engl. *hard example mining*) [18]. Jedan takav primjer funkcije gubitka s dobrim empirijskim rezultatima je žarišni gubitak (engl. *focal loss*) [19]. Da bi prikazali koliko funkcija gubitka utječe na konačnu evaluaciju, dizajnirali su vlastitu „jednostavnu“ mrežu zvanu RetinaNet. Njom su prikazali da jednostupanjski detektori mogu dostići (ili čak prestići) preciznost dvostupanjskih detektora, ali ne na temelju inovacija u dizajnu mreže, nego zbog nove funkcije gubitka.

Drugo rješenje je korištenje kaskade klasifikatora. Svaki stupanj u ovoj kaskadi obično je dizajniran da bude relativno brz, a može biti i relativno neprecizan. Rani stupnjevi podešeni su za visoko prisjećanje (engl. *recall*) tako da se pozitivni rezultati ne izgube, ali istovremeno uklanja veliki dio lažno pozitivnih (engl. *false positive*) rezultata. Bitno je napomenuti da predloženi okviri nisu nasumični, moguće je da skoro odgovaraju istinitim okvirima te na taj način odbacuju veliku većinu lakih negativnih primjera [20]. Postupno, to ublažava neravnotežu i u kasnijim fazama se vide uravnoteženiji podaci, dolazimo do 1-2 tisuće mogućih okvira od praktički beskonačno mnogo mogućih okvira. U prvoj se fazi odbacuje veliki broj lakih pozadinskih primjera dok zadržava sav prednji plan. Drugi stupanj sada ima uravnoteženiji problem, isto tako odbija mnoge pozadinske točke dok zadržava one u prvom planu. Ponavlja se sve dok konačni klasifikator ne radi na razumno uravnoteženom režimu podataka. Ovo klasično rješenje postalo je poznato u eri prije dubokog učenja 2001. godine [21]. U modernoj eri dubokog učenja, i dalje ima široku upotrebu. Primjeri uključuju R-CNN obitelj modela [22].

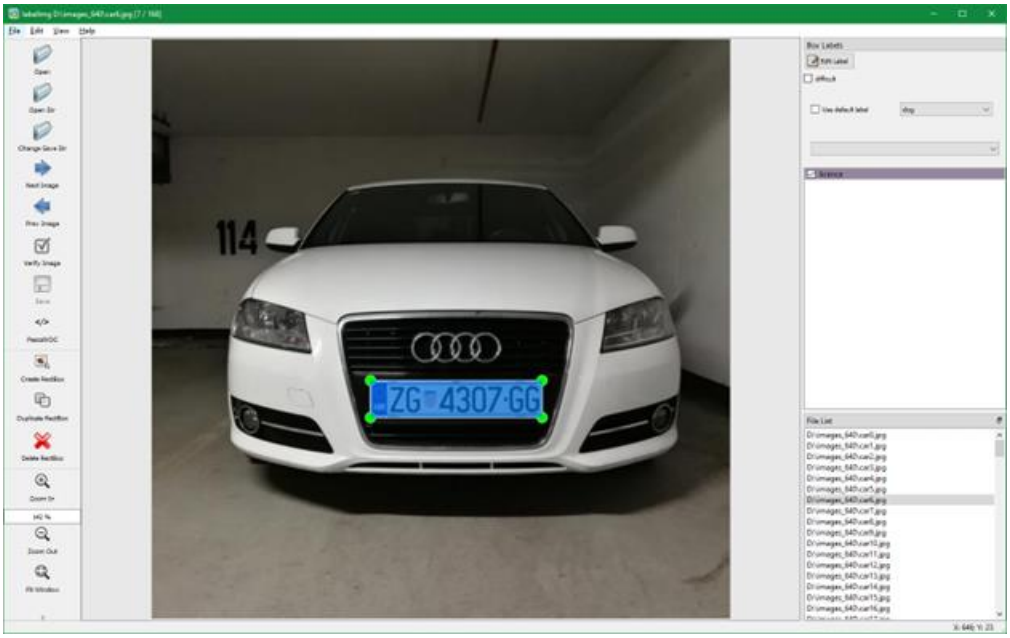
Treći pristup neravnoteži također je vrijedan spomena. U ovom pristupu kvantizacija je izuzetno agresivna tako da prema dizajnu mreže se ne dobije mnogo izlaznih okvira. Za prikazivanje konkretnog primjera, u prvoj verziji YOLO arhitekture [23] dobije se 98 izlaznih okvira u slučaju prepoznavanja jedne klase (zadnji sloj mreže je 7×7×2, gdje je 2 naša klasa i pozadina). S nekoliko izlaznih okvira stupanj neravnoteže ograničen je dizajnom mreže.

## 2.2. Priprema podataka

Da bismo trenirali model za određenu svrhu, prvo nam trebaju obilježeni podaci. U kontekstu računalnog vida, radi se o slikama - specifično slikama automobila s vidljivom registarskom pločicom, snimljenim pod kutom koji nije previše izražen (kako ne bi bilo problema s prepoznavanjem znamenaka/teksta).

Za učinkovito treniranje modela ključno je da podaci koji se koriste za treniranje budu što sličniji onima s kojima će model kasnije raditi u stvarnom svijetu. No, nije dovoljno samo

uslikati fotografije i smjestiti ih u određene datoteke. Za probleme detekcije objekata, poput ovoga, potrebno je obilježiti klase predmeta koje planiramo otkriti. Upotrebom Python biblioteke LabelImg označavamo vlastite fotografije pravokutnim okvirima, kao što je prikazano na Slika 10.




**Slika 10. Primjer označavanja u Python biblioteci LabelImg**

Različiti modeli koriste različite formate obilježavanja podataka. Na primjer, modeli bazirani na TensorFlow-u (Google-ovi modeli) koriste Pascal VOC (engl. *Visual Object Classes*) format, prikazan na Slika 11. S druge strane, YOLO modeli koriste YOLO (DarkNet) format, koji je naizgled jednostavniji jer koristi samo broj klase, normalizirane koordinate centra okvira, njegovu širinu i visinu. Normalizacijom podataka zapisa okvira se postiže lakša i brža pretvorba ulaznih podataka u različite veličine. Postoje raznorazne skripte koje nam omogućuju brzo pretvaranje podataka iz jednog formata u drugi, a i web aplikacije koje nam daju mogućnosti odabira u kojem formatu da označavamo podatke (npr. RoboFlow). U ovom slučaju postoji samo jedan objekt (registarska pločica) i zapisane su koordinate gornjeg lijevog i donjeg desnog ugla predstavljene pikselima.

```xml
<annotation>
    <folder>images_640</folder>
    <filename>car6.jpg</filename>
    <path>D:\images_640\car6.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>640</width>
        <height>640</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>licence</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>242</xmin>
            <ymin>377</ymin>
            <xmax>432</xmax>
            <ymax>418</ymax>
        </bndbox>
    </object>
</annotation>
```

**Slika 11. Izgled Pascal VOC formata za konkretan primjer s prethodne slike**

U računalnom vidu, često se preporučuje da slike budu kvadratne kako bi se izbjeglo potrebno obrezivanje tijekom predobrade. Uobičajena rezolucija za treniranje modela ovisi o korištenom skupu podataka. Na primjer, VGG16 model trenira se na ImageNet skupu podataka koji sadrži slike rezolucije 224×224.

Pri finom podešavanju tijekom prijenosnog učenja, mijenjamo ili dodajemo neke hiperparametre u konfiguracijskoj datoteci koja dolazi s modelom. Unutar njega, između ostalog, postoje i parametri za povećanje podataka (engl. *dana augmentation*) poput ovih navedenih na Slika 12.

```
  data_augmentation_options {
    random_crop_image {
      min_aspect_ratio: 0.5
      max_aspect_ratio: 1.7
      random_coef: 0.25
    }
  }


  data_augmentation_options {
    random_adjust_hue {
    }
  }

  data_augmentation_options {
    random_adjust_contrast {
    }
  }

  data_augmentation_options {
    random_adjust_saturation {
    }
  }

  data_augmentation_options {
    random_adjust_brightness {
    }
  }
```

**Slika 12. Razne opcije povećanja podataka u modelima baziranim na TensorFlow biblioteci**

Važno je napomenuti da, iako je augmentacija korisna, treba biti pažljiv s odabirom metoda augmentacije. Na primjer, rotacija od 90 stupnjeva nije primjerena za naše slike automobila, budući da takve slike u stvarnom svijetu nećemo često susretati.

Kako je poznato, model ne obrađuje zapravo slike kao što ih mi vidimo, već nekakve brojeve raspoređene u tenzore. U suštini, neuronske mreže neovisno o količini podataka konvergiraju gradijente na iste načine, razlika je u brzini pronalaženja tih gradijenata – upravo to postižemo raspoređivanjem brojeva, odnosno podataka, u tenzore. Naši tenzori su 4D, gdje su dimenzije (različito raspoređene ovisno o modelu):

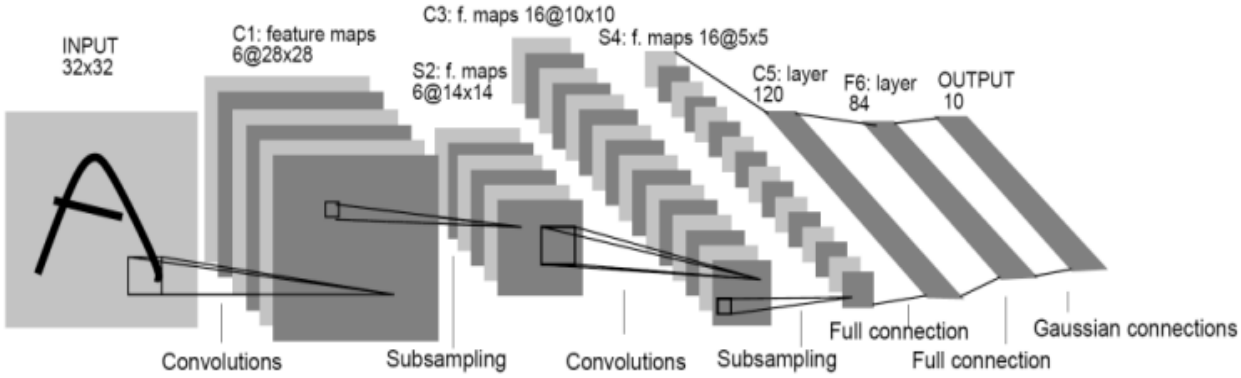
- Veličina serije (engl. *batch size*) - uobičajene vrijednosti su od 1 do 256, najčešće potencije broja 2.
- Visina slike - odgovara broju redaka piksela u slici ili prvom broju rezolucije.
- Širina slike - odgovara broju stupaca piksela ili drugom broju rezolucije.
- Broj kanala - ovisno o tome je li slika u boji (3 kanala, npr. RGB – engl. *Red Green Blue*) ili crno-bijela (engl. *grayscale* – 1 kanal).

U ovom kontekstu, brojevi unutar tenzora obično variraju između 0 i 255, kada koristimo 8-bitnu dubinu boje. Često se koristi sloj unutar modela koji normalizira ove vrijednosti, odnosno normalizacija u rasponu od 0 do 1 (ili -1 do 1, ovisno o modelu). Bez normalizacije, slike koje imaju širok raspon boja imaju veći utjecaj na promjenu parametara modela od onih s manjim rasponom.

Prije početka treniranja modela, ako koristimo TensorFlow bazirane modele, potrebno je zapisati sve podatke u .record datoteke s ciljem standardizacije ulaznih podataka i optimiziranja izvođenja [24], dok YOLO modeli, bazirani na PyTorchu, koriste običan tekstualni zapis. Datoteke se i u ovom slučaju odvajaju na skupove treninga, evaluacije i testiranja.

## 2.3. Konvolucijske neuronske mreže

Najbitniji gradivni blok korištenih modela za računalni vid je višeslojna konvolucija. Ideja je popularizirana u [25], gdje je predstavljena LeNet-5 mreža za prepoznavanje, odnosno klasifikaciju (jer prikazuje jedino znamenke i ništa drugo) znamenaka.




**Slika 13. Arhitektura LeNet-5 konvolucijske neuronske mreže za prepoznavanje znamenaka [25]**

Umjesto povezivanja svake jedinice u sloju sa svakom jedinicom u prethodnom sloju, kao što to rade „obični" slojevi neuronskih mreža, konvolucijske mreže organiziraju svaki sloj u mape značajki (engl. *feature maps*), koji se može zamisliti kao paralelne ravnine kao što je prikazano na Slika 13.

U Keras biblioteci koju koristi TensorFlow, pri kreiranju CNN-a funkcija Conv2D uzima nekoliko argumenata koje valja spomenuti.

- Filter. Broj mapa značajki, odnosno koliko puta će se konvolucija odviti na ulaz.
- Veličina kernela. Kvadratna matrica s neparnim brojem redaka i stupaca, vrijednosti se kreću najčešće od 3×3 do 11×11.
- Iskoraci (engl. *strides*). Broj ili par brojeva koji određuje za koliko koraka će se kernel pomaknuti po ulazu, po visini i širini, najčešće vrijednosti su od 1 do veličine kernela.
- Ispunjavanje (engl. *padding*). Povećanje dimenzija ulaza po visini i širini s vrijednostima 0, sa svih strana, s ciljem da pikseli na rubovima ulaznih slika ne budu parcijalno procesirani.

  Primjer s gornje slike, uzmemo li da nam je veličina ulazne slike $W \times W$, filter veličine $K \times K$, iskorak $S$ i ispunjavanje $P$, slijedi:

$$Veličina\ izlaza = \frac{W - K + 2P}{S} + 1 \qquad \textbf{(2.1)}$$

Odnosno na primjeru sa slike gore, za dimenziju slike nakon prolaska kroz prvu konvolucijsku mrežu:

$$28 = \frac{32 - 5 + 2 * 0}{1} + 1 \qquad \textbf{(2.2)}$$

Prilikom izgradnje složenih arhitektura, poput onih o kojima se raspravlja u ovom radu, korištenje konvolucijskih slojeva uključuje više od konvolviranja preko jednostavnog dvodimenzionalnog ulaza. S obzirom da je naš ulaz obično slika u RGB formatu, tretira se kao 2D signal s tri kanala. Kada se $K \times K$ kernel primijeni na ovaj 2D ulaz koji se sastoji od elemenata $x_{i,j}$, on izračunava jedan sloj jedinica $h_{i,j}$ kao:

$$h_{i,j} = a\left(\beta + \sum_{m=1}^{K}\sum_{n=1}^{K} \omega_{m,n}\, x_{i+m-2,j+n-2}\right) \qquad \textbf{(2.3)}$$

Ovdje su $\omega_{m,n}$ unosi konvolucijskog kernela. U biti, ovo predstavlja težinski zbroj preko $K \times K$ ulaznog područja.

Postoje slučajevi kada je primarni cilj promijeniti broj kanala između slojeva bez mijenjanja drugih svojstava. Glavni primjer je „usko grlo“ (engl. *bottleneck*) opisan u ResNet radu [26]. Da bismo to postigli, koristimo konvoluciju s veličinom kernela 1×1. Dimenzije konvolucijskog filtra bile bi $1 \times 1 \times C_i \times C_o$, gdje $C_i$ predstavlja broj ulaznih kanala, a $C_o$ označava broj izlaznih kanala.

Da bismo bolje razumjeli takvu konvoluciju, uzmimo navedeni ResNet primjer, gdje se broj kanala mijenja s 256 na 64 pomoću 1×1 filtra. Vizualiziramo jedan 1×1×256 filter kao dugu iglu koja ulazi u mapu značajki ulaza kako bismo uzorkovali svih 256 kanala na određenoj lokaciji (jedan od ulaznih piksela). Kada ova „igla“ uzme skalarni produkt s ulazom (koji je također 1×1×256 u tom pikselu), daje skalarnu vrijednost. Budući da postoje 64 takve „igle“, svaka s različitim težinama, svaka od njih generira jedinstvenu skalarnu vrijednost kada se skalarno množi s ulazom. Nakon što se rasporede, proizvode 1×1×64 izlaz za tu određenu lokaciju, na taj način dolazimo od $W \times W \times C_i$ do $W \times W \times C_o$.

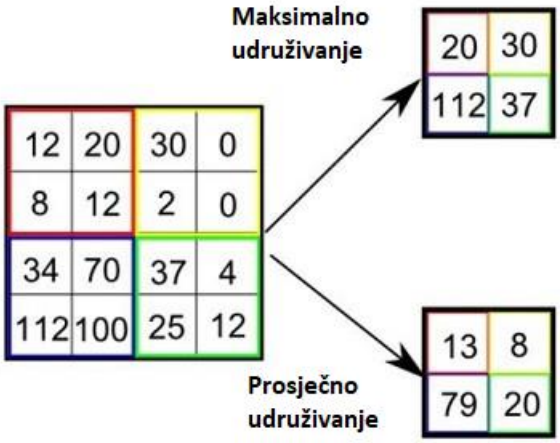
Da bi se vizualiziralo što se događa sa slikom koja prolazi kroz neki model, može se predočiti mape značajki kroz pojedine slojeve u modelima. Jedan od starijih i jednostavnijih modela, tadašnji pobjednik ILSVRC-a (engl. *ImageNet Large Scale Visual Recognition Challenge*) 2014. godine, VGG16 (od *Oxford Visual Geometry Group*), se može vizualizirati

pomoću Keras biblioteke i vlastite slike. Uz navedeni proces valja dodati i sažetak arhitekture modela koji je prikazan na Slika 14. Prikazano je 16 slojeva konvolucija i 3 potpuno povezana sloja (engl. *fully connected layers*) koji sadrže 138 milijuna parametara za treniranje.

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_16 (InputLayer)       [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

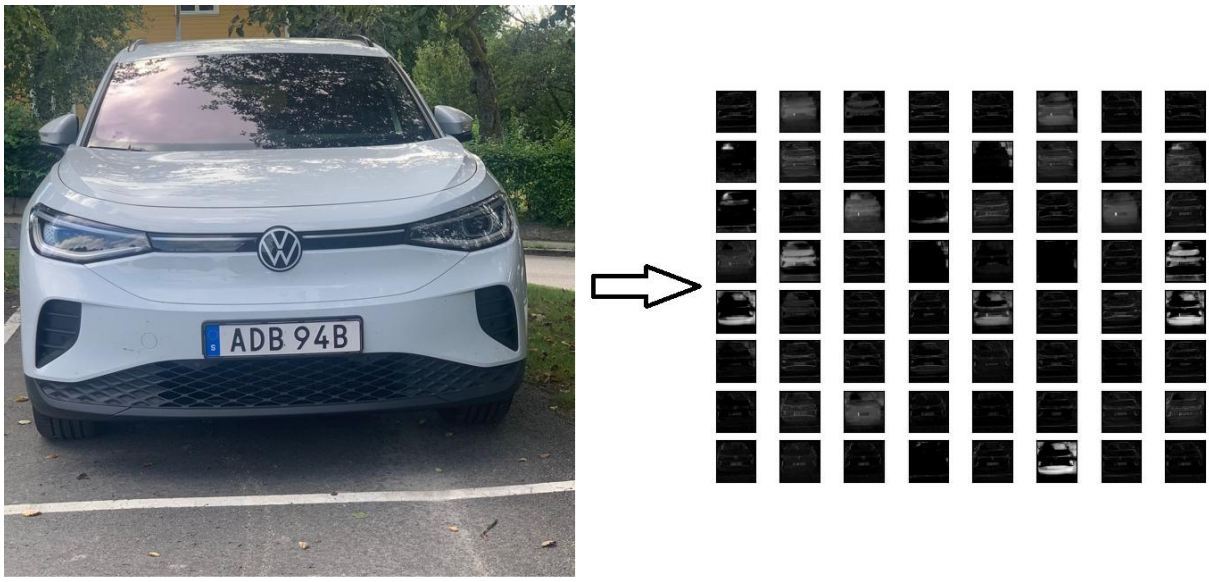
**Slika 14. Sažetak VGG16 modela**

Između nekih konvolucijskih slojeva koriste se slojevi za udruživanje (engl. *pooling*), u ovom slučaju maksimalno (koristi se i prosječno udruživanje, samo rjeđe u slojevima koji nisu pri kraju modela). U ovom modelu udruživanje se koristi za smanjenje visine i širine slike dok prolazi kroz mrežu, konkretno maksimalno udruživanje s filterom 2×2 i iskorakom 2, da bi se dobile duplo manje dimenzije. Primjer dvije navedene vrste udruživanja je prikazan na slici dolje.




**Slika 15. Maksimalno i prosječno udruživanje s iskorakom 2 i 2×2 filterom**

Za pregled značajki nakon prvog sloja konvolucijske mreže dobivamo mapu značajke dubine 64, sličica veličine 224×224 (jednakih ulazu) prikazanu pomoću biblioteke matplotlib na slici ispod za jednu od vlastitih fotografija automobila. Prije ulaska u mrežu, slika je smanjena na rezoluciju 224×224.

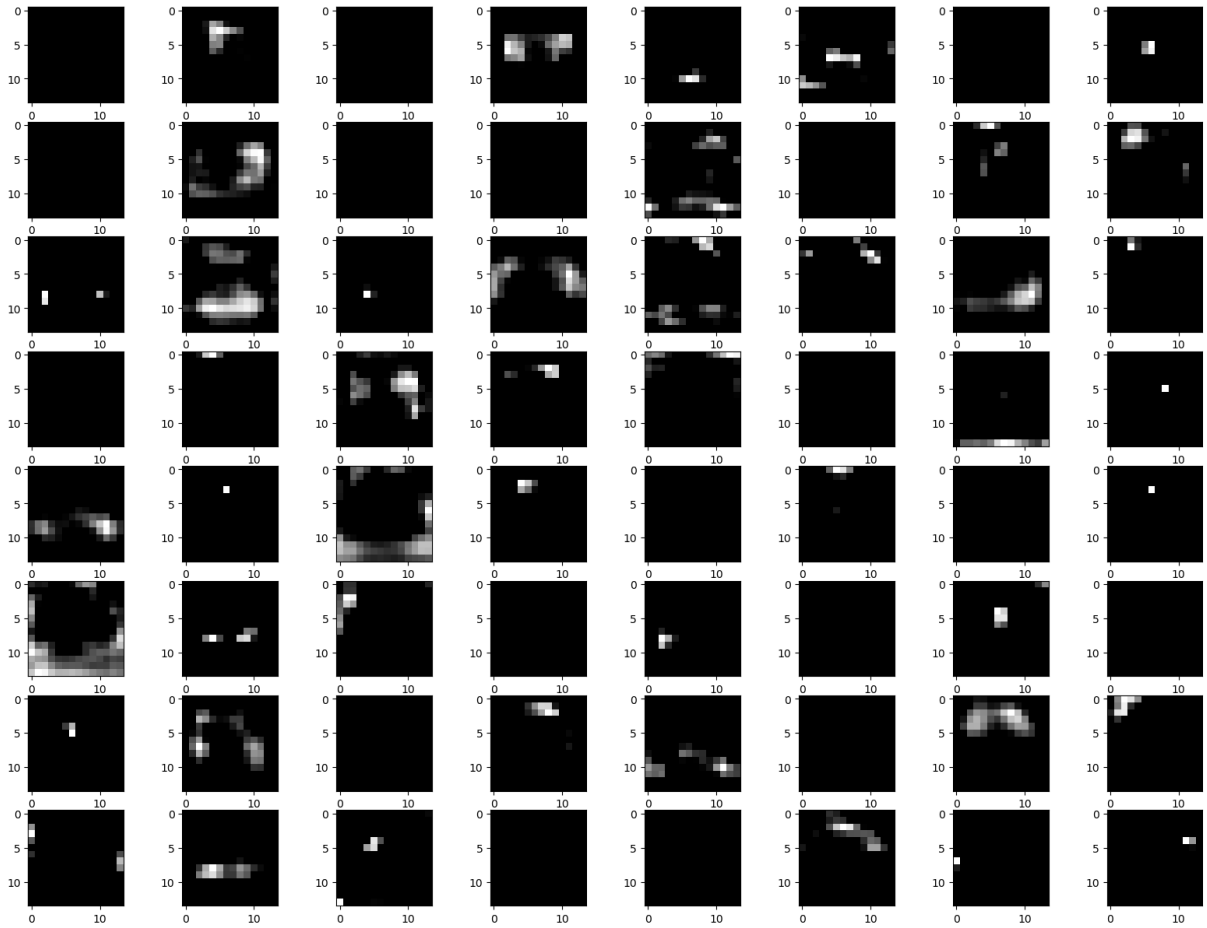



**Slika 16. Primjer prolaza kroz prvi sloj VGG16**

Kada se dublje analizira rad neuronske mreže, konkretnije slojevi prije smanjenja rezolucije ulaznih podataka, može se vidjeti kako model pokušava izdvojiti poneke značajke. Konkretno model VGG16 bi se u ovom slučaju koristio za klasifikaciju slika. Što se ide dublje u značajke, to one postaju niže rezolucije i same značajke postaju veće. Na nekim mapama značajki može se vidjeti izolacija specifičnih dijelova automobila, poput

vjetrobranskog stakla, područka oko registarske pločice i pozadine fotografije. Na Slika 17 prikazane su mape značajki iz posljednjeg konvolucijskog sloja prije udruživanja, što znači da su ove mape veličine 14×14 i nisu prikazane sve jer ih je 512.

BLOCK_17



**Slika 17. Izlaz iz zadnje konvolucijske mreže modela VGG16, prikazane su 64 mape značajki**

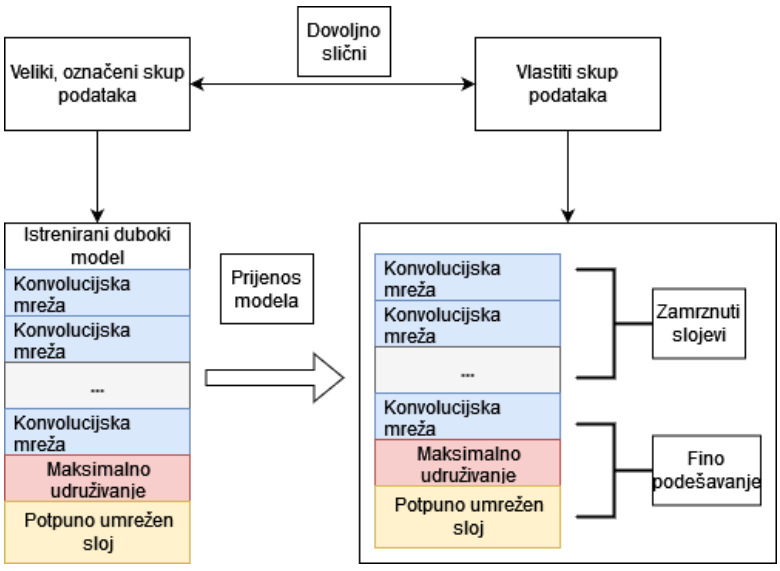
## 2.4. Metode

Suvremeni modeli korišteni u modernim sustavima računalnog vida koriste i omogućuju nam prijenosno učenje (engl. *transfer learning*), stoga primjenjujemo takav pristup pri ovom zadatku. Razlog zbog kojeg ne pravimo vlastite modele je manjak znanja i računalne infrastrukture. Korišteni su primarno modeli iz YOLO obitelji, konkretno verzije 7 [27] i 8 [28], te EfficientDet obitelj [29] koja nije detaljno objašnjena. U ovom kontekstu, pojam obitelj se koristi za varijacije u veličinama mreža i veličini ulaznih podataka, odnosno slika. Oni su već trenirani na COCO 2017 skupu podataka koji sadrži 330 tisuća slika, 1,5 milijuna objekata raspodijeljenih u 80 klasa. To što su već trenirani znači da su im težine i

pristranosti (engl. *weights and biases*) optimizirani za, u ovom slučaju, otkrivanje, klasifikaciju i segmentaciju objekata. Općenito, arhitekture modela za prepoznavanje objekata možemo razdvojiti na kičmu (engl. *backbone*) i glavu (engl. *head*), dok poneki autori dodaju između njih i vrat (engl. *neck*).

### *2.4.1. Prijenosno učenje*

Prijenosno učenje je tehnika u strojnom učenju koja omogućuje da se model obučen na jednom zadatku prilagodi i koristi za drugi, sličan zadatak. Ideja koja stoji iza prijenosa učenja jest iskoristiti znanje koje je model stekao u jednom zadatku i primijeniti ga na drugi zadatak, umjesto treniranja novog modela od nule (Slika 18). Uzimamo težine iz već trenirane (engl. *pre-trained*) kičme nekog drugog modela, koja je pri treniranju na velikom skupu podataka naučila raspoznati općenite značajke poput rubova, tekstura i boja. Glava modela na početku treninga ima nasumične težine u mreži jer se prijenos predviđanja okvira generalno ne prenosi dobro s jednog modela na drugi. Ovisno o veličini vlastitog skupa podataka, odabiremo vrstu finog podešavanja (engl. *fine-tuning*) odabranog modela. U slučaju malog skupa podataka (<1000 slika), kičma ostaje zamrznuta (engl. *frozen*), odnosno ne trenira se, stoga se trenira samo glava detektora. S druge strane, ako je skup podataka donekle velik (10000+ slika), nije nužno zamrznuti kičmu modela. Završni slojevi glave prilagođavaju se kako bi uzeli u obzir drugačiji izlazni broj klasa.




**Slika 18. Prijenosno učenje**

Tijekom finog podešavanja, uobičajeno je koristiti manju stopu učenja (engl. *learning rate*) za kičmu, a veće za glavu, jer drastična ažuriranja prethodno treniranih težina mogu uništiti prepoznavanje korisnih značajki. Postoji više vrsta finog podešavanja, poput ranije spomenutog smrzavanja kičme i treniranja isključivo glave modela, treniranja cijelog modela ili čak postepeno odmrzavanje slojeva počevši od zadnjih nekoliko slojeva. Kriteriji za odabir su veličina skupa podataka, sličnost skupu na kojem je model već treniran te računalni resursi kojima raspolažemo. U ovom radu upotrebljavamo treniranje cijelog modela s progresivno manjom stopom učenja.

### *2.4.2. Načini evaluacije modela*

Da bismo razumjeli rezultate treninga, a i što nam sve ove evaluacije govore tijekom istog, bitno je navesti načine na koje se kvaliteta modela procjenjuje.

#### *2.4.2.1. Osnovni načini*

Preciznost (engl. *precision*) je mjera koja nam govori koliko je detekcija modela istinito u odnosu na sve pozitivne detekcije. Dakle, to je broj istinitih pozitivnih (engl. *true positive*) detekcija podijeljeno zbrojem istinitih pozitivnih i lažnih pozitivnih (engl. *false positive*) detekcija. Na primjer, ako prepoznamo 8 trulih jabuka, od kojih je 5 zapravo trulo, a 3 nisu, iz navedene formule dobijemo vrijednost preciznosti od 5/8.

$$Preciznost = \frac{Istiniti\ Pozitivni}{Istiniti\ Pozitivni + Lažni\ Pozitivni} \quad \textbf{(2.4)}$$

Prisjećanje (engl. *recall*) mjeri koliko je istinitih pozitivnih slučajeva model uspio prepoznati. To je broj istinitih pozitivnih rezultata podijeljen zbrojem istinitih pozitivnih i lažno negativnih detekcija. Za primjer uzmimo da imamo 10 trulih jabuka, a točno smo prepoznali 5 trulih jabuka, a 5 nismo, što nam daje prisjećanje od 5/10. Ova metoda je itekako bitna gdje lažna negativna greška ima posljedice. Recimo, u medicinskim testovima, lažna negativna detekcija (neuspjeh identifikacije bolesti kada je prisutna) nosi više težine nego lažna negativna detekcija (identificiranje bolesti kada je nema).

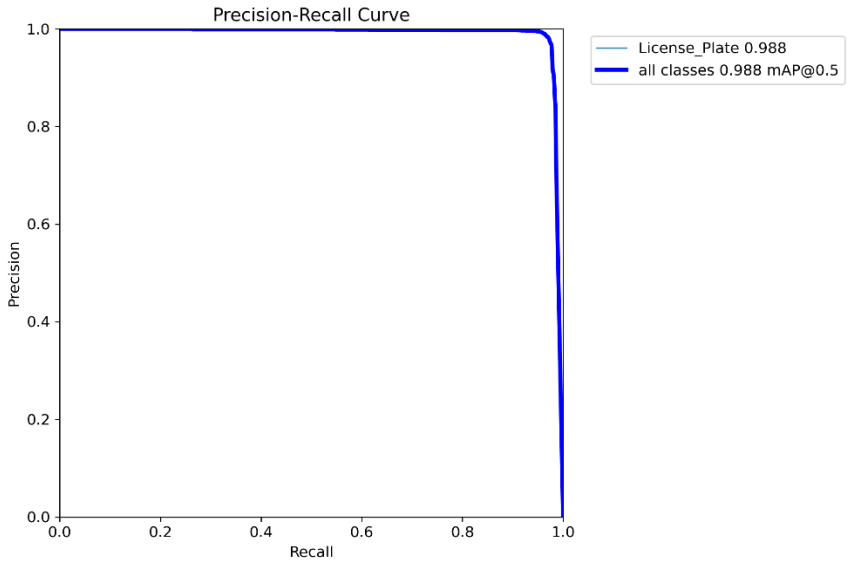
$$Prisjećanje = \frac{Istiniti\ Pozitivni}{Istiniti\ Pozitivni + Lažni\ Negativni} \quad \textbf{(2.5)}$$

F1 rezultat (engl. *F1 score*) je harmonijska sredina preciznosti i prisjećanja. Koristi se kada su obje prethodne metrike važne. Prikazuje se ovom jednadžbom:

$$F_1 = 2 * \frac{Preciznost * Prisjećanje}{Preciznost + Prisjećanje} \quad \textbf{(2.6)}$$

### *2.4.2.2. Načini kod prepoznavanja objekata*

Prosječna preciznost (AP – engl. *Average Precision*) u ovom kontekstu je površina ispod grafa preciznost-prisjećanje. Uobičajeno je da joj se dodaje IoU prag pri kojem se mjere preciznost i prisjećanje. Na slici ispod je prikazan vrlo dobar primjer jedne takve krivulje.




**Slika 19. Preciznost - Prisjećanje krivulja na modelu treniranom za detekciju registarskih pločica**

Prosjek prosječnih preciznosti (mAP – engl. *mean average precision*): Kada imamo više klasa za prepoznavanje, AP se može izračunati za svaku klasu zasebno, a mAP je prosjek vrijednosti za sve klase. Daje nam jednu metriku koja sažima izvedbu detektora za sve klase.

mAP@X: Ovdje se „X“ odnosi na IoU prag pri kojem se mAP računao. Standardne vrijednosti su 0,50 i 0,50:0,95.

mAP@0,50:0,95 znači da je mAP izračunat od prosjeka više IoU pragova s korakom od 0,05. Ovo je standard za procjenu modela prepoznavanja objekata te kada se u znanstvenim radovima spominje samo „AP“ ili „mAP“, misli se na ovaj kriterij.

Od 2017. godine, kada je COCO skup podataka objavljen, koriste se tzv. „COCO metrike“. Najpoznatija od njih je ova prethodna, a koriste se još:

- AP50 – AP pri IoU=0,50
- AP75 – AP pri IoU=0,75
- $AP^{small}$ – AP za male objekte, odnosno površinom manje od $32^2$ piksela
- $AP^{medium}$ – AP za srednje objekte, $32^2 <$ površina $< 96^2$
- $AP^{large}$ – AP za velike objekte, površina $> 96^2$
- $AR^{max=1}$ – AR pri maksimalnom broju detekcija 1
- $AR^{max=10}$ – AR pri max. broju detekcija 10
- $AR^{max=100}$ – AR pri max. broju detekcija 100
- $AR^{small}$, $AR^{medium}$, $AR^{large}$ – isto kao i AP, ovisno o površini objekta,

tamo gdje IoU nije naveden, misli se na 0,50:0,95, a objekt je istiniti okvir, dakle onaj koji je definiran pri označavanju podataka.

## 2.5. YOLOv7 [27]

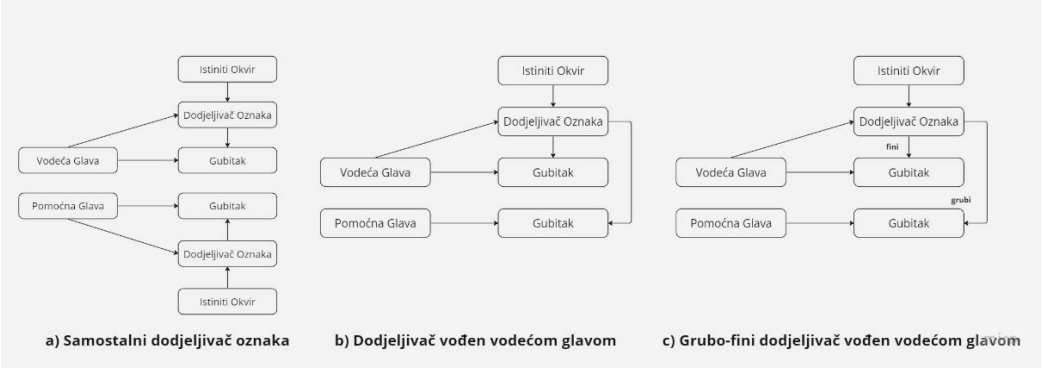
YOLOv7 je nadmašio sve poznate detektore objekata do datuma objavljivanja (srpanj 2022.) i brzinom i preciznošću u rasponu od 5 FPS (engl. *frames per second* – sličica u sekundi) do 160 FPS, a njegova najveća postignuta točnost je 56,8% AP na preko 30 FPS na V100 GPU (NVIDIA Tesla V100 Tensor Core GPU). Detekcija objekata u stvarnom vremenu često je neophodna komponenta u sustavima računalnog vida, u područjima kao što su autonomna vožnja, robotika, analiza medicinske slike itd. Obično je uređaj koji izvršava detekciju u stvarnom vremenu neki mobilni CPU (engl. *central processing unit*, odnosno procesor) ili GPU (engl. *graphics processing unit*, grafički procesor). Za razliku od ostalih uobičajenih detektora objekata, ovo istraživanje modela bilo je usmjereno na optimizaciju arhitekture i optimizaciju procesa treninga. Na taj se način točnost detektora poboljšava, bez povećanja troškova zaključivanja (engl. *inference* – korak gdje je model već istreniran i

koristi se za namijenjenu svrhu). Predložene optimizacijske metode i moduli nazvani su *bag-of-freebies* (vreća besplatnih proizvoda), kao što je ranije spomenuto u radu YOLOv4 [30].

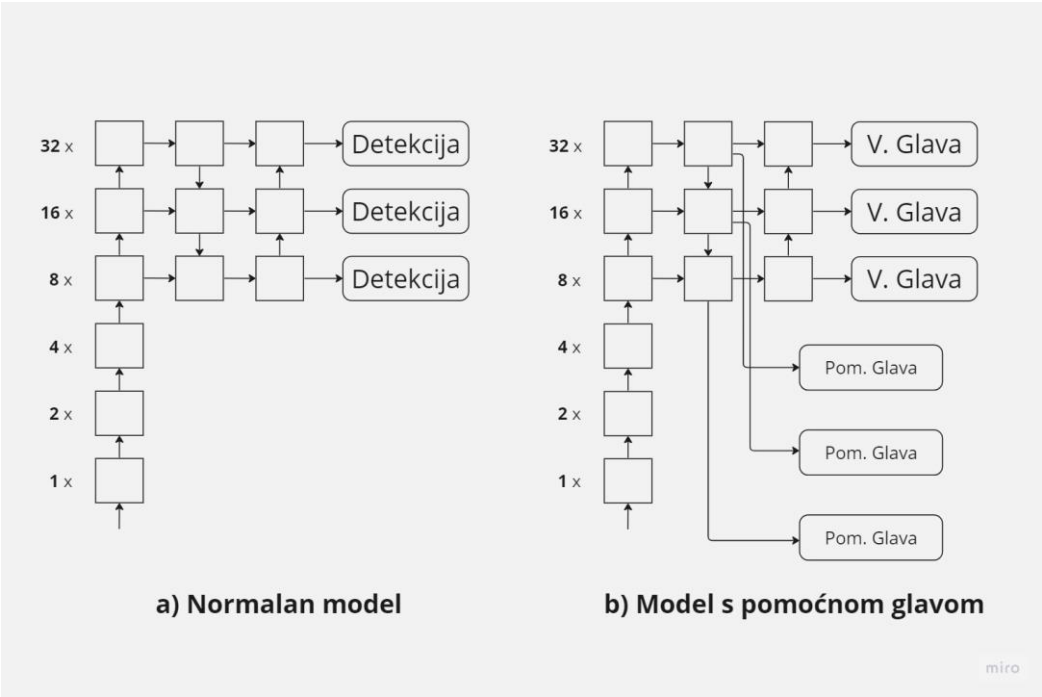
Skaliranje modela način je povećanja ili smanjenja već dizajniranog modela i prilagođavanja različitim računalnim uređajima. Razlike između skaliranih modela nalaze se u razlučivosti (veličini ulazne slike), dubini (broju slojeva), širini (broju kanala – engl. *channels*) i stupnju (broju piramida značajki – engl. *feature pyramid map - FPN*). Dizajniranje učinkovite arhitekture, kako je opisano u većini literature, uglavnom uzima u obzir broj parametara, količinu računalne gustoće (da računalne operacije koje obavljamo su bitne) i količinu izračuna. Kako bi mreža učinkovito učila i konvergirala, predlaže se prošireni ELAN (mreže učinkovite slojne agregacije – engl. *efficient layer aggregation networks*) temeljen na ELAN-u. ELAN se fokusira na kontrolu maksimalne udaljenosti koju gradijent mora prijeći tijekom širenja unazad (engl. *backpropagation*), a da se pritom ne rasprši ili smanji. Poznato je da gradijenti u vrlo dubokim mrežama mogu postati iznimno mali (nestati) ili iznimno veliki (eksplodirati) što može ometati učenje.

Tradicionalno dodjeljivanje oznaka (engl. *label assignment*), tijekom treninga dubokih mreža, koristilo je temeljne bilješke istinitosti (vlastito označene podatke) za izravno generiranje „tvrdih oznaka“, kao što su „auto“ ili „nije auto“, bez dvosmislenosti. S napretkom u istraživanju uvedene su „meke oznake“. Za razliku od tvrdih oznaka koje su apsolutne (npr. 1 ili 0), meke oznake daju vrijednosti poput 0,8 ili 0,3, ukazujući na različite stupnjeve pouzdanosti ili relevantnosti. Na primjer, YOLO [23] koristi IoU predviđenog okvira i istinitog okvira za dodjelu meke oznake za „objektnost“. Ovo mjeri kolika je vjerojatnost da određena regija ili okvir sadrži objekt.

S ovom evolucijom u dodjeli od tvrdih do mekih oznaka, pojavljuju se novi izazovi. Jedan od njih je kako dodijeliti meke oznake različitim dijelovima mreže kada se koristi duboki nadzor [31] s više glava za predviđanje (kao što su pomoćna i vodeća glava – engl. *auxiliary and lead head*, vidjeti Slika 21). Duboki nadzor uključuje dodavanje pomoćnih zadataka tijekom treninga modela. Na primjer, umjesto korištenja jedino izlaza završnog sloja za izračunavanje gubitaka i povratnog širenja, gubitak se također može izračunati korištenjem izlaza iz srednjih slojeva. To može pomoći u boljem protoku gradijenta i može potencijalno poboljšati učenje dubljih mreža, čak i za arhitekture koje obično dobro konvergiraju.




**Slika 20. Primjeri različitih vrsta dodjeljivača oznaka**

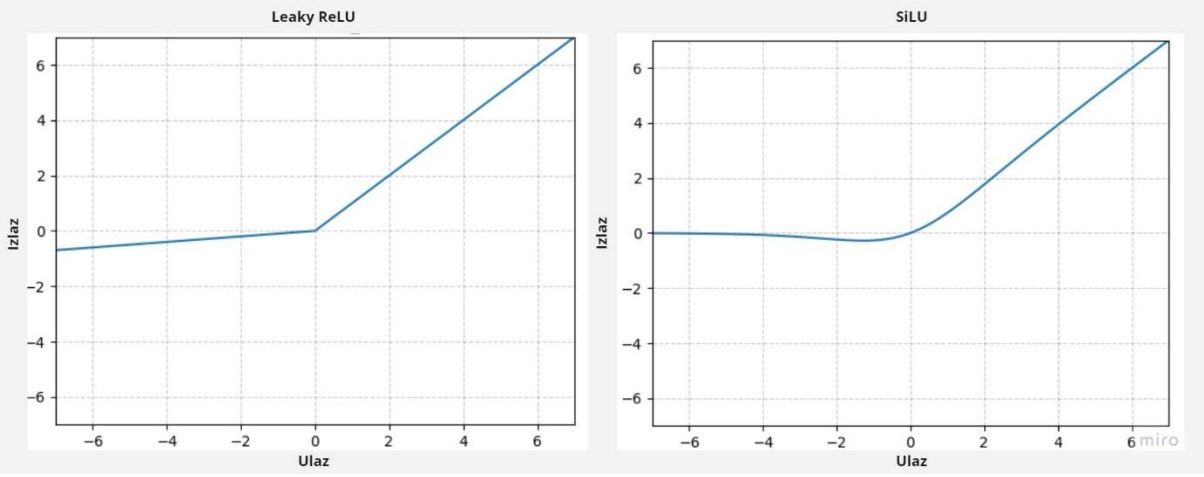



**Slika 21. Razlika između a) normalnog modela i b) modela s pomoćnom glavom**

Metoda predložena u radu uključuje korištenje predviđanja iz vodeće glave (konačni izlaz) za usmjeravanje dodjele oznaka za pomoćnu glavu (srednji izlaz) i samu vodeću glavu. Dodjeljivač oznaka prvenstveno se temelji na predviđanjima glavnog voditelja i temeljnoj istini, kao što je prikazano na Slika 20a. Pretpostavlja se da vodeća glava ima jaču sposobnost učenja od pomoćne glave, pa se vjeruje da su meke oznake izvedene iz njezinih predviđanja reprezentativnije za pravu distribuciju podataka. Dopuštajući pomoćnoj glavi da uči iz informacija koje je glavna glava već shvatila, vodeća glava se može usredotočiti na učenje svih preostalih informacija koje još nije uhvatila (Slika 20b). Proizvode se dva seta mekih oznaka: „gruba oznaka“ i „fina oznaka“ (Slika 20c). Fina oznaka odražava mekanu oznaku

koju proizvodi standardni dodjeljivač vođene glavom. S druge strane, gruba oznaka nastaje blažim dodjeljivanjem pozitivnih uzoraka, dopuštajući da više područja na slici budu označeni kao da sadrže objekte. Zašto to učiniti? Sposobnost učenja pomoćne glave nije tako snažna kao vodeće glave, stoga je prioritet maksimiziranje prisjećanja (sposobnost modela da identificira sve relevantne instance) za pomoćnu glavu da bi se spriječio gubitak vitalnih informacija o učenju. U međuvremenu, za vodeću glavu rezultati se filtriraju kako bi se postigla veća preciznost. Međutim, ako se gruboj oznaci i finoj oznaci prida jednaka važnost, konačna predviđanja mogu biti ispod optimalnih. Kako bi se to izbjeglo, uvedena su ograničenja u dekoderu kako bi se ograničio utjecaj dodatnih pozitivnih područja slike danih iz grube oznake, čime se osigurava da fina oznaka uvijek ima veći prioritet u učenju.

Dizajn modela uzima u obzir raspoloživu računsku snagu. Na osnovnom modelu korištene su različite metode skaliranja kako bi se dobili dublji, širi modeli. Ranije spomenuti E-ELAN koristi se samo za YOLOv7-E6E, drugi najveći i najprecizniji model. Najmanji model, YOLOv7-tiny, koristi propusnu ReLU (engl. *leaky rectifited linear unit* – propusna ispravljena linearna jedinica) aktivacijsku funkciju, dok ostali modeli koriste SiLU (engl. *sigmoid linear unit* – sigmoidna linearna jedinica) aktivacijsku funkciju. Da bismo pokazali koliko odabir aktivacijske funkcije utječe na potrebnu računalnu snagu, možemo usporediti modele YOLOv7-tiny (engl. *tiny* - sitan) i YOLOv7-tiny-SiLU, koji imaju jednak broj parametara. Ipak, ako uspoređujemo po FLOPs (engl. *floating point operations per second* – broj operacija s pomičnim zarezom po sekundi), čime mjerimo računalnu složenost iliti broj operacija koje model zahtijeva pri predviđanju, dolazimo do brojki od 3,5 milijardi, odnosno 13,8 milijardi pri upotrebi SiLU modela. Prikazane su na slici ispod, gdje propusna ReLU funckija ima negativan nagib od 0.1.




**Slika 22. Propusna ReLU i SiLU aktivacijske funkcije**

Uvedena je još jedna tehnika za poboljšanje efikasnosti modela, tzv. reparametrizacija (engl. *reparametrization*). Radi se o transformiranju treniranog modela u svrhu implementacije, čime postaje učinkovitiji, a u nekim segmentima i precizniji u prepoznavanju. Uključuje pojednostavljenje modela spajanjem ili mijenjanjem određenih operacija kako bi se smanjili računalni troškovi.

### *2.5.1. Treniranje YOLOv7 modela*

Kako bismo u potpunosti shvatili nijanse svakog modela treniranog pomoću ove arhitekture, bitno je upoznati se s određenim hiperparametrima - vrijednostima postavljenim prije početka treninga. Dodatno, razumijevanje šireg procesa treninga i napretka je korisno, dajući pogled na trening modela „izvana“. Trening uključuje definiranje broja epoha, tj. koliko će puta naš model vidjeti sve slike u našem skupu podataka, a broj se kreće od obično 50-100 kada se koriste unaprijed uvježbane težine do preporučenih 300 kada se trenira od nule. Veličina serije i veličina slike glavni su hiperparametri koji će odrediti računalno opterećenje. Veličina serije odnosi se na broj primjeraka iz dijela za trening (u ovom slučaju, broj slika) koji se koriste u jednoj iteraciji (tj. jedno ažuriranje parametara modela) ili „broj slika koje model obrađuje odjednom“. Općenito, veće veličine serije povećavaju upotrebu memorije, daju stabilnu konvergenciju zbog točnijih procjena gradijenta, ali mogu zapeti u neoptimalnim minimumima. S druge strane, manja veličina serije može proizvesti „bučna“ (engl. *noisy*) ažuriranja gradijenata, što dovodi do oscilacija u funkciji gubitka, ali taj šum može pomoći u izbjegavanju lokalnih minimuma [32]. Veličina slike određuje ulaznu rezoluciju svake slike koja se koristi tijekom treninga, uglavnom je to kvadrat (640×640 za manje modele, 1280×1280 za veće). Važno je trenirati model na približno istoj veličini slike na kojoj će izvesti predviđanje. Na odluku o korištenju kvadratne razlučivosti za unos utječu slojevi udruživanja u CNN-ovima, skup podataka COCO koristi samo kvadratne slike (prijenosno učenje), omjer širine i visine slike i održavanje proporcija objekta kroz slojeve.

Sekundarni hiperparametri uključuju stopu učenja, optimizator i tehnike povećanja podataka. Stopa učenja određuje veličinu koraka tijekom svake iteracije dok se kreće prema minimumu funkcije gubitka. Visoka vrijednost može uzrokovati brzu konvergenciju treninga, ali može premašiti minimum i rezultirati neoptimalnim rješenjem ili čak odstupiti. Niska vrijednost osigurava opreznije korake, što čini trening sporijim ili bi potencijalno mogao zapeti u lokalnim minimumima. Danas se koristi podesiva stopa učenja koja se definira hiperparametrom konačne brzine učenja na koju se dolazi nakon određenog broja epoha

(također se može definirati!). Prilagodba možda čak i nije linearna, u nekim se slučajevima koristi kosinusna stopa učenja.

Kako bismo optimizirali obuku modela, često moramo odabrati pravi optimizator. Među najraširenijima su SGD (engl. *Stochastic Gradient Descent* – stohastično spuštanje gradijenta), Adam (engl. *Adaptive Moment Estimation* – procjena adaptivnog momenta, gdje se moment može definirati) i AdamW. Ovi optimizatori usmjeravaju kako se težine modela ažuriraju tijekom treninga na temelju gradijenata izvedenih iz funkcije gubitka. Ovdje su korišteni ili SGD ili AdamW.

Povećanje podataka umjetno povećava veličinu i raznolikost skupa podataka za trening primjenom različitih transformacija na izvorne slike, što može pomoći u poboljšanju generalizacije modela. Tijekom obuke, naš model efektivno vidi nešto drugačiji skup podataka tijekom svake epohe. Uobičajena povećanja uključuju:

- Rotacija - rotiranje cijele slike za određeni kut
- Pomicanje (engl. *translation*) - pomicanje slike vodoravno ili okomito
- Skaliranje - povećavanje ili smanjivanje slike
- Okretanje - okrenuti sliku naopako ili lijevo-desno
- HSV - podešavanje nijanse, zasićenosti ili vrijednosti,

i one naprednije kao što su:

- Mozaik - predstavljen u radu YOLOv4, spaja 4 slike u jednu (Slika 23)
- Copy-paste - kopirajte područje sa slike i zalijepite ga na drugu sliku
- Mix-up – kombiniraju se dvije slike linearno pomoću omjera miješanja kako bi se proizvela nova slika (Slika 24).

**Slika 23. Primjer mozaika generiran tijekom treninga**

**Slika 24. Istovremeni mix-up i mozaik**

Tijekom treninga, nadziremo naš model tako što ga trenutno provjeravamo u odnosu na naš skup podataka za evaluaciju, koji obično sadrži 10-20% našeg ukupnog skupa podataka. Ovo pruža uvid u nadzirane gubitke, preciznost, prisjećanje, mAP@50 i mAP@0,50:0,95. Budući da se model nije susreo sa slikama za provjeru valjanosti tijekom treninga, ovaj proces u biti procjenjuje njegovu dosadašnju izvedbu. S alatima kao što je CometML možemo vizualno procijeniti brojne metrike. Redoviti nadzor je značajan za razlučivanje odgovara li model previše (engl. *overfitting*) ili premalo (engl. *underfitting*) ili

jesu li hiperparametri prikladno odabrani. CometML nudi grafikone svih metrika treninga i provjere valjanosti, pomažući u zaključivanju putanje treninga.

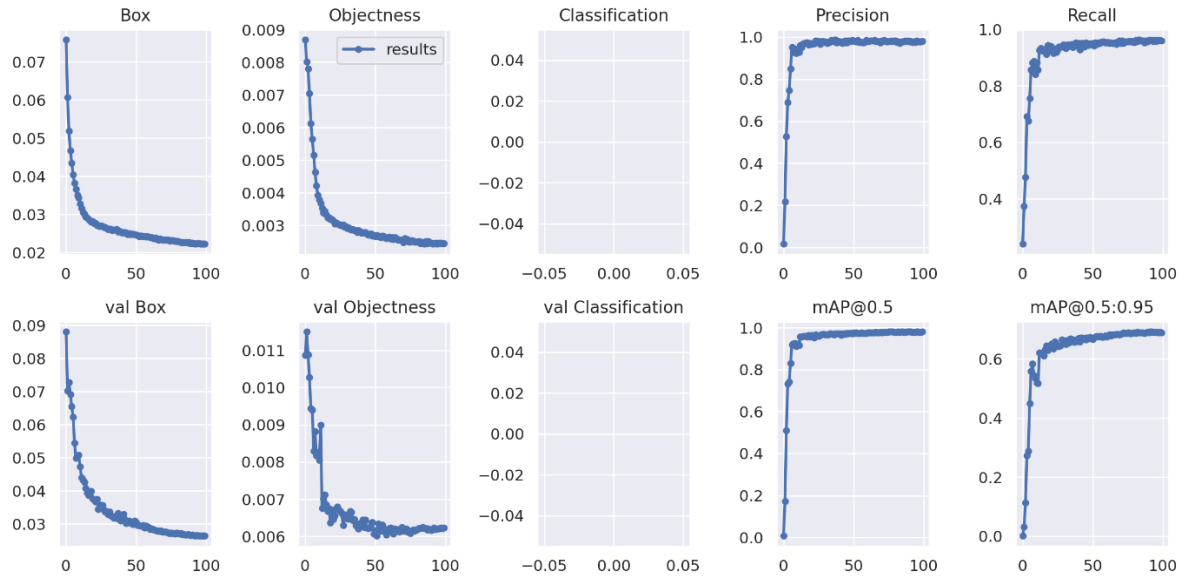
Tu su i 3 dodatna hiperparametra koje valja spomenuti, a to su box, cls i dfl. Svi oni kontroliraju relativnu važnost svakog gubitka tijekom treninga, jer funkcija gubitka ima više komponenti.

Box određuje gubitak povezan s koordinatama graničnog okvira, posebno njegovom lokacijom i veličinom. Cilj mu je osigurati usklađenost predviđenih okvira sa istinitim okvirima. To uključuje središnje koordinate, širinu i visinu okvira. YOLO koristi srednju kvadratnu pogrešku (engl. *mean squared error*) za ovu metriku.

Cls mjeri gubitak klasifikacije. Nakon što se projicira granični okvir, model klasificira sadržani objekt. Ovaj gubitak osigurava da identificirana klasa objekta odgovara stvarnoj oznaci. Obično je oblikovan kao kategorički unakrsni gubitak entropije (engl. *categorical cross-entropy*) u potencijalnim klasama.

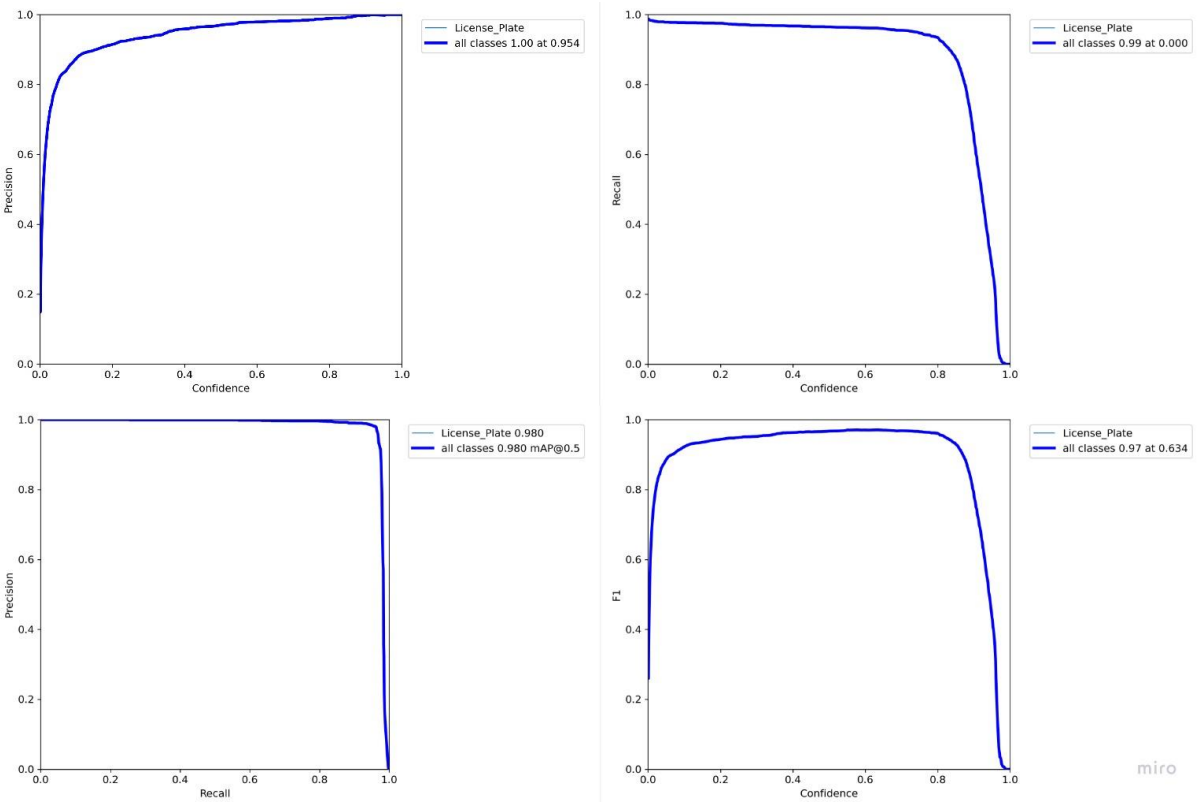
Dfl se odnosi na žarišni gubitak o kojem smo ranije govorili, bavećі se pitanjem neravnoteže klase tijekom treninga. Čak i ako je samo jedan objekt cilj detekcije, oznaka „pozadine“ ostaje ključna.

Među nekoliko sitnih modela koji su trenirani, onaj koji je dao najbolje rezultate iznenađujuće je koristio uglavnom zadane hiperparametre, uz iznimke broja epoha i veličine serije. Ovaj je model treniran tijekom 100 epoha koristeći veličinu serije od 64, koristeći GPU u oblaku (engl. *cloud*). Cijeli proces treninga trajao je oko 1 sat. Prilikom ocjenjivanja izvedbe, primarna metrika koja je razmatrana bila je mAP@0,50:0,95 iz validacijskog skupa. Iako su neke druge metrike pokazale bolje rezultate s različitim težinama, to je ostala primarna odrednica. Vizualni prikaz primarnih metrika ilustriran je u popratnim grafikonima (Slika 25).




**Slika 25. Proces treninga sitnog YOLOv7 modela**

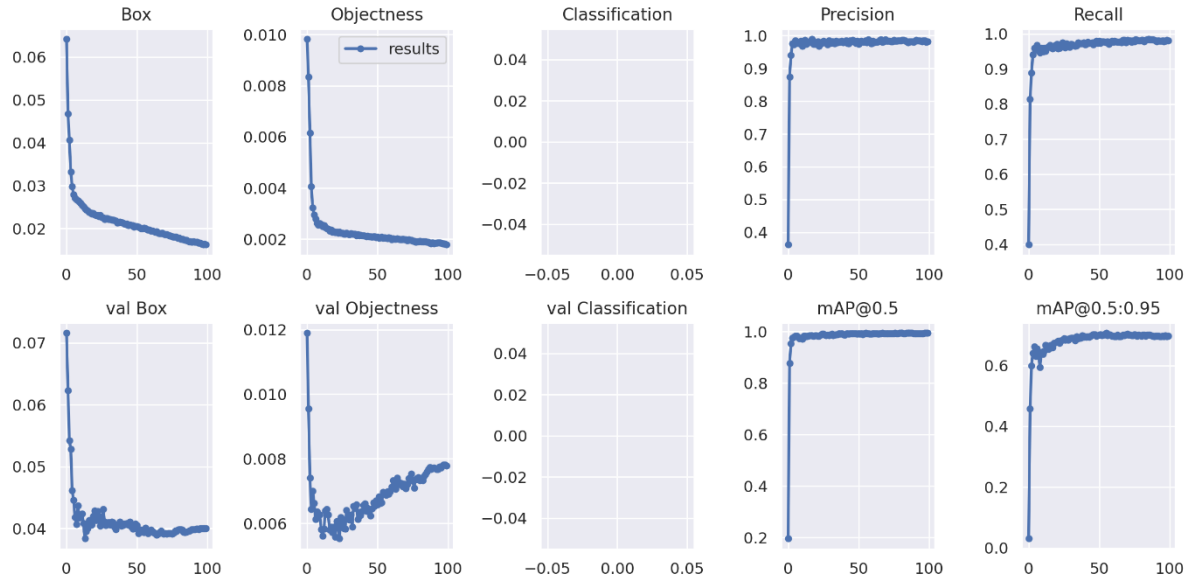
Dodatno, uključene su i krivulje preciznosti-pouzdanosti, prisjećanja-pouzdanosti, preciznosti-prisjećanja i F1 krivulje za holistički prikaz (Slika 26). Dok se procjenjuju razine pouzdanosti, vrijedi primijetiti kako os pouzdanosti daje uvid u strogost naših predviđanja.




**Slika 26. Dodatni grafikoni**

Optimalne težine treninga spremaju se u mapu „weights“, gdje najbolje težine zauzimaju oko 12 megabajta prostora za pohranu. Model je treniran na skupu podataka koji se sastoji od 8823 slike, raspoređenih u 6176 za trening, 1765 za evaluaciju i 882 za testiranje (u omjeru 70:20:10). Većina ovih slika ima jednu registarsku pločicu. Za vizualizaciju arhitekture modela, biblioteka Netron se preporučuje. Međutim, zbog složenosti arhitekture, koja obuhvaća 263 sloja, jednostavan vizualni prikaz nije izvediv.

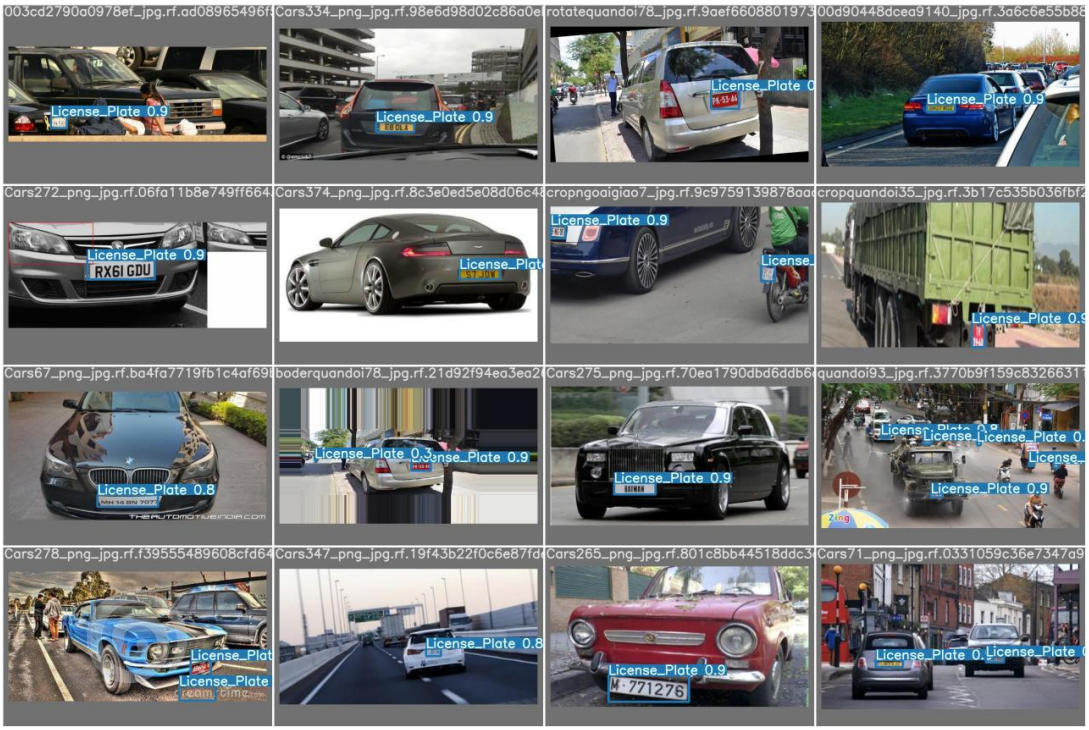
Usporedno gledano, najbolji "ne-sitni" model pokazao je neznatno superiorne rezultate, ali je složeniji s 415 slojeva, 37 milijuna parametara i gradijenata, te zahtijeva gotovo 100 dodatnih GFLOPs-a u računalnoj snazi, što je rezultiralo time da datoteka s težinama zauzima 73 megabajta. Model se trenirao više od 3 sata tijekom 100 epoha, održavajući veličinu serije od 64. Bitno je napomenuti da su parametri povećanja malo drugačiji u odnosu na sitni model i daje se više pažnje „cls“ i „obj“ gubitku. Relevantni grafikoni za ovaj model prikazani su u nastavku (Slika 27).




**Slika 27. YOLOv7 rezultati**

Također, valja pokazati rezultate na konkretnim primjerima. Neki su prikazani na slikama ispod (Slika 28 i Slika 29).


**Slika 28. Rezultati na test skupu podataka, isključivo auti u garažama**




**Slika 29. Raznovrsnija i teža predviđanja**

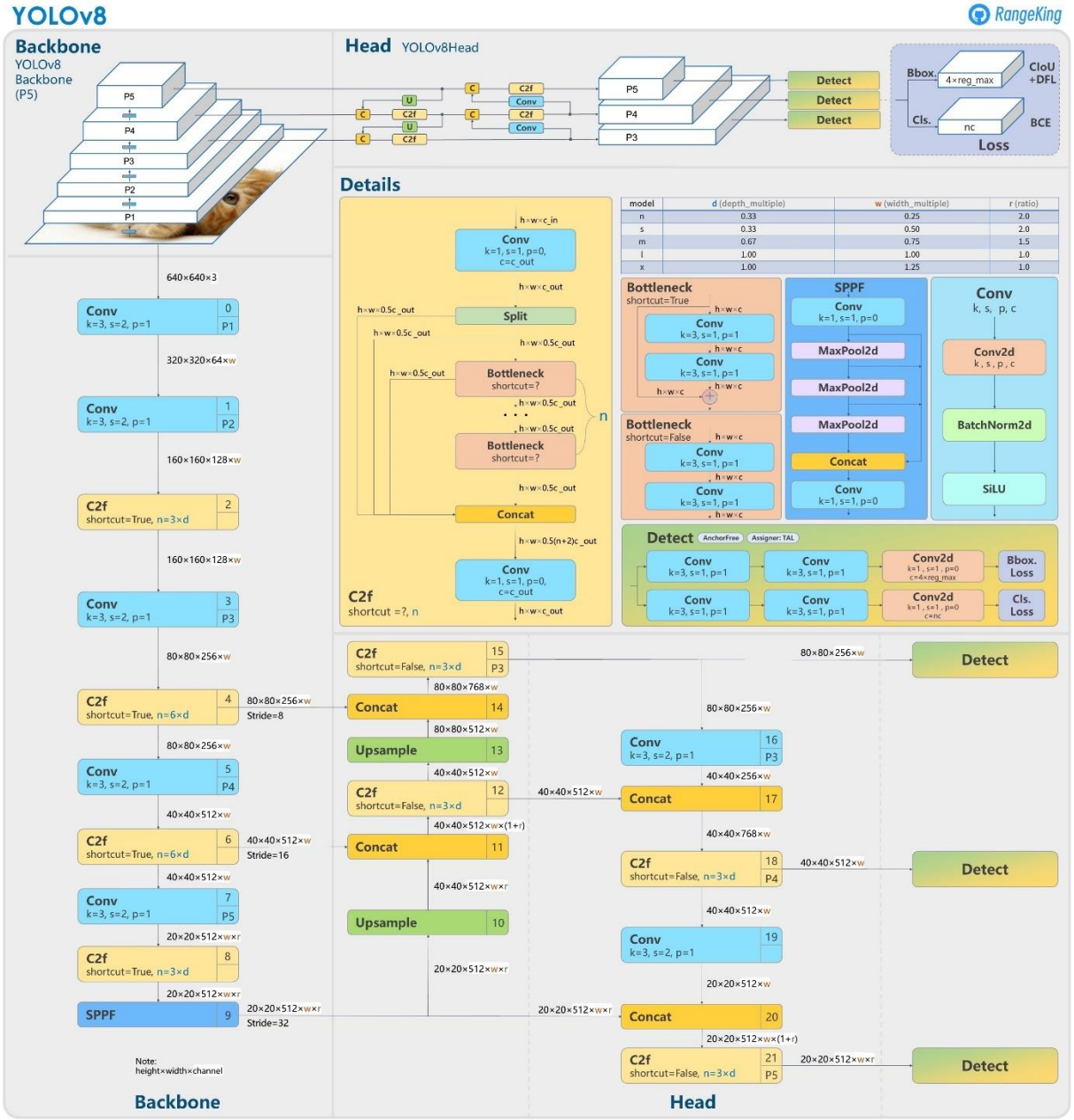
## 2.6. YOLOv8 [28]

Iako naziv sugerira da je nasljednik YOLOv7, YOLOv8 je razvio drugi tim, a ne postoji službeni znanstveni rad koji detaljno opisuje njegovu arhitekturu. Izradio ga je isti tim koji stoji iza YOLOv5, a njegova arhitektura je poslužila kao temelj, uz nekoliko poboljšanja.

Prva od razlika je detekcija bez sidra (engl. *anchor-free detection*). Da se bolje shvati ovaj koncept, korisno je najprije razumjeti prepoznavanje temeljeno na sidrima. Upotrijebljeni u ranijim verzijama YOLO-a, usidreni okviri unaprijed su definirani granični okviri koji služe kao početne referentne točke za predviđanje graničnih okvira. Za svaku lokaciju na mapi značajki, model predviđa prilagodbe (u smislu mjerila, omjera širine i visine slike i položaja) ovih usidrenih okvira kako bi bolje odgovarali stvarnim objektima prisutnima na slici. Ovi usidreni okviri obično se definiraju analizom okvira istine u skupu podataka za trening. Nasuprot tome, u pristupu bez sidra, model izravno predviđa središte objekta i dinamički prilagođava mjerilo. To znači da model uči predviđati veličine objekata bez utjecaja unaprijed definiranih oblika sidra. Ovaj pristup pojednostavljuje proces ne-maksimalnog potiskivanja (NMS) budući da skraćuje početna predviđanja okvira. Dok modeli temeljeni na sidrima generiraju višestruka predviđanja za svako sidro na svakoj lokaciji.

Napravljena je značajna izmjena glavnog građevnog bloka, prelaskom s C3 na C2f. Početni konvolucijski sloj s kernelom 6×6 smanjen je na 3×3. Ova se promjena može vidjeti na glavnom dijagramu arhitekture (Slika 30), gdje „f“ označava broj značajki, „e“ predstavlja brzinu proširenja, a CBS je blok koji sadrži konvoluciju, skupnu normalizaciju (engl. *batch normalization*) i sigmoidnu linearnu jedinicu (SiLU). U C2f, izlazi iz svih modula uskog grla (definirani s dvije 3×3 konvolucije s preostalim vezama, koncept posuđen od ResNeta) su spojeni. To je u suprotnosti s C3, gdje je korišten samo izlaz iz konačnog uskog grla.

Što se tiče povećanja podataka, točnije mozaika, empirijski je pokazano da šteti rezultatu treninga ako se vrši tijekom cijelog treninga. Kako bi se to zaobišlo, povećanje mozaika je onemogućeno za zadnjih 10 epoha.




**Slika 30. YOLOv8 detaljna arhitektura [33]**

Promatrajući dijagram, P1 do P5 predstavljaju razine mapa značajki, gdje značajke imaju određenu rezoluciju u odnosu na ulaznu sliku. Postupno se pojavljuju kroz kičmu arhitekture, od izvorne veličine slike (P1) pa sve do 1/32 slike (P5). Analiziranjem arhitekture vidi se da P3 prepoznaje „velike“ objekte, P4 prepoznaje „srednje“ objekte, a P5 „male“ objekte, što znači da mreža zapravo vraća više vrsta detekcija.

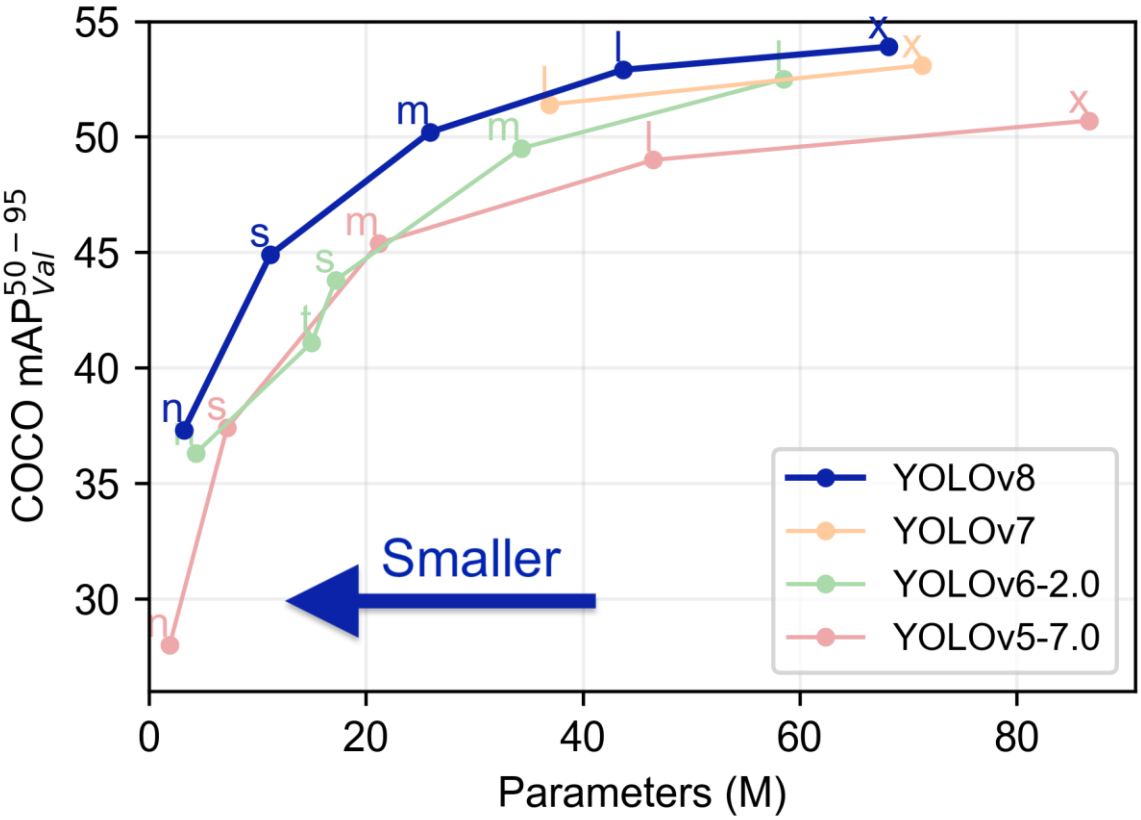
SPPF (engl. *spatial pyramid pooling fast* – brzo objedinjavanje prostorne piramide) blok, brza implementacija SPP bloka [34], rješava problem gdje tradicionalni CNN-ovi zahtijevaju unos fiksne veličine, što znači da se slikama obično mora promijeniti veličina ili

ih se obreže. SPP omogućuje mreži prihvaćanje slika bilo koje veličine i pruža izlaz fiksne veličine, bez obzira na ulaznu veličinu. Ovaj blok dijeli mapu značajki u skup ćelija u više razmjera, gdje se maksimalno udruživanje (*max pooling*) izvodi neovisno u svakoj pojedinoj ćeliji. To daje fiksni broj značajki za svaku veličinu ćelije. Sva udružena obilježja iz ćelija različitih veličina zatim se spajaju u obliku vektora obilježja fiksne duljine.

Blok ulančavanja (engl. *concatenation*) jednostavno spaja značajke iz različitih slojeva, skoro uvijek po dimenziji kanala, što se i vidi na dijagramu.

### *2.6.1. Treniranje YOLOv8 modela*

Strojno učenje je iterativni proces, što će biti pokazano na ovom modelu. Korištena su 2 skupa podataka: isti kao i u v7 i njegova proširena verzija. Daleko najviše treninga je odrađeno na „nano“ modelu, nekoliko na malom i srednjem i najprecizniji trening je očito bio pomoću velikog modela (i najsporiji). Postoji još „x“ model, najveći i jedini na kojem trening nije odrađen jer je na COCO skupu imao neznatno veću točnost. Usporedba svih navedenih je prikazana na Slika 31.

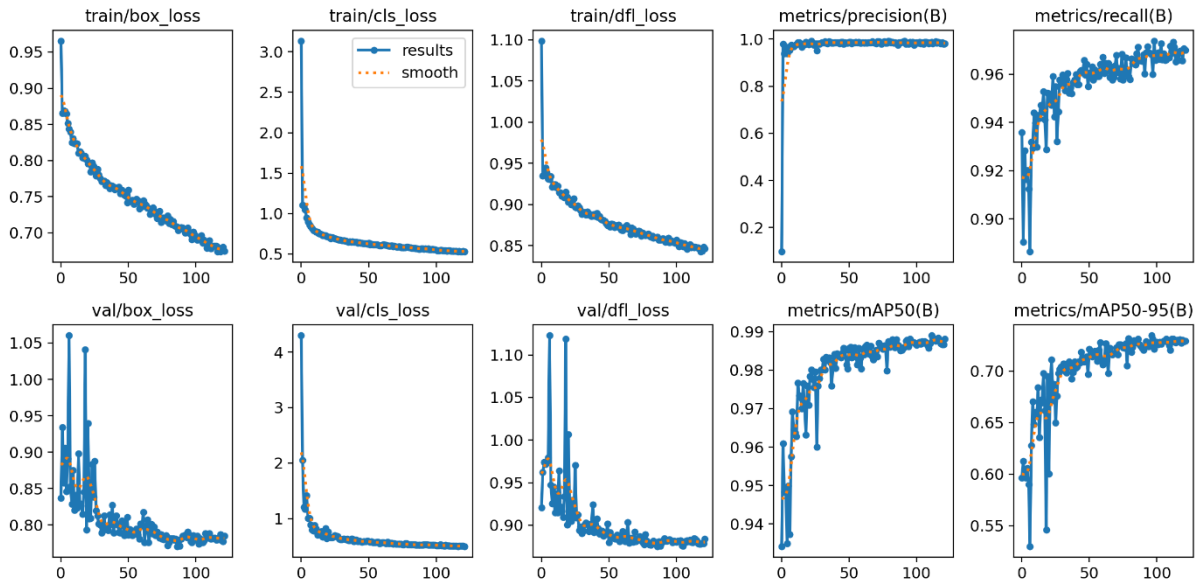



**Slika 31. Usporedba YOLO modela na COCO evaluacijskom skupu podataka [28]**

Raspon epoha treninga je bio između 20 i otprilike 120, „otprilike“ jer postoji hiperparametar koji je često nebitan (ovisno o vrijednosti) pod nazivom strpljenje (engl. *patience*). Određuje broj epoha bez poboljšanja evaluacije prije nego što se trening zaustavi, a ovdje je broj bio relativno strog (točnije 25). Strpljenje se smatra jednim od načina sprječavanja pretreniranja.

Promjena povećanja podataka nije imala pozitivan utjecaj na trening, dok promjena usredotočenosti pojedinog gubitka može dati neznatno bolje rezultate. Također veličina slika pri ulazu je brže dolazila do vrlo dobrih rezultata (>0.7 mAP0.50:0.95)

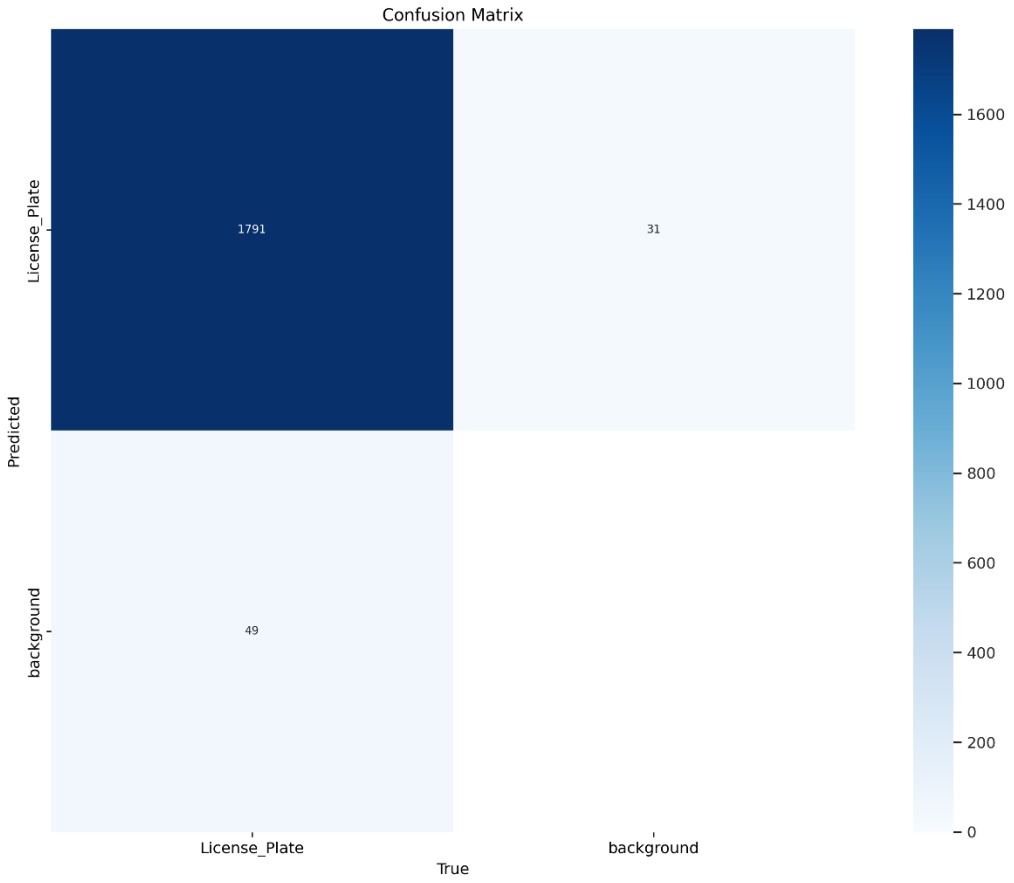
Nano model se sastoji od 225 slojeva, 3157200 parametara i zahtjeva 8,9 GFLOPs-a, a datoteka najboljih težina sadrži 6MB podataka. Najbolji dobiveni rezultat je na manjem skupu podataka pri vrijednosti mAP0,50:0,95 od 0,735. Svi primarni grafikoni su vidljivi na Slika 32.




**Slika 32. YOLOv8 nano metrike najboljih rezultata**

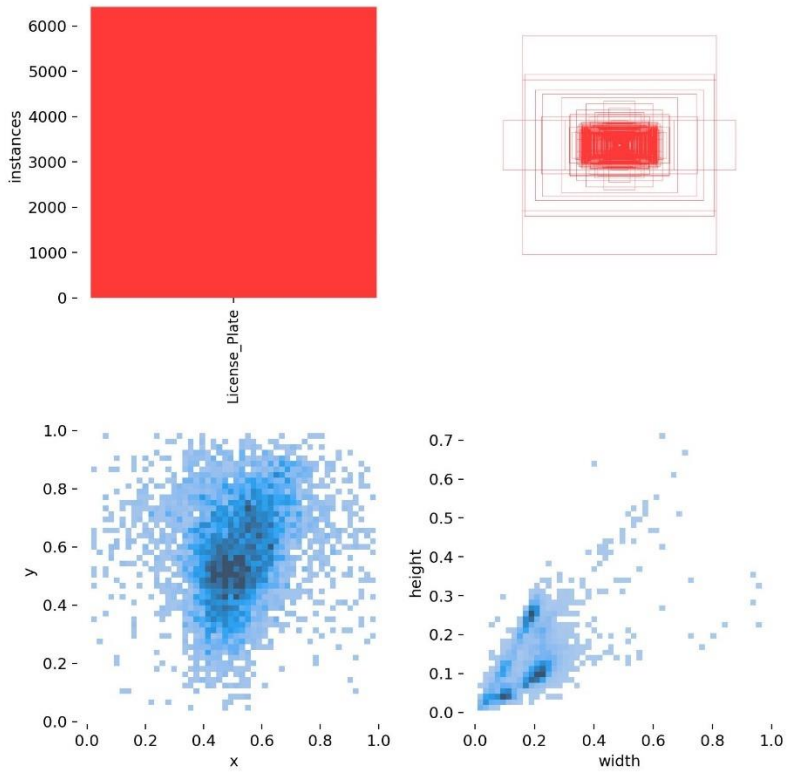
Ostali grafikoni poput preciznosti, prisjećanja i F1 rezultata su vrlo slični onima u YOLOv7, stoga ovdje nisu prikazani. S obzirom da je ovaj model više softverski nego znanstveno orijentiran, na kraju treninga postoji više dobivenih informacija. To su matrica zbunjenosti (engl. *confusion matrix*), grafikon oznaka i korelogram oznaka (engl. *label correlogram*).

Matrica zbunjenosti se primarno koristi za opisivanje performansi modela na evaluacijskom setu podataka. Za binarni problem poput ovoga, matrica je tablica veličine 2×2 koja prikazuje odnos među detekcijama i istinitim podacima. Gornji red sadrži istinite pozitivne i lažne pozitivne, dok donji red prikazuje lažne pozitivne i istinite negativne rezultate. Konkretan primjer prikazan je na Slika 33. Matrica postaje kompleksnija i korisnija što je broj klasa veći. Isto tako se njene dimenzije povećavaju s brojem klasa, gdje je dimenzija kvadratne matrice *broj_klasa+1*.

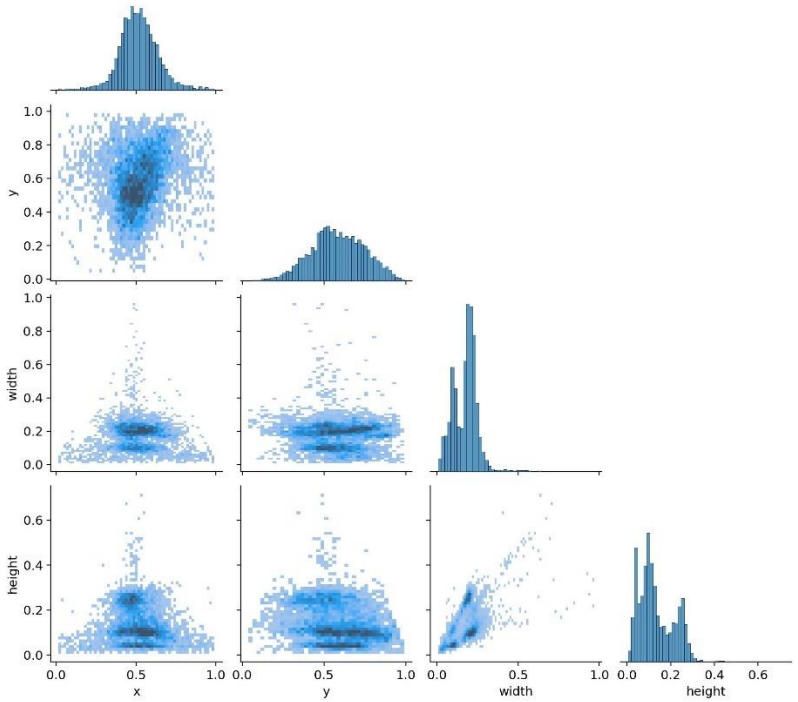



**Slika 33. Matrica zbunjenosti**

Grafikon oznaka (Slika 34) i korelogram oznaka (Slika 35) prikazuju lokaciju oznaka u skupu podataka za trening na cijelom skupu odjednom. S obzirom da su oznake tekstualni podaci, moguće ih je prezentirati na ovaj način. Korelogram oznaka je grupa 2D histograma koji prikazuju svaku os oznaka naspram jedne od ostalih osi.



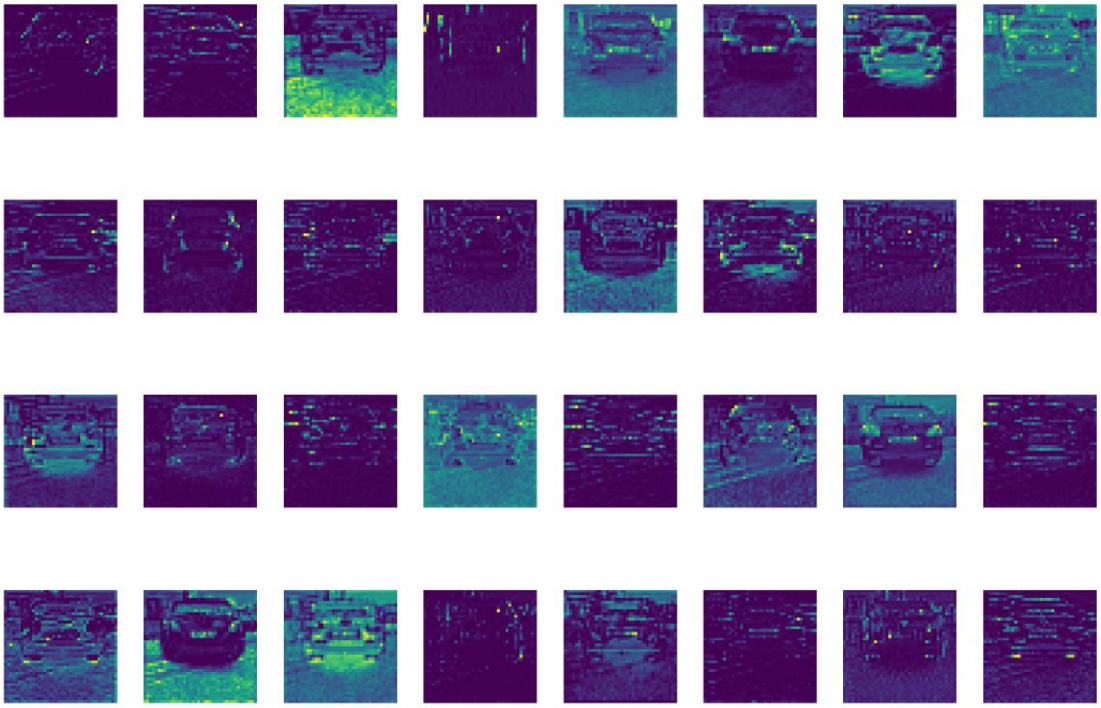

**Slika 34. Grafikon oznaka manjeg skupa podataka**



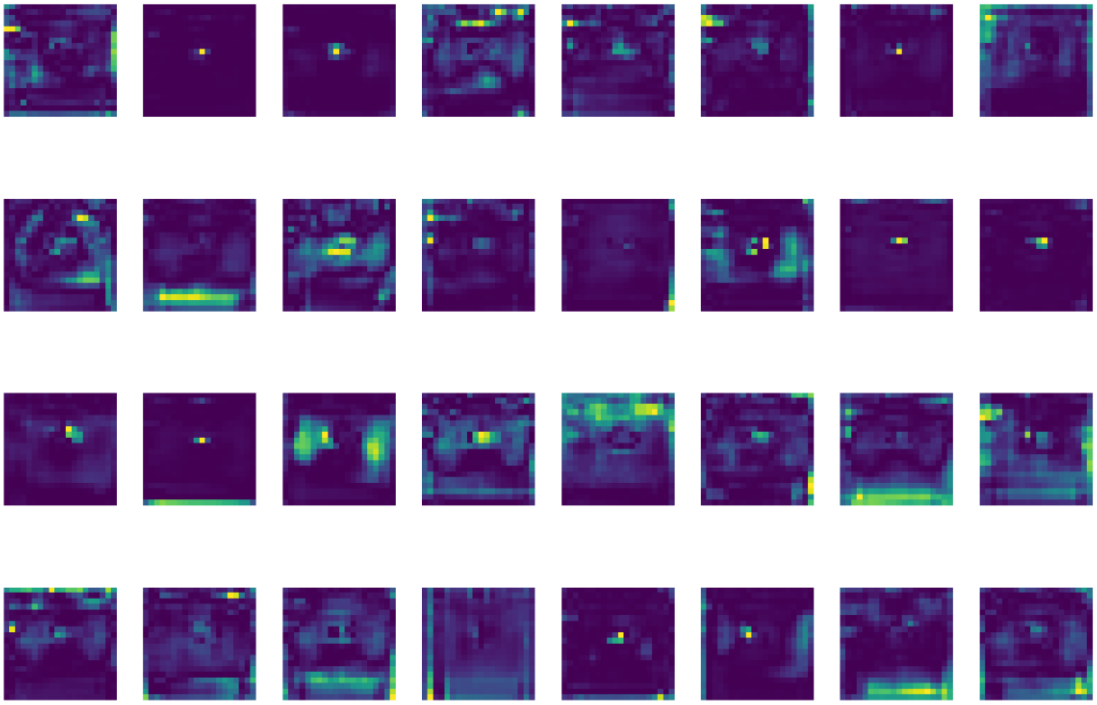

**Slika 35. Korelogram oznaka manjeg skupa podataka**

Pri prepoznavanju, moguće je vizualizirati prolazak kroz svaki od blokova u arhitekturi (ukupno 22). Logično je da se mape značajki ne mijenjaju kroz blokove koji ne sadrže konvolucije (*Upsample*, *SPPF*, *Concat*). Za primjer pokazano je nekoliko mapa značajki koji su izlazi iz blokova navedenih u opisu slike (Slika 36, Slika 37, Slika 38). Vidi se da negdje nisu prikazane sve značajke, npr. izlaz iz 6. bloka (broj 5 na Slika 30, jer počinjemo od 0) sadrži 128 kanala (512×*w* gdje je *w* za nano model 0,25, očitano iz tablice sa iste slike), ali prikazano je samo 32. Izlaz iz prvog bloka sadrži 16 mapa značajki i sve su prikazane na slici. Kao što je opisano na slici arhitekture, mape značajki prolaskom kroz mrežu postaju sve manje i manje po rezoluciji i uglavnom dublje po kanalima. Ulazna rezolucija je 640×640 i rezultat predviđanja je prikazan na Slika 39.

**Slika 36. Mape značajki koje prvi blok daje kao izlaz**



**Slika 37. Dio mapa značajki koje šesti blok daje kao izlaz**



**Slika 38. Izlaz iz zadnjeg bloka, rezolucije su 20×20**


**Slika 39. Konačan rezultat predviđanja**

Zanimljivo je primijetiti kako predviđanje na slikama znatno veće rezolucije od one na kojima je trening odrađen daje prilično loše rezultate na „velikim“ oznakama, odnosno onima koje zauzimaju velik broj piksela na slici (Slika 40). Usporedbe radi prikazano je nekoliko slika gdje je lijeva rezolucije treninga (640×640), a desna je izvorne rezolucije, odnosno 2976×2976. Na Slika 41 se vidi da model prepoznaje registarske pločice koje su one rezolucije na kojoj je i učio. Na lijevoj slici je jedina detektirana pločica veličine 26×10 piksela, dok su one u pozadini relativno neprepoznatljive. Na desnoj slici su one u pozadini mnogo veće nego ona koja je detektirana na lijevoj, npr. najveća je 70×20 piksela.


**Slika 40. Predviđanje na slici znatno veće rezolucije od one na kojoj je model treniran**


**Slika 41. Predviđanje na 640×640 i 2976×2976 slikama**

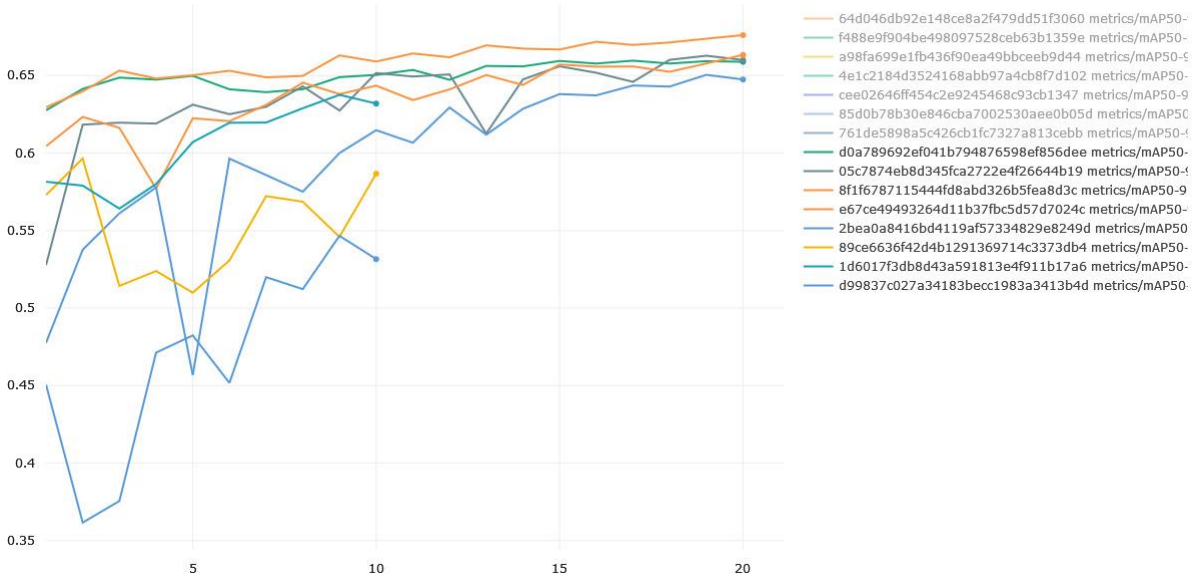
## 2.7. Optimizacija hiperparametara

Kao što je već navedeno, strojno učenje je iterativni proces, a optimizacija hiperparametara pruža mogućnost pronalaženja najboljih hiperparametara za određenu svrhu.

Najčešće se traže optimalna stopa učenja, optimizator, neka povećanja, gubitci, a ako se model trenira od nule, mijenja se i veličina kernela u konvolucijskim slojevima ili njihov broj.

Moguće je uzimati hiperparametre iz neke već definirane grupe (npr. optimizator može biti SGD ili AdamW), nasumičnim odabirom iz nekog raspona ili pomoću optimizacijske metode koja na temelju prethodno odabranih hiperparametara pokušava naći što bolju vrijednost.

Ovdje je na YOLOv8 nano modelu tijekom 20 epoha trenirano nekoliko modela pomoću optimizacije hiperparametara pomoću biblioteke RayTune. Najveći problem pri ovom postupku je ogromna potrošnja računalne snage, a ne može se ni prenijeti na modele koje prepoznaju drugačije objekte. Modeli su nadzirani pomoću navedenog CometML alata koji je napravljen upravo za ovakve stvari, dobiveni rezultati su prikazani na slici ispod gdje svaki model sadrži nasumični naziv.




**Slika 42. Optimizacija hiperparametara YOLOv8 nano modela tijekom 20 epoha gdje je y-os mAP0,50:0,95**

# 3. Prepoznavanje teksta registracije

## 3.1. Optičko prepoznavanje znakova

Sustavi za optičko prepoznavanje znakova (OCR – engl. *optical character recognition*), posebno oni koji koriste duboko učenje za slike prirodnih scena, obično prihvaćaju pristup u dva koraka. Početna faza može se usporediti s otkrivanjem objekta, pri čemu je „objekt“ tekst. U ovoj fazi sustav identificira i ocrtava granične okvire oko potencijalnih područja teksta. Nakon što se ta područja točno odrede, sljedeći korak uključuje prepoznavanje znakova unutar tih okvira, njihovo prevođenje u strojno čitljiv tekst. Stoga, iako se OCR može promatrati kao specijalizirani oblik otkrivanja objekata, on nosi dodatnu kompleksnost prepoznavanja znakova nakon otkrivanja.

U današnjoj digitalnoj eri, sa sveprisutnošću digitalnih fotoaparata, osobito u mobilnim telefonima, skenirane slike nisu jedini kandidati za pretvorbu teksta. Ova promjena naglašava važnost faze „detekcije“ jer tekst više ne predstavlja cijelu sliku.

## 3.2. PP-OCRv3

Model koji se koristi je PP-OCRv3, treća verzija PaddleOCR-a [35]. Kako bismo razumjeli što se događa, analiziramo njegovu arhitekturu i napredak. S obzirom da su primarne platforme za implementaciju takvih modela tzv. rubni uređaji (engl. *edge devices*), poput mobilnih telefona, koriste se lagane verzije standardnih modela. To uključuje laganu kičmu i laganu glavu. Dok su prethodnici modela, PP-OCRv1 i v2, koristili CRNN (engl. *Convolutional Recurrent Neural Network* – konvolucijska ponavljajuća neuronska mreža) za prepoznavanje teksta — kombinaciju CNN-a za ekstrakciju značajki i RNN-a za interpretaciju značajki — PP-OCRv3 integrira sitnu verziju SVTR modela [36] (engl. *Single Visual Text Recognition* – jednovizualno prepoznavanje teksta), temeljenog na transformatorima (engl. *transformer*), s PP-LCNet-om koji se temelji na CNN-u. Ovaj odabir dizajna omogućuje modelu da uzme u obzir globalni kontekst slike, što je odmak od korištenja isključivo CNN-a, koji se ističe u obuhvaćanju lokalnog konteksta.

Kako bi raspoznao različite veličine teksta prisutne na slikama, model koristi RSE-FPN (engl. *Residual Attention Feature Pyramid Network* – piramidalna mreža značajki). FPN-ovi se skoro uvijek koriste u modelima za otkrivanje objekata. Oni konstruiraju piramidu mape značajki spajanjem značajki niske rezolucije, semantički bogatih, sa značajkama visoke

rezolucije, semantički slabim [37]. Ugrađeni mehanizam pažnje zatim dodjeljuje težine različitim ulaznim segmentima.

Slično konvencionalnim modelima prepoznavanja objekata, PP-OCRv3 koristi prednosti naprednih tehnika povećanja podataka. Za razliku od otkrivanja registarskih pločica gdje je izbjegavano proširenje copy-paste, u ovom kontekstu pokazalo se itekako korisnim, čime je tekst uklopljen u prirodne scene (Slika 43).


**Slika 43. Primjer copy-paste u detekciji teksta. Zeleni okviri predstavljaju zalijepljeni tekst, a crveni izvorni tekst u slici. [38]**

Kako bi se poboljšala učinkovitost modela, upotrijebljeno je nekoliko strategija za smanjenje njegove veličine i povećanje brzine zaključivanja. To uključuje kvantizaciju (npr. prijelaz s preciznosti float32 na float16 ili čak int8) i izostavljanje blokova „stiskanja i pobude“ (engl. *squeeze and excitation*), osim u krajnjim slojevima modela gdje se njegova računalna kompleksnost ipak isplati.

### 3.3. Metode predobrade

Nakon otkrivanja registarske pločice na slici, izolira se to područje, fokusirajući se isključivo na njega za prepoznavanje teksta. Ovo se u biti bavi komponentom otkrivanja teksta unutar OCR procesa. Budući da je slika snimljena digitalnom kamerom, može se naići

na prepreke kao što su geometrijska distorzija, neravnomjerno osvjetljenje i šum. Kako bi se poboljšala točnost OCR modela, provode se neki koraci predobrade kako bi se minimizirale te nesavršenosti. Sve metode u ovom i sljedećem poglavlju koriste biblioteku OpenCV.

Kao što je navedeno u radu koji se bavi predobradom za digitalno snimljene slike, medijalno zamućenje (engl. *median blur*) se izbjegava, a povećanje veličine poboljšava točnost [39].

Pretvaranjem slike u sive nijanse (engl. *grayscale*), smanjuje se računska složenost eliminacijom kanala boja (s 3 na 1). Slika se umjetno povećava koristeći linearnu interpolaciju za nepostojeće piksele, budući da registarska pločica zauzima vrlo malo područje u pikselima (u većini slučajeva, najviše 5% cijele slike). Kvaliteta snimljene slike, uvjeti osvjetljenja, korišteni algoritmi kompresije i druge varijable mogu uzrokovati prisutnost šuma, što je vidljivo na primjeru (Slika 44). Koristi se Gaussovo zamućenje kako bi se uklonio šum na slici, čime se izračunava težinski prosjek intenziteta tog piksela i intenziteta njegovih susjeda. Težine su određene Gaussovom funkcijom, što znači da vlastiti intenzitet piksela ima najveću težinu, a težina se smanjuje kako se udaljava od centralnog piksela. Cijeli proces se odvija operacijom konvolucije slike i Gaussovog kernela (koji pridaje veće težine pikselima bliže centru).


**Slika 44. Umjetno dodavanje šuma**

Zatim se koristi binarizacija za transformaciju slike u binarni format, pri čemu se razlikuje crno i bijelo. Otsu-ova metoda služi kao tehnika koju smo odabrali za ovaj korak, koja određuje „optimalni“ prag koji maksimizira varijancu između dvije klase piksela: prednjeg plana i pozadine. Usporedba originala i binarne verzije prikazana je na Slika 45, premda visoka kvaliteta originalne slike ne zahtijeva nikakvu predobradu.


**Slika 45. Primjena binarizacije**

Konačno, ovisno o boji teksta (crna ili bijela), predobrada može završiti operacijom morfološkog „zatvaranja“, koja uključuje dilataciju (engl. *dilatation*) nakon koje slijedi erozija (engl. *erosion*). Kada se radi o crnom tekstu, dilatacija učinkovito sužava znakove, koristeći kernel (često kvadratni ili kružni) za konvoluciju slike. Nasuprot tome, erozija proširuje znakove. Ove morfološke tehnike su korisne za smanjenje šuma, popunjavanje praznina i povezivanje fragmentiranih dijelova na slici.

Cijeli proces i rezultati su vizualizirani kroz skup slika (Slika 46), u ovom slučaju očitavanje teksta nije bilo točno u izvornoj slici (rezultat „OS-288-1“), dok je najbolji rezultat na slici 32d, gdje se dobije „.OS 288.IU“, što bi nakon neke vrste filtriranja dalo točan rezultat.


**Slika 46. Koraci u predobradi registarske pločice: a) originalna slika 103×39 piksela, b) povećana, zamućena i binarizirana slika u sivim nijansama, c) dilatacija, d) erozija, e) zatvaranje, odnosno dilatacija pa erozija, f) otvaranje, odnosno erozija pa dilatacija**

## 3.4. Transformacija perspektive

Kako bi se osiguralo precizno prepoznavanje i u slučajevima koji se ne bi trebali dogoditi pri sustavima poput garaža ili rampi, ključno je ispraviti sva geometrijska iskrivljenja koja mogu nastati zbog razlika u kutu između kamere i automobila. Iako su prethodno obrađeni neki koraci, ovdje su dublje opisani oni specifični za transformaciju perspektive.

Ako slika ima razlučivost ispod određenog praga, npr. 500 piksela (iako se to idealno izbjegne s visokokvalitetnim kamerama), u početku je treba povećati. Slika se zatim pretvara u sive tonove. Naknadno se koristi prilagodljivi prag (engl. *adaptive thresholding*) kako bi se pojačala tamna područja, posebice tekst i obrub registarske pločice. Gaussovo zamućenje pomaže u uklanjanju šuma, nakon čega slijedi binarizacija i morfološke operacije.

Prelazeći na napredne strategije, koristi se otkrivanje kontura kako bi se izolirale najistaknutije konture na slici. Jednostavno rečeno, kontura je kontinuirana krivulja koja povezuje točke istog intenziteta ili boje [40]. Iz najveće konture, aproksimacija poligona olakšava izdvajanje četverokuta, pružajući potrebna četiri kuta za transformaciju perspektive. Ova transformacija daje ispravljeni pogled na registarsku pločicu, kao da se promatra frontalno (Slika 47).


**Slika 47. Transformacija na temelju četiri točke (engl. *four point transform*)**

Pod optimalnim uvjetima, kada je automobil postavljen izravno ispred rampe, ova transformacija bi mogla biti suvišna. Na temelju specifikacija fotoaparata i kvalitete slike, promjena faktora poput kernela prilikom zamućivanja ili morfoloških operacija je vjerojatno nužna.

# 4. Hardverska realizacija

## 4.1. Hardver za implementaciju

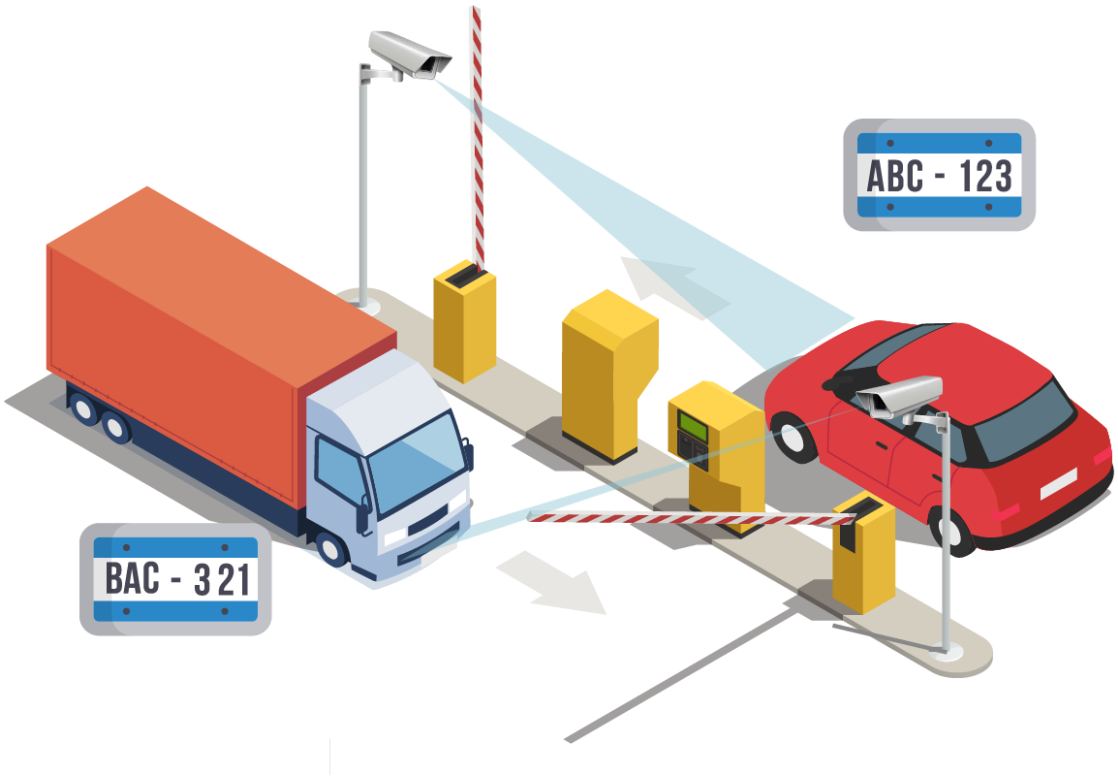
Izbor odgovarajućeg hardvera za implementaciju sustava prepoznavanja registarskih pločica temelji se na različitim faktorima kao što su proračun, procesorska snaga, željeno korisničko iskustvo, brzina prepoznavanja i točnost. Jedna je mogućnost pokrenuti model izravno na rubnom uređaju (engl. *edge device*), nudeći brže vrijeme odziva, ali zahtijevajući više računalnih resursa. Alternativno, prepoznavanje u oblaku omogućuje da se računalni zadaci prebace na znatno jaču platformu. Iako ovo uvodi malo kašnjenja, nudi prednosti robusnije obrade i centraliziranog upravljanja, pod uvjetom da rubni uređaj održava stabilnu internetsku vezu. Međutim, primarni nedostatak korištenja oblaka je cijena: svaki API (engl. *application programming interface* – sučelje za programiranje aplikacija, način na koji računalni programi komuniciraju) poziv se naplaćuje, a ti se troškovi mogu brzo akumulirati.

Kamera bi trebala biti postavljena na visini i pod kutom koji osigurava jasan, neometan pogled na registarsku pločicu. Za standardna parkirna mjesta sa samo rampom idealan je frontalni, malo udaljen od središta pogled. S druge strane, za garaže bi centralizirano postavljanje na stropu moglo biti najbolje rješenje, s obzirom na skup podataka na kojem su ovi modeli trenirani.

Moguće vrste kamera uključuju:

- IP kamera: prenosi video putem mreže, što ju čini pogodnom za daljinski nadzor i kontrolu.
- USB kamera: kompatibilna s rubnim uređajima kao što su Raspberry Pi ili NVIDIA Jetson.
- CCTV kamera s DVR/NVR: tradicionalni sustavi sigurnosnih kamera.

Za situacije s ograničenim osvjetljenjem, kao što su noćne operacije, ključno je da kamera ima infracrveno osvjetljenje ili mogućnost noćnog snimanja, pogotovo na loše osvijetljenim parkiralištima. S obzirom da postoje tvrtke koje se bave ovakvim stvarima, prikazana je njihova implementacija kamera na ovakvim sustavima na Slika 48.

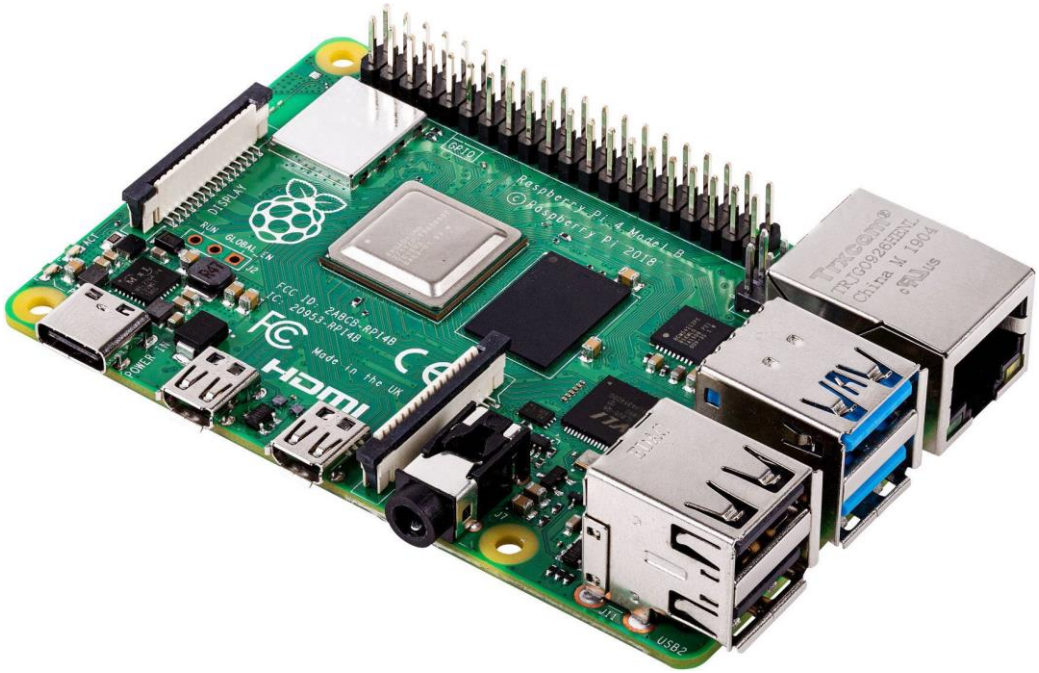



**Slika 48. Sustav detekcije registarskih pločica [41]**

Kada se automobil zaustavi ispred parkirne rampe radi snimanja registarske pločice, postoji nekoliko dostupnih metoda. Međutim, softverski pristupi su obično najisplativiji jer eliminiraju potrebu za dodatnim hardverom. Implementacija jednostavnog algoritma za detekciju pokreta — otkrivanje promjena u uzastopnim video okvirima pri smanjenoj brzini sličica u sekundi (npr. 5 FPS) — može se postići pomoću OpenCV-a. Ako se softverska rješenja ispostave nepouzdanima, razni senzori mogu odrediti je li automobil stao ispred rampe.

U slučaju da postoji potreba za pohranom informacija, poput prepoznatih registarskih tablica, tekstualnih podataka ili vremenskih oznaka, način prijenosa i pohrane tih podataka postaje bitan faktor. Na lokalnoj mreži to se može postići putem Wi-Fi ili Ethernet kabela. Povezivanje s postojećim sustavom za upravljanje parkiranjem može automatizirati procese poput ulaska, izlaska i plaćanja, ali to također uvodi dodatnu kompleksnost. S obzirom na ograničene mogućnosti pohrane rubnih uređaja, centralizirani sustav mogao bi biti poželjniji za velika parkirališta. Treba uzeti u obzir da je prikupljanje podataka u skladu sa smjernicama GDPR-a.

Korištenje pojednostavljenog modela, poput YOLOv8n (gdje „n“ označava „nano“), moglo bi omogućiti implementaciju na Raspberry Pi (Slika 49). Najnovija inačica, verzija 4 (izdana 2019.), sadrži 1,8 GHz Cortex-A72 ARM CPU, do 8 GB RAM-a i utor za SD karticu. Također ima specijalizirani 15-pinski konektor za kameru, što još malo povećava cjelokupnu brzinu procesa. Iako ovaj specifični model nije testiran, vrijeme prepoznavanja za YOLOv8 nano model je otprilike jedna sekunda [42].



**Slika 49. Raspberry Pi 4**

Uzimajući u obzir sve korake i potrebna vremena za njihovo izvršavanje, cijeli proces - od detekcije automobila, preko identifikacije registarskih tablica, do prepoznavanja teksta na njima - trajao bi otprilike 2 sekunde. ako je to relativno brzo, to vrijeme može biti dodatno smanjeno korištenjem naprednijih rubnih uređaja, kao što je NVIDIA Jetson, koji također integrira CUDA (engl. *Compute Unified Device Architecture*) podršku. CUDA je bitan dio uređaja jer omogućuje paralelno procesiranje na NVIDIA GPU-ovima, dok su tradicionalni CPU-ovi dizajnirani za serijsko procesiranje, odnosno izvršavanje instrukcija redom.

## 4.2. ONNX

Otvorena razmjena neuronskih mreža (ONNX - engl. *Open Neural Network Exchange*)

je standardni format dizajniran za predstavljanje modela strojnog učenja. Njegova primarna svrha je olakšati prijelaz modela između različitih okvira (engl. *framework*) dubokog učenja. Ovi formati su namijenjeni za implementaciju na krajnji hardver, a da pritom što bolje zadrže točnost.

Prije pojave ONNX-a, prijelaz modela treniranog u određenom okruženju, kao što je PyTorch, u drugo okruženje kao što je TensorFlow, zahtijevao je opsežno prilagođeno kodiranje pretvorbe. Ova je složenost također bila očigledna pri prilagodbi modela za različite platforme ili specijalizirane mehanizme za prepoznavanje, poput NVIDIA-inog TensorRT-a. Ovaj izazov potaknuo je Microsoft i tvrtku sada poznatu kao Meta (bivši Facebook) da uvedu ONNX.

U biti, služi kao posrednik između različitih okruženja dubokog učenja, pojednostavljujući procese razmjene modela. Većina glavnih okvira ima ugrađene pomoćne programe koji omogućuju izvoz njihovih izvornih modela u ONNX format. Nakon što se model izveze u ovom formatu, može se podvrgnuti zaključivanju na mnoštvu platformi. Također, ONNX format služi za optimizacije prilagođene različitim hardverima, uključujući CPU-ove, GPU-ove, TPU-ove (engl. *tensor processing unit*) i rubne uređaje.

Za detaljnije potrebe vezane uz hardver ili performanse, poboljšanja nakon izvoza mogu se postići pomoću alata kao što je ONNX grafičke optimizacije (engl. *graph optimizations*).

Ovo naravno nije jedini format koji se koristi. Za TensorFlow modele odavno već postoji tzv. TFLite format, optimiziran za slabije uređaje i operativne sustave poput Android-a, iOS-a, Linux-a i sličnih.

Svi modeli prije korištenja u proizvodnji se pretvaraju u formate poput ovoga. Nakon pretvorbe u ONNX, moguće je detaljnije vizualizirati bilo koju arhitekturu u već spomenutoj biblioteci netron, zbog „univerzalnog“ formata zapisa.

# 5. ZAKLJUČAK

Rad detaljno opisuje cijeli proces detekcije i očitavanja teksta registarskih pločica. Sagledava se područje koje nas sve više okružuje i koje, uz dovoljno računalne snage i kreativnosti, može rješavati čovjeku monotone zadatke. Jedan od tih zadataka je upravo ovaj, koji se koristi u mnogim industrijama – od parkirališta do naplate cestarina.

Problem detekcije objekata opisan je kao specijalizirani problem strojnog učenja. Pomoću prijenosnog učenja na modelima treniranim za opće prepoznavanje objekata, trenira se model specifično za detekciju registarskih pločica vozila. Relativno velik skup podataka proširuje se pomoću raznih metoda augmentacije podataka, a koje su relevantne za ovu vrstu zadatka. Detaljno su opisani načini na koje model „provlači“ ulaznu sliku kroz svoju arhitekturu te se objašnjava svrha svakog njenog bloka . Korišteni su najnoviji jednostupanjski modeli YOLOv7 i YOLOv8. Preciznosti modela evaluirana je na odvojenom skupu podataka, na kojem su postignute najveće točnosti od 0.692 i 0.735 prema strogom kriteriju mAP@0,50:0,95.

Očitavanje teksta, nakon izolacije registarske pločice, primarno ovisi o kvaliteti slike. Visokokvalitetan pogled sprijeda gotovo ne generira pogreške, dok faktori poput udaljenosti i orijentacije vozila predstavljaju poseban izazov. Metodama predobrade poboljšava se preciznost očitavanja, a transformacijom perspektive nastoji postići frontalni pogled.

Cjelokupni proces zaključivanja traje kratko jer su odabrani kompaktni modeli. Predložena je i hardverska implementacija za sustave parkirališta i garaža. Iako detekcija registarske pločice i očitavanje njezina teksta koriste odvojene modele, ovakav se sustav može objediniti u jedan model.

# Literatura

[1] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed., Springer, 2021.

[2] B. M. A., Mind As Machine: A History of Cognitive Science, Oxford University Press, 2006.

[3] M. D., Vision: A Computational Investigation into the Human Representation and Processing of Visual Information, W.H. Freeman, 1982.

[4] W. T. a. A. E. H. Freeman, »The design and use of steerable filters,« *IEEE Transactions on Pattern Analysis and Machine Intelligence*, svez. 13, br. 9, pp. 891-906, 1991.

[5] S. F. C. C. F. a. M. J. Belongie, »Spectral partitioning with indefinite kernels using the Nystrom extension.,« European Conference on Computer Vision (ECCV), 2002.

[6] S. Yeung, »Introduction to Computer Vision,« 2015. [Mrežno]. Available: https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html. [Pokušaj pristupa Rujan 2023].

[7] Y. e. a. LeCun, »The MNIST database,« 1998. [Mrežno]. Available: http://yann.lecun.com/exdb/mnist/. [Pokušaj pristupa Rujan 2023].

[8] B. a. F. D. J. Olshausen, »How close are we to understanding V1?,« *Neural Computation*, svez. 17, pp. 1665-1699, 2005.

[9] L. e. a. Von Melchner, »Visual behaviour mediated by retinal projections directed to the auditory pathway,« *Nature*, svez. 404, br. 6780, pp. 871-876, 2000.

[10] R. D. E. e. al., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge: MIT Press, 1986.

[11] G. E. a. S. T. J. Hinton, »Learning and relearning in Boltzmann machines,« *Parallel Distributed Processing*, svez. 1, pp. 282-317, 1986.

[12] I. B. Y. a. C. A. Goodfellow, Deep Learning, MIT Press, 2016.

[13] Y. J. Y. L. H. A. R. S. I. J. M. R. David Tsai, »Large-Scale Image Annotation using Visual Synset,« Google Research, 2011.

[14] [Mrežno]. Available: https://towardsdatascience.com/coco-data-format-for-object-detection-a4c5eaf518c5. [Pokušaj pristupa Rujan 2023].

[15] [Mrežno]. Available: https://sh-tsang.medium.com/review-imagenet-real-are-we-done-with-imagenet-38bca7d13531. [Pokušaj pristupa Rujan 2023].

[16] J. e. a. Shotton, »Real-time human pose recognition in parts from single depth images,« *IEEE ComputerSociety Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[17] N. M. a. C. D. J. Su, »The affective growth of computer vision,« *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[18] A. G. a. R. G. A. Shrivastava, »Training region-based object detectors with online hard example mining,« CVPR, 2016.

[19] T.-Y. L. e. al., »Focal Loss for Dense Object Detection,« Facebook AI Research (FAIR), 2018.

[20] K. E. A. v. d. S. T. G. A. W. M. S. J. R. R. Uijlings, »Selective Search for Object Recognition,« Int J Computer Vision, 2013.

[21] J. M. Viola P., »Rapid Object Detection using a Boosted Cascade of Simple Features,«

CVPR, 2001.

[22] K. H. R. G. a. J. S. Shaoqing Ren, »Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,« 2016.

[23] S. D. R. G. A. F. Joseph Redmon, »You Only Look Once: Unified, Real-Time Object Detection,« University of Washington, Allen Institute for AI, Facebook AI Research, 2016.

[24] »TensorFlow documentation,« [Mrežno]. Available: https://www.tensorflow.org/tutorials/load_data/tfrecord#tfrecords_format_details. [Pokušaj pristupa Rujan 2023].

[25] B. e. a. LeCun, »Gradient-Based Learning Applied to Document Recognition,« 1998.

[26] X. Z. S. R. J. S. Kaiming He, »Deep Residual Learning for Image Recognition,« Microsoft Research, 2015.

[27] A. B. a. H.-Y. M. L. Chien-Yao Wang, »YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,« Institute of Information Science, 2022.

[28] Ultralytics, »Ultralytics YOLOv8,« 2023. [Mrežno]. Available: https://github.com/ultralytics/ultralytics. [Pokušaj pristupa Rujan 2023].

[29] R. P. Q. V. L. Mingxing Tan, »EfficientDet: Scalable and Efficient Object Detection,« Google Research, 2020.

[30] C.-Y. W. H.-Y. M. L. Alexey Bochkovskiy, »YOLOv4: Optimal Speed and Accuracy of Object Detection,« arXiv, 2020.

[31] L. e. al., »Deeply-Supervised Nets,« arXiv, 2014.

[32] D. M. J. N. M. S. P. T. P. T. Nitish Shirish Keskar, »On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima,« arXiv, 2017.

[33] G. u. RangeKing, 2023. [Mrežno]. Available: https://github.com/RangeKing. [Pokušaj pristupa Rujan 2023].

[34] X. Z. S. R. a. J. S. Kaiming He, »Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,« 2015.

[35] e. a. LI, »PP-OCRv3: More Attempts for the Improvement of Ultra Lightweight OCR System,« Baidu Inc., 2022.

[36] e. a. Du, »SVTR: Scene Text Recognition with a Single Visual Model,« Fudan University, Baidu Inc., 2022.

[37] D. P. G. R. Lin Tsung-YI, »Feature Pyramid Networks for Object Detection,« FAIR, Cornell University and Cornell Tech, 2017.

[38] e. a. Du, »PP-OCRv2: Bag of Tricks for Ultra Lightweight OCR System,« Baidu Inc., 2021.

[39] S. G. a. W. R. Wojciech Bieniecki, »Image Preprocessing for Improving OCR Accuracy,« MEMSTECH, 2007.

[40] OpenCV, »OpenCV: Contours,« 2023. [Mrežno]. Available: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html. [Pokušaj pristupa Rujan 2023].

[41] Axiomtek. [Mrežno]. Available: https://www.axiomtek.com/. [Pokušaj pristupa Rujan 2023].

[42] D. Eliuseev, »YOLO object detection on the Raspberry Pi,« 2023. [Mrežno]. Available: https://towardsdatascience.com/yolo-object-detection-on-the-raspberry-pi-6de3629256fa. [Pokušaj pristupa Rujan 2023].