Detekcija slobodnih parking mjesta pomoću YOLO algoritma

Tomašević, Jure

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

Detekcija slobodnih parking mjesta pomoću YOLO algoritma

Jure Tomašević

Split, travanj 2024.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu
Prirodoslovno-matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Detekcija slobodnih parking mjesta pomoću YOLO algoritma

Jure Tomašević

SAŽETAK

U ovom radu napravljeno je istraživanje o razvoju detekcije objekata na području detekcije objekata slobodnih mjesta. Predstavljena je You Only Look Once obitelj algoritama i njene karakteristike s naglaskom na YOLOv5 verziju algoritma. U njemu je izrađen model za prepoznavanje koji je za uvježbavanje, validaciju i testiranje koristio skupove podataka PKLot i CNRPark+EXT. Također, detaljno je opisan postupak izrade modela i dodatne mogućnosti koje YOLO nudi.

Ključne riječi: detekcija i klasifikacija objekata, YOLO, slobodna parking mjesta, PKLot, CNRPark+EXT

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 48 stranica, 22 grafičkih prikaza, 4 tablice i 82 literaturni navod. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Saša Mladenović,** *redoviti profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Neposredni voditelj:

Dino Nejašmić, *predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Prof. Dr. sc. Saša Mladenović,** *redoviti profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Doc. Dr. Sc. Goran Zaharija, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dino Nejašmić, *predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: travanj 2024

Basic documentation card

Graduate Thesis

University of Split
Faculty of Science
Department of Computer Science
Ruđera Boškovića 33, 21000 Split, Croatia

Detection of vacant parking space using YOLO algorithm

Jure Tomašević

ABSTRACT

In this paper, research on the development of object detection in the context of free parking space has been conducted. There is a representation of You Only Look Once family of algorithms, with a focus on YOLOv5 version of algorithm. Recognition model was created within the paper which utilized the PKLot and CNRPark+EXT datasets for training, validation and testing. Also, paper provides a detailed description of the model creation process and additional capabilities that YOLO offers.

Key words: Object detection and classification, YOLO, vacant parking spaces, PKLot, CNRPark+EXT

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 48 pages, 22 figures, 4 tables and 82 references

Original language: Croatian

Mentor: **Saša Mladenović, Ph.D.** *Full Professor of Faculty of Science, University of Split*

Supervisor: **Dino Nejašmić,** *Lecturer of Faculty of Science, University of Split*

Reviewers: **Saša Mladenović, Ph.D.** *Full Professor of Faculty of Science, University of Split*

Goran Zaharija, Ph.D. *Assistant Professor of Faculty of Science, University of Split*

Dino Nejašmić. *Lecturer of Faculty of Science, University of Split*

Thesis accepted: April 2024

Sadržaj

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam diplomski rad s naslovom Detekcija slobodnih parking mjesta pomoću YOLO algoritma izradio samostalno pod voditeljstvom redovitog profesora Saše Mladenovića. U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u diplomskom radu na uobičajen, standardan način citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Student:

Jure Tomašević

Uvod

Svakodnevni smo svjedoci sve većeg broja automobila koji nas okružuju. S rastućim brojem vozila stvaraju se razni problemi: povećava se rizik od nesreća, zagađuje se zrak, a uzrokuju se i velike gužve pogotovo po većim sredinama. Nedostatkom efikasnih sustava upravljanja problemi s parkiralištem su postali sve učestaliji i kompleksniji: vozači se susreću s dugim potragama za slobodnim parkirnim mjestom koje najčešće i ne nađu pa im kao parkiralište posluže i površine čija namjena često nije parkiranje.

Razmatrajući problem s parkiranjem, umjetna inteligencija ima potencijal pružiti efikasne strategije upravljanja parkiranjem implementacijom sustava prepoznavanja slobodnih parkirališnih mjesta. Takvo što je moguće postići na dva načina:

- Korištenjem senzora za očitavanje stanja parkirnog mjesta
- Korištenjem kamere koja bi obradom snimke mogla dostavljati informacije o stanju na parkiralištu

Samo korištenje senzora, iako ima svoje prednosti može biti veoma skupo za implementaciju i zahtjeva mnogo vremena kako bi se sustav mogao pustiti u rad. S druge strane, osim što je korištenje kamere mnogo jeftinija opcija, dobar dio parkirališta zbog sigurnosti već koristi kamere pa su samim time i praktičnije. Osim toga, lakše pokrivaju veće područje što ih čini lakšim za održavanje i mogu biti multifunkcionalne (mogu se koristiti za sigurnost, čitanje registracijskih tablica itd.).

Umjetna inteligencija može analizirati podatke koji se prikupe pomoću kamere i omogućiti vozačima informacije o dostupnosti slobodnih parking mjesta na određenom parkiralištu preko aplikacija ili navigacijskih sustava bio on dio sustava za naplaćivanje ili ne. Također, prepoznavanjem slobodnih mjesta može se izraditi sustav koji bi mogao osigurati dinamičnu naplatu cijena parkiranja ovisno o popunjenosti prostora što bi potaklo bolju raspodjelu parkirnih mjesta i smanjilo broj zauzetih mjesta za duže vremensko razdoblje. Implementacija ovih tehnologija može rezultirati smanjenjem vremena potrage za parking mjestom što bi za rezultat imalo i smanjenje prometnih gužvi te manjak stresa kod svakodnevnih gradskih vožnji. U ovom radu cilj je izraditi model koji će detektirati slobodna mjesta na parkiralištu korištenjem YOLOv5 algoritma koji u zadacima prepoznavanja

objekata ima jedan od najboljih omjera preciznosti i brzine što bi koristilo kod pravovremenog ažuriranja informacija o pojedinom parkiralištu. Uz to, prednost YOLOv5 algoritma je u njegovoj skalabilnosti: model je fleksibilan i može se prilagoditi različitim postavkama parkirališta kao i različitim vrstama kamere i okruženjima. YOLOv5 je lako dostupan i ima dobru podršku zajednice te se njegovi modeli mogu prilagoditi specifičnim situacijama u budućnosti. Nakon YOLOv5 objavljeni su i noviji algoritmi iz YOLO obitelji, no kako je u ovom radu cilj prikazati funkcioniranje YOLO algoritama na najlakši mogući način tu se YOLOv5 iskazao kao najbolji kandidat (nekoliko dana prije završetka pisanja ovog rada objavljen je YOLOv9, no zbog svoje rane faze razvoja nije testiran za korištenje). Za uvježbavanje, validaciju i testiranje modela koristit će se javni skupovi podataka koji se koriste na problemu detekcije slobodnih parking mjesta.

1. Odabir tehnologija i alata

Posljednjih godina svjedoci smo strelovitog razvoja u području strojnog učenja, a pogotovo računalnog vida čiji će razvoj biti predstavljen u nastavku ovog rada. Taj razvoj pratio je i razvoj mnoštva alata i okvira koji se danas svakodnevno upotrebljavaju pri razvijanju rješenja koja se koriste u svakodnevnim životima ljudi. Tehnologije koje se koriste u ovom radu su donekle uvjetovane samim algoritmom koji se koristi – YOLOv5 [1] je napravljen u PyTorch-u te samim time zahtjeva korištenje Pythona i njegovih biblioteka kako bi funkcionirao. Također, autori YOLOv5 potiču korištenje Roboflow-a [2] kako bi svoje skupove podataka prilagodili što brže i jednostavnije. Sustav bilježnica je iznimno popularan alat u svijetu strojnog učenja, a autori nude već gotove bilježnice kako bi korisnik mogao što prije krenuti sa uvježbavanjem svojih modela. Zbog limitiranih osobnih resursa, odabrano je korištenje Google Colaboratory-a [3] čije korištenje omogućava besplatno korištenje Google-ovih resursa.

1.1. Python

Python je interpreterski, objektno-orijentirani programski jezik visoke razine sa dinamičnom semantikom kojeg je stvorio Gudio van Rossum 1991. godine [4]. Danas se najviše koristi pri izradi skripti i automatizaciji poslova, no prednjači i u područjima analize podataka, strojnog učenja i umjetne inteligencije.

Postoji nekoliko razloga zašto Python prevladava u navedenim područjima:

- Veliki broj biblioteka – biblioteka je skup modula koji se sastoje od prethodno napisanog koda koji omogućava korisnicima da lakše ostvare određene funkcionalnosti. Neke od dostupnih koje se koriste pri radu sa različitim vrstama podataka su: Sckit-learn [5], Pandas [6], Keras [7], TensorFlow [8], Matplotlib [9], NLTK [10] itd.
- Lak početak kodiranja – Python je programski jezik koji je najbliži svakodnevnom engleskom jeziku, što i znanstvenicima koji nemaju previše doticaja s programiranjem, a posao im zahtjeva njihovo korištenje, omogućava relativno brz pristup alatima koje nudi umjetna inteligencija

- Fleksibilnost – nudi programerima opciju biranja onog stila programiranja u kojem se osjećaju najbolje: imperativno, funkcionalno, objektno-orijentirano, proceduralno
- Neovisnost o platformama – korištenje Pythona za strojno učenje omogućava pokretanje na svim dostupnim platformama uz manje izmjene.
- Vizualizacije – neke od mnogih Python-ovih biblioteka su i one koje se koriste kao vizualizacijski alati što olakšava prezentaciju podataka s kojima se radilo
- Zajednica – postoji mnogo resursa koji olakšaju korištenje Pythona među kojima su i razni forumi na kojima se može potražiti pomoć vezano za bilo kakvo pitanje koje bi korisnik mogao imati i ta zajednica posljednjih godina konstantno raste.

Iako postoje različita okruženja za izradu programa koja podržavaju programski jezik Python, za područje strojnog učenja uobičajeno je koristiti okruženje koje osigurava laku izradu prototipa. Kao industrijski standard nametnule su se razne inačice interaktivnih Jupyter bilježnica.

1.2. Jupyter bilježnica

Jupyter bilježnica je *open source* web aplikacija omogućava kombiniranje koda, vizualizacije, obogaćenog teksta i drugih značajki koje su korisne u zadacima koje zahtijevaju obradu podataka [11]. Kompatibilan je sa tri programska jezika: Julia, Python i R po čemu je i dobio ime, a postoje i dodaci koji osiguravaju podršku i za druge programske jezike. Same bilježnice mogu se pokrenuti na lokalnom uređaju ili u oblaku, no kako operacije koje se provode tokom uvježbavanja zahtijevaju velik broj računalnih resursa koja lokalna računala često nemaju, bar ne za izvođenje u nekom relativno kratkom vremenu. Zato se u ovom radu odlučio koristiti pristup rada u oblaku, a odabran je Google Colaboratory koji nudi besplatan pristup računalnim resursima, koristi prethodno instalirane biblioteke te ima integraciju sa Google Drive-om preko kojeg se mogu povezati potrebni podaci.

1.3. Google Colaboratory

Google Colaboratory (Colab) je Google-ov servis koji omogućava izradu i korištenje Jupyter bilježnica korištenjem web preglednika. Kod, obogaćeni tekst ili naredbe jezgre koje se nalaze u bilježnicama se mogu lako dijeliti i simultano mijenjati između više korisnika i

dokumentirati u jednu sveobuhvatnu bilježnicu koja može sadržavati kod, obogaćeni tekst i grafove. Sam servis je napravljen kako bi bio prikladan kako za edukaciju tako i za eksperimentiranje sa podatkovnom znanošću, strojnim učenjem i umjetnom inteligencijom s obzirom na Google-ovu predanost razvitku polja umjetne inteligencije [3].

Za razliku od svog konkurenta, Jupyter Notebook-a, Colab nudi mogućnost lakšeg timskog rada, ne zahtjeva lokalne instalacije jer se sve događa na cloud-u, lakše dijeljenje dokumenata preko Google Drive-a i omogućava besplatan pristup Google-ovim računalnim resursima kao što su GPU i TPU(Googleov aplikativno specifičan integrirani sklop izrađen za brzo rješavanje kompleksnih matrica i vektorskih operacija).

Kako je ideja ovog rada što lakša i brža izrada prototipa modela i odabrani su najpristupačniji alati kako bi se postigla lakoća postoji još jedan alat koji olakšava prikupljanje i organizaciju i procesuiranje podataka, a to je Roboflow.

1.4. Roboflow

Roboflow je web aplikacija koji korisnicima olakšava rješavanje problema u području računalnog vida. Omogućuje skup alata koji programerima pomažu u izradi aplikacija računalnog vida, neovisno o tome koliko je njihovo iskustvo i sposobnosti. Iako nudi treniranje i korištenje modela te njihovo postavljanje na uređaje u kojima će se koristiti, Roboflow sadrži širok skup mogućnosti kada su u pitanju skupovi podataka: omogućava njihovo prikupljanje, organiziranje, označavanje, procesuiranje (formatiranje slika, razne augmentacije kako bi se povećao korišteni skup podataka) i provjeru ispravnosti skupa podataka[2]. Također, skupove podataka je moguće javno dijeliti, a Roboflow omogućuje izvoz skupova podataka u raznim formatima ovisno o algoritmima koji se koriste. S obzirom da su oba skupa podataka postavljena na Roboflow-u, odatle su preuzeti zbog jednostavnosti korištenja.

Kako bi se razumjelo kako funkcionira detekcija i klasifikacija objekata, dobro je razumjeti kako se to područje razvijalo kroz godine. U svrhu toga, napravila se pretraga za pregledom razvoja područja. Istaknuo se članak „*Object Detection in 20 Years: A survey*“ koji je opisan u nastavku ovog rada. U njemu su autori napravili temeljitu analizu razvoja područja kroz pregled dostupne literature i istraživačkih radova.

2. Općenito o detekciji i klasifikaciji objekata

Detekcija objekata je tehnika računalnog vida koja služi kako bi se locirali i identificirali objekti određene klase(kao što su ljudi, vozila, životinje i slično) na slikama ili video zapisima. Postoji nekoliko pristupa rješavanja problema detekcije objekata [12]:

- Usporedba geometrijskih obilježja – kod ovakvog pristupa, jedna ili više značajki se izdvajaju i ciljani objekti modelirani su po svojstvima tih značajki. Te značajke mogu biti oblici, veličine ili boje objekata.
- Detekcija objekata temeljena na predlošku – ako je dostupan predložak koji opisuje određeni objekt, detekcija postaje proces usklađivanja značajki između predložaka i slika koje se analiziraju.
- Detekcija objekata korištenjem metoda dubokog učenja – korištenje konvolucijskih neuronskih mreža omogućilo je prilagodbu ekstrahiranih značajki što je rezultiralo u značajnom rastu točnosti. S vremenom su se razvila dva načina korištenja ovih metoda – detektori s jednom i detektori s dvije razine

U zadnja dva desetljeća, razvoj detekcije objekata je prošao kroz dva perioda: period tradicionalne detekcije objekata koji je trajao do 2014. godine i period detekcije temeljen na dubokom učenju koji je počeo nakon toga. U članku „*Object Detection in 20 Years: A survey*“ [12] napravljen je pregled razvoja detekcije objekata od njegovih početaka:

U 90-im godinama prošlog stoljeća većina algoritama za detekciju objekata izgrađena je na temelju ručno izrađenih značajki. Zbog nedostatka učinkovite reprezentacije slika u to vrijeme ljudi su morali dizajnirati sofisticirane prikaze značajki.

2001. godine P. Viola i M. Jones [13] su postigli detekciju ljudskih lica u stvarnom vremenu i njihov algoritam je bio nekoliko desetaka puta brži od drugih algoritama u to vrijeme uz usporedivu preciznost. Njihov detektor je prolazio kroz svaku moguću lokaciju i mjere na slici kako bi odlučio nalazili se ljudsko lice na slici.

U 2005. godini Dalal i Triggs [14] su predstavili HOG(*Histogram of Oriented Gradients*) deskriptor značajki što je bio značajan napredak jer su ponudili efektivan način reprezentiranja i detekcije oblika objekata i struktura u slikama.

2008. godine je objavljen *Deformable Part-based Model* [15] koji se pojavio kao proširenje na HOG detektor te je koristio „dekompoziciju objekta“. Iako su današnji detektori puno precizniji, DPM je imao veliki utjecaj u njihovom radu.

Nakon dosta godina slabog napretka, 2014. godine R. Girshick sa suradnicima [16] je predstavio *Region-based Convolution Neural Network*(RCNN) koji je označio početak rapidnog rasta i razvoja detekcije objekata. Ideja RCNN je da počinje sa izdvajanjem skupa prijedloga za moguće objekata. Zatim se svaki prijedlog skalira na fiksnu veličinu slike i šalje se u konvolucijski model kako bi se izvadile značajke. Nakon toga, korist se linearni *Support Vector Machine* klasifikator [17] kako bi predvidio prisustvo objekta u svakoj od regija i kako bi ih razdijelio u kategorije. Unatoč velikom napretku koji je omogućio, RCNN je imao očite nedostatke: velik broj preklapajućih prijedloga je vodio do iznimno spore detekcije.

Kasnije iste godine K. He i suradnici [18] objavili su *Spatial Pyramid Pooling Networks* (SPPNet) koji je riješio probleme koje je imao RCNN. Najveći doprinos bio je SPP sloj koji je omogućio generiranje reprezentacije fiksne veličine, neovisno o veličini slike bez ponovnog skaliranja. Kada se koristi SPPNet, mape značajki se iz cijele slike mogu računati samo jednom a nakon toga se reprezentacije proizvoljnih regija generiraju kako bi služile za treniranje detektora što značajno ubrzava detekciju bez gubitka preciznosti. 2015. godine R. Girshick je objavio *Fast* RCNN [19] detektor koji je donio nova unaprjeđenja u odnosu na RCNN i SPPNet po pitanju brzine i preciznosti, no i dalje je bio limitiran sa prijedlozima detekcija pa se i dalje postavljalo pitanje „Može li se generirati prijedloge objekata korištenjem CNN modelom?“. *Faster* RCNN [20] je bio prvi algoritam koji je radio prijedloge regija i radio detekciju. Prijedloge regija je radio pomoću *Feature Pyramid Network-a* (FPN) [21] koji se pokazao otporan na detekciji objekata koji se pojavljuju u raznim veličinama.

Svi navedeni algoritmi su bili dio takozvane detekcije s dvije razine: prvo se rade prijedlozi regija u kojima se objekt može nalaziti što smanjuje broj regija koje je potrebno analizirati, a zatim se provodi dublja analiza u kojoj algoritam izvodi klasifikaciju i daljnju lokalizaciju nad predloženim regijama. Naravno, ti koraci, iako su precizni zahtijevaju vrijeme, no neki zadaci u stvarnom životu zahtijevaju trenutnu reakciju.

Stoga je J. Redmon 2015. godine objavio *You only look once* (YOLO) [22] koji je napravio revoluciju uvođenjem novog koncepta računalnog vida: dok su se do tada problemi detekcije objekata smatrali klasifikacijskim problemima, YOLO detekciju objekata smatra

regresijskim problemom. YOLO je detektor s jednom razinom što znači da predviđa pravokutnike koji okružuju objekt u jednom koraku bez korištenja prijedloga regija i na taj način detekciju radi mnogo brže od detektora s dvije razine uz lošije rezultate po pitanju preciznosti, pogotovo za manje objekte. Kasnije verzije YOLO-a su poradile na preciznosti i brzini.

Kasnije je isti autor objavio još dvije verzije YOLO-a, a nakon toga se zbog etičkih razloga prestao baviti računalnim vidom. Nakon toga, razni autori su objavljivali druge verzije YOLO-a: A. Bochakovskiy objavio je YOLOv4 [23] ili YOLOv7 [24], C. Li YOLOv6 [25], a G. Joacher YOLOv5 [1] i YOLOv8 [26], najnoviju verziju YOLO-a koja se pokazala najučinkovitijom od dosad objavljenih. No, ono što YOLOv5 čini posebnim je jednostavnost korištenja: za razliku od ostalih verzija, YOLOv5 je napravljen na način da ga početnici mogu vrlo lako i brzo početi koristiti, dok njegove performanse, iako lošije od kasnijih verzija, i dalje pokazuju poprilično dobre rezultate.

Single Shot MultiBox Detector (SSD) [27] iz 2015. godine je također imao za cilj poboljšati manjkavosti YOLO algoritma. Poboljšao je preciznost i brzinu uvođenjem više-referentnih i više-rezolucijskih tehnika detekcije što je popravilo preciznost detekcije, pogotovo kod manjih objekata. Glavna razlika između SSD i drugih detektora je što SSD detektira objekte različitih veličina na različitim slojevima dok se kod drugih detekcija vodila samo na najvišim slojevima.

2017. Lin i suradnici [28] su se proučavali razloge zbog veće preciznosti detektora s dvije razine objavili su RetinaNet. Kao glavni razlog preciznijih rezultata detektora s dvije razine su naveli ekstremnu klasnu neravnotežu prednjeg i pozadinskog plana tijekom treninga. Kod RetinaNet-a su koristili funkciju gubitka zvanu fokalni gubitak [28], koja je nastala kao preinaka postojeće *cross entropy* funkcije gubitka [29] kako bi detektor stavio više fokusa na krivo klasificirane primjere tokom treniranja. Fokalni gubitak je omogućio detektorima s jednom razinom da ostvare usporedive rezultate po pitanju preciznosti zadržavajući visoku razinu detekcije.

Sve dosad spomenute metode su koristile prethodno definirane pravokutnike(eng. *anchor box*) kako bi omogućili klasifikacijske i regresijske značajke. Kako objekti u stvarnosti dolaze u različitim varijacijama lokacija, veličine, broja i slično, trebao se postavili velik broj referentnih pravokutnika kako bi bolje odgovarali svim mogućim oblicima koji se mogu pojaviti. No, mreža bi patila zbog prisutne neravnoteže kategorija i mnogo ručno dizajniranih hiperparametara i dugu konvergencije vremena. Stoga su H. Law i njegovi suradnici [30]

odbacili prethodne paradigme detekcije i pristupili su problemu na način da su označili ključne točke, odnosno kutove pravokutnika. Nakon dobivanja ključnih točaka, odvaja ih i ponovno grupira te točke pomoću dodatnih informacija kako bi se formirao pravokutnik vezan za objekt. CornerNet je tom preinakom nadmašio većinu detektora s jednom razinom u to vrijeme.

2019. X. Zhao je sa suradnicima [31] predstavio CenterNet, u kojem objekt smatra jednom točkom koja se nalazi u njegovom središtu i iz te točke određuje ostale atribute (veličina, orijentacija, lokacija itd.)

Carion N. je 2020. objavio *Detection transformer* (DETR) [32] – novi pristup detekciji objekata s transformirajućom neuronskom mrežom koja nakon tradicionalne konvolucijske mreže koristi transformerski enkoder [33] za predviđanje objekata. Ovaj pristup je bio prvi koji nije koristio ni prethodno definirane kvadrate ni prethodno definirane točke.

Postoji veliki broj metoda za detekciju objekata koje se mogu koristiti u različitim domenama. Prilikom odabira metode vodio sam se zahtjevima prema sklopovlju te dostupnom dokumentacijom i primjerima dobre prakse. Kako ovo istraživanje ima i praktičnu primjenu cilj je odabrati rješenje koje se nalazi u stabilnoj fazi razvoja, te rješenje koje ne pokazuje pretjeranu kompleksnost u postavljanju i korištenju. YOLOv5 je odabran zbog svoje stabilne faze razvoja, učinkovitosti u praktičnim scenarijima i lakoće implementacije.

2.1. Klasifikacija objekata

Zadatak klasifikacije objekata je tehnika nadziranog učenja gdje svaka instanca dodjeljuje klasi, što se označava vrijednošću određenim atributom klase. Cilj atributa je da ima kategoričke vrijednosti od kojih svaka odgovara klasi. Svaki se primjer sastoji od dva dijela, skupa vrijednosti atributa i ciljane vrijednosti atributa pri čemu se skupovi vrijednosti atributa koriste za predviđanje vrijednosti ciljanog atributa. Važno je napomenuti da za razliku od regresije kod koje su ciljane varijable kontinuirane, klasifikacijske ciljane varijable imaju diskretne vrijednosti. Zadaci klasifikacije mogu biti različiti [34]:

- Binarna klasifikacija – predstavlja klasifikaciju između dvije, međusobno razdvojene klase. U tom se slučaju klase određuju u binarnom formatu: 0 ili 1. Ovaj tip klasifikacije se koristi u ovom radu

- Više klasna klasifikacija – ima više međusobno odvojenih klasa, a cilj je previđanje kojoj od klasa pripada ulazni primjer.
- Više označna klasifikacija – kod ove vrste klasifikacije za svaki ulazni primjer se bira između 0 ili više klasa koje ne moraju biti međusobno razdvojene – npr. objekt može u isto vrijeme biti prepoznat kao vozilo i kao motor.
- Hijerarhijska klasifikacija – kod ovakve vrste klasifikacija klase se organiziraju u hijerarhijskoj strukturi prema sličnosti, pri čemu klase više razine predstavljaju generalnije kategorije, a one niže razine konkretnije i specifičnije klase

Za primjenu u ovom radu odabran je problem određivanja slobodnih parking mjesta u gradovima kao jedan od ključnih elemenata za olakšavanje prometne situacije koja u velikim gradovima stvara sve veće probleme zbog porasta broja vozila, a problem je kojem već neko vrijeme svjedočimo i u našoj državi.

Jedna od vodećih svjetskih tvrtki za prikupljanje i analizu podataka o stanju u prometu, Inrix, objavila je 2017. godine globalnu analizu ovog problema koja je obuhvatila 8700 gradova u 100 država. Širu analizu su napravili za tri države: Sjedinjene američke države, Ujedinjeno Kraljevstvo i Njemačku.

Rezultati za Njemačku, koja nam je geografski i kulturološki najbliža, pokazali su da vozači potroše 44 sata godišnje na traženje parking mjesta (do 65 sati u većim gradovima), a tijekom traženja potroše 40,4 milijarde eura [35].

Jedine informacije koje su dostupne za Hrvatsku su vezane za Zagreb, koji se uklapa u provedeno istraživanje s obzirom na veličinu grada, u kojem je prosječno vrijeme traženja parkirnog mjesta 48 sati godišnje. Uzimajući u obzir da se broj vozila dan danas neprekidno povećava, možemo zaključiti da navedene brojke danas dodatno rastu.

Po svemu navedenom, riječ je o problemu koji uz to što stvara svakodnevnu napetost i nervozu kod vozača, uzima svoj doprinos i na ekonomiji država. Iz tog razloga, ovaj je slučaj odabran za praktičnu primjenu provedenog istraživanja u ovom radu.

3. Pregled područja istraživanja problema slobodnih parking mjesta

Korištenjem PRISMA smjernica [36] izdvojeni su pregledni radovi kako bi se smanjio ukupan broj radova te su uključene i meta analize od kojih se prema području interesa ističe pregledni rad: „*A System Review on Computer Vision-Based Parking Lot Management Applied on Public Datasets*“ [37] u kojem je napravljen pregled skupova podataka korištenih na problemima detekcija slobodnih mjesta na parkiralištu. Osim toga, ovaj pregledni rad sadrži velik broj članaka relevantnih i za ovaj rad. Iz navedenog razloga je prvo dodatno istražena literatura koja se koristila u ovom preglednom radu nakon čega se proširio skup radova s onima koji su objavljeni nakon njega. Kako bi se članci mogli filtrirati uvedeni su uključujući i isključujući kriteriji kako bi se pronašli članci koji imaju najveću povezanost sa radom:

Uključujući kriterij:

- Članci koji koriste dva ili više javnih skupova podataka u svom radu

Isključujući kriterij:

- Članci koji koriste dronove/snimanje iz zraka za prikaz parkirališta
- Članci koji koriste skupove podataka koji nisu javno dostupni

Navedeni članak sadržava 105 naslova od kojih 42 nisu bila vezana za temu, a 52 su odbačena po kriterijima koji su navedeni u nastavku teksta te je tako ostalo devet članaka koji su povezani s temom.

Kako je taj rad napravljen u ožujku 2021. godine, noviji članci su pronađeni u citatnoj bazi *Web of science*. Pretraga je bila ograničena na period od 1.1.2021 do 20.6.2023, a obuhvaća radove koji su objavljivani u kategorijama „*Computer Science & AI*“ i „*Computer vision & graphics*“. Za pretragu su se koristile ključne riječi:

- *Parking lot occupany detection*
- *Robust parking space detection*
- *Parking space detection*

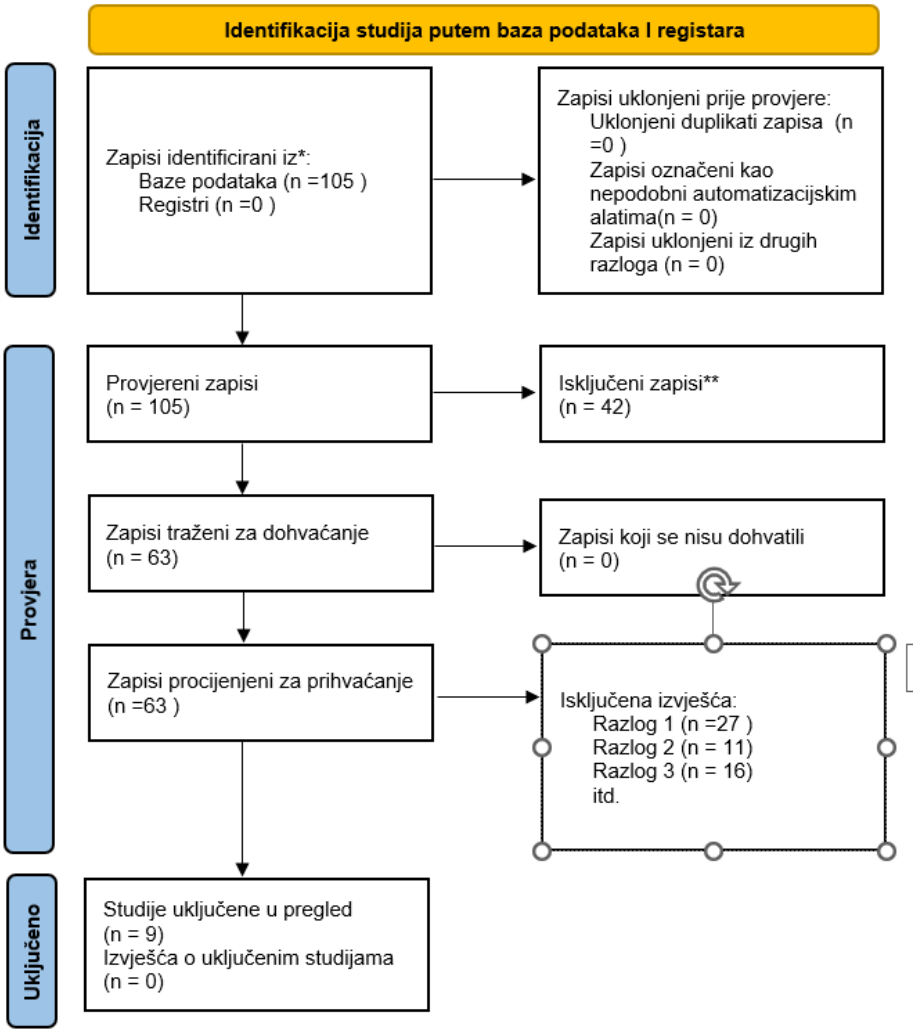
Od pronađenih 41 članka, 23 je odbačeno jer nije bilo vezano za navedenu temu, a još sedam članaka nije bilo javno dostupno. Nadalje, prethodno navedenim kriterijima je odbačeno još

devet članaka. Te su razmatranje uzeta dva članka. Kako se istraživanje se vršilo po PRISMA smjernicama te su generirana dva dijagrama, jedan na temelju navedenog članka, a drugi korištenjem baze Web of Science koji su prikazani na slici 3.1 i slici 3.2.

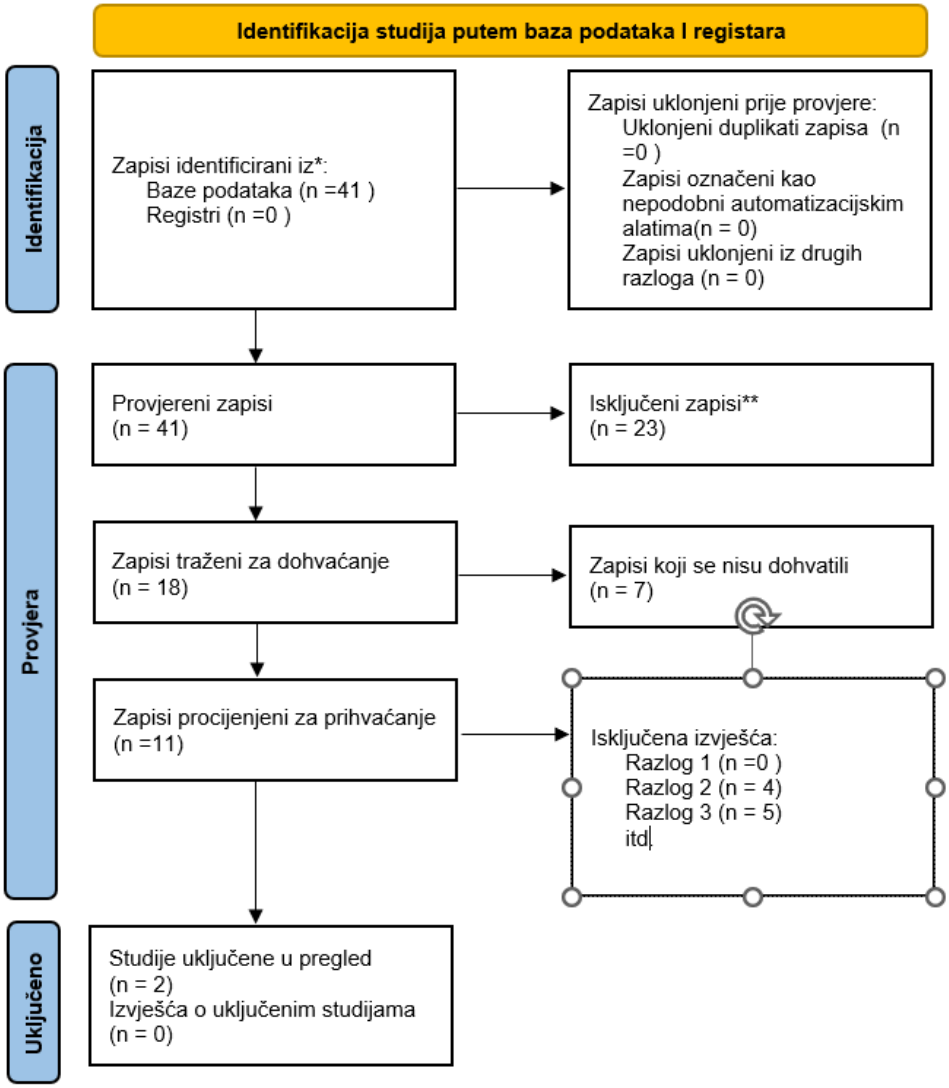
Kod mnogih skupova podataka postoji problem jer su snimani samo na jednom mjestu što može uzrokovati preuvježbanost modela s obzirom na to da je pozadina, uz određene promjene većinu vremena, ista. Također, kod korištenja jednog od razmatranih skupova podataka postoji problem vozila koja su parkirana kroz dugi vremenski period te tako mogu utjecati na uvježbavanje, a ovisno o načinu preraspodjele skupa isto vozilo na jednakom mjestu bi moglo naći u skupu za uvježbavanje i u skupu za testiranje. Kako je važno voditi računa o navedenom postavljen je uključujući kriterij za radove koji koriste dva ili više javno dostupna skupa podataka u svom radu jer je na taj način osigurana generalizacija modela. Navedeno ne znači kako se ne mogu koristiti određeni privatni skupovi podataka ukoliko je korišten za testiranje preciznosti modela.

Uz snimanje parkinga pomoću fiksne kamere instalirane na parkiralištu, može se koristiti i drugačiji pristup – snimanje parkinga iz zraka. Iako učinkovita i u pregledu istraženosti dosta korištena metoda [37], ova metoda predstavlja drugačiju vrstu problema od one koja je odabrana za ovaj rad, detekcije objekata pomoću fiksne kamere. Snimanje iz zraka omogućava snimanje velikih površina čime osigurava sveobuhvatan uvid u područje koje snima. Može osigurati dodatne mogućnosti kao što je navođenje do najbližeg slobodnog mjesta i sl. Ipak, postavljanje i održavanje takvih sustava može biti skupo za održavanje, a i u našem okruženju se u pravilu ne koriste, stoga je odabran sustav koji je realniji u našem podneblju.

Postoji nekoliko problema kod korištenja privatnih skupova podataka: jedan od temeljnih principa u znanstvenom istraživanju je mogućnost neovisnog ponavljanja rezultata i njihovog vrednovanja. Postavlja se pitanje što kada istraživanje nije ponovljivo? Nadalje, ostaje nejasna kvaliteta i etičnost samog skupa podataka koji može narušavati privatnost, kršiti autorska prava ili koristiti nereprezentativan uzorak te na taj način dobiti pristrane rezultate. Također, cilj istraživanja je ostvariti napredak u određenoj grani istraživanja, no ukoliko se ne koriste javno dostupni skupovi podataka, istraživači ograničavaju pristup svojim postignućima za razliku od javno dostupnih skupova podataka koji promoviraju suradnju i zajednički napredak.



Slika 3.1 PRISMA flow dijagram za referentni pregledni članak



Slika 3.2 PRISMA *flow* dijagram pregledanih radova

Nakon primjene kriterija isključivanja, ostalo je dvanaest članaka koji su odgovarali postavljenim uvjetima.

S obzirom na to da su u svom radu autori predstavili skup podataka koji se koristio u velikom broju radova koji su uslijedili, a i razvili su mrežu koju su kasnije mnogi drugi autori koristili za svoje eksperimentalne radove, rad Giuseppe Amata i suradnika iz 2016. godine [38] jedan je od najvažnijih radova u ovom području. U njemu su autori predstavili skup podataka CNRPark, preteču skupa podataka korištenog u ovom radu. Također, uvježbali su dvije različite konvolucijske mreže, mAlexNet i mLeNet, koje su pojednostavljene verzije postojećih arhitektura AlexNet[39] i LeNet [40]. mAlexNet je algoritam koji su kasnije koristili i drugi autori u svojim radovima na ovu temu pa ga se pobliže opisuje.

AlexNet je u vrijeme objavljivanja (2012.) bila jedna od najvećih konvolucijskih neuronskih mreža koja sadrži pet konvolucijskih i tri potpuno povezana sloja. Izlaz zadnjeg, potpuno povezanog sloja, spaja se na *softmax* koji može producirati tisuću oznaka klasa. Uz to koristi *Rectified Linear Unit* (ReLU) [41] kao aktivacijsku funkciju, omogućio je treniranje na više *Graphic Processing Unit-a* (GPU), *Local Response normalizaciju* [39], preklapajuće slojeve sažimanja [40] i augmentacije podataka i sloj isključivanja [42].

MAlexNet se temelji na AlexNet-u, a autori su iz originalne arhitekture uklonili treći i četvrti konvolucijski i potpuno povezani sloj, nakon svakog konvolucijskog sloja slijedi *max pooling*, LRN(*local response normalization*)[39] i ReLU osim u trećem sloju gdje nema LRN. Broj filtera na konvolucijskim slojevima 1-3 i broj neurona u četvrtom sloju je značajno smanjen kako bi odgovarao problemima s dimenzijama. Za potpuno povezan četvrti i peti sloj, nije preuzeta regulacija sloja isključivanja koja se inače koristi za izbjegavanje preuvježbanosti.

Godinu dana nakon objave ovog članka, isti autori uz dodatak Carla Meghinija su objavili i drugi članak na istu temu [43]. Njihova ideja je bila napraviti rješenje koje detekciju izvršava direktno na kameri bez korištenja centralnog servera što bi povećalo brzinu donošenja odluka zbog smanjenja vremena utrošenog na komunikaciju sa serverom. U njemu su objavili CNRPark+EXT skup podataka koji se koristi u ovom radu, a koji koristi devet kamera kako bi dobili slike iz različitih kutova. Autori su se ponovno koristili arhitekturom mAlexNet.

A.B. Ebren i B. Bolat su 2020. objavili rad u kojem su koristili ResNet [44] arhitekturu koja je imala jednu od najdubljih mreža ikad pokrenutih na ImageNet [39] skupu podataka te je odigrala ključnu ulogu u popularizaciji veoma dubokih neuronskih mreža [45]. Autori su u svom radu uvježbavali samo zadnji sloj ResNet-a i njihove težine koristili na ImageNet skupu podataka metodom *transfer learning*-a [46].

U članku objavljenom 2021. godine V. Dhuri i suradnici su koristili VGG16 [47] konvolucijsku mrežu kako bi iz slika izvukli značajke, a zatim bi vektorizirane izlaze proslijedili u potpuno povezane slojeve za binarnu klasifikaciju [48]. U svom su radu primijenili transfer learning.

F. Dornaika i suradnici su 2019. objavili rad u kojem su predstavili piramidalni deskriptor [49] na više razina koji usvaja piramidalnu reprezentaciju generiranu iz originalne slike. Kod svih klasifikacija se koristi nelinearni *Support Vector Machine* klasifikator [17], a

klasifikacija se izvodi korištenjem različitih deskriptora. Isti su autori iste godine objavili i članak u kojem su testirali razne deskriptore te predložili nekoliko varijanti *Local Binary Pattern*-a (LBP) [50], a za klasifikaciju su koristili Support Vector Machine klasifikator.

Članak autora M.S. Faraga i suradnika objavljen je 2020. godine. U svom radu su uspoređivali tradicionalne metode s AlexNet mrežom i predložili svoju izvedbu konvolucijske neuronske mreže [51].

U radu M. A. Merzouga i suradnika autori su predstavili cijeli sistem automatskog prepoznavanja slobodnih parking mjesta i komunikacije sa vozačima [52]. Za automatsko prepoznavanje parking mjesta koristi se MobileNetV2 [53] arhitektura koja je slična običnoj neuronskoj konvolucijskoj mreži, osim konvolucijskog dijela. MobileNet koristi odvojive konvolucije koje se sastoje od dvije operacije: dubinske i točkaste. Model je treniran na MobileNet arhitekturi i nakon toga uspoređen brzinom zaključivanja sa dvije druge metode: ResNet-50 [44] i YOLOv3 [54] od kojih se pokazao bržim.

M. Kolhar i A. Alameen u svom radu iz 2021.godine su uz automatsko prepoznavanje parking mjesta radili i na automatskom prepoznavanju registracijskih oznaka [55]. Za prepoznavanje mjesta, autori su se koristili svoju varijaciju već spomenutog mAlexNet koju su nazvali mmAlexNet.

U radu S. Nurullayev-a i Sang-Woong Lee-a iz 2019. godine autori su koristili arhitekturu koju su nazvali „dilatirana konvolucijska neuronska mreža“ [56]. Svoju su strukturu autori nazvali CarNet, a dilatiranu su konvoluciju koristili kako bi izbjegli preduboko učenje, kako bi se model fokusirao na veće značajke, zanemarujući manje.

2020. godine A. Varghese i G. Sreelekha su objavili rad gdje su radili na detekciji označenih i neoznačenih parking mjesta [57]. Kod označenih parking mjesta koristili su *Bag of features* [58] koji mapira lokalne značajke u „torbu“ odabranih, neporedanih predložaka značajki. Za detekciju su koristili *Support Vector Machine* klasifikator.

Li Wang i suradnici su u svom radu iz 2023. godine predložili novu značajku za ekstrakciju značajki koji su nazvali „*Global perception feature extractor*“ (GPFE) [59]. Autori su u svom radu testirali mAlexNet i ShuffleNet [60], konvolucijsku mrežu koja je dizajnirana za mobilne uređaje sa ograničenjima u računalnim operacijama. Prvo su testirali detekciju samo na mrežama, a nakon toga uz dodatak GPFE sa i bez dodatka modula klasifikacijske susretljivosti.

2021. godine su S. Goumiiri i suradnici u svom radu izradili konvolucijsku mrežu koja se sastoji od samo jednog konvolucijskog sloja nakon kojeg slijedi max-pooling sloj. [61]

Rezultati koje su autori postigli u svojim radovima uz algoritme i skupove podataka koje su koristili se nalaze u tablici 3.1:

Tablica 3.1 Rezultati dobiveni u pregledanim radovima

Rad	Algoritam	**Skupovi podataka**	Rezultati
Amato (2016)	mAlexNet	PKLot i CNRPark	90.4%
Amato (2017)	mAlexNet	PKLot i CNRPark+EXT	90.13%-99.64%
Bolat (2020)	Resnet	PKLot i CNRPark	97.36%
Dhuri (2021)	VGG16	PKLot, CNRPark, CNRPark+EXT, privatni skup podataka	87.17%
Dornaika (2019)	SVM	PKLot, CNR, hibridni skup podataka	90.19%
Farag (2020)	Vlastita neuronska mreža	PKLot, CNRall	79-84%
Kolhar (2021)	mmAlexNet	PKLot, CNRPark	93.71%-99.54%
Nurullayev (2019)	CarNet	PKLot, CNRPark	94.77%
Varghese (2019)	SVM	PKLot, CNRPark, CNRPark+EXT	65.31% - 82.88%
Wang (2023)	mAlexNet/ShuffleNet	PKLot, CNRPark+EXT	89.21%-91.21%
Goumiri (2021)	Jednoslojna CNN	PKLot, CNRPark, CNRPark+EXT	98.75%-99.84%

Jedna od najvažnijih stavki kod zadataka strojnog učenja je zasigurno odabir skupa podataka koji će se koristiti za uvježbavanje i evaluaciju modela. Skup podataka koristimo kako bi model mogao naučiti uzorke i odnose u skupovima podataka, te zatim to znanje iskoristiti za rješavanje zadatka kojeg smo mu zadali. Odabir lošeg skupa podataka rezultirati će u nepreciznim modelima koji imaju velike greške pri detekciji i klasifikaciji ili model koji ima pristrana predviđanja uzrokovana preuvježbanosti ili podvježbanosti što posljedično dovodi do protraćenih resursa, bilo vremenskih ili financijskih. Stoga je odabir skupa podataka veoma važna stavka pri izradi modela.

4. Odabrani skupovi podataka

S obzirom na važnost samih podataka i njihove kvalitete, ovo poglavlje detaljnije opisuje skupove podataka koji su korišteni pri ovom radu kako bi se lakše razumjelo zašto su odabrani baš navedeni skupovi. Uz dva korištena skupa podataka opisan je i COCO [62] skup podataka na kojem su prethodno uvježbane težine u prethodno uvježbanim verzijama YOLOv5 [1]:

4.1. COCO

COCO je skraćenica od „*Common Objects in Context*“ i riječ je o masivnom skupu podataka koji se koristi detekciju objekata, segmentaciju i generiranje opisa slike. Razvio ga je Microsoft u suradnji sa nekoliko akademski institucija 2014. godine, a danas je poznat kao referentni je skup podataka za uvježbavanje i procjenu algoritama u zadacima koji se odnose na prepoznavanje i lokalizaciju objekata. Skup podataka sadrži 328 000 slika klasificiranih u osamdeset različitih klasa. Iako se ne koristi izravno u ovom radu, prethodno uvježbani YOLOv5 modeli koji se koriste u ovom radu su uvježbavani na ovom skupu podataka.

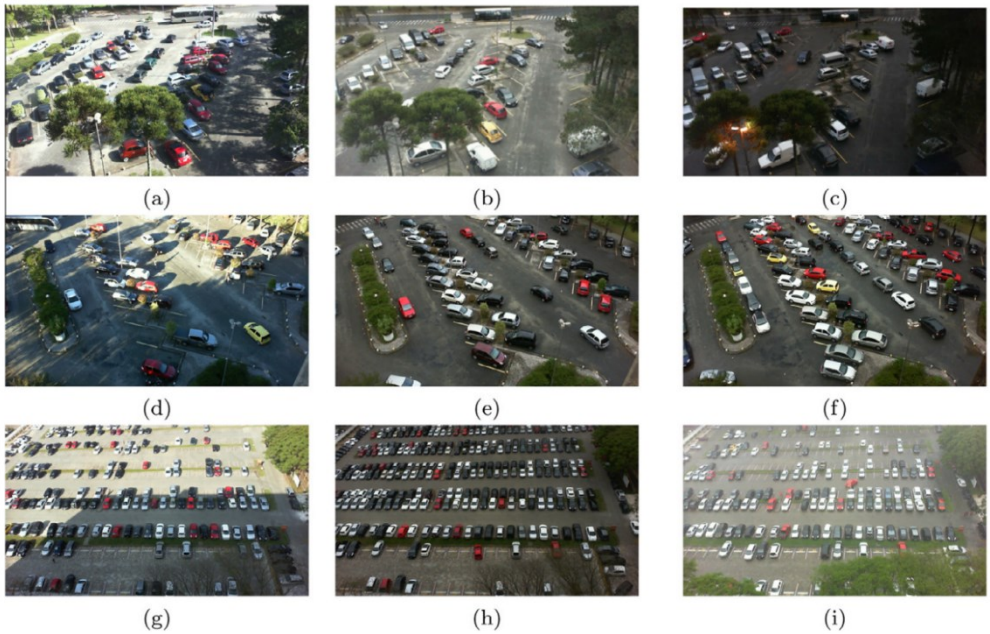
Što se tiče skupova podataka koji su specifični za problem prepoznavanja slobodnih parking mjesta, nakon temeljitog pregleda literature odabrani su slijedeći skupovi podataka kao najprimjereniji za konkretnu primjenu na studiju slučaja:

4.2. PKLot

PKLot [63] je skup podataka koji su 2015 objavili Paulo Almeida, Luiz Oliveira, Alceu S. Britto Jr., Eunelson J. Silva Jr i Alessandro Koerich i nastao je kao proširenje na prethodni PKLot skup podataka objavljen 2013. godine. Sadrži 12 417 slika različitih parkirališta na kojima se nalazi 695 899 parking mjesta koji su manualno provjereni i označeni. Sve slike su prikupljene sa parkirališta na Federal University of Parana (UFPR) i Pontificial Catholic University of Parana (PUCPR), oba locirana u Curitibi u Brazilu. Protokol kojim su se pri izradi služili autori je slijedeći:

- Akvizicija slike: proces se izvršavao u petominutnim intervalima u više od trideset dana snimanjem kamerom Microsoft LifeCam, HD-5000. koja se nalazila na vrhu zgrade kako bi se minimizirala mogućnost mogućih prepreka između kamere i

vozila. Glavni cilj je bio prikupljanje slika u različitim vremenskim uvjetima (oblačno, sunčano i kišovito) što se postiglo ponovnim snimanjem svakih 5 minuta prilikom promjene vremena. Na taj način dobio se dovoljan broj fotografija prikupljenih u različitim uvjetima (npr. lagana kiša, velika kiša, stanje nakon kiše). Autori nisu prikupili noćne snimke kako je tokom noći na parkiralištu jako loša vidljivost. Slike su spremane u JPEG formatu sa kompresijom bez gubitka u rezoluciji 1280x720 piksela. Organizirani su u tri podskupa: UFPR04, UFPR05 I PUCPR. Prva dva podskupa sadržavaju fotografije iz različitih kutova istog parkinga sa četvrtog i petog kata UFPR zgrade. PUCPR skup podataka sadrži fotografije slikane sa desetog kata administracijske zgrade PUCPR. Na slici 4.1. se vide primjeri fotografija iz sva tri podskupa sa različitim vremenskim uvjetima.



Slika 4.1 Primjeri fotografija iz PKLot skupa podataka

- Označavanje: za svaku sliku parkirališta kreirana je „*extensible markup language*“ (XML) datoteka koja sadrži pozicije i stanje svakog parkirnog mjesta. Napravljen je interaktivni alat za označavanje slika. Takav alat dopušta pregled svake slike i definiranje granica parking mjesta i njegovog statusa. Različite podmape su se koristile kako bi se ručno podijelile fotografije ovisno o vremenskim uvjetima.
- Segmentacija: izdvojena su parking mjesta za svaku sliku parkirališta koristeći informaciju iz odgovarajućih XML datoteka. Važno je napomenuti da su označena samo ocrtana parking mjesta, ne i ona koja su nepropisno parkirana.

Na slici 4.2 su prikazane karakteristike svakog od podskupova ovog skupa podataka:

Parking lot	Weather condition	# Of days	# Of images	# Of parking spaces		
				Occupied	Empty	Total
UFPR04	Sunny	20	2098	32,166 (54.98%)	26,334 (45.02%)	58,400
(28 parking spaces)	Overcast	15	1408	11,608 (29.47%)	27,779 (70.53%)	39,387
	Rainy	14	285	2351 (29.54%)	5607 (70.46%)	7958
	Subtotal		3791	46,125 (43.58%)	59,720 (56.42%)	105,845
UFPR05	Sunny	25	2500	57,584 (57.65%)	42,306 (42.35%)	99,890
(45 parking spaces)	Overcast	19	1426	33,764 (59.27%)	23,202 (40.73%)	56,966
	Rainy	8	226	6078 (68.07%)	2851 (31.93%)	8929
	Subtotal		4152	97,426 (58.77%)	68,359 (41.23%)	165,785
PUCPR	Sunny	24	2315	96,762 (46.42%)	111,672 (53.58%)	208,433
(100 parking spaces)	Overcast	11	1328	42,363 (31.90%)	90,417 (68.10%)	132,780
	Rainy	8	831	55,104 (66.35%)	27,951 (33.65%)	83,056
	Subtotal		4474	194,229 (45.78%)	230,040 (51.46%)	424,269
TOTAL			12,417	337,780 (48.54%)	358,119 (51.46%)	695,899

Slika 4.2 Karakteristike skupa podataka PKLot

Ono što ovaj skup podataka čini popularnim je:

1. Sve slike koje pokrivaju različite vremenske uvjete su fotografirane bez kontrole osvjetljenja
2. Fotografije su fotografirane na različitim parkiralištima koja sadržavaju različite značajke
3. Kamere su pozicionirane na različitim visinama.
4. Na slikama su obuhvaćeni mnogi realni problemi kao što su prisutnost sjena, prevelika izloženost suncu, slabo osvjetljenje u kišnim danima, razlika u perspektivi itd.
5. Fotografije vozila su karakteristične onima u realnom svijetu s obzirom na to da se sustavi nadzora nalaze na visini
6. Velik broj potencijalnih načina korištenja

Ovaj skup podataka preuzet je sa stranice Roboflow [2], a došao je raspoređen u tri mape: train, valid i test koje u sebi sadrže 8 691,2483 i 1242 slike parking mjesta.

4.3. CNRPark+EXT

Ovaj skup podataka su objavili autori teksta „*Car Parking Occupancy Detection Using Smart Camera Networks and Deep Learning*“ [38] u kojem su objavili i ovaj skup podataka koji se sastoji od otprilike 150 000 označenih parking mjesta fotografiranih na parkiralištu od 164 mjesta. Nastao je kao proširenje već postojećeg skupa podataka CNRPark, manjeg skupa koji se sastojao od otprilike 12 000 označenih mjesta koje su slikane u srpnju 2015.

sa dvije različite kamere, A i B, koje su postavljene tako da imaju različite uglove snimanja. CNRPark+EXT proširuje taj skup podataka sa s fotografijama prikupljenima od studenog 2015. do veljače 2016 u različitim vremenskim uvjetima sa 9 kamera koje su imale različite kutove snimanja. Kao i u prethodnom, i u ovom skupu podataka se na fotografijama mogu pronaći različiti vremenski uvjeti, kao i određene zapreke(stabla, rasvjeta itd.) koje mogu uzrokovati smetnje pri prepoznavanju slobodnih mjesta. Skup podataka se sastoji od 4287 slika veličine 1000x750 piksela koje su slikane na 23 različita dana te imaju 144 965 označenih parking mjesta. Različite raspodjele podskupova skupa podataka se mogu vidjeti na slici 4.3. Također, skup sadrži CSV datoteke koje daju informacije o pozicijama parking mjesta i vozila na slikama. Postoji i verzija koja sadrži fotografije individualnih mjesta veličine 150x150 piksela. Primjeri slika iz ovog skupa se mogu vidjeti na slici 4.4.

SUBSETS	FREE SPACES	BUSY SPACES	TOTAL
CNRParkOdd	2201	3970	6171
CNRParkEven	1980	4433	6413
CNRPark-EXT TRAIN	46877	47616	94493
CNRPark-EXT VAL	5232	13415	18647
CNRPark-EXT TEST	13549	18276	31825
CNRPark-EXT SUNNY	25665	37513	63178
CNRPark-EXT OVCST	21067	23176	44243
CNRPark-EXT RAINY	18926	18618	37544
CNRPark-EXT *C1*	6407	9308	15715
CNRPark-EXT *C2*	1454	2641	4095
CNRPark-EXT *C3*	4101	5370	9471
CNRPark-EXT *C4*	7219	9357	16576
CNRPark-EXT *C5*	9582	11256	20838
CNRPark-EXT *C6*	9462	10646	20108
CNRPark-EXT *C7*	10595	10519	21114
CNRPark-EXT *C8*	11237	12847	24084
CNRPark-EXT *C9*	5601	7363	12964

Slika 4.3 Raspodjela skupa CNRPark-EXT

Slika 4.4 Primjer slika iz skupa CNRPark-EXT

5. YOLO

Kada je riječ o detekciji objekata, postoji mnogo modela dostupnih za korištenje, a kao najpopularniji se ističu YOLOv5 i YOLOv8 [26]. YOLOv8 se istakao kao algoritam sa najboljim performansama što se tiče brzine i preciznosti među YOLO algoritmima. No, iako je YOLOv5 izgubio titulu najbržeg detektora koju je imao kada se pojavio i dalje se smatra jednim od bržih dostupnih algoritama, a uz to je i danas model koji je među navedenim najviše prilagođen korisnicima sa svojom implementacijom na PyTorch [64] okviru što ga dan danas čini jednim od najpopularnijih izbora u rješavanju zadataka detekcije i klasifikacije objekata. Također, neke od glavnih odlika koje se navode kod YOLOv8 su procjena položaja, segmentacije objekata i veliko poboljšanje detekcije sitnih objekata [65], ali takvi alati ne koriste ili nemaju smisla u primjeni koja je odabrana u ovom radu.

YOLOv5

U lipnju 2020. godine, mjesec dana nakon objave YOLOv4 [23], Glenn Joacher je objavio YOLOv5 model. Samo objavljivanje je izazvalo kontroverze zbog korištenja imena YOLO iz nekoliko razloga:

1. Joacher nije originalni YOLO autor (iako je pomogao u razvijanju YOLOv3)
2. Nije objavljen nikakav znanstveni članak, nego samo GitHub repozitorij
3. Smatrano je da sama implementacija nije uvela dovoljno noviteta

Usprkos tome, YOLOv5 je zbog svoje iznimne brzine i jednostavnosti veoma brzo postao veoma popularan te i danas ima aktivnu zajednicu.

Ovisno o potrebama YOLOv5 dostupan je za korištenje u pet glavnih verzija: *nano* (n), *small* (s), *medium* (m), *large* (l), *extra large* (x). Verzije se uglavnom razlikuju u veličini modela i kompleksnosti, tako da svaka nudi određeni omjer brzine i preciznosti, ovisno o potrebama korisnika. Razlike između uvježbavanja na svakoj od verzija prikazane su u tablici 5.1. Rezultati su preuzeti iz tablice koja je preuzeta sa službenog GitHub repozitorija [1]:

Tablica 5.1 Rezultat različitih verzija YOLOv5 algoritma

Model	mAP_{50-95}	mAP_{50}	Speed CPU	Speed V100 b1	Speed V100 b32	params	FLOPs @640
YOLOv5n	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	50.7	68.9	766	12.1	4.8	86.7	205.7

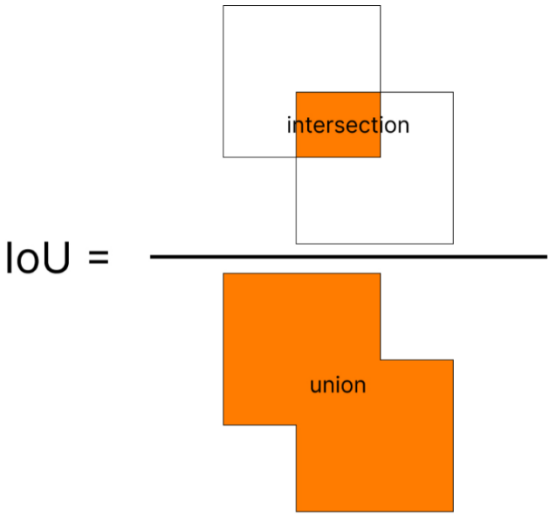
Svaka verzija je uvježbana na COCO skupu podataka čije slike imaju dimenzije 640x640 na tristo epoha na zadanim postavkama. Nano i small modeli koriste hiperparametre sa niskom razinom augmentacije, a ostali koriste hiperparametre sa visokim stupnjem augmentacije. Prva dva stupca odgovaraju mAP metrici sa 50 i 50-95 IoU pragom. Treći stupac daje vrijeme zaključivanja na CPU sa pojedinačnom serijom s iskazanim vremenom u milisekundama, četvrti vrijeme zaključivanja na NVIDIA V100GPU sa pojedinačnom serijom, peti na istoj postavci, ali sa veličinom serije 32. Stupac params daje broj parametara u svakom modelu izražen u milionima te broj aritmetičkih operacija s pomičnim zarezom (FLOPs) u milijardama za ulaz od 640x640 piksela.

Rad YOLO algoritma zasniva se na tri tehnike [22]:

- Slika na ulazu se podijeli na SxS rešetku i svaka ćelija rešetke ima istu dimenziju. Ako se centar objekta nalazi u određenoj ćeliji, ta ćelija je odgovorna za detekciju tog objekta
- Regresija granica pravokutnika – određivanje granica pravokutnika koji odgovaraju kvadratima koji označavaju objekte na slici. YOLO određuje atribute granica pravokutnika sa regresijskim modulom kao vektor u slijedećem formatu: $Y=[p_c,b_x,b_y,b_h,b_w,c]$ gdje je p_c ocjena vjerojatnosti da ćelija sadržava objekt, b_x i b_y su x i y koordinate centra graničnih pravokutnika s obzirom na ćeliju koja je omotava. B_h i b_w su odgovarajuća širina i dužina s obzirom na ćeliju koja je omotava, a c je

vektor koji svakoj mogućoj klasi dodjeljuje vjerojatnost mogućih klasa i dužina ovisi o broju klasa koje se mogu pojaviti na slici.

- *Intersection Over Union* (IoU) [66] – IoU popularna je metrika za mjerenje preciznosti lokalizacije i računanja lokalizacijskih greški u modelima detekcije objekata. Kako bi se izračunao IoU između previđenih i stvarnih granica pravokutnika uzima se površina njihovog presjeka (0-1), što je preklapanje površina veće to je IoU bliži 1. Vizualno je IoU prikazan na slici 5.1.



Slika 5.1 Intersection over Union

Kada se navedene tehnike ukomponiraju, proces detekcije se vrši na slijedeći način: fotografija se proslijedi u konvolucijsku mrežu kako bi izvadila značajke iz dane slike, nakon čega se značajke prosljeđuju kroz potpuno povezane slojeve koje predviđaju vjerojatnosti pripadnosti klasi i koordinate granica pravokutnika. Nakon toga se slika dijeli na ćelije i svaka je zadužena za previđanje skupa granica pravokutnika i vjerojatnosti klase. Izlaz iz mreže je skup granica pravokutnika i vjerojatnosti klasa za svaku ćeliju. Granice pravokutnika se zatim filtriraju i koristi se algoritam *non-max suppression* (NMS) [67] kako bi se uklonili preklapajući pravokutnici i odabrali oni koji najbolje označavaju objekt. Konačni izlaz je skup predviđenih granica pravokutnika i oznaka klasa za svaki objekt na slici.

YOLO model se sastoji od tri dijela:

- Temeljni dio (eng. *Backbone*) – odgovoran je za ekstrakciju korisnih značajki iz ulazne slike. Tipično je konvolucijska mreža koja se trenira na klasifikacijskim zadacima velikih razmjera kao što je ImageNet [39]. Temeljni dio hijerarhijski hvata

značajke u različitim razmjerima – značajke niže razine se vade u ranijim slojevima, a one više razine se vade u dubljim slojevima.

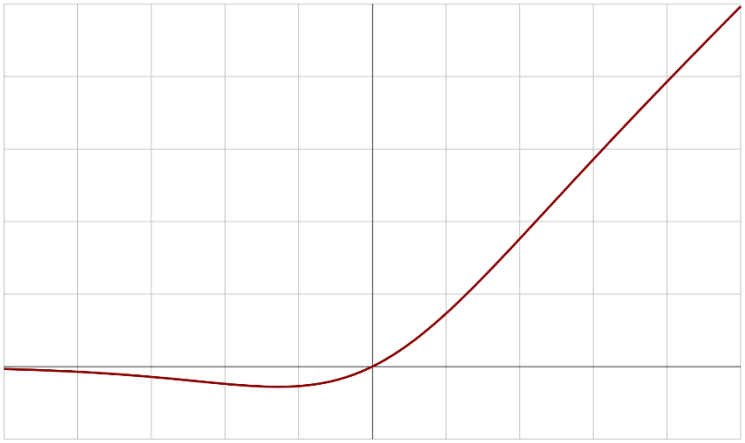
- Vrat (eng. *Neck*) - napredna komponenta koja spaja temeljni dio i glavu. Prikuplja i poboljšava značajke koje je izvodio temeljni dio, često se fokusirajući na poboljšanje prostornih i semantičkih informacija na različitim razinama. Može sadržavati dodatne konvolucijske mreže, *feature pyramid networks* (FPN) [21] ili druge mehanizme za poboljšanje reprezentacije značajki.
- Glava (eng. *Head*) - posljednja komponenta, zadužena za predviđanja koja se temelje na značajkama koje su omogućili temeljni dio i vrat. Obično se sastoji od jedne ili više pod mreža specifičnih za zadatak koje su zadužene za klasifikaciju, lokalizaciju i, u novije vrijeme, segmentaciju instanci i procjenu položaja glava procesuira značajke koje omogućava vrat, i radi procjene za svakog kandidata objekta. Na kraju ima post-processing korak kao što je NMS [67] koji filtrira preklapajuća predviđanja i zadržava najsigurnije detekcije.

5.1. Arhitektura

YOLOv5 koristi CSP-Darknet53 [23] arhitekturu za ekstrakciju značajki iz slike. Darknet53 [54] korišten je u YOLOv3 [54], a dobio je *Cross Stage Parital* (CSP) [68] dizajn. CSP je uveden iz nekoliko razloga: poboljšava tok gradijenta kako bi se izbjegao problem redundantnih gradijenata, omogućava kombiniranje značajki iz ranijih i kasnijih razina mreže i doprinosi izradi efektivnijih i preciznijih modela za detekciju.

Također, YOLO koristi *Path Aggregation Network* (PANet) [69] kako bi poboljšao tok informacija i lokalizaciju piksela u zadacima predviđanja maske.

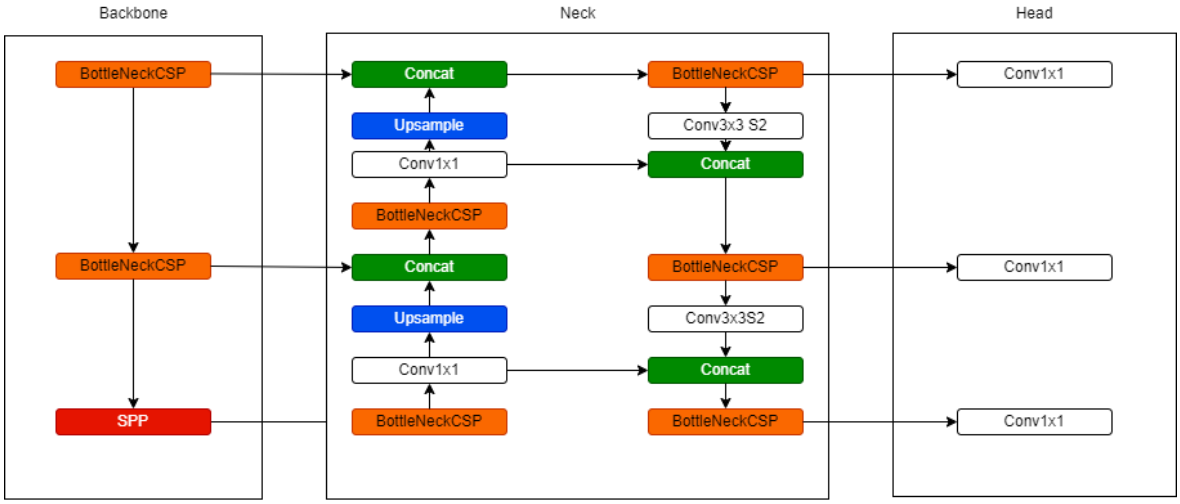
Temeljni dio se sastoji od pet konvolucijskih slojeva, četiri C3 *bottleneck*-a i *Spatial Pyramid Pooling Fast* sloja, adaptirane verzije Spatial Pyramid Pooling-a [18]. Konvolucijski slojevi koriste *Sigmoid Linear Unit* (SiLU) aktivacijsku funkciju (koristi se i naziv *Swish*) čiji se graf može vidjeti na slici 5.2.



Slika 5.2 Sigmoid Linear Unit aktivacijska funkcija

Spatial Pyramid Pooling Fast (SPPF) je verzija *Spatial Pyramid Pooling*, modificirana verzija SPP-a[18]. SPP računa mape značajki iz cijele slike u jednom koraku i zatim radi pooling. SPPF je brži zbog manjih veličina jezgre i manje skrivenih kanala. Koristi se kako bi poboljšao performanse detekcije na manjim objektima.

Glava u YOLOv5 sadrži četiri serije konvolucijskih slojeva koji se koriste za transformiranje mapa značajki iz Temeljni dio u odgovarajući format. Nakon prva dva konvolucijska sloja slijedi *Upsampling* [70] sloj koji se koristi kako bi se povećala prostorna razlučivost mapa značajki. Zatim iza svakog Upsampling sloja, kao i konvolucijskog gdje nema *Upsampling*-a slijedi konkatenacijski sloj koji spaja mape značajki (P3, P4, P5) iz temeljnog dijela sa mapama značajki iz prethodnih slojeva, što u konačnici spaja informacije iz nekoliko razina što je korisno pri detekciji objekata različitih veličina. Nakon konkatenacije, mape se dodatno procesiraju pomoću C3 bloka koji se sastoji od konvolucijskih slojeva koji nadalje čiste i vade relevantne značajke iz kombiniranih mapa značajki. Arhitektura je vizualno prikazana na slici 5.3, a napravljena je po uzorku na sliku koja se nalazi među diskusijama na službenim GitHub stranicama [71], a za koju je autor potvrdio da je točan prikaz.



Slika 5.3 YOLOv5 arhitektura

YOLO od svoje druge verzije [72] koristi sustav prethodno definiranih pravokutnika (eng. *anchor boxes*) To su predefinirani granični pravokutnici određenih širina i dužina. Definiraju se kako bi mogli prepoznati objekte različitih veličina i pomažu kod objekata čije se sredine nalaze u istoj ćeliji. Za razliku od starijih verzija, YOLOv5 koristi metodu dinamičkih prethodno definiranih pravokutnika. Uključuje korištenje algoritma za grupiranje kako bi grupirao originalne granične pravokutnike i onda korištenjem centara grupa kao prethodno definiranih pravokutnika. To omogućava prethodno definiranim pravokutnicima kako bi bili usklađeniji sa veličinama i oblikom detektiranih objekata.

Funkcija gubitka se u YOLOv5 sastoji od 3 vrijednosti, a izražena je u jednadžbi 1:

$$\boldsymbol{gubitak} = \boldsymbol{l_{box}} + \boldsymbol{l_{cls}} + \boldsymbol{l_{obj}} \quad \text{Jednadžba 1}$$

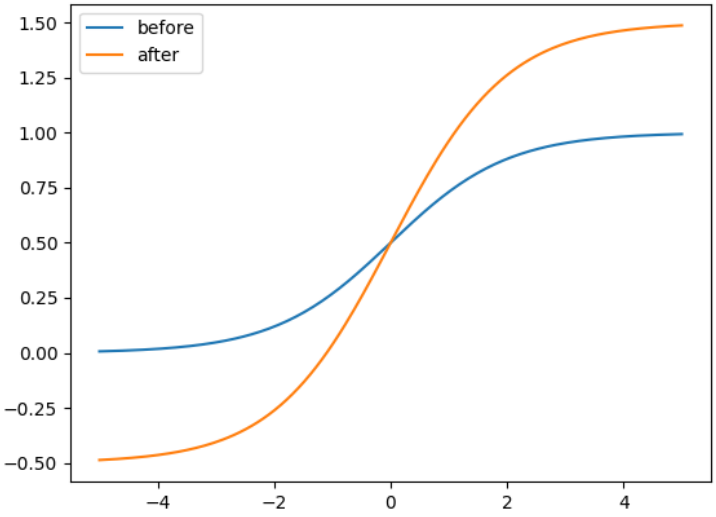
gdje je l_{box} funkcija gubitka regresije graničnih pravokutnika i za njeno računanje se koristi *Complete Intersection over Union* [73] funkcija gubitka i l_{cls} klasifikacijska funkcija gubitka, a l_{obj} funkcija gubitka preciznosti za koje se koristi *Binary Cross Entropy* [29] funkcija gubitka.

YOLOv5 je popravio određene manjkavosti prethodnih verzija algoritma: prethodni bi algoritmi imali probleme u detekciji objekata koji se nalaze na rubovima slike zbog jednadžbi koje su koristile za previđanje graničnih pravokutnika. Nove su jednadžbe pomogle pri rješavanju tog problema povećavanjem raspona pomaka središnje točke iz 0-1 na -0.5-1.5 kako bi pomak mogao biti 1 ili 0. Uz to, omjeri skaliranja nisu bili ograničeni što je znalo dovoditi do nestabilnosti u uvježbavanju.

Korištene formule su navedene u jednadžbi 2:

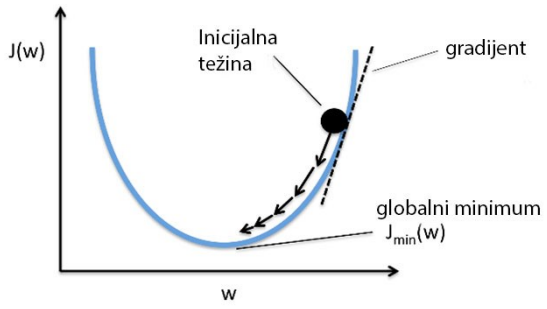
$$b_x = (2 \times \sigma(t_x - 0.5) + c_x \times b_y = \left(2 \times \sigma\left(t_y\right) - 0.5\right) + c_y \times b_{w=} = p_w \times \left(2 \times \sigma(t_w)\right)^2 \times b_h = p_h \times \left(2 \times \sigma(t_h)\right)^2 \text{ Jednadžba 2}$$

Usporedbom pomaka centralne točke prije i nakon skaliranja na slici 5.4 se može vidjeti da je raspon pomaka centralne točke prilagođen iz (0,1) u (0.5, 1.5). Stoga pomak lako može biti nula ili jedan.



Slika 5.4 Graf usporedbe pomaka centralne točke prije i nakon skaliranja

Bitna stavka kod uvježbavanja je i optimizacija: ona služi kako bi pokušali minimizirati greške koje algoritam radi u detekciji za vrijeme uvježbavanja. YOLOv5 ima implementirana dva optimizacijska algoritma: Stohastički gradijentni spust [74] i Adam [75]. Stohastički gradijentni spust je metoda koja koristi manju skupinu nasumično odabranih podataka te uz fiksnu stopu učenja kroz nekoliko koraka traži točku u kojoj će postići najmanju vrijednost gubitka. Postoje određeni problemi koji se mogu pojaviti korištenjem stohastičkog gradijentnog spusta: uzimanjem premale stope učenja može se postići prespora konvergencija te sam pronalazak minimuma traje presporo, a može i zapeti u lokalnom minimumu te ga označiti kao globalni. Također , uzimanjem prevelike stope učenja, može doći do divergencije što može rezultirati neuspješnim učenjem i nestabilnosti modela. Na slici 5.5 je ilustriran rad stohastičkog gradijentnog spusta.



Slika 5.5 Stohastički gradijentni spust

Adam s druge strane ima prilagodljivu dužinu koraka. On pamti svoje optimizacije te na temelju njih ažurira vrijednosti što mu olakšava pronalazak globalnog minimuma. No, s obzirom da je riječ o složenijem algoritmu, rad mu je sporiji. Eksperimentiranjem se pokazalo kako Adam za zadatak ovog rada ne radi razliku u odnosu na zadani stohastički gradijentni spust, pa je zbog brzine rada on izabran za ovaj rad.

5.2. Metrike

Postoji niz metrika koje YOLOv5 koristi kako bi se mogla evaluirati uspješnost prepoznavanja objekata. Uz njih, postoji velik broj drugih metrika koja se koriste za evaluaciju, a metrike koje koristi YOLOv5 uspoređene su sa ostalim metrikama u radu „*A systematic analysis of performance measures for classification tasks*“ [76], a metrike koje se koriste su detaljnije opisane u nastavku:

5.2.1. Preciznost

Preciznost je metrika koja mjeri točnost pozitivnih predviđanja modela. Laički rečeno, odgovara na pitanje „Od svih objekata koji su detektirani kao pozitivni, koliko ih je stvarno pozitivno?“, odnosno cilj ove metrike je smanjiti pogreške u predviđanju pozitivnih primjera. Matematički, preciznost je izražena u jednadžbi 3 :

$$Preciznost = \frac{Stvarno\ pozitivni}{Stvarno\ pozitivni + Lažno\ pozitivni}$$ Jednadžba 3

Stvarno pozitivni je broj točno predviđenih pozitivnih primjera, dok je lažno pozitivni broj primjera koji su predviđeni kao pozitivni ali su ustvari negativni. No, ova metrika nije dostatna za samostalno funkcioniranje jer ne uzima u obzir lažno negativne primjere pa se tako uz samo jedan stvarno pozitivni procijenjeni primjer bez predviđanja ostalih primjera može postići savršena ocjena.

5.3. Odziv

Odziv (eng. *Recall*) je metrika koja mjeri koliko često model točno previđa pozitivne primjere iz svih stvarno pozitivnih primjera, odnosno za cilj ima maksimizirati broj pozi. Matematički, odziv se iskazuje kako je navedeno u jednadžbi 4:

$$Odziv = \frac{Stvarno\ pozitivni}{Stvarno\ pozitivni + Lažno\ negativni}$$ Jednadžba 4

Lažno negativni su svi primjeri koji su predviđeni kao negativni, ali su ustvari pozitivni. Također, i ova metrika ima problem: ukoliko se svi primjeri procjene kao pozitivni, neće biti negativnih, a rezultat će biti savršen – jedan.

5.4. F1

Kako prethodne dvije metrike, iako korisne, imaju očite manjkavosti po pitanju njihovog iskorištavanja, a u isto vrijeme jedna metrika onemogućava iskorištavanje druge, određena je metrika F1 koja je harmonijska sredina preciznosti i odziva dana u jednu vrijednost. Cilj F1 je maksimizirati prethodne dvije metrike tako da se dobije precizan i kompletan model. Harmonijska sredina se koristi kako bi se više „kaznila“ manja vrijednost ukoliko bi se rezultati značajno razlikovali. Matematički je prikazana u jednadžbi 5:

$$F1 = 2 \times \frac{Preciznost \times Odziv}{Preciznost + Odziv}$$ Jednadžba 5

5.5. Srednja prosječna preciznost

Srednja prosječna preciznost [77] se koristi na zadacima detekcije objekata, a za to se služi matricom konfuzije i već spomenutim metrikama: IoU, preciznost i odziv. Matrica konfuzije [78] sastoji se od četiri već spomenuta atributa: stvarno pozitivno, stvarno negativno, lažno pozitivno i lažno negativno, a prikazana je u tablici 5.2.

Tablica 5.2 Matrica konfuzije

	Stvarno	
Predviđeno	Stvarno pozitivni	Lažno pozitivni
	Lažno negativni	Stvarno negativni

.

U matrici se vrijednosti mogu prikazati brojem pronađenih primjera za svaku vrijednost ili postotkom u odnosu na sve promatrane primjere.

Srednja prosječna vrijednost se računa traženjem prosječne preciznosti za svaku klasu, a zatim se računa prosjek prosječnih preciznosti kroz sve ostale klase. Taj se postupak događa u nekoliko koraka:

1. Generiraju se rezultati predviđanja
2. Rezultati previđanja se pretvaraju u oznake klasa
3. Računaju se atributi za matricu predviđanja
4. Pomoću matrice predviđanja računaju se preciznost i odziv te se zatim računa površina ispod grafa preciznost – odziv krivulje
5. Računa se prosječna preciznost

Matematički, formula je prikazana u jednadžbi 6:

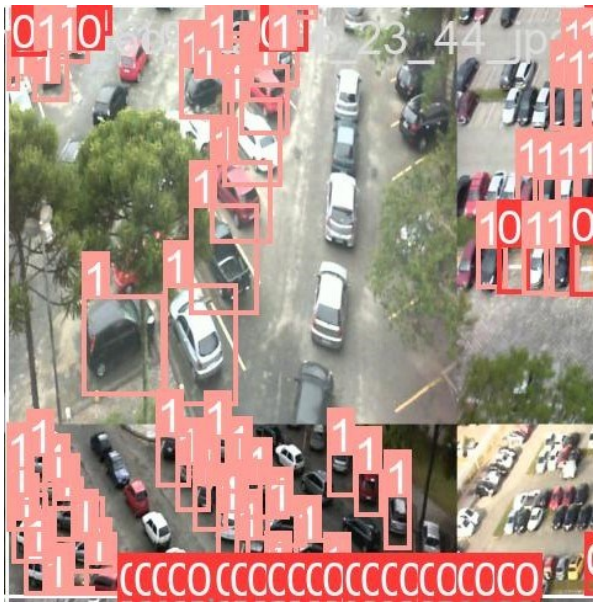
$$mAP = \frac{1}{n}\sum_{k=n}^{k=n} AP_k \quad \text{Jednadžba 6}$$

gdje je n broj klasa, a AP_k prosječna preciznost klase k.

5.6. Augmentacije

Augmentacije su bitan faktor YOLOv5 algoritma. Augmentacije su postupak u kojem se originalna slika na određeni način modificira kako bi model mogao bolje identificirati objekte i generalizirati dane zadatke. Augmentacije mogu izmijeniti osvjetljenje i boje na slikama promjenom tona, zasićenosti i svjetline boje. Također, mogu promijeniti izgled i orijentaciju slike korištenjem tehnika kao što su rotacije, zrcaljenja, smicanja, rezanja, mijenjanja perspektive u slici i slično. Uz njih, YOLOv5 koristi i posebne tehnike kao što su mozaik [23], MixUp [79] i copy-paste [80]. Bitno je naglasiti da u uvježbavanju nijedna

slika ne pojavljuje u istom obliku dvaputa (odatle i ime *You only look once*), nego samo može doći nekoliko modificiranih verzija iste slike. Autori YOLOv5 stavljaju velik naglasak na mozaik augmentaciju koju je autor YOLOv5 uveo u YOLOv3. Mozaik u jednu sliku kombinira nekoliko različitih slika koje nasumično postavlja unutar slike kako bi se napravio uzorak koji kombinira različite pozadine, objekte i veličine u jednoj slici kako bi se model bolje funkcionirao u različitim kontekstima što posljedično smanjuje preuvježbanost. Slike koje se pojave mogu se pojaviti u različitim oblicima, bilo da su zakrenute ili da im je promijenjena boja ili perspektiva. Na slici 5.6 je prikazan primjer mozaičke augmentacije koji se pojavio u uvježbavanju u ovom radu. Može se primijetiti kako se slika sastoji od 5 različitih slika koje su neravnomjerno poredane i kojih sve imaju različite pozadine.



Slika 5.6 Mozaik augmentacija

Novitet kod YOLOv5 u odnosu na starije verzije je što bi se prije kao augmentacija povećavala veličina cijele slike i zatim bi se proslijedila na uvježbavanje što bi koristilo mnogo memorije. No, u slučaju YOLOv5 se zadržava veličina slike dok se kod povećanja slika izreže tako da bude u originalnoj veličini, a kod smanjenja se stvara crna pozadina u dijelovima gdje je slika prije postojala. S tim se uspijeva zadržati varijabilnost, dok se u isto vrijeme smanjuje korištenje memorije za uvježbavanje.

Nakon pregleda literature i predstavljenih skupova podataka te uzimajući u obzir složenost primjene razmatranih algoritama za realizaciju modela koji za cilj ima pokazati mogućnost primjene odredio se plan izrade okruženja za rad nakon kojeg je napravljena evaluacija

performansi dostupnih YOLOv5 modela na problemu određivanja slobodnih parking mjesta. U sklopu evaluacije, cilj je bio optimizirati parametre za ciljane rezultate.

Kako bi YOLOv5 mogao funkcionirati kako je zamišljeno, potrebno je instalirati

Okruženje je postavljeno kako slijedi:

- Razvojno okruženje: Google Colaboratory[3]
- Skupovi podataka: PKLot i CNRPark-EXT
- Algoritam:YOLOv5
- Metrika za ocjenu postignutog rezultata prepoznavanja: Srednja prosječna preciznost

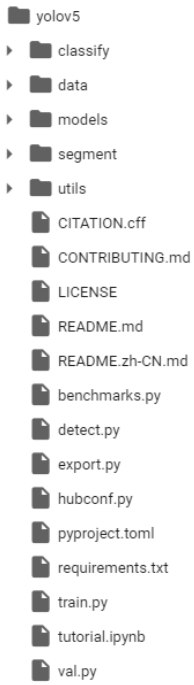
6. Postupak uvježbavanja i evaluacije

Kao što je već spomenuto, za izradu modela koristio se Google Colaboratory, za koji YOLOv5 nudi već gotovu bilježnicu kako bi ga novi korisnici brzo mogli testirati na svojim podacima. Svoju su bilježnicu podijelili na nekoliko koraka:

- Instalacija YOLOv5 i alata potrebnih za njihovo korištenje
- Prikupljanje skupa podataka
- Uvježbavanje YOLOv5 kako bi prepoznavao objekte na prikupljenom skupu podataka
- Evaluacija performansi YOLOv5 modela
- Izvođenje testnog zaključivanja kako bi se model prikazao na djelu

Instalacija YOLOv5 se sastoji od kloniranja YOLOv5 repozitorija sa GitHub-a u Colaboratory okruženje, instalacije biblioteka koje YOLOv5 koristi pri radu i Roboflow-a kako bi se s moglo preuzeti skupove podataka korištene u radu. Također, kako bi YOLOv5 mogao raditi na način na koji je to zamišljeno, moraju se instalirati biblioteke koje su navedene u datoteci requirements.txt dostupnoj u repozitoriju[81], a podijeljene su na nekoliko kategorija kao što su bazne biblioteke ili one za vizualizaciju, a po izboru postoje i druge grupe koje se mogu instalirati.

Kloniranjem repozitorija dobije se struktura datoteka na slici 6.1:



Slika 6.1 Struktura datoteka u YOLOv5

U radu s YOLOv5 koristi se prethodno određeni datotečni sustav: datoteke za treniranje, validaciju i testiranje moraju sadržavati dvije pod datoteke – *images* i *labels*. U mapi *images* se nalaze slike dok se u *labels* nalaze tekstualne datoteke od kojih svaka odgovara slici u mapi *images*. U datoteci *labels* nalaze se oznake graničnih pravokutnika za svaki objekt na slici i oznake su normalizirane na veličinu slike i vrijednosti koje mogu poprimiti se nalaze u rasponu od nula do jedan. Format u kojem su napisani je: identifikacijski broj klase – koordinata X centra objekta – koordinata Y centra objekta – dužina box-a – visina box-a. Na slici 6.2 se može vidjeti izgled takvog formata.

```
1 0.23515625 0.28828125 0.0359375 0.0625
0 0.259375 0.290625 0.03515625 0.06640625
```

Slika 6.2 Format definiranih graničnih pravokutnika

Konfiguracija podataka navedena je u YAML datoteci u kojoj se nalaze informacije o mjestima gdje se nalaze mape za uvježbavanje, validaciju i testiranje, broj klasa i njihova imena.

Datoteke konfiguracije modela određuju njegovu arhitekturu. Postoji nekoliko verzija YOLOv5 arhitektura koje su prethodno navedene, a razlikuju se po broju parametara koje primaju: nano verzija koristi najmanji broj parametara što je čini najbržom, ali najmanje preciznom, dok extra large verzija koristi najveći broj parametara te je najpreciznija, ali i

najsporija od navedenih verzija (iako je i dalje brža od većine drugih algoritama). Arhitekture su prilagođene za uvježbavanje na veličinama slika 640*640 piksela koje se i koriste u ovom radu.

Uvježbavanje se može vršiti od nule ili se mogu koristiti modeli koji su prethodno uvježbani na COCO skupu podataka.

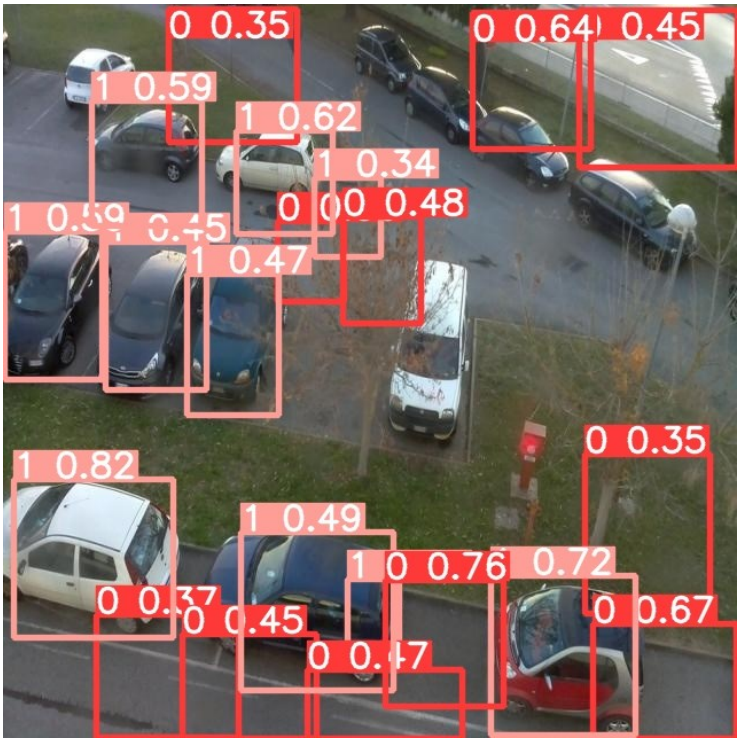
Skupovi podataka su preuzeti sa Roboflow stranice koja nudi opciju pripreme slika i oznaka kako bi odgovarali YOLOv5 formatu što je skratilo vrijeme potrebno za promjenu veličina, označavanje i prilagodbu originalnih fotografija. Kada se pronađe skup podataka koji se želi koristiti Roboflow omogućava preuzimanje originalnih veličina slika ili njihovo preuzimanje u rezoluciji 640x640. Kako ova dva skupa podataka imaju različite originalne rezolucije, zbog jednostavnosti, a i zbog toga što su u pravilu dimenzije promatranih objekata velike u odnosu na veličinu slike, preuzete su slike dimenzija 640x640.

Nakon odabira veličina slike, Roboflow nudi različite formate preuzimanja, od formata koji su prilagođeni verzijama YOLO-a od YOLOv3 do YOLOv8 do formata za druge algoritme i biblioteke. Također, nudi i razne opcije procesiranja slike i augmentacija, no te opcije nisu korištene u ovom radu kako sami skupovi podataka imaju dovoljan broj slika snimanih iz različitih kutova pa nema potrebe za širenjem skupa podataka.

S obzirom na velik broj fotografija koju skupovi podataka sadrže – lako je postići preuvježbanost modela, a uz to, uvježbavanje traje predugo te otežava korištenje Colaboratory-a s obzirom na ograničenja pri njegovom korištenju. Iz navedena dva razloga napravljeno je uzorkovanje. U radu je napravljeno nasumično uzorkovanje korištenjem takozvanog *seed*-om koji se koristi kako bi se u slučaju ponovnog pokušavanja moglo pristupiti istim slikama koje su se koristile u radu.

Početna ideja je bila da se uvježbavanje vrši na skupu podataka PKLot, i da se zatim model testira na CNRPark+EXT, no iako je u tom slučaju model donekle uspijevao prepoznavati automobile, odnosno zauzeta parking mjesta (iako je i ta detekcija imala problema koji bi zahtijevali određena podešavanja), nepremostiv problem je bio prepoznavanje slobodnih parking mjesta s obzirom na sličnost izgleda koje ima slobodno parking mjesto sa dijelom parkinga koji se koristi za prolaz vozila. Uz to, loša vidljivost linija koje označavaju parking na skupu za testiranje, pogotovo u određenim uvjetima osvjetljenja kao što su kiša ili sumrak u kojima se ne može odrediti razlika između parking mjesta i prolaza, utječu na razlučivost slobodnog parking mjesta. Također, primijećen je i problem skaliranja uzrokovan skoro pa

identičnim izgledom pozadine i traženog objekta. Na slici 6.3 se može primjer detekcije kada se za uvježbavanje koristio samo skup podataka PKLot. Iz svih navedenih razloga je u skup za uvježbavanje podijeljen na 90% slika iz skupa podataka PKLot, a 10% na slike iz skupa CNRPark+EXT kako bi model mogao prepoznati stvarna parking mjesta. Ukupan broj slika za uvježbavanje je 849, a na njima se nalazi 24991 objekata za prepoznavanje od kojih je 12801 primjera slobodnih parking mjesta, a 12190 primjera zauzetih parking mjesta. Drugi pristup rješavanju ovog problema je mogao biti označavanje parking mjesta, no to bi rad svelo na klasifikaciju što nije bio cilj rada.



Slika 6.3 Primjer grešaka kada se uvježbavanje radilo samo na jednom skupu

YOLOv5 koristi skup hiperparametara koji se mogu mijenjati kako bi se model bolje prilagodio danom zadatku:

1. Lr0,lrf – inicijalna i konačna stopa učenja za proces optimizacije.
2. Momentum – određuje ažuriranje optimizacijskog algoritma, veće vrijednosti dovode do brže konvergencije, ali mogu premašiti optimalne vrijednosti.
3. Weight Decay – regularizacijska tehnika koja sprječava preučenost.
4. Postavke za zagrijavanje – parametri koji se odnose na fazu zagrijavanja u početnim fazama uvježbavanja. Postupno povećavaju stope učenja, moment i pristranost.

5. Težine funkcija gubitka – parametri koji kontroliraju doprinos različitih komponenti za sveukupnu funkciju gubitka.
6. IoU prag, Anchor prag – pragovi koji se koriste za uvježbavanje, utječu kako se granični pravokutnici generiraju i evaluiraju.
7. Postavke augmentacije slika – parametri koji kontroliraju razne augmentacijske tehnike kao što su HSV augmentacije, rotacije, translacije, skaliranja, smicanje, zrcaljenja, mozaik, MixUp i copy-paste.

Kod preuzimanja YOLOv5 repozitorija omogućeno je korištenje šest različitih vrsta hiperparametara od kojih su dva podešena za treniranje na Object365 odnosno VOC skupovima podataka dok su preostala podešena za različite vrste augmentacija – bez augmentacija, sa niskom, srednjom i viskom razinom augmentacije. Također, pri uvježbavanju je moguće pokrenuti evoluciju hiperparametara kojom se pomoću genetičkog algoritma hiperparametri optimiziraju za dani zadatak. Evoluiranje se provodilo nad skupom za treniranje koji se koristi u ovom radu, na YOLOv5s prethodno utreniranom modelu. Iako je preporuka izvođenje deset epoha na tristo generacija, u ovom radu je, zbog vremenskog ograničenja Google Colaboratory-a napravljeno 250 generacija.

Eksperimenti sa svim navedenim postavkama parametara su pokazali najbolje rezultate korištenjem hiperparametara sa visokom razinom augmentacija koje YOLOv5 sadrži u svom repozitoriju, a njihove vrijednosti se mogu naći na slici 6.4.

```yaml
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.1  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.3  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 0.7  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.9  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.1  # image mixup (probability)
copy_paste: 0.1  # segment copy-paste (probability)
```

Slika 6.4 Postavke hiperparametara korištenih u radu

Uvježbavanje u YOLOv5 se pokreće pomoću skripte train.py kojoj se prosljeđuju argumenti koji daju informacije o uvjetima u kojima se uvježbavanje radi kao što su: veličini slika u skupu podataka, broju epoha koje će sadržavati treniranje, veličini serije, odakle podaci dolaze, koje parametre koristiti i slično. Primjer naredbe koja se pokreće se može vidjeti na slici 6.5.

```
!python train.py --img 640 --batch 16 --epochs 150 --data /content/yolov5/data.yaml --weights yolov5m.pt --cache --hyp '/content/yolov5/data/hyps/hyp.scratch-high.yaml'
```

Slika 6.5 Naredba za pokretanje uvježbavanja

Tokom uvježbavanja YOLO u realnom vremenu ažurira podatke o trenutnoj epohi, memoriji koju koristi GPU, gubitcima graničnih pravokutnika, klasifikacije i preciznosti, broj detektiranih objekata u epohi, veličinu slike, metrike prikupljene za sve klase, broj slika koje se koriste za evaluaciju, preciznost, Recall, srednju prosječnu preciznost sa pragom 50 i sa pragom od 50 do 95(mAP50 i mAP50-95 na slici 6.6).

```
AutoAnchor: 6.12 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Plotting labels to runs/train/exp2/labels.jpg...
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/train/exp2
Starting training for 150 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      0/149      6.26G     0.1108     0.2028    0.01614        196        640: 100% 54/54 [00:37<00:00,  1.42it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:10<00:00,  2.64s/it]
                   all        120       6604       0.16      0.401      0.123     0.0297

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      1/149      7.49G    0.08625     0.2098    0.01162        102        640: 100% 54/54 [00:30<00:00,  1.79it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.37it/s]
                   all        120       6604      0.416      0.654      0.482       0.15

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      2/149      7.49G    0.08223     0.1981   0.008329        170        640: 100% 54/54 [00:28<00:00,  1.90it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:06<00:00,  1.55s/it]
                   all        120       6604      0.316      0.776      0.385      0.159

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      3/149      7.49G    0.07631     0.1989   0.006481         31        640: 100% 54/54 [00:29<00:00,  1.84it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.49it/s]
                   all        120       6604      0.714      0.806      0.824      0.396

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      4/149      7.49G     0.0728     0.1793   0.005649         20        640: 100% 54/54 [00:28<00:00,  1.87it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.52it/s]
                   all        120       6604      0.534      0.784      0.682      0.289
```

Slika 6.6 Uvježbavanje modela

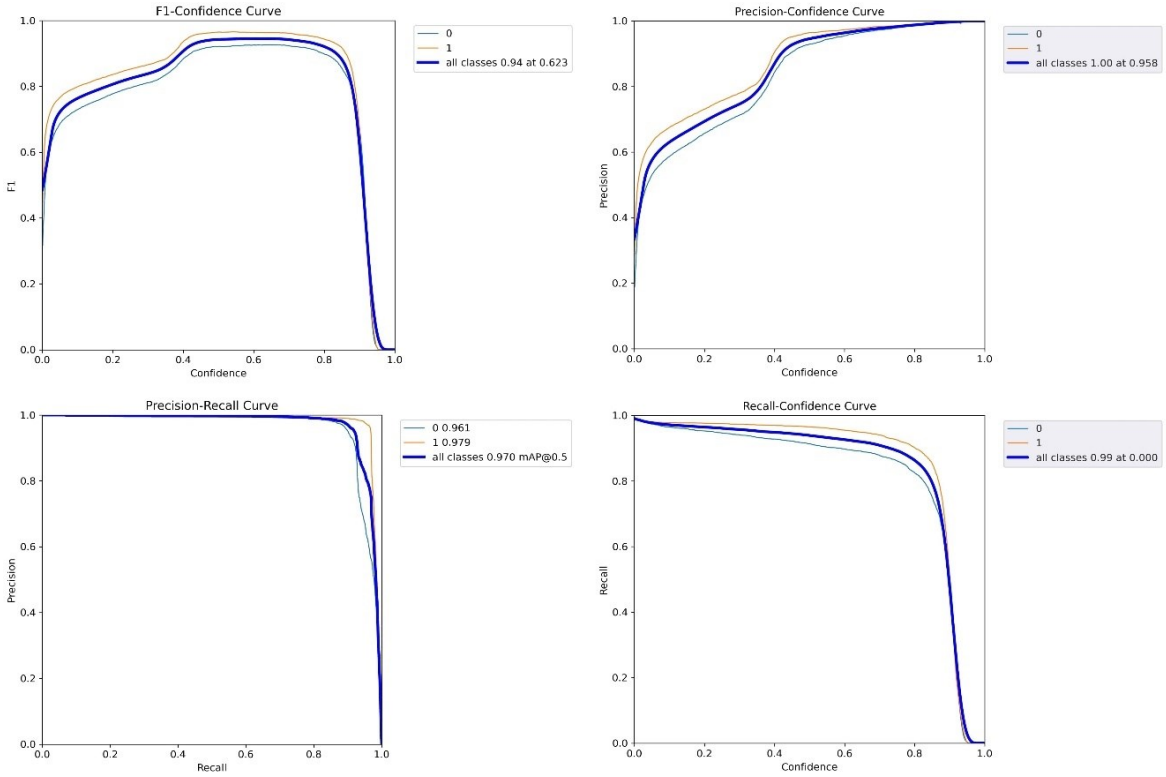
Kako se može vidjeti na slici, modeli koji se koristi za usporedbu rezultata se uvježbava na 150 epoha s obzirom na to da veći broj epoha (300,450) nije rezultirao poboljšanim rezultatima. Što se tiče veličine serije, zbog ograničenja memorije koja su se pojavila na većim verzijama algoritma odabrana je veličina 16 kako bi sve verzije radile u jednakim uvjetima.

Skripta val.py se koristi za evaluaciju modela pri čemu koristi slike i oznake iz test datoteke. Nakon izvršavanja prikazuje metrike koje prikazuju učinak modela prikazan na slici 6.7. Za svaku klasu se prikažu metrike preciznosti, odziva, mAP50 i mAP50-95. Također, val.py donosi informacije o brzini zaključivanja na različitim razinama, pretprocesuiranju, zaključivanju i NMS-u po slici za specificiranu veličinu slike. Na slici se vidi kako izgleda izvršavanje val.py skripte.

```
Model summary: 267 layers, 46113663 parameters, 0 gradients, 107.7 GFLOPs
test: Scanning /content/yolov5/test/labels.cache... 243 images, 0 backgrounds, 0 corrupt: 100% 243/243 [00:00<?, ?it/s]
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 8/8 [00:17<00:00,  2.18s/it]
                 all        243       8210      0.967      0.901      0.942      0.628
                   0        243       3563      0.967      0.859       0.91      0.619
                   1        243       4647      0.967      0.943      0.973      0.637
Speed: 0.3ms pre-process, 29.1ms inference, 6.7ms NMS per image at shape (32, 3, 640, 640)
Results saved to runs/val/exp2
```

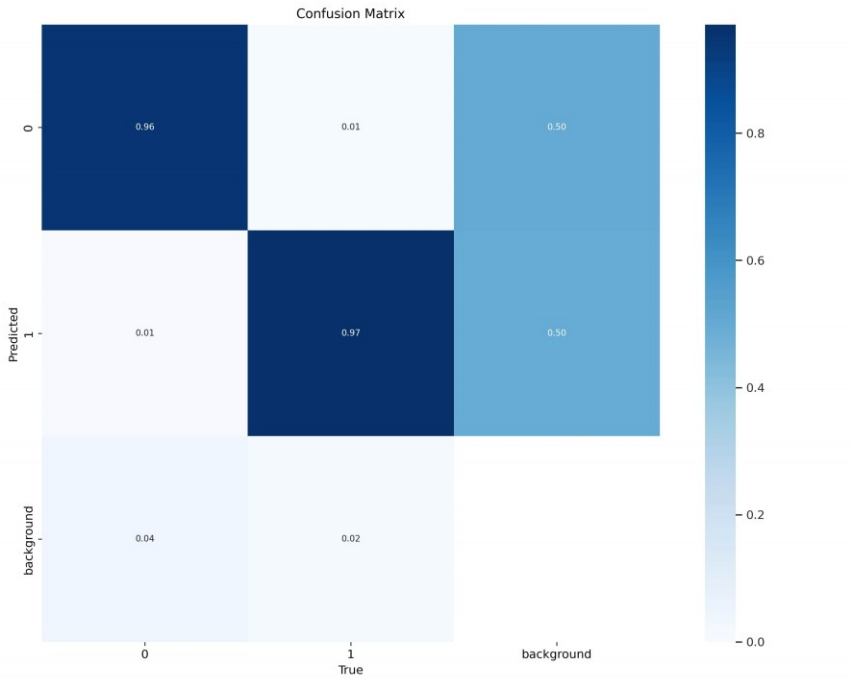
Slika 6.7 Prikaz izgleda evaluacije korištenjem val.py skripte

Od rezultata koje dobije, val.py radi i nekoliko grafova koje pohranjuje u datoteku kako bi bile dostupne za pregled i daljnje korištenje. Uz njih, generira nekoliko serija slika prikazanih na slici 6.8 koje služe za analizu i pregled.



Slika 6.8 Grafovi metrika koje YOLOv5 kreira pri evaluaciji

Na slici su prikazani grafovi koje skripta val.py generira: promjenu preciznosti, odziva i F1 ovisno o tome kako se mijenja prag pouzdanosti za detekciju objekta. Uz njih generira se graf preciznosti i odziva koji pokazuje međusoban odnos rasta i pada te dvije metrike. Uz navedene grafove generira se matrica konfuzije koja je prikazana na slici 6.9, a uz postojeće klase za objekte sadrži i klasu pozadine kako bi se mogla prikazati razdioba lažno pozitivnih i lažno negativnih primjera. Svaki stupac se nadopunjuje kako bi završni zbroj bio jednak jedan pa se na taj način prikazuje razdioba detekcije stvarnog objekta.



Slika 6.9 Matrica konfuzije koju generira val.py

Kako je već spomenuto, model je uvježbavan na pet glavnih verzija YOLOv5 i njihovim prethodno uvježbanim inačicama. Uvježbavanje se vršilo u 150 epoha, sa veličinom serije 16 i kao optimizacijska funkcija je zbog brzine korišten stohastički gradijentni spust. Pri radu je korištena besplatna verzija Colaboratory-a u kojoj se koristi T4 Tesla GPU koji sadrži 16GB memorije.

Rezultati mAP i brzine zaključivanja koji su se dobili evaluacijom na testnom skupu podatka, za navedene verzije nalaze se u tablici 6.1. Prvo su navedene verzije koje koriste težine prethodno uvježbane na COCO skupu podataka, a zatim verzije koje su se uvježbavale bez prethodno uvježbavanih težina. Rezultati srednje prosječne preciznosti su prikazani sa različitim pragovima IoU – 50 i računanjem različitih pragova od 0.5 do 0.95 sa stopom 0.05. Brzina se odnosi na brzinu zaključivanja po slici, a kako se u službenom YOLOv5 repozitoriju pri računanju zaključivanja nije uključena brzina koja se potroši na NMS algoritam ovdje se također ne uzima u obzir, a sama brzina se izražava u milisekundama.

Tablica 6.1 Rezultati evaluacije YOLOv5

Verzija	mAP@50	mAP@50-95	Brzina
YOLOv5n	0.833	0.41	6.1
YOLOv5s	0.917	0.499	11.8
YOLOv5m	0.945	0.563	19.0
YOLOv5l	0.948	0.565	32.8
YOLOv5x	0.954	0.575	49.9
YOLOv5n.pt	0.928	0.519	7.2
YOLOv5s.pt	0.964	0.592	10.4
YOLOv5m.pt	0.981	0.668	19.2
YOLOv5l.pt	0.984	0.697	29.7
YOLOv5x.pt	0.986	0.717	51.5

Generalno, može se primijetiti kako prethodno uvježbavani modeli pokazuju bolje rezultate nego njihove odgovarajuće verzije koje nisu koristile uvježbavane težine što je i logično s obzirom da se u osamdeset klasa na kojima su uvježbavani ima i različitih vrsta vozila. Navedeno također stavlja naglasak važnosti transfer learning-a [46] pri detekciji i klasifikaciji objekata.

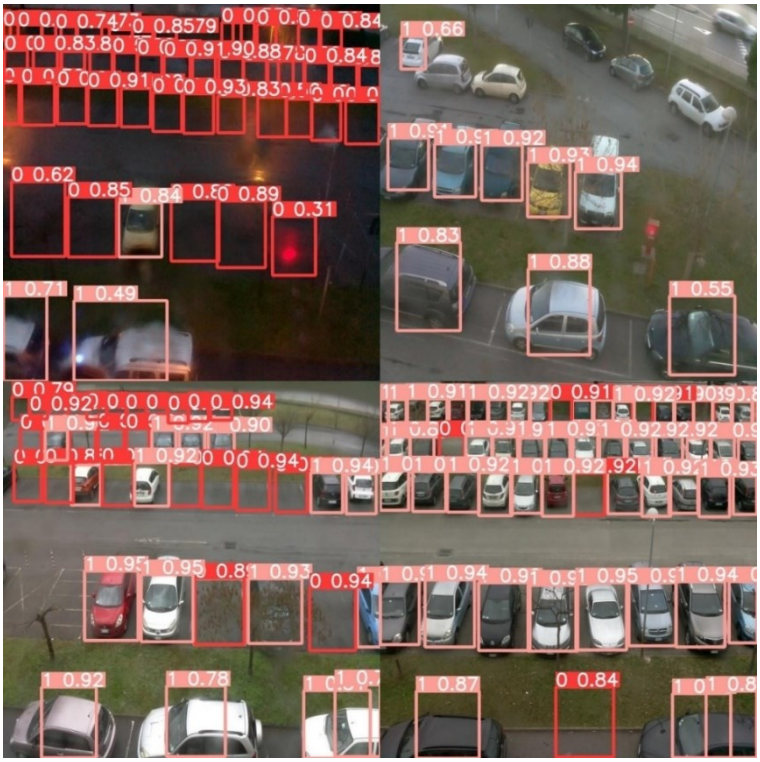
Kako se veličina modela povećava, tako rastu obje mAP mjere što pokazuje kako arhitekture s više parametara uspješno prepoznaju kompleksnije uzorke u podacima bez pojave mogućih šumova.

Kako YOLOv5 skripta za evaluaciju kao zadane vrijednosti pouzdanosti koristi dosta male vrijednosti (do 0.01) , a to radi kako bi se napravilo više predviđanja, što posljedično dovodi do većeg odziva. No, više previđanja dovodi do manje sigurnosti te do porasta lažno pozitivnih primjera i lošije predviđenih graničnih okvira. Povećanjem praga IoU (0.5-0.95)

uvode se stroži kriteriji kako bi se detekcija ocijenila kao točna. Iz tog razloga postoji velika razlika između mAP50 i mAP50-95.

Što se tiče brzine, manji modeli donose zaključke do nekoliko puta manje u odnosu na velike, što je velika prednost ukoliko bi se model koristio kada bi se radilo na snimanju parkirališta u realnom vremenu. No, model se može koristiti i na način da se fotografije s kamere uzimaju periodički (period može biti od nekoliko sekundi do nekoliko minuta, zavisno od tome koliko je sam parking korišten u toku dana) te se zatim slika analizira, za što vrijeme zaključivanja igra manju ulogu te su modeli s većim brojem parametara korisniji s obzirom na svoju preciznost. Kao najbolji omjer preciznosti i brzine pokazuje se model srednje veličine s prethodno uvježbanim težinama koji pokazuje neznatno lošije rezultate (ispod jedan posto razlike) od većih modela a zaključke donosi duplo brže od njih.

Kako bi se mogao provjeriti rad modela na djelu, autori su napravili detect.py skriptu koja koristi drugačije postavke rada i drugačije pragove IoU i pouzdanosti kako bi se model optimizirao za rad u stvarnom svijetu. Cilj mu je da pokaže kako se model ponaša u stvarnom svijetu pa detekcije radi brže nego skripta za evaluaciju jer ne računa nikakve metrike nego samo prikazuje i pohranjuje krajnji rezultat. Na slici 6.10 su prikazani rezultati koje detect.py pokazuje, a odabrani su određeni specifični slučajevi korištenjem modela koji se pokazao najpreciznijim pri testiranju modela – prethodno uvježbanim YOLOv5x modelom.



Slika 6.10 Generirani rezultati skripte detect.py

Prva slika pokazuje da model dobro detektira zauzeta i slobodna mjesta u uvjetima koji se ne smatraju jednostavnim jer uz sumrak gdje je osvijetljene smanjeno pada i kiša te se mogu razaznati i kapi na kameri. Unatoč tome model unatoč manjim ocjenama sigurnosti uspijeva detektirati i klasificirati parkirališna mjesta. Druga slici vidimo kako je model naučio razlikovati vozila koja su propisno parkirana, dok zanemaruje vozila koja se nalaze na mjestima koja nisu predviđena za parkiranje. Treća slici može se primijetiti kako model vozilo koje je zakriveno i čak teško zamjetno golim okom prepoznaje sa poprilično visokom sigurnošću. Također, na donje dvije slike se može primijetiti kako model za vozilo na istom parkirnom mjestu detektira dva puta, što može indicirati to da se vozilo nalazi u dvjema ćelijama te da svaka ćelija detektira navedeno vozilo zasebno.

Zaključak

U ovom se radu prikazana je upotreba YOLOv5, jednog od najboljih i najčešće korištenih algoritama za detekciju objekata na zadatku detekcije i klasifikacije slobodnih parking mjesta. Prvo se dao pregled razvoja računalnog vida, a zatim pregled istraženosti područja za navedeni problem koristeći PRISMA smjernice kako bi se napravila izdvojili referentni radovi.

Za uvježbavanje modela korišteni su javni skupovi podataka koji se, kako se ispostavilo analizom istraženosti ovog problema koriste u većini drugih radova. Uspoređene su postojeće YOLOv5 verzije algoritma u dva oblika: korištenjem težina koje su prethodno uvježbane na COCO skupu podataka i onih koje su se uvježbavale „od početka“. Korištenjem standardne metrike za evaluaciju rezultata – srednja prosječna preciznost, dobili su se rezultati koji su pokazali kako čak i manje verzije YOLOv5 algoritma mogu efikasno detektirati i klasificirati slobodna i zauzeta parking mjesta što model čini pogodnim za postavljanje na fiksnim kamerama bez potrebe za spajanjem na poslužitelje koji bi zatim obrađivali slike koje dobiju tako da bi se informacije o dostupnosti mjesta mogle koristiti i lokalno. Važno je napomenuti da je bilo potrebno u skup za uvježbavanje dodati određen broj fotografija (10%) kako bi, s obzirom na slabu vidljivost oznaka na parking mjestu, sustav ipak naučio prepoznati takve oznake. Također, pokazalo se kako se model uspijeva prilagoditi raznim vremenskim uvjetima kao što su kiša, oblačno vrijeme ili mrak. Uz to, model uspijeva raspoznati mjesta čak i kada postoje zapreke kao što su rasvjetni stupovi ili krošnje stabala.

Svakodnevno korištenje ovakvih modela bilo bi korisno za različite interesne skupine: vlasnici parkirališta (javni ili privatni) mogli bi, zavisno o podataka popunjenosti parkinga, efikasnije upravljati prostorom kojeg imaju ili zavisno o obrascima korištenja regulirati svoje cijene i time povećati svoje prihode. Također, ukoliko je riječ o frekventnim mjestima kao što su bolnice ili aerodromi, korisnicima bi se mogao ponuditi sustav automatskog navođenja kako bi imali ugodnije iskustvo korištenja parkirališta i trošili manje vremena na pronalazak slobodnog mjesta. Lokalne uprave, pogotovo u većim gradovima, mogle bi razviti sustav pametnog parkiranja tako što bi, ovisno o zauzeću pojedinih parkirališta mogli preusmjeriti svoje građane i goste na najbliže parkiralište sa slobodnim mjestima što ima sljedeće prednosti: uz to što će smanjiti gužve i zagađenost u gradu te smanjuje razinu stresa koja se

često stvori kod ljudi koji duže vrijeme traže parking mjesto. Krajnji korisnici, sami vozači, bi imali koristi: od navedenu uštede dragocjenog vremena i živaca, ali i financijsku uštedu s obzirom na moguću manju potrošnju goriva smanjenjem prijeđenog puta do parkirnog mjesta i potencijalno manjih gužvi u gradu koje su uzrok povećavanju potrošnje goriva po prijeđenom kilometru. Kod korišten u radu dostupan je na [82].

Reference

[1] G. Jocher, 'YOLOv5 by Ultralytics'. May 2020. doi: 10.5281/zenodo.3908559.

[2] 'Roboflow: Give your software the power to see objects in images and video'. Accessed: Feb. 21, 2024. [Online]. Available: https://roboflow.com/

[3] 'colab.google'. Accessed: Feb. 21, 2024. [Online]. Available: https://colab.google/

[4] 'What is Python? Executive Summary', Python.org. Accessed: Feb. 21, 2024. [Online]. Available: https://www.python.org/doc/essays/blurb/

[5] 'scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation'. Accessed: Mar. 18, 2024. [Online]. Available: https://scikit-learn.org/stable/

[6] 'pandas - Python Data Analysis Library'. Accessed: Mar. 18, 2024. [Online]. Available: https://pandas.pydata.org/

[7] 'Keras: Deep Learning for humans'. Accessed: Mar. 18, 2024. [Online]. Available: https://keras.io/

[8] 'TensorFlow'. Accessed: Mar. 18, 2024. [Online]. Available: https://www.tensorflow.org/

[9] 'Matplotlib — Visualization with Python'. Accessed: Mar. 18, 2024. [Online]. Available: https://matplotlib.org/

[10] 'NLTK :: Natural Language Toolkit'. Accessed: Mar. 18, 2024. [Online]. Available: https://www.nltk.org/

[11] 'The Jupyter Notebook — Jupyter Notebook 7.1.0 documentation'. Accessed: Feb. 21, 2024. [Online]. Available: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

[12] X. Zou, 'A Review of Object Detection Techniques', in *2019 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, Aug. 2019, pp. 251–254. doi: 10.1109/ICSGEA.2019.00065.

[13] P. Viola and M. Jones, 'Rapid object detection using a boosted cascade of simple features', in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Dec. 2001, p. I–I. doi: 10.1109/CVPR.2001.990517.

[14] N. Dalal and B. Triggs, 'Histograms of oriented gradients for human detection', in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Jun. 2005, pp. 886–893 vol. 1. doi: 10.1109/CVPR.2005.177.

[15] P. Felzenszwalb, D. McAllester, and D. Ramanan, 'A discriminatively trained, multiscale, deformable part model', in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2008, pp. 1–8. doi: 10.1109/CVPR.2008.4587597.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, 'Rich feature hierarchies for accurate object detection and semantic segmentation'. arXiv, Oct. 22, 2014. doi: 10.48550/arXiv.1311.2524.

[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik, 'A training algorithm for optimal margin classifiers', in *Proceedings of the fifth annual workshop on Computational learning*

theory, in COLT '92. New York, NY, USA: Association for Computing Machinery, Srpanj 1992, pp. 144–152. doi: 10.1145/130385.130401.

[18] K. He, X. Zhang, S. Ren, and J. Sun, 'Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition', vol. 8691, 2014, pp. 346–361. doi: 10.1007/978-3-319-10578-9_23.

[19] R. Girshick, 'Fast R-CNN'. arXiv, Sep. 27, 2015. doi: 10.48550/arXiv.1504.08083.

[20] S. Ren, K. He, R. Girshick, and J. Sun, 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'. arXiv, Jan. 06, 2016. doi: 10.48550/arXiv.1506.01497.

[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, 'Feature Pyramid Networks for Object Detection'. arXiv, Apr. 19, 2017. doi: 10.48550/arXiv.1612.03144.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection'. arXiv, May 09, 2016. doi: 10.48550/arXiv.1506.02640.

[23] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, 'YOLOv4: Optimal Speed and Accuracy of Object Detection'. arXiv, Apr. 22, 2020. doi: 10.48550/arXiv.2004.10934.

[24] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, 'YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors'. arXiv, Jul. 06, 2022. doi: 10.48550/arXiv.2207.02696.

[25] C. Li *et al.*, 'YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications'. arXiv, Sep. 07, 2022. doi: 10.48550/arXiv.2209.02976.

[26] G. Jocher, A. Chaurasia, and J. Qiu, 'Ultralytics YOLO'. Jan. 2023. Accessed: Mar. 14, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[27] W. Liu *et al.*, 'SSD: Single Shot MultiBox Detector', vol. 9905, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, 'Focal Loss for Dense Object Detection'. arXiv, Feb. 07, 2018. doi: 10.48550/arXiv.1708.02002.

[29] A. Mao, M. Mohri, and Y. Zhong, 'Cross-Entropy Loss Functions: Theoretical Analysis and Applications'. arXiv, Jun. 19, 2023. doi: 10.48550/arXiv.2304.07288.

[30] H. Law and J. Deng, 'CornerNet: Detecting Objects as Paired Keypoints'. arXiv, Mar. 18, 2019. doi: 10.48550/arXiv.1808.01244.

[31] X. Zhou, D. Wang, and P. Krähenbühl, 'Objects as Points'. arXiv, Apr. 25, 2019. doi: 10.48550/arXiv.1904.07850.

[32] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, 'End-to-End Object Detection with Transformers'. arXiv, May 28, 2020. doi: 10.48550/arXiv.2005.12872.

[33] A. Vaswani *et al.*, 'Attention Is All You Need'. arXiv, Aug. 01, 2023. doi: 10.48550/arXiv.1706.03762.

[34] 'What is image classification? Basics you need to know | SuperAnnotate'. Accessed: Feb. 21, 2024. [Online]. Available: https://www.superannotate.com/blog/image-classification-basics

[35] INRIX, ‘Smart Parking – A Silver Bullet for Parking Pain’, Inrix. Accessed: Feb. 21, 2024. [Online]. Available: https://inrix.com/blog/parkingsurvey/

[36] M. J. Page *et al.*, ‘The PRISMA 2020 statement: an updated guideline for reporting systematic reviews’, *BMJ*, vol. 372, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.

[37] P. R. L. de Almeida, J. H. Alves, R. S. Parpinelli, and J. P. Barddal, ‘A Systematic Review on Computer Vision-Based Parking Lot Management Applied on Public Datasets’, *Expert Syst. Appl.*, vol. 198, p. 116731, Jul. 2022, doi: 10.1016/j.eswa.2022.116731.

[38] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, ‘Car parking occupancy detection using smart camera networks and Deep Learning’, in *2016 IEEE Symposium on Computers and Communication (ISCC)*, Jun. 2016, pp. 1212–1217. doi: 10.1109/ISCC.2016.7543901.

[39] A. Krizhevsky, I. Sutskever, and G. Hinton, ‘ImageNet Classification with Deep Convolutional Neural Networks’, *Neural Inf. Process. Syst.*, vol. 25, Jan. 2012, doi: 10.1145/3065386.

[40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, ‘Gradient-based learning applied to document recognition’, *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[41] V. Nair and G. E. Hinton, ‘Rectified linear units improve restricted boltzmann machines’, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, in ICML’10. Madison, WI, USA: Omnipress, Lipanj 2010, pp. 807–814.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ‘Dropout: A Simple Way to Prevent Neural Networks from Overfitting’, *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jun. 2014.

[43] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, ‘Deep learning for decentralized parking lot occupancy detection’, *Expert Syst. Appl.*, vol. 72, pp. 327–334, Apr. 2017, doi: 10.1016/j.eswa.2016.10.055.

[44] K. He, X. Zhang, S. Ren, and J. Sun, ‘Deep Residual Learning for Image Recognition’. arXiv, Dec. 10, 2015. doi: 10.48550/arXiv.1512.03385.

[45] A. B. Ebren and B. Bolat, ‘Determining the Occupancy of Vehicle Parking Areas by Deep Learning’, in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, Jun. 2020, pp. 1–4. doi: 10.1109/ICECCE49384.2020.9179399.

[46] S. J. Pan and Q. Yang, ‘A Survey on Transfer Learning’, *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[47] K. Simonyan and A. Zisserman, ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’. arXiv, Apr. 10, 2015. doi: 10.48550/arXiv.1409.1556.

[48] V. Dhuri, A. Khan, Y. Kamtekar, D. Patel, and I. Jaiswal, ‘Real-Time Parking Lot Occupancy Detection System with VGG16 Deep Neural Network using Decentralized Processing for Public, Private Parking Facilities’, in *2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Feb. 2021, pp. 1–8. doi: 10.1109/ICAECT49130.2021.9392506.

[49] F. Dornaika, K. Hammoudi, M. Melkemi, and T. D. A. Phan, ‘An efficient pyramid multi-level image descriptor: application to image-based parking lot monitoring’, *Signal Image Video Process.*, vol. 13, no. 8, pp. 1611–1617, Nov. 2019, doi: 10.1007/s11760-019-01512-6.

[50] A. Hadid, ‘The Local Binary Pattern Approach and its Applications to Face Analysis’, in *2008 First Workshops on Image Processing Theory, Tools and Applications*, Nov. 2008, pp. 1–9. doi: 10.1109/IPTA.2008.4743795.

[51] M. Farag, M. Din, and H. Elshenbary, ‘Deep learning versus traditional methods for parking lots occupancy classification’, *Indones. J. Electr. Eng. Comput. Sci.*, vol. 19, p. 964, Aug. 2020, doi: 10.11591/ijeecs.v19.i2.pp964-973.

[52] M. A. Merzoug, A. Mostefaoui, and A. Benyahia, ‘Smart IoT Notification System for Efficient In-City Parking’, in *Proceedings of the 15th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, in Q2SWinet’19. New York, NY, USA: Association for Computing Machinery, Studeni 2019, pp. 37–42. doi: 10.1145/3345837.3355954.

[53] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ‘MobileNetV2: Inverted Residuals and Linear Bottlenecks’. arXiv, Mar. 21, 2019. doi: 10.48550/arXiv.1801.04381.

[54] J. Redmon and A. Farhadi, ‘YOLOv3: An Incremental Improvement’. arXiv, Apr. 08, 2018. doi: 10.48550/arXiv.1804.02767.

[55] M. Kolhar and A. Alameen, ‘Multi Criteria Decision Making System for Parking System’, *Comput. Syst. Sci. Eng.*, vol. 36, pp. 101–116, Jan. 2021, doi: 10.32604/csse.2021.014915.

[56] S. Nurullayev and S.-W. Lee, ‘Generalized Parking Occupancy Analysis Based on Dilated Convolutional Neural Network’, *Sensors*, vol. 19, no. 2, Art. no. 2, Jan. 2019, doi: 10.3390/s19020277.

[57] A. Varghese and G. Sreelekha, ‘An Efficient Algorithm for Detection of Vacant Spaces in Delimited and Non-Delimited Parking Lots’, *IEEE Trans. Intell. Transp. Syst.*, vol. PP, pp. 1–11, Aug. 2019, doi: 10.1109/TITS.2019.2934574.

[58] S. O’Hara and B. A. Draper, ‘Introduction to the Bag of Features Paradigm for Image Classification and Retrieval’. arXiv, Jan. 17, 2011. doi: 10.48550/arXiv.1101.3354.

[59] L. Wang *et al.*, ‘Global Perception-Based Robust Parking Space Detection Using a Low-Cost Camera’, *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1439–1448, Feb. 2023, doi: 10.1109/TIV.2022.3186035.

[60] X. Zhang, X. Zhou, M. Lin, and J. Sun, ‘ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices’. arXiv, Dec. 07, 2017. doi: 10.48550/arXiv.1707.01083.

[61] S. Goumiri, D. Benboudjema, and W. Pieczynski, ‘One Convolutional Layer Model For Parking Occupancy Detection’, in *2021 IEEE International Smart Cities Conference (ISC2)*, Sep. 2021, pp. 1–5. doi: 10.1109/ISC253183.2021.9562946.

[62] T.-Y. Lin *et al.*, ‘Microsoft COCO: Common Objects in Context’. arXiv, Feb. 20, 2015. doi: 10.48550/arXiv.1405.0312.

[63] P. R. L. de Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich, 'PKLot – A robust dataset for parking lot classification', *Expert Syst. Appl.*, vol. 42, no. 11, pp. 4937–4949, Jul. 2015, doi: 10.1016/j.eswa.2015.02.009.

[64] 'PyTorch', PyTorch. Accessed: Mar. 14, 2024. [Online]. Available: https://pytorch.org/

[65] Ultralytics, 'Home'. Accessed: Feb. 24, 2024. [Online]. Available: https://docs.ultralytics.com/

[66] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, 'Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression'. arXiv, Apr. 14, 2019. Accessed: Mar. 14, 2024. [Online]. Available: http://arxiv.org/abs/1902.09630

[67] J. Hosang, R. Benenson, and B. Schiele, 'Learning non-maximum suppression'. arXiv, May 09, 2017. doi: 10.48550/arXiv.1705.02950.

[68] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, 'CSPNet: A New Backbone that can Enhance Learning Capability of CNN'. arXiv, Nov. 26, 2019. Accessed: Feb. 27, 2024. [Online]. Available: http://arxiv.org/abs/1911.11929

[69] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, 'Path Aggregation Network for Instance Segmentation'. arXiv, Sep. 18, 2018. doi: 10.48550/arXiv.1803.01534.

[70] A. Youssef, 'Image Downsampling and Upsampling Methods'.

[71] 'Overview of model structure about YOLOv5 · Issue #280 · ultralytics/yolov5', GitHub. Accessed: Feb. 29, 2024. [Online]. Available: https://github.com/ultralytics/yolov5/issues/280

[72] J. Redmon and A. Farhadi, 'YOLO9000: Better, Faster, Stronger'. arXiv, Dec. 25, 2016. doi: 10.48550/arXiv.1612.08242.

[73] Z. Zheng *et al.*, 'Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation'. arXiv, Jul. 05, 2021. Accessed: Mar. 14, 2024. [Online]. Available: http://arxiv.org/abs/2005.03572

[74] L. Bottou, 'Stochastic Gradient Descent Tricks', in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., Berlin, Heidelberg: Springer, 2012, pp. 421–436. doi: 10.1007/978-3-642-35289-8_25.

[75] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization'. arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

[76] M. Sokolova and G. Lapalme, 'A systematic analysis of performance measures for classification tasks', *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.

[77] E. Zhang and Y. Zhang, 'Average Precision', in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds., Boston, MA: Springer US, 2009, pp. 192–193. doi: 10.1007/978-0-387-39940-9_482.

[78] K. M. Ting, 'Confusion Matrix', in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2010, pp. 209–209. doi: 10.1007/978-0-387-30164-8_157.

[79] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, 'mixup: Beyond Empirical Risk Minimization'. arXiv, Apr. 27, 2018. doi: 10.48550/arXiv.1710.09412.

[80] G. Ghiasi *et al.*, ‘Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation’. arXiv, Jun. 23, 2021. doi: 10.48550/arXiv.2012.07177.

[81] ‘yolov5/requirements.txt at master · ultralytics/yolov5 · GitHub’. Accessed: Feb. 24, 2024. [Online]. Available: https://github.com/ultralytics/yolov5/blob/master/requirements.txt

[82] ‘jtomasevic123/dipl’, GitHub. Accessed: Apr. 03, 2024. [Online]. Available: https://github.com/jtomasevic123/dipl