Detekcija i praćenje vozila u prometu korištenjem YOLOv8

Tus, Ivana

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka, Faculty of Engineering / Sveučilište u Rijeci, Tehnički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:190:290258

Rights / Prava: Attribution 4.0 International/Imenovanje 4.0 međunarodna

Download date / Datum preuzimanja: **2025-03-29**

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Engineering

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni diplomski studij računarstva

Diplomski rad

Detekcija i praćenje vozila u prometu korištenjem YOLOv8

Rijeka, rujan 2024.

Ivana Tus
0069085174

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Sveučilišni diplomski studij računarstva

Diplomski rad

Detekcija i praćenje vozila u prometu korištenjem YOLOv8

Mentor: prof.dr.sc. Jonatan Lerga

Rijeka, rujan 2024.

Ivana Tus
0069085174

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA DIPLOMSKE ISPITE

Rijeka, 13.03.2024.

Zavod: Zavod za računarstvo

Predmet: Kodiranje i kriptografija

ZADATAK ZA DIPLOMSKI RAD

Pristupnik: **Ivana Tus (0069085174)**

Studij: Sveučilišni diplomski studij računarstva (1400)

Modul: Programsko inženjerstvo (1441)

Zadatak: **Detekcija i praćenje vozila u prometu korištenjem YOLOv8 / Detection and Tracking of Vehicles in Traffic Using YOLOv8**

Opis zadatka:

Potrebno je razviti sustav računalnog vida za detekciju i praćenje vozila iz slika korištenjem YOLOv8. Isti je potrebno testirati na snimkama prometa u Rijeci pri različitim prometnim uvjetima. Temeljem detektiranog stanja u prometu potrebno je predložiti alternativne rute u slučaju prometnih gužvi ili radova na cesti (koristiti OpenStreetMap bazu.) Po provedenoj detekciji objekata potrebno je analizirati, vizualizirati i interpretirati dobivene rezultate.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20.03.2024.

Predsjednik povjerenstva za diplomski ispit:
prof. dr. sc. Miroslav Joler

Mentor:
prof. dr. sc. Jonatan Lerga

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, rujan 2024.

_____

Ivana Tus

Zahvala

Zahvaljujem obitelji i prijateljima na ukazanoj podršci i poticaju tokom studija i izvan studija.
Zahvaljujem prof. dr. sc. Jonatanu Lergi na vodstvu, uputama i savjetima pri pisanju diplomskog rada.
Zahvaljujem svim kolegama, asistentima i profesorima na razmjeni znanja, pomoći i savjetima kroz studij.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Umjetna inteligencija naziv je koji se najčešće pridaje računalnim sustavima koji imaju način snalaženja u novim situacijama. U današnje doba umjetna se inteligencija koristi u razne svrhe te se u potpunosti uplela u živote običnih ljudi. Dijeli se na razne grane poput obrade prirodnoga jezika (eng. *natural language processing*, skraćeno NPL), robotike, stručnih sustava, neuronskih mreža te strojnog učenja (eng. *machine learning*), kojime se bavi ovaj rad. Strojno učenje definira se kao grana umjetne inteligencije koja se bavi oblikovanjem algoritama koji svoju učinkovitost poboljšavaju na temelju empirijskih podataka. Računalni vid jedno je od najpopularniji područja u kojem se upotrebljava strojno učenje. Od prvih modela baziranih na ključnim riječima (koji su i dalje zahtijevali ljudski rad kako bi se slika povezala s ključnim riječima), područje se razvilo do današnjih modela baziranih na dubokim konvolucijskim neuronskim mrežama (eng. *deep convolutional neural network*, skraćeno DCNN). Modeli DCNN-a zahtijevaju obuku na (po mogućnosti velikoj) bazi podataka slika u odabranim područjima obuke i pružaju više od 90 posto točnih identifikacija objekata. Modeli DCNN-a zahtijevaju malo podešavanja nakon pravilne obuke, što eliminira veći dio ručnog rada nakon obuke modela. Neki koncepti poput praćenja objekata i prepoznavanja objekata isprepletani su s područjem računalnog vida, posebno u kontekstu analize videozapisa. Na čelu razvoja računalnog vida je programski jezik Python i njegove mnoge podržane knjižnice, od kojih je OpenCV najvažnija za računalni vid. Najpopularniji model za detekciju i praćenje objekata danas, baziran na ovoj posebnoj knjižnici, je YOLO (skraćeno od eng.

You Only Look Once, odnosno hrv. *samo jednom pogledaš*), razvijen od strane američke tvrtke Ultralytics. YOLO se široko koristi u raznim područjima. Ovaj projekt kombinira područje računalnog vida sa statističkom analizom podataka kako bi istražio načine na koje razni objekti mogu biti potpuno analizirani (putem detekcije i praćenja) automatskim radom računala, što je od velike koristi u području analize prometa, jer se u video zapisima raznih prometnih područja može pronaći mnogo prometnih podataka. U analizi prometa, ovi sustavi se ističu u detekciji i praćenju vozila, pješaka i drugih objekata, omogućavajući automatsko praćenje prometa i sigurnosti. Gradovi poput Singapura su uveli sustave računalnog vida za automatsko praćenje stanja na cestama i prometnog toka, što je rezultiralo učinkovitijim prometnim sustavima. Osim u prometu, industrije poput maloprodaje također imaju koristi od ovih tehnologija za praćenje kupaca i analizu ponašanja.

1.1 Struktura rada

Rad je podijeljen na klijentsku i poslužiteljsku stranu koje su međusobno povezane pomoću platforme za računarstvo u oblaku. Za poslužiteljsku stranu korišteno je prijenosno računalo s 64-bitnim operativnim sustavom Linux Ubuntu 22.04.3 LTS, procesorom AMD Ryzen 7 5700u with radeon graphics × 16, integriranom grafičkom karticom RENOIR (renoir, LLVM 15.0.7, DRM 3.54, 6.5.0-35-generic) te 16 gigabajta radne memorije, dok se klijentska strana pokretala na uređajima s operativnim sustavom Android, verzija 11 ili više s mogućnošću pristupa lokaciji i kameri uređaja.

1.1.1 Poslužiteljska strana

Poslužiteljska strana sastoji se od obrade video snimaka računalnim vidom koji detekcijom i praćenjem objekata izvlači podatke o prometu na video snimci. Ti podaci potom se zapisuju te statistički obrađuju kako bi se dobila analitika i opis prometa na snimci. S obzirom da je računalni vid implementiran pomoću programskog jezika Python, radi jednostavnosti je i statistička obrada podataka implementirana u istom jeziku. Pokretanje Python skripti od kojih se sastoji poslužiteljska strana automa-

tizirano je pomoću *bash* skripte koja jednostavno pokreće početnu Python skriptu koja nastavlja s radom kompletnog sustava poslužiteljske strane. Po završetku rada sustava za obradu podataka, rezultati se šalju na platformu za računarstvo u oblaku kako bi bili dostupni klijentskoj strani.

1.1.2 Klijentska strana

Klijentska strana sastoji se od nativne Android aplikacije razvijene u Java programskom jeziku. Aplikacija je prilagođena za Android verziju 7.0 Nougat i novije čime je ciljano tržište 97.4% svih Android uređaja. Aplikacija se može koristiti na svakom Android mobilnom ili tablet uređaju s predviđenom verzijom operacijskog sustava, te za pravilan rad zahtjeva pristup internetu, lokaciji uređaja, koristi se ugrađenom kamerom samog uređaja i njegovom ugrađenom pohranom. Kako bi se aplikacija mogla koristiti kamerom i unutarnjom pohranom, korisnik mora dati ekspliticna dopuštenja, a ukoliko to odbije učiniti, neke funkcionalnosti aplikacije neće biti dostupne.

1.1.3 Motivacija i ciljevi rada

S obzirom na učestale građevinske radove te promjene u tokovima prometnica u gradu Rijeci, motivacija rada je pružiti sustav koji na temelju dotadašnjih podataka može uputiti korisnika u stanje prometa na određenoj prometnici tokom određenog dijela dana. Takav sustav uvelike bi olakšao i ubrzao prometni tranzit na odredište ili u krajnem slučaju uputio korisnika na ranije kretanje zbog duljeg predviđenog vremena putovanja.

Poglavlje 2

Statistička obrada podataka

2.1 Deskriptivna statistika

Deskriptivna statistika odnosi se na osnovne opisne karakteristike promatranog skupa podataka. Deskriptivna statistika prikazuje mjere centralne tendencije skupa kao što su prosječna srednja vrijednost skupa podataka, medijan skupa podataka i mod skupa podataka, te standardnu devijaciju skupa podataka. Prosječna srednja vrijednost predstavlja središnju numeričku vrijednost skupa, odnosno prosjek svake numeričke vrijednosti koja se nalazi u skupu, te se računa pomoću jednadžbe:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.1}$$

Kao što je vidljivo u jednadžbi, prosječna srednja vrijednost dobivena je dijeljenjem sume svih numeričkih vrijednosti skupa s brojem numeričkih vrijednosti skupa. Ekstremne vrijednosti (tzv. eng. *outliers*) mogu imati značajni utjecaj na prosječnu srednju vrijednost skupa. Medijan skupa podataka smatra se ona vrijednost od koje je pedeset posto vrijednosti skupa podataka veće i pedeset posto vrijednosti skupa podataka manje, što medijan čini mjerom središta distribucije vrijednosti skupa podataka. Medijan je otporniji na utjecaj ekstremnih vrijednosti od prosječne srednje vrijednosti. Nema eksplicitnu jednadžbu kojom se računa, već se pronalazi tako da se vrijednosti skupa poredaju po veličini, a medijan se nalazi u samoj sredini takvog skupa. Mod skupa podataka je ona numerička vrijednost koja se najčešće ponavlja

u skupu podataka te predstavlja dominantnu vrijednost unutar samog skupa. Mod se računa pomoću frekvencija ponavljanja vrijednosti unutar skupa podataka, odnosno mod predstavlja vrijednost s najvećom frekvencijom ponavljanja. Standardna devijacija ne predstavlja mjeru centralne tendencije, već standardno (prosječno) odstupanje vrijednosti od prosječne srednje vrijednosti skupa. Uzima se kao mjera raspršenosti skupa jer pokazuje koliko dobro mjere centralne tendencije mogu opisati skup (što je standardna devijacija manja, to su mjere centralne tendencije bolje opisale skup podataka). Jednadžba za izračun standardne devijacije izgleda na sljedeći način:

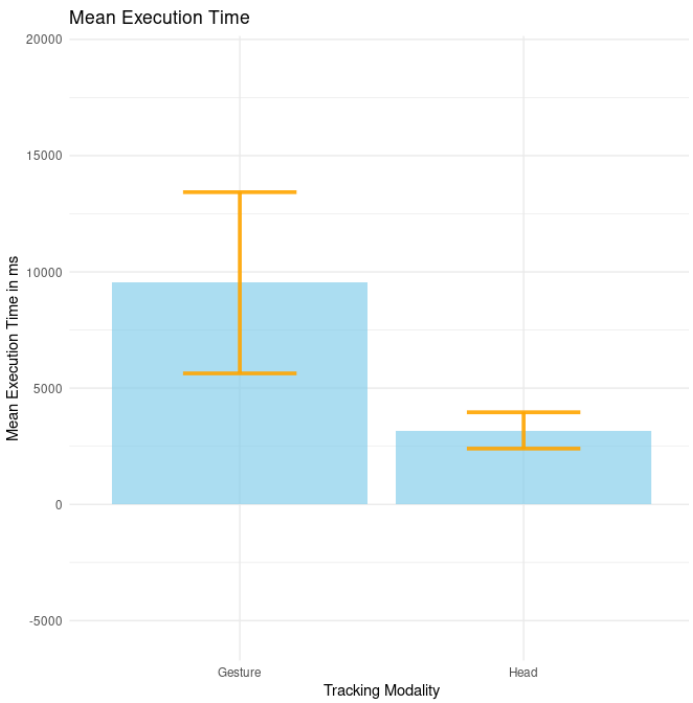
$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2.2}$$

Može se primjetiti da jednadžba sadrži izračunatu prosječnu srednju vrijednost skupa podataka od koje se računa razlika za svaku vrijednost u skupu. Standardna devijacija nam ukazuje na postojanje ekstremnih vrijednosti u skupu.

2.1.1 Vizualizacija deskriptivne statistike

Deskriptivna statistika je učestalo prikazana vizualizacijom iste pomoću različitih grafova. Svaka komponenta deskriptivne statistike ima standardni način vizualizacije. Prosječna srednja vrijednost uobičajeno su prikazane zajedno na stupičastom grafu s intervalima povjerenja. Visina stupičastog grafa odnosi se na izračunatu prosječnu srednju vrijednost, dok intervali povjerenja prikazuju standardnu devijaciju od prosječne srednje vrijednosti. Na grafu je obavezno na osi ordinati naznačiti numeričke vrijednosti te mjernu jedinicu mjerenja, dok je na osi apscisi potrebno naznačiti skup podataka na koji se stupičasti graf odnosi. Primjer pravilnog prikaza stupičastog grafa s intervalima povjerenja prikazan je slikom 2.1.

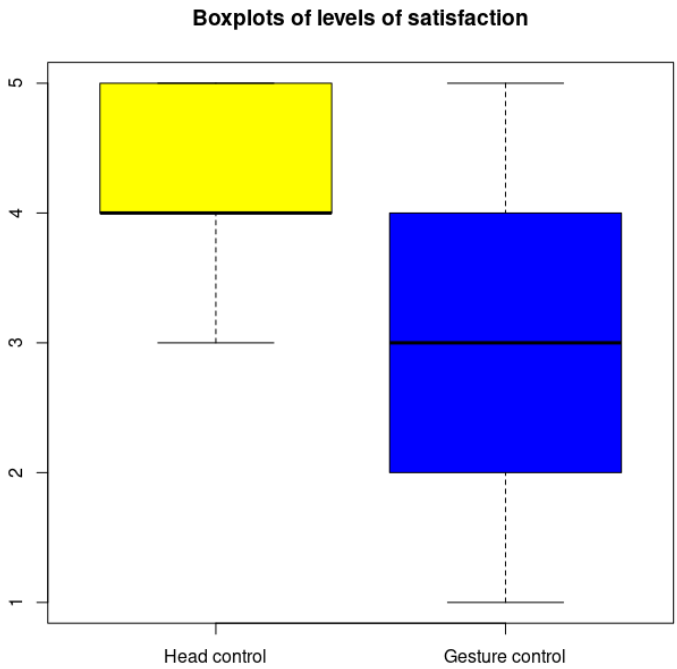
Osim stupičastog grafa, često korišteni graf je kutijasti dijagram koji prikazuje distribuciju vrijednosti unutar skupa podataka. Kutijasti dijagram sastoji se od od pravokutnika koji ukazuje na distribuciju vrijednosti skupa podataka te tri vodoravne linije koje redoslijedom od niže prema višem prikazuju najnižu (ekstremnu) vrijednost u skupu podataka, medijan te najvišu (ekstremnu) vrijednost u skupu podataka. Osim ekstremnih vrijednosti i medijana, kutijasti dijagram prikazuje i kvartile skupa



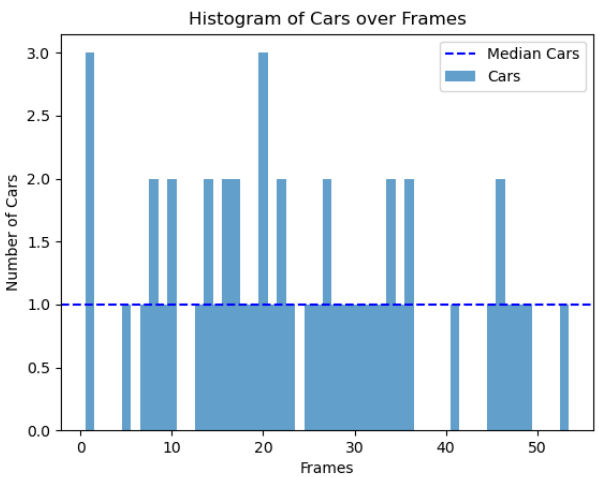
Slika 2.1 Primjer stupičastog grafa na kojemu su prikazane prosječna srednja vrijednost (plava) i standardna devijacija (narančasta).

podataka. Kvartili skupa podataka odnose se na podjelu skupa podataka na četiri dijela ovisno o vrijednosti podataka, a s obzirom da poredaju podatke po veličini, smatra ih se statistikom reda podataka. Primjer prikaza kutijastog dijagrama nalazi se na slici 2.2.

S obzirom da se mod odnosi na vrijednost s najvećom frekvencijom pojavljivanja u skupu podataka, najčešći način prikaza je histogramima frekvencija. Histogram je stupičasti dijagram čija os apscisa označava vrijednost za koju je histogram nacrtan, dok os ordinata označava broj ponavljanja (frekvenciju). Modom se smatra vrijednost čiji je stup histograma najviši, odnosno ima najveći broj ponavljanja. Primjer histograma (dodatno s prikazom medijana) nalazi se na slici 2.3.



Slika 2.2 Primjer kutijastog dijagrama na kojemu su prikazani medijan, ekstremne vrijednosti te distribucija vrijednosti u skupu podataka.



Slika 2.3 Primjer histograma na kojemu su prikazani frekvencija ponavljanja vrijednosti u skupu podataka te medijan.

2.2 Inferencijalna statistika

Inferencijalna statistika odnosi se na statističke testove, odnosno na ispitivanje istinitosti određenih hipoteza koji se odnose na skupove podataka. Pojam hipoteza odnosi se na pretpostavku karakteristike skupa podataka. Svaka hipoteza statističkog testa ima alternativnu hipotezu (uobičajeno suprotnu u karakteristici skupa) te se ispituje istinitost jedne od njih, nazivaju se nulta i alternativna hipoteza. Kako bi se moglo govoriti o inferencijalnoj statistici, potrebno je razumijevanje nekolicine pojmova poput testne statistike, p-vrijednosti, intervala povjerenja i standardne pogreške. Testna statistika je naziv za numeričku vrijednost koja se dobije izračun iz uzorka podataka tokom statističkog testiranja. Testna statistika razlikuje se za svaki statistički test. P-vrijednost je proizvod testiranja hipoteza putem različitih statističkih testova[1]. Važnost p-vrijednosti je prihvaćanje ili odbacivanje nulte hipoteze u odnosu na to je li p-vrijednost veća ili manja od granične vrijednosti 0.05. Ukoliko je p-vrijednost veća od 0.05 prihvaćena je nulta hipoteza, no u slučaju da je p-vrijednost manja od 0.05 odbacuje se nulta hipoteza i prihvaća alternativna. Interval povjerenja je mjera za nesigurnost istraživača u statistici uzorka kao procjena parametra populacije, ako se proučava manji uzorak od cijele populacije.[1] Standardna pogreška je standardna devijacija srednje vrijednosti uzorka podijeljena s kvadratnim korijenom veličine uzorka, što pokazuje očekivanu varijaciju srednje vrijednosti uzorka ako se eksperiment ponovi s različitim uzorcima iz iste populacije, odnosno skupine podataka.[1] Uz razumijevanje spomenutih pojmova, potrebno je i razlikovanje parametarskih i neparametarskih testova. Isti skup podataka nije uobičajeno podvrgnuti i parametarskim i neparametarskim testovima, već samo jednoj skupini. Parametarski testovi imaju parametre, odnosno preduvjete koje skup podataka mora ispuniti kako bi statistički test dao validan rezultat. Neparametarski testovi alternative su parametarskim testovima kada skup (ili skupovi) podataka nad kojima se test izvršava ne zadovoljavaju preduvjete parametarskog testa. Dok parametarski testovi uobičajeno promatraju prosječne srednje vrijednosti skupova podataka, neparametarski statistički testovi orijentiraju se prema usporedbi medijana i rangova zbog čega su otporniji na ekstremne vrijednosti unutar skupova te ne zahtijevaju određenu distribuciju podataka.

2.2.1 Korišteni statistički testovi

Kako bi se odredili prikladni statistički testovi za korištenje, potrebno je poznavati određene informacije o podacima nad kojima se testovi izvršavaju. Ideja inferencijalne statističke analize u ovom radu bila je usporediti količinu prometa između prometnica te ustanoviti na kojoj se prometnici promet može smatrati gušćim. Kao prometne sudionike uzeti su pješaci, automobili, autobusi, kamioni, motocikli, bicikli i vlakovi, odnosno postoji sedam različitih grupa podataka. Grupe se uspoređuju zasebno, odnosno ne uspoređuju se podaci o pješacima jedne prometnice s podacima o automobilima ili vlakovima drugih prometnica. Korišteni test za usporedbu dviju prometnica, odnosno dvije skupine podataka prikladan je parametarski upareni t-test te njegova neparametarska alternativa Wilcoxon Signed Rank test. Testovi prikladni za usporedbu dvije skupine podataka objašnjeni su u potpoglavlju 2.2.2. Za usporedbu više od dvije skupine podataka prikladan parametarski test je ANOVA (eng. *analysis of vairance*) test koji uspoređuje prosječne srednje vrijednosti triju ili više nezavisnih skupina podataka. Kako je napomenuto ANOVA je parametarski test koji zahtijeva da podaci nad kojima se izvršava ispunjavaju sljedeće preduvjete: normalna distribucija podataka, homogeničnost podataka i sferičnost podataka. Ukoliko bilo koji od preduvjeta nije zadovoljen, ANOVA neće dati validne rezultate. Kako bi se provjerili preduvjeti, koriste se testovi iz Python knjižice SciPy te vanjske knjižice za Android Apache Commons Math. Za provjeru normalnosti distribucije često se koristi vizualna provjera pomoću histograma koji prikazuje distribuciju podataka. Stupići histograma u slučaju približne normalne distribucije su viši u sredini te se snižavaju prema rubovima (odnosno visina stupića prati krivulju normalne distribucije). Postoje i statistički testovi za provjeru normalne distribucije poput Kolmogorov-Smirnov testa i Shapiro-Wilk testa. U ovom slučaju korišten je Shapiro-Wilk test iz SciPy knjižice čija nulta hipoteza pretpostavlja normalnu distribuciju podataka te se ona odbacuje ukoliko je p-vrijednost testa manja 0.05. Sferičnost se provjerava Mauchly testom, koji testira hipotezu da su varijance razlika između uvjeta jednake.[14] Nulta hipoteza jest da su podaci sferični te se ta hipoteza odbacuje ukoliko je dobivena p-vrijednost testa manja od 0.05. Homogenost podataka provjerava se Levene testom koji pretpostavlja homogenost varijanci među grupama. Pretpostavka se odbacuje ukoliko je dobivena p-vrijednost testa manja od 0.05. Mauchly test i Levene

test korišteni su uz pomoć Apache Commons Math knjižice za Android. Ako su uvjeti zadovoljeni, provodi se ANOVA test. ANOVA mjeri varijance unutar grupa podataka (unutargrupna varijanca) i među grupama podataka (međugrupna varijanca). Testna statistika ANOVA-e (F-vrijednost) računa se kao omjer međugrupne varijance i unutargrupne varijance. Ukoliko je testna statistika veća od kritične, koja ovisi o broju uspoređenih skupina podataka i o broju podataka unutar skupine (stupnjevi slobode), ili je dobivena p-vrijednost manja od 0.05, nulta hipoteza se odbacuje, odnosno postoji statistički značajna razlika među skupinama podataka. U slučaju da bilo koja od skupina podataka koje želimo usporediti ne zadovoljava preduvjete ANOVA-e, podaci se podvrgavaju neparametarskoj alternativi za ANOVA test zvanoj Friedman testom. Friedman test za razliku od ANOVA testa ne uspoređuje prosječne srednje vrijednosti, već rangove podataka unutar skupina. Friedman test upućuje na značajne (ili neznačajne) razlike u rangovima podataka među različitim skupovima. podaci se rangiraju unutar svake skupine te idu po principu od najmanje vrijednosti (najniži rang) do najveće vrijednosti (najviši rang). Testna statistika Friedman testa dobiva se jednadžbom:

$$\chi_F^2 = \frac{12}{n \cdot k \cdot (k+1)} \left(\sum_{j=1}^{k} R_j^2 \right) - 3n(k+1) \tag{2.3}$$

pri čemu je značenje vrijednosti sljedeće:

- n: broj subjekata (broj parova u promatranju)
- k: broj grupa,
- R_j: zbroj rangova za svaku grupu.

Dobivena vrijednost uspoređuje se s kritičnom vrijednosti iz hi-kvadrat distribucije na zadanoj razini značajnosti, a ukoliko je dobivena vrijednost veća od kritične, ili je p-vrijednost manja od 0.05, odbacuje se nulta hipoteza, odnosno postoji značajna razlika među rangovima skupina podataka. Ukoliko ANOVA ili Friedman test odbacuju svoje nulte hipoteze, daju na znanje da postoji statistički značajna razlika među uspoređenim skupinama podataka, no ne prilažu informacije o konkretnim razlikama zbog čega se zahtjeva daljnja analiza zvana post-HOC analiza koja utvrđuje konkretne odnose među skupovima podataka. Iako se post-HOC analiza izvodi na

svim kombinacijama uspoređenih skupina podataka, korištenje isključivo testova za usporedbu dviju grupa bez izvođenja testova poput ANOVA-e ili Friedman-a nije preporučljivo s obzirom da se s povećanjem broja izvođenih testova povećava i mogućnost pogreške tipa 1 (pogrešno odbacivanje nulte hipoteze).[14]

2.2.2 Post-HOC analiza

Kako je prethodno spomenuto, post-HOC analiza izvršava se samo ukoliko je statistički test nad više od dva skupa podataka utvrdio da postoji značajna razlika među skupovima nad kojima je test izvršen. U post-HOC analizi skupovi podataka se uspoređuju u odnosu jedan na jedan u svim kombinacijama kako bi se dobio konkretan odnos među svim skupovima podataka. Najčešće se koriste upareni i neupareni t-test kao parametarski statistički testovi te Wilcoxon-Signed Rank i Mann-Whitney U test kao neparametarske alternative. S obzirom da se u ovom radu radi o uparenim podacima, korišteni su upareni t-test i Wilcoxon-Signed Rank test. Ukoliko su oba skupa podataka normalno distribuirana, uspoređuju im se prosječne srednje vrijednosti pomoću uparenog t-testa. Upareni t-test ima nekoliko bitnih vrijednosti koje se promatraju: testna statistika (t-vrijednost), stupanj slobode i p-vrijednost. Stupanj slobode dobiva se oduzimanjem jedinice od broja uzoraka, odnosno podataka u oba skupa koja se uspoređuju, dok se t-vrijednost dobiva sljedećom jednadžbom:

$$t = \frac{\bar{d}}{SE} = \frac{\bar{d}}{\frac{s_d}{\sqrt{n}}} \tag{2.4}$$

u kojoj je sljedeće značenje vrijednosti:

- t: t-vrijednost,
- $\bar{d}$: srednja razlika među uparenim uzorcima,
- SE: standardna pogreška srednje razlike,
- s_d: standardna devijacija razlika među uparenim uzorcima,
- n: broj uparenih uzoraka.

Wilcoxon Signed Rank test koristi se kao neparametarska alternativa uparenog t-testa pri čemu uspoređuje srednje rangove uparenih uzoraka između dvije skupine

podataka. Kao i upareni t-test, rezultati Wilcoxon Signed Rank testa sastoje se od bitnih vrijednosti poput testne statistike (z-vrijednost) i p-vrijednosti. Wilcoxon Signed Rank test koristi se kada jedan ili oba skupa podataka nad kojima se izvršava statistički test nisu podložni normalnoj distribuciji.

$$T- = \sum(negativnirangovi) \tag{2.5}$$

$$T+ = \sum(pozitivnirangovi) \tag{2.6}$$

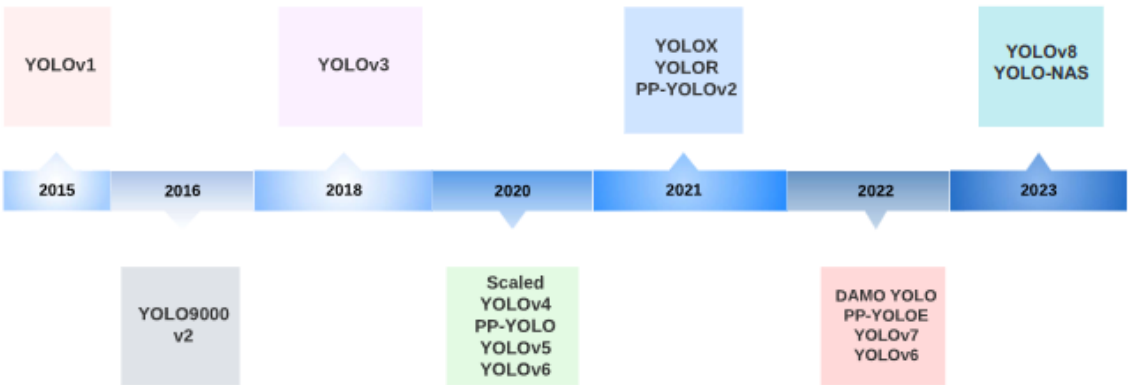
$$T = \sum_{i=1}^{n} R_i \tag{2.7}$$

- T: testna statistika Wilcoxon signed rank testa,
- R_i: rang apsolutnih vrijednosti razlika između parova uzoraka,
- W: suma rangova pozitivnih ili negativnih razlika (ovisno o testu),
- n: broj parova uzoraka.

Poglavlje 3

Modeli računalnog vida

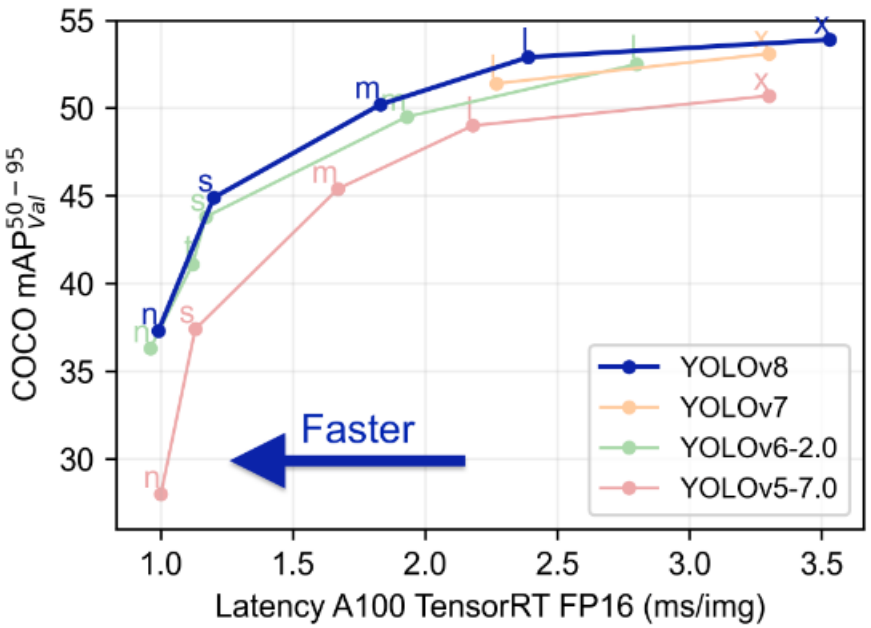
U novije doba, duboke neuronske mreže prevladavaju u razvoju računalnog vida. Neke od arhitektura koje su često korištene za prepoznavanje slika po sadržaju su ResNet, VGG16 i GoogleNet. I dok su spomenute arhitekture korištene u više različitih svrha, YOLO arhitektura je dizajnirana specifično za svrhu prepoznavanja objekata. YOLO arhitektura se prvi puta spominje 2015. godine kada je predstavljen model YOLO verzije 1 (YOLOv1).[2] YOLO arhitektura razvijana je od strane više autora, no najpoznatiji i najnoviji modeli razvijeni su od strane tvrtke Ultralytics čiji je rad orijentiran prema razvoju YOLO arhitekture za računalni vid.



Slika 3.1 Povijest izdanih YOLO modela od 2015. godine do 2023. godine.[2]

3.1 Različiti YOLO modeli za detekciju objekata

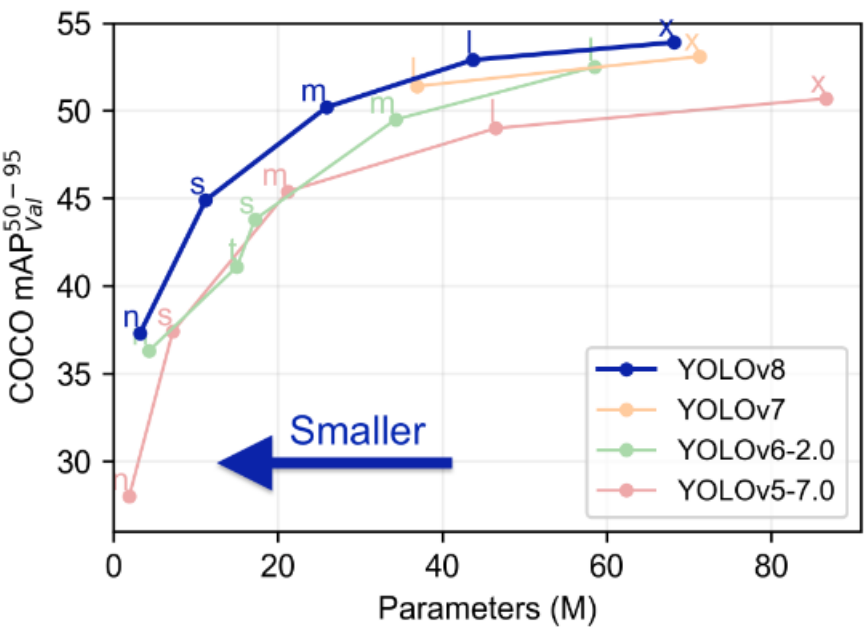
Pri odabiru modela računalnog vida koji će biti korišten za detekciju objekata na video snimkama, treba obratiti pažnju na više elemenata. Bitna je brzina detekcije, zadaci za koje je sam model namijenjen, performanse poput točnosti, latencije i broja parametara koje zahtjeva. Uz odabir verzije modela koja se koristi za detekciju objekata, potrebno je odabrati i modalitet samog modela koji je prigodan za tražene zadatke. Na slici 3.2 prikazan je dijagram usporedbe latencije YOLO modela verzije 5, 6, 7 i 8. Latencija je vrijeme između traženja podataka te odgovora samog modela koji pruža podatke. Što je to vrijeme kraće, znači da je model brži u prepoznavanju objekata, stoga se traži što kraće vrijeme.



Slika 3.2 Usporedba latentnosti različitih YOLO modela.[2] Na osi apscisa prikazana je latencija (vremensko kašnjenje od početka do završetka procesa) kada je analiziranje slike odrađeno pomoću NVIDIA A100 TensorRT grafičke procesorske jedinice sa 16-bitnim načinom preciznosti s pomičnim zarezom, dok os ordinata odnosi na srednju prosječnu preciznost korištenjem COCO protokola za procjenu na COCO skupu podataka. Svaka točka predstavlja drugačiji modalitet određenog YOLO modela.

Na dijagramu se može primjetiti da model YOLOv8 ima učestalo manju latenciju

u odnosu na druge modele zbog čega se može nazvati bržim od preostalih modela. Još jedna prednost YOLO modela verzije 8 nad drugim modelima je broj parametara koje model zahtjeva kako bi ispravno detektirao i klasificirao objekte. Manje potrebnih parametara znači manju potrošnju memorije te brže izračune interferencije zbog čega je sam model brži u detekciji objekata. Na dijagramu prikazanom slikom 3.3 prikazana je usporedba broja parametara između YOLO modela verzije 5, 6, 7 i 8. Vidljivo je da je broj parametara prosječno manji za model YOLOv8 s većom preciznošću postignutom na COCO bazi podataka.



Slika 3.3 Usporedba broja parametara različitih YOLO modela.[2] Os apscisa prikazuje broj parametara, dok os ordinata prikazuje srednju prosječnu preciznost korištenjem COCO protokola evaluacije na COCO skupu podataka. Svaka točka predstavlja drugačiji modalitet rada određenog YOLO modela.

Jedan od najbitnijih elemenata pri odabiru modela računalnog vida jest preciznost (točnost) modela. Na slikama 3.2 i 3.3 os ordinata prikazuje prosječnu preciznost modela korištenjem COCO protokola na COCO bazi podataka za treniranje i testiranje modela. Na obje slike je vidljivo da se graf koji prikazuje model YOLOv8 nalazi iznad ostalih grafova koji prikazuju druge YOLO modele čime se može zaključiti da model YOLOv8 ima veću prosječnu preciznost od drugih YOLO modela.

3.2 Prilagodba modela računalnog vida

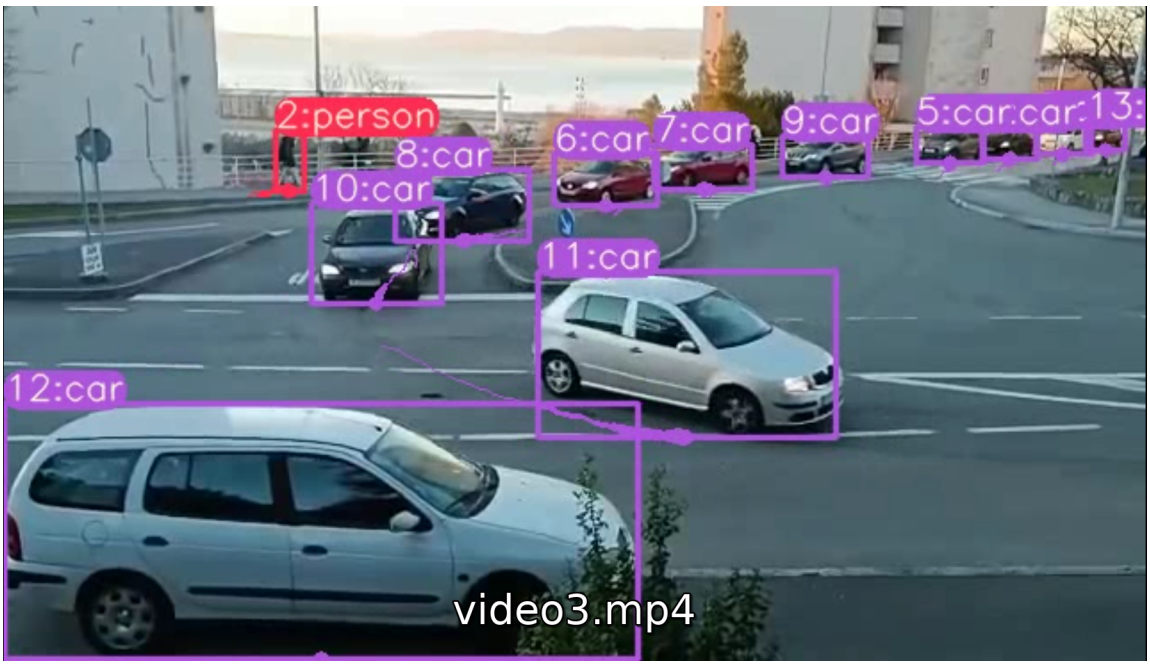
YOLO modeli se distribuiraju kao prethodno trenirani modeli koji su spremni za trenutnu upotrebu. YOLO modeli trenirani su na više baza podataka od kojih su najpoznatije COCO i ImageNet baze podataka koje su postale standard za treniranje i testiranje modela računalnog vida. YOLO model prepoznaje i klasificira razne objekte u desetke različitih klasa zbog čega je potrebno filtrirati prepoznavanje objekata na objekte koji mogu biti sudionici prometa. Objekti koji su klasificirani kao sudionici prometa su ljudi (pješaci), automobili, autobusi, kamioni, motocikli, bicikli i vlakovi. Svaki od spomenutih objekata nalazi se među objektima koje YOLOv8 prepoznaje i samostalno klasificira. Sam filtar dodan je u obradu nakon operacije detektiranja (eng. *post-processing*) te uklanja detektirane objekte koji ne pripadaju prometnim sudionicima iz liste detektiranih objekata, nakon čega se te detekcije više ne obrađuju, odnosno ne zapisuju te ne označavaju klasifikacijskim okvirom na rezultatnom videu. Filtar je prikazan pojednostavljenim pseudokodom 1.

Algorithm 1 Pojednostavljeni pseudokod filtra liste detektiranih objekata koji služi kako bi u listi ostali samo sudionici prometa

```
for each i, detektirani-objekt in enumerate(lista-detektiranih-objekata) do
  for each objekt in detektirani-objekt do
    klasa ← klasa detektiranog objekta
    if klasa ∉ {covjek, automobil, autobus, kamion, motocikl, bicikl, vlak}
    then
      if lista-detektiranih-objekata contains objekt then
        remove(lista-detektiranih-objekata, objekt)
      end if
    end if
  end for
end for
```

Nakon filtriranja liste detektiranih objekata, objekti poput biljaka, zgrada i prometnih znakova koji su učestali u snimkama prometnica neće utjecati na zapis i broj detektiranih objekata kako bi se nad dobivenim podacima mogla odraditi točnija analiza podataka. Na slici 3.4 prikazan je isječak iz rezultantne video snimke na-

kon upotrebe filtra. Vidljivo je da prometni znak i biljke na snimci nisu uokvireni klasifikacijskim okvirom, što znači da je filtar uspješno uklonio te detekcije iz liste.



Slika 3.4 Isječak iz video snimke s prikazon detektiranih objekata nakon upotrebe filtra za uklanjanje detekcija objekata koji nisu prometni sudionici.

3.3 Zapis podataka generiranih računalnim vidom

YOLOv8, uz mogućnost detekcije objekata, pruža i mogućnost praćenja objekata. U zapisivanju detekcija objekata, zapisano je i u kojem okviru videa su detektirani te njihov identifikacijski broj što omogućava praćenje konkretnog broja sudionika prometa u samoj video snimci te njihovo zadržavanje unutar video snimke. Kako bi se osigurala trajnost zapisa detekcija, one se zapisuju u zasebnu *csv* (eng. *comma-separated values*, hrv. *zarezom odvojene vrijednosti*) datoteku koja se kasnije koristi za analizu podataka. Ovime se osigurava da dotada obrađeni podaci ne nestanu ukoliko iz bilo kojeg razloga dođe do prekida izvršavanja programa. Datoteka se također stvara unutar programa te se, nakon statističke obrade i slanja rezultata na

udaljeni server, briše programski kako bi se oslobodila memorija za daljnu upotrebu. Primjer zapisa detekcija u datoteci prikazan je tablicom 3.1.

Frame	**ID**	**Object**
4	1	person
7	2	car
10	3	bus
13	4	truck
16	5	motorcycle
19	6	bicycle
22	7	train

Tablica 3.1 Primjer zapisa detekcija u tablici programski generirane *csv* datoteke.

Poglavlje 4

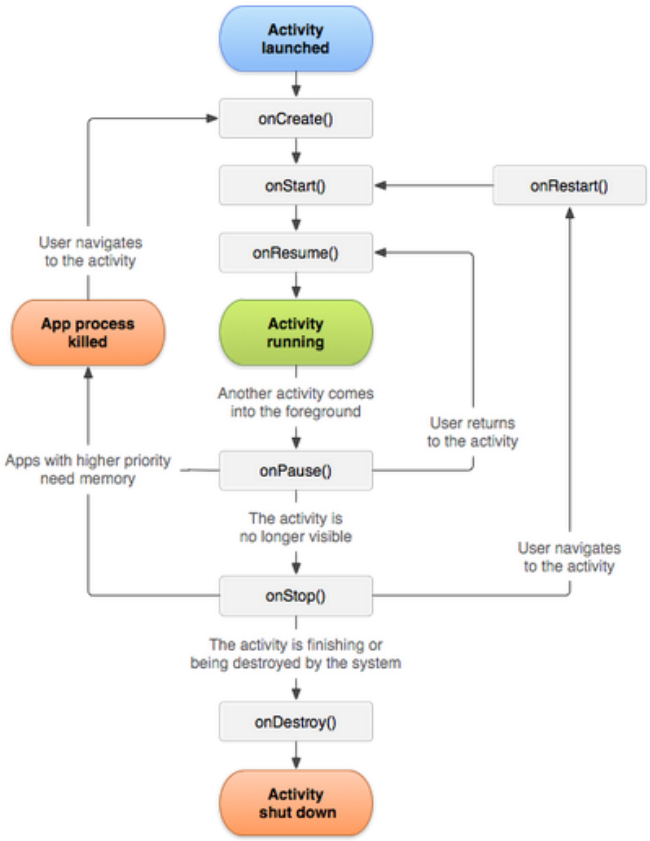
Klijentska aplikacija

Klijentska aplikacija je aplikacija za Android operativni sustav te ju je moguće koristiti na svim mobilnim i tablet Android uređajima s verzijom operativnog sustava 7.0 ili novijom. Aplikacija ima osnovne funkcionalnosti očitanja trenutne lokacije korisnika (uz korisničko dopuštenje), prikaza lokacije unesene adrese, prikaza jedne ili više ruta kojima je moguće doći od trenutne lokacije korisnika do željene lokacije te statističke usporedbe različitih ruta do lokacije pomoću deskriptivne i inferencijalne statistike. Koristi se podacima dobivenim pomoću poslužiteljskog dijela projekta, odnosno obradom video zapisa računalnim vidom, kako bi statistika bila predstavljena korisniku. Deskriptivna statistika očitana je s Firebase platforme za računarstvo u oblaku, dok se inferencijalna statistika odrađuje unutar same aplikacije. Omogućava korisniku da postavi stalne oznake na lokacije koje često posjećuje poput doma, posla, škole i slično, a također je moguće postaviti i jedinstvene podsjetnike unutar aplikacije koji služe za podsjetnik na polazak prema odredištu, a korisnik ih samostalno postavlja, uređuje i briše. Kako bi podaci bili što točniji i opširniji, korisnicima mogu sudjelovati u proširenju baze podataka na način da samostalno pošalju video snimke za obradu kroz aplikaciju, pri čemu se automatizirano šalju svi potrebni podaci za pravilno izvođenje statističke analize podataka te kategorizaciju spremanja podataka.

4.1 Android Studio okruženje za razvoj nativnih Android aplikacija

Android je u današnje vrijeme dominantan operativni sustav za mobilne i tablet uređaje s više od 70% tržišnog udjela u svijetu.[4] Android je razvijen od strane tvrtke Google te je objavljen 2007. godine. 2013. godine objavljeno je službeno razvojno okruženje za Andorid operativni sustav pod nazivom Android Studio. Android Studio razvijen je na temelju JetBrains-ovog IntelliJ IDEA programa te je prilagođen za razvoj nativnih Android aplikacija. Android Studio podržava razvoj u programskim jezicima Kotlin (primaran), Java i C++. Unutar Android Studio okruženja moguće je kreirati različite datoteke i resurse. Najvažnije datoteke koje je potrebno kreirati za svaku aplikaciju su aktivnosti i nacrti (eng. *layout*). Aktivnost je klasa koja sadrži sva ponašanja za određeni zaslon koji se prikazuje trenutno na uređaju. Ponašanja su organizirana unutar funkcija i klasa unutar aktivnosti. Aktivnosti imaju standardan životni ciklus koji se sastoji od sedam različitih standardnih funkcija koje se uvijek izvode istim redoslijedom i kao reakcija na iste okidače. Graf na slici 4.1 prikazuje pojednostavljen tijek životnog ciklusa aktivnosti.

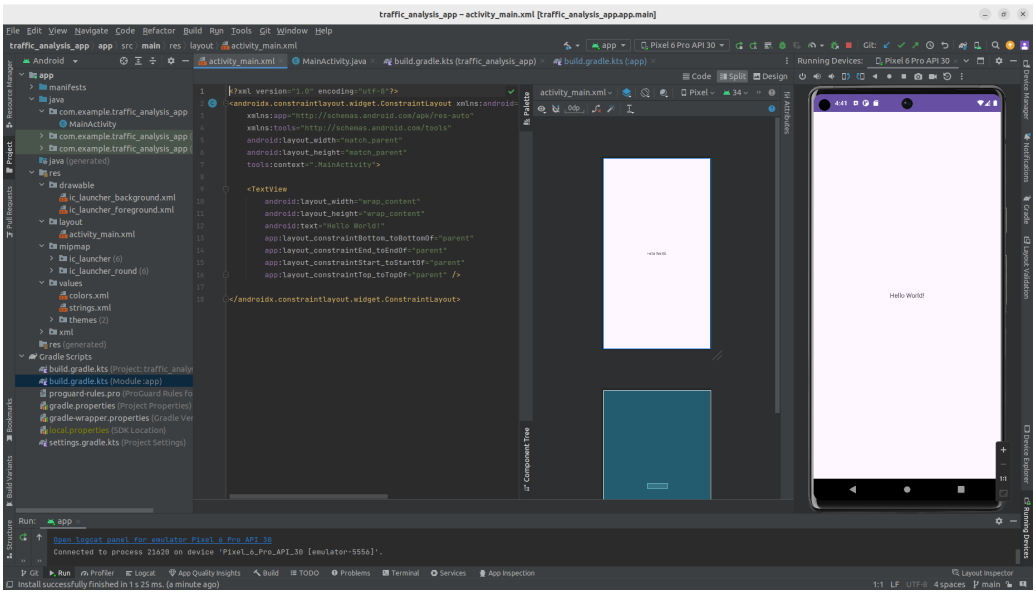
Nacrti su datoteke napisane *xml* (skraćeno od eng. *Extensible Markup Language*) označnim jezikom te sadrže raspored i definiciju svih elemenata koji se nalaze na zaslonu. Nacrti su glavni resurs koji prikazuje sve ostale resurse koji se nalaze unutar projekta. Nacrti i aktivnosti su međusobno zavisni s obzirom da nacrt sadrži definiciju i raspored svih elemenata na zaslonu, dok aktivnost sadrži ponašanje elemenata. Datoteka koje sadrže konfiguraciju cijelog projekta, odnosno aplikacije nazvana je manifestom. U manifestu se definiraju sve aktivnosti unutar aplikacije, sva dopuštenja koja korisnik mora dati kako bi aplikacija funkcionirala te osnovne karakteristike aplikacije poput ciljane Android verzije, početne aktivnosti i ikone aplikacije. Posljednje datoteke koje je neophodno spomenuti su *gradle* datoteke. *Gradle* je skup alata koji služi za izradu i automatizaciju procesa izrade aplikacije. Te datoteke sadrže konfiguraciju same aplikacije te sve njezine ovisnosti za implementaciju, a bez njihove sinkronizacije pri svakoj promjeni je nemoguće pokrenuti aplikaciju. Jedna od korisnih značajki samog Android Studio razvojnog okruženja jest emulator Android uređaja. Emulator je virtualni Android uređaj koji je prilagodljiv te



Slika 4.1 Pojednostavljeni prikaz životnog ciklusa aktivnosti unutar Android Studio razvojnog okruženja za nativne Android aplikacije.[5]

prikazuje postojeće Android uređaje. Emulator omogućuje testiranje i pokretanje same aplikacije unutar Android Studio okruženja bez potrebe za vanjskim Android uređajem.

Verzija Android Studio razvojnog okruženja korištena za razvoj klijentske aplikacije je Android Studio 2022.3.1 Giraffe iz srpnja 2023. godine, a programski jezik korišten za razvoj aplikacije je Java.



Slika 4.2 Snimka zaslona korisničkog sučelja Android Studio razvojnog okruženja. Na lijevoj strani sučelja nalazi se struktura projekta sa svim datotekama samoga projekta. Središnji dio je uređivač koda koji je u ovom slučaju razdvojen na programski i vizualni dizajnerski dio *xml* datoteke. Na desnoj strani emulatorom je pokrenuta aplikacija koja se trenutno razvija.

4.2 Firebase platforma za računarstvo u oblaku

Računarstvo u oblaku postaje sve veći dio razvoja u svim aspektima računarstva. Korištenje udaljenih poslužitelja koji se ne moraju lokalno održavati, uvijek su dostupni te imaju veće mogućnosti (poput različitih usluga i veće memorije) od lokalnih poslužitelja uvelike olakšava razvoj programskih proizvoda. Iako ima više pružatelja usluga računarstva u oblaku, najpoznatiji i najveći pružatelji su Amazon, platforma Amazon Web Services (skraćeno AWS), Microsoft, platforma Microsoft Azure, te Google, platforma Google Cloud. Google Cloud najčešće je korištena platforma u edukacijske svrhe s obzirom na pristupačnost, prepoznatljivost i raširenost Google usluga te samu jednostavnost korištenja. Google Cloud sastoji se od mnogih dijelova te nudi razne usluge poput platforme kao usluge, servisa kao usluge te infrastrukture kao usluge. Jedan od najpopularnijih dijelova Google Cloud platforme je Firebase. Firebase je platforma koja je prilgođena za razvoj mobilnih (Android i iOS) i web

aplikacija te pruža različite usluge i servise poput analitike, autentifikacije korisnika te slanja poruka i obavijesti unutar aplikacije. Dvije usluge Firebase platforme korištene u ovom radu su Firebase Realtime Database, odnosno baza podataka u stvarnom vremenu bazirana na JSON formatu koja pruža pohranu tesktualnih i numeričkih podataka pod različitim ključevima u stvarnom vremenu, te Firebase Storage, koja je baza podatka velike memorije za pohranu svih vrsta podataka te datoteka i direktorija. Unutar Firebase platforme stvara se Firebase projekt koji se potom povezuje s razvijanom mobilnom i/ili web aplikacijom putem para ključeva kako bi samo autorizirani korisnik (odnosno programer aplikacije) mogao upravljati komunikacijom između aplikacije i Firebase platforme. Firebase platforma je besplatna za razvoj aplikacija malih razmjera, odnosno korištenje platforme se počinje naplaćivati tek nakon određene količine korištenja same platforme. Zbog mogućnosti povezivanja Firebase projekta s više različitih aplikacija, u ovome se projektu koristi kao poveznica između klijentske strane (mobilna Android aplikacija) i poslužiteljske strane (Python program).

4.3 OpenStreetMap baza podataka

OpenStreetMap je besplatna, legalno dostupna baza geolokacijskih podataka otvorenog koda. OpenStreetMap (skraćeno *OSM*) nije u vlasništvu jednog entiteta već se oslanja na zajednicu korisnika koja ju održava razvojem, ažuriranjem, novim zapisima te potvrdom točnosti zapisanih informacija. Posebna značajka OSM baze podataka je raširenost zajednice u raznim područjima s velikim brojem podružnica među kojima je i OSM Hrvatska. To znači ažurne informacije i preciznu pokrivenost područja u kojima su zajednice aktivne. OSM baza je nastala kao odgovor na težak i dugotrajan proces dobivanja licenci za korištenje podataka iz britanske nacionalne baze geografskih podataka, a popularnost joj raste od 2012. godine kada je Google počeo naplaćivati korištenje najpoznatije baze geografskih podataka Google Maps. OSM baza podataka se najčešće koristi za izradu elektroničkih (digitalnih) mapa unutar različitih web i mobilnih aplikacija. Integracija OSM baze podataka je vrlo jednostavna te je od zaklade OpenStreetMap traženo samo da se naznači i kreditira OpenStreetMap baza podataka i njeni doprinositelji za korištene informacije.

Poglavlje 5

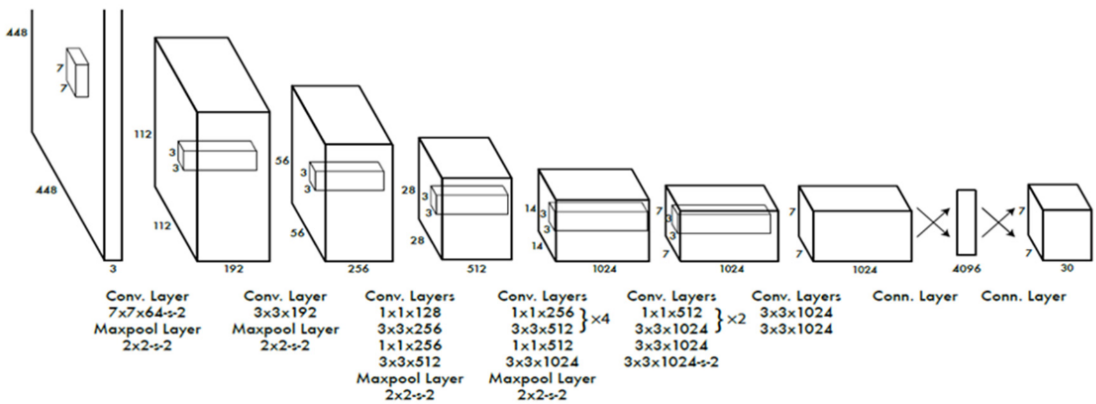
Implementacija

Implementacija rada izvršena je u dva odvojena dijela: implementacija poslužiteljske strane te implementacija klijentske aplikacije, svaki sa zasebnim programskim jezikom. Poslužiteljska strana implementirana je Python programskim jezikom, dok se klijentska strana sastoji od nativne mobilne Android aplikacije. Povezivanje poslužiteljske strane i klijentske aplikacije može se smatrati trećim zasebnim dijelom te je izvršeno pomoću tehnologije računarstva u oblaku.

5.1 Implementacija poslužiteljske strane

Zadatak poslužiteljske strane jest obraditi video snimke poslane klijentskom aplikacijom preuzete s pohrane u oblaku pomoću računalnog vida te dobivene rezultate potom statistički obraditi. Video snimke se preuzimaju pomoću Python skripte automatski pokrenute *bash* skriptom u Linux Ubuntu sustavu. Automatizacija pokretanja *bash* skripte odrađena je pomoću 'cron' servisa za periodično pokretanje skripti u Linux sustavu. Kako bi se zakazalo pokretanje skripte, potrebno je u terminalu stvoriti novi zadatak, odnosno *cronjob*, koji specificira po kojem rasporedu se pokreće skripta definirana svojim putem (potreban je apsolutni put do skripte). Python skripta pokrenuta ovom *bash* skriptom ima isključivo funkcionalnost pristupa Firebase platformi i preuzimanju video zapisa dostupnih na njoj (objašnjeno u 5.3), nakon čega pokreće glavnu Python skriptu za obradu video snimaka prilagođenim

modelom YOLOv8 za detekciju i klasifikaciju objekata računalnim vidom. Obrada računalnim vidom sastoji se od preuzimanja gotovog prethodno treniranog YOLO modela. Preuzimanje takvog modela sa sobom povlači kompliciranu arhitekturu direktorija i datoteka s različitim namjenama i ulogama. Veliki dio datoteka se sastoji od Python skripti za pokretanje različitih *YAML* i *cfg* konfiguracijskih datoteka, no također postoje i razne baze podataka te sami modaliteti modela u *pt* (skraćeno od *Place-Text*) datotekama. Konfiguracija direktorija i datoteka povezana je sa samom arhitekturom YOLO modela za računalni vid koja se sastoji od posebno razvijene mreže slojeva konvolucijskih neuronskih mreža s namjerom korištenja isključivo za zadatke računalnog vida. Arhitektura modela prikazana je slikom 5.1.



Slika 5.1 Arhitektura YOLO strukture neuronskih mreža.[9]

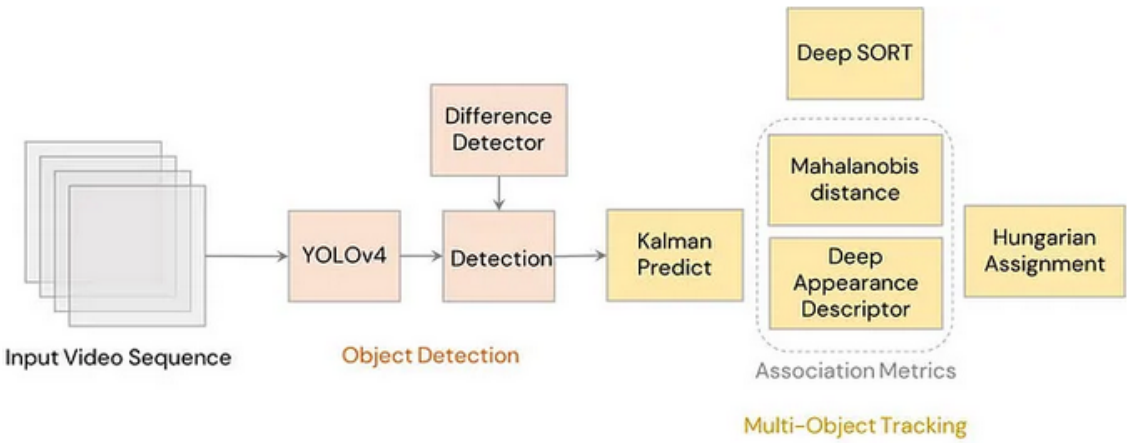
Iako je YOLO arhitektura vrlo složena, unutar jedne Python skripte pokreću se svi podprocesi potrebni za obradu video snimaka računalnim vidom. Ta skripta, osim što se sastoji od više različitih funkcija koje su potrebne u obradi vide snimaka, pokreće i procese unutar drugih Python skripti, ali i preuzima potrebne podatke iz drugih već spomenutih datoteka. Kako bi se pojednostavila konfiguracija tako složene arhitekture, korišten je programski okvir Hydra koji stvara mogućnost hijerarhijske konfiguracije sastavljene iz više različitih izvora. Prednost Hydra alata jest mogućnost nadjačavanja postojeće konfiguracije s konfiguracijom prilagođenom potrebama trenutnog korisnika. Primjer kako je Hydra korištena unutar ovoga projekta kako bi se pokrenula detekcija i praćenje objekata prikazana je na sljedeći način:

Listing 5.1 Pokretanje detekcije i praćenja objekata pomoću Hydra konfiguracije

```
@hydra.main(config_path=str(DEFAULT_CONFIG.parent),
    config_name=DEFAULT_CONFIG.name, version_base=None)
def predict(cfg):
    init_tracker()
    cfg.model = cfg.model or "yolov8n.pt"
    cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)
    # cfg.source is the directory containing the video files
    source_directory = hydra.utils.to_absolute_path(cfg.source)
    print(source_directory)
    for filename in os.scandir(source_directory):
        print(filename)
        if filename.is_file():
            predictor = DetectionPredictor(cfg)
            predictor(filename.path)
            global_instance.filename = filename.name
            info = filename.name.split('_')
```

Cijela YOLO arhitektura bazirana je na specijalnoj knjižici za rad s računalnim vidom pod nazivom OpenCV. OpenCV najpoznatija je, najopširnija i najkorištenija knjižica za rad s računalnim vidom te je dostupna za više različitih platformi, no najkorištenija upravo u Python programskom jeziku. YOLO se služi s nekoliko tisuća njezinih algoritama kako bi stvorio vlastite algoritme i metode detekcije i praćenja objekata. U prethodnom kodu može se primjetiti da je prvi pokrenuti proces funkcija *init_tracker()*. Unutar te funkcije inicijalizira se *Deep SORT* (skraćeno od eng. *Simple Online and Realtime Tracking with Deep Association Metric*) algoritam za praćenje objekata. Ulazni podaci za *Deep SORT* algoritam su video zapisi koji su obrađeni algoritmom za detekciju objekata (u ovom slučaju YOLOv8 baziran na OpenCV knjižici). Ovaj algoritam služi isključivo za praćenje već detektiranih objekata na snimci. Konfiguracija *Deep SORT* algoritma prikazana je na slici 5.2.

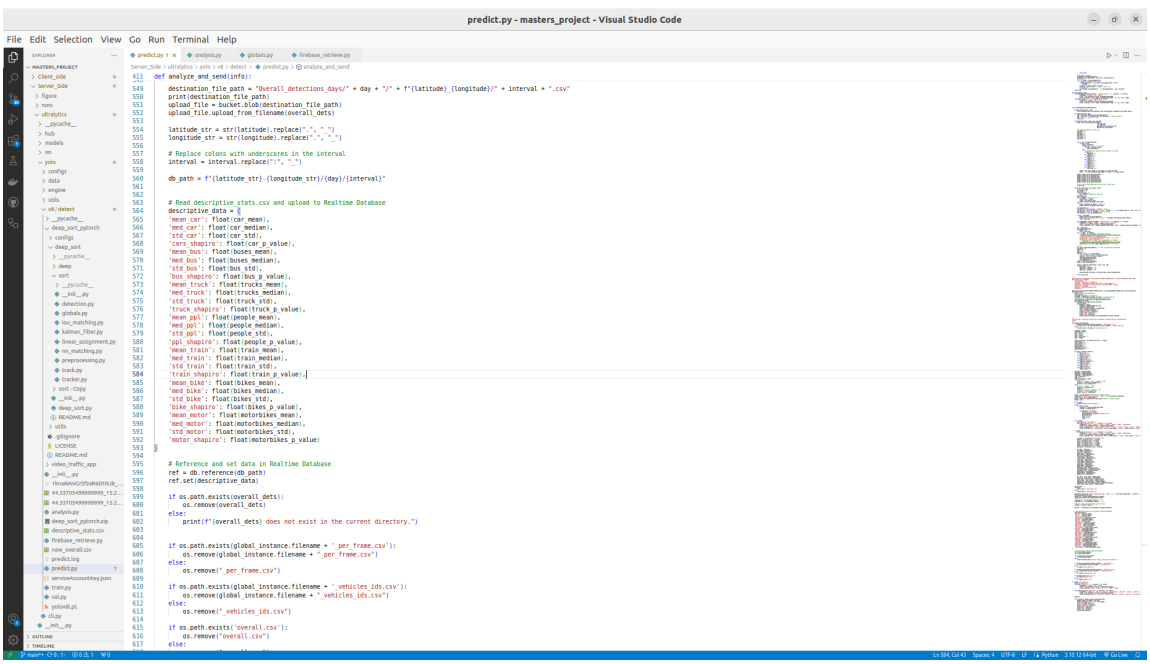
Unutar funkcije za inicijalizaciju algoritma *init_tracker()* inicijaliziraju se i *csv* datoteke koje se koriste za zapis detektiranih objekata te njihovo praćenje. Unutar prikazanog koda bitno je i istaknuti i poziv prilagođenoj klasi DetectorPredictor



Slika 5.2 Prikaz ulaza i konfiguracije *Deep SORT* algoritma.[21]

za svaki vremenski okvir trenutno obrađivanog video zapisa. Klasa DetectorPredictor nasljeđuje klasu BasePredictor iz YOLO modela te ju proširuje za uključivanje *Deep SORT* algoritma. Ova klasa služi za zadatak detekcije i klasifikacije objekata s video zapisa uz definiciju funkcija za pretprocesiranje i naknadnu obradu (eng. *post-processing*). Unutar funkcije za naknadnu obradu uključen je filtar za obradu isključivo prometnih sudionika spomenut u 3.2 i prikazan pseudokodom 1. U ovoj klasi također se zapisuju informacije o detektiranim objektima u *csv* datoteke potrebne za kasniju statističku obradu. Osim spomenutih funkcija i klasa čije funkcionalnosti su detekcija i praćenje objekata te zapis rezultata istih za kasniju statističku obradu, postoji cijeli niz funkcija za obradu vizualnog izlaza programa. Vizualni izlaz programa jest video snimka s iscrtanim identifikacijskim kutijama oko detektiranih objekata kako je već i prikazano na 3.4. Funkcije vizualizaciju sastoje se od funkcija za definiciju koordinata rubova detekcijske kutije u svakom vremenskom okviru video snimke, funkcije za određivanje boje detekcijske kutije ovisno o klasi detektiranog objekta, funkcije za ispis identifikacijske oznake svakog detektiranog objekta te funkcija za iscrtavanje samih identifikacijskih kutija i repa praćnog objekta. Za razvoj samog poslužiteljskog dijela projekta, odnosno uređivanje koda korišten je Microsoft Visual Studio Code te njegova ekstenzija za pokretanje Python koda. Python skripte također se mogu otvoriti i pokrenuti unutar svakog uređivača teksta s podrškom za pokretanje Python koda. Unutar Microsoft Visual Studio Code

programa s lijeve strane vidljiva je i arhitektura direktorija i datoteka potrebnih za razvoj projekta. Početak svake Python skripte sastoji se od uvoza svih potrebnih modula i knjižica te definiranja puteva do drugih potrebnih skripti kako bi se kod mogao nesmetano pokrenuti. U Microsoft Visual Studio Code programu također je napisana i *bash* skripta koja služi za automatizirano pokretanje analize video snimaka. *Bash* se odnosi na Bourne Again SHell, odnosno na jezik koji razumije Unix ljuska koja je pokrenuta u terminalu (komandnoj liniji) Linux Ubuntu 22.04 operacijskog sustava unutar kojega je projekt pokretan. *Bash* skripta pokrenuta je unutar samog terminala automatiziranim putem ugrađenog planera, kako je već spomenuto, te pokreće Python skriptu pod nazivom *firebase_retrieve* koja potom nakon odrađenog zadataka preuzimanja video snimaka iz pohrane na oblaku pokreće daljnje skripte za obradu video snimaka. Na slici 5.3 prikazan je kod pokretačke Python skripte poslužiteljskog dijela projekta otvoren u Microsoft Visual Studio Code programu.



Slika 5.3 Korisničko sučelje Microsoft Visual Studio programa. U lijevom odijeljku vidljiva je struktura direktorija i datoteka u trenutno otvorenom projektu. U glavnom dijelu ekrana prikazan je programski kod koji se trenutno uređuje. U slučaju pokretanja koda unutar programa, prikazan je terminal u donjem dijelu ekrana u zasebnom prozoru.

Nakon obrade računalnim vidom i zapisom detekcija objekata kako je objašnjeno u 3.3, izvršava se statistička analiza podataka u Python programskom jeziku. Deskriptivna statistika izvršava se automatski nakon obrade računalnim vidom te se bazira na rezultatima detekcije prometnih sudionika u svakoj zasebnoj video snimci. Pri statističkoj obradi deskriptivnom statistikom u obzir se uzima broj detekcija različitih objekata kroz video snimku te se ažuriraju već postojeći podaci (ukoliko ih ima na oblaku) za određenu lokaciju u određenom vremenskom intervalu određenog dana. Podršku za deskriptivnu statistiku omogućava NumPy knjižica. Unutar NumPy knjižice postoje sljedeće automatizirane funkcije za izračun mjera centralne tendencije:

- *np.mean* - funkcija za izračun srednje prosječne vrijednosti skupine podataka,
- *np.median* - funkcija za izračun medijana skupine podataka,
- *np.std* - funkcija za izračun standardne devijacije skupine podataka,
- *np.mode* - funkcija za izračun moda skupine podataka.

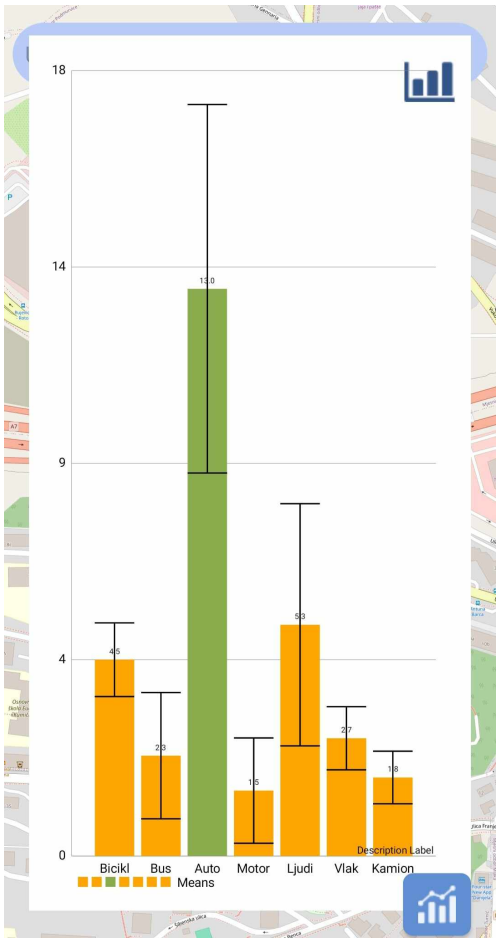
Kako bi se mogle koristiti spomenute funkcije, skupina podataka mora biti unutar NumPy polja, u ovom slučaju jednodimenzionalno polje s Integer tipom numeričkih podataka. S obzirom da broj očitanih detekcija može biti samo cijeli broj (nula ili veći), Integer je primjeren tip numeričkog podatka za korištenje. Svaka lokacija ima podatke podijeljene po danima u tjednu te po vremenskim intervalima, npr. ponedjeljak od 8:00 do 8:30 ili subota od 14:30 do 15:00. Na taj se način uzimaju samo podaci koji su važni za korisnika u trenutku potrebe, te se ne miješaju s podacima koji se odnose na različito vremensko razdoblje od trenutka potrebe. Osim deskriptivne statistike, u poslužiteljskom dijelu radi se provjera normalnosti distribucije svake grupe podataka Shapiro-Wilk testom. Shapiro-Wilk test nalazi se unutar SciPy knjižice. Nakon ažuriranja postojećih podataka deskriptivne statistike i provjere normalnosti distribucije, podaci se šalju na oblak kako bi u svakom trenutku korisnik imao pristup najnovijim podacima. S obzirom na potencijalnu usporedbu dvaju ili više puteva, ostatak inferencijalne statističke analize odvija se unutar klijentske aplikacije, a bazirana je na podacima koji su uneseni na prethodno objašnjen način.

5.2 Implementacija klijentske strane

Klijentska aplikacija izvedena u Android operativnom sustavu pomoću programskog jezika Java ima funkcionalnost korištenja prikupljenih podataka o statističkoj analizi prometa pomoću usporedbe različitih puteva od početne lokacije do završne lokacije ovisno o količini prometa koja je prethodno zabilježena na određenim prometnicama. Kako je za statističku analizu potrebno imati bazu podataka, u ovom slučaju video snimaka s različitih prometnica u različito vrijeme, mobilna aplikacija pruža mogućnost korisniku da pridonese stvaranju dovoljno velikog broja video snimaka kako bi statistička analiza mogla što vjerodostojnije prikazati karakteristike prometa na određenoj lokaciji kroz različite dane i vremenske intervale u danu. Glavna funkcionalnost aplikacije je prikaz različitih puteva od početne (trenutne) lokacije korisnika do ciljane lokacije uz usporedbu podataka o prometu na različitim prometnicama. Ciljana lokacija dobiva se upisom adrese lokacije u tražilicu unutar aplikacije ili duljim držanjem lokacije na virtualnoj mapi na kojoj se prikazuje i trenutna lokacija, kao i putevi od početne do ciljane lokacije. Pritiskom na oznaku ciljane lokacije (takozvani marker) otvara se skočni prozor s mogućnošću odabira da se lokacija spremi u memoriju Android uređaja pod vlastitom oznakom ili da se pokažu upute do lokacije. Ukoliko se odabere opcija spremanja lokacije, odabrana lokacija sprema se u SharedPreferences objekt s oznakom geografske širine i dužine te s upisanim nazivom lokacije od strane korisnika. Pri svakom sljedećem učitavanju OpenStreetMap karte na Android uređaju bit će prikazana oznaka spremljene lokacije. U slučaju odabira opcije prikaza uputa do lokacije, pokreće se funkcija koja prikazuje jedan ili više potencijalnih puteva do željene lokacije. Svaki od puteva prikazan je iscrtanom linijom u boji te je moguće usporediti određene dijelove puteva pomoću podataka statističke analize. Rute između dvije lokacije dobivaju se pomoću poziva programskom sučelju aplikacije (eng. *application programming interface*, skraćeno *API*) OpenStreetMap koje vraća JSON (skraćeno od eng. *JavaScript Object Notation*) objekt s geotočkama ruta između dvije lokacije. API poziv obrađen je pomoću knjižnice Retrofit koja služi za pojednostavljenje HTTP (skraćeno od eng. *Hypertext Transfer Protocol*) poziva unutar Android aplikacije. Pritiskom na dugme za usporedbu prometnica, može se odabrati dio puteva čiji podaci se žele usporediti. Moguće je izabrati jednu ili više prometnica, pri čemu je za jednu prometnicu moguće isključivo

pregledati deskriptivnu statistiku prometnice, dok je za više odabranih prometnica moguće uvesti statističke testove za usporedbu. Podaci su, kao što je spomenuto u 5.1, podijeljeni po danima u tjednu, lokacijama te vremenskim intervalima od trideset minuta kroz dan. Stoga se uspoređuju podaci koji se poklapaju po danu i vremenskom intervalu koji je određen vremenom u koje se pokušava pristupiti podacima. podaci se prikazuju grafički iscrtavanjem stupičastih dijagrama s intervalima povjerenja koji prikazuju srednje prosječne vrijednosti skupina podataka (visina stupčastog grafa) i standardnu devijaciju skupine podataka (intervali povjerenja) kako bi korisnik dobio vizualnu usporedbu količine prometa. Za crtanje dijagrama korištena je knjižica MPAndroidChart koja podržava iscrtavanje linijski grafova, stupičastog dijagrama, tortnog grafikona, raspršeng grafikona, radarskog grafikona (paukova mreža) te mjehurastog grafikona.[20] Iako ima podršku za iscrtavanje stupičastih dijagrama, knjižica nema podršku za iscrtavanje intervala povjerenja čije crtanje je moralo biti ručno isprogramirano kako bi se podaci pravilno prikazali. Kako bi se prikazali intervali povjerenja, potrebno je stvoriti novu Java klasu koja nasljeđuje postojeću klasu stupičastog dijagrama iz MPAndroidChart knjižice te se nadograđuje s metodom za izračunavanje pozicija i iscrtavanjem intervala povjerenja. Primjer iscrtanog grafa prikazan je na slici 5.4.

Za usporedbu podataka o više prometnica služe statistički testovi koji ovise o podacima, odnosno o broju skupina podataka koji se uspoređuje, karakteristikama podataka i udovoljavanju parametara statističkih testova. Testovi među kojima se bira su upareni t-test ili Wilcoxon Signed Rank test u slučaju usporedbe dvije skupine podataka, odnosno te ANOVA ili Friedman test u slučaju više od dvije skupine, a svaki od statističkih testova te njihove primjene objašnjeni su u 2.2.1. Inferencijalna statistika testovima odrađena je unutar same aplikacije pomoću knjižice Apache Commons Math. Apache Commons Math knjižica omogućava automatiziranu izvedbu parametarskih testova poput ANOVA-e i uparenog t-testa, dok su rezultati njihovih neparametarskih alternativa Friedman testa i Wilcoxon Signed Rank testa ručno izračunati. Ručni izračun baziran je na jednadžbama zapisanim u 2.2.1 i 2.2.2. Kako bi se provjerili preduvjeti za korištenje parametarskog testa ANOVA, potrebno je provjeriti preduvjete sferičnosti i homogenosti podataka pomoću Mauchly i Levene testa. Spomenuti testovi objašnjeni su u 2.2.1, a također nisu podržani unutar



Slika 5.4 Snimka zaslona s prikazanim stupičastim dijagramom i intervalima povjerenja koji prikazuju rezultate deskriptivne statističke analize odabrane prometnice.

Apache Commons Math knjižice zbog čega su također ručno izračunati. Jednadžbe potrebne za izračun rezultata Mauchly testa:

- izračun srednje vrijednosti za svaku mjeru:

$$\mu_j = \frac{1}{n}\sum_{i=1}^{n} X_{ij} \tag{5.1}$$

- izračun kovarijacijske matrice C:

$$C_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} (X_{ki} - \mu_i)(X_{kj} - \mu_j) \tag{5.2}$$

- izračun determinante det(C) za 2x2 matricu:

$$\det(\mathbf{C}) = C_{11} \cdot C_{22} - C_{12} \cdot C_{21} \tag{5.3}$$

- izračun traga kovarijacijske matrice:

$$\mathrm{tr}(\mathbf{C}) = \sum_{i=1}^{p} C_{ii} \tag{5.4}$$

- izračun Mauchlyjeve testne statistike W:

$$W = \left(\frac{\det(\mathbf{C})}{\left(\frac{\mathrm{tr}(\mathbf{C})}{p} \right)^p} \right)^{\frac{n}{2}} \tag{5.5}$$

- izračun hi-kvadrat vrijednosti:

$$\chi^2 = -(n-1) \left(p + 1 - \frac{2p}{3} \right) \log(W) \tag{5.6}$$

- izračun stupnjeva slobode:

$$\mathrm{df} = \frac{p(p-1)}{2} \tag{5.7}$$

- izračun p-vrijednosti Mauchlyjevog testa:

$$p\text{-value} = 1 - F_{\chi^2}(\chi^2, \mathrm{df}) \tag{5.8}$$

Pri čemu su:

- n: broj uzoraka,
- p: broj mjera,
- X: matrica podataka dimenzija n×pn×p, gdje svaki redak odgovara uzorku, a svaki stupac odgovara ponovljenoj mjeri.

Jednadžbe potrebne za izračun rezultata Levene testa:

- izračun srednje vrijednosti unutar svake grupe:

$$\bar{X}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} X_{ij} \tag{5.9}$$

- izračun apsolutne devijacije unutar svake grupe:

$$Z_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left| X_{ij} - \bar{X}_i \right| \tag{5.10}$$

- izračun sume kvadrata devijacija SS_Z i sume kvadrata grupnih sredina SS_{group}

$$\mathrm{SS}_Z = \sum_{i=1}^{k} Z_i^2 \tag{5.11}$$

$$\mathrm{SS}_{\mathrm{group}} = \sum_{i=1}^{k} n_i Z_i^2 \tag{5.12}$$

- izračun globalne srednje vrijednosti:

$$\bar{Z} = \frac{1}{k} \sum_{i=1}^{k} Z_i \tag{5.13}$$

- izračun brojnika za F-test:

$$\mathrm{Numerator} = \frac{(N-k) \cdot \mathrm{SS}_{\mathrm{group}} - \mathrm{SS}_Z}{k-1} \tag{5.14}$$

- izračun nazivnika za F-test:

$$\mathrm{Denominator} = \frac{\mathrm{SS}_Z - N \cdot \bar{Z}^2}{N-k} \tag{5.15}$$

- izračun vrijednosti testne statistike (F-test):

$$F = \frac{\mathrm{Numerator}}{\mathrm{Denominator}} \tag{5.16}$$

- izračun p-vrijednosti:

$$p\text{-value} = 1 - F_{\mathrm{dist}}(F, k-1, N-k) \tag{5.17}$$

Pri čemu su:

- N: ukupan broj opažanja,
- k: broj skupina podataka.

Uobičajeni prikaz rezultata statističkih testova je pomoću tablice poput 5.1 bez interpretacije, no unutar aplikacije prikazani su pomoću teksta zajedno s interpretacijom rezultata kako bi korisnik koji nije stručnjak mogao s razumjevanjem očitati rezultate testa.

	Levene Statistic	**df1**	**df2**	**Sig.**
Beck Depression Inventory (Sunday)				
Based on Mean	3.644	1	18	.072
Based on Median	1.880	1	18	.187
Based on Median and with adjusted df	1.880	1	10.076	.200
Based on trimmed mean	2.845	1	18	.109
Beck Depression Inventory (Wednesday)				
Based on Mean	.508	1	18	.485
Based on Median	.091	1	18	.766
Based on Median and with adjusted df	.091	1	11.888	.768
Based on trimmed mean	.275	1	18	.606

Tablica 5.1 Primjer rezultata testa homogenosti varijance.[14]

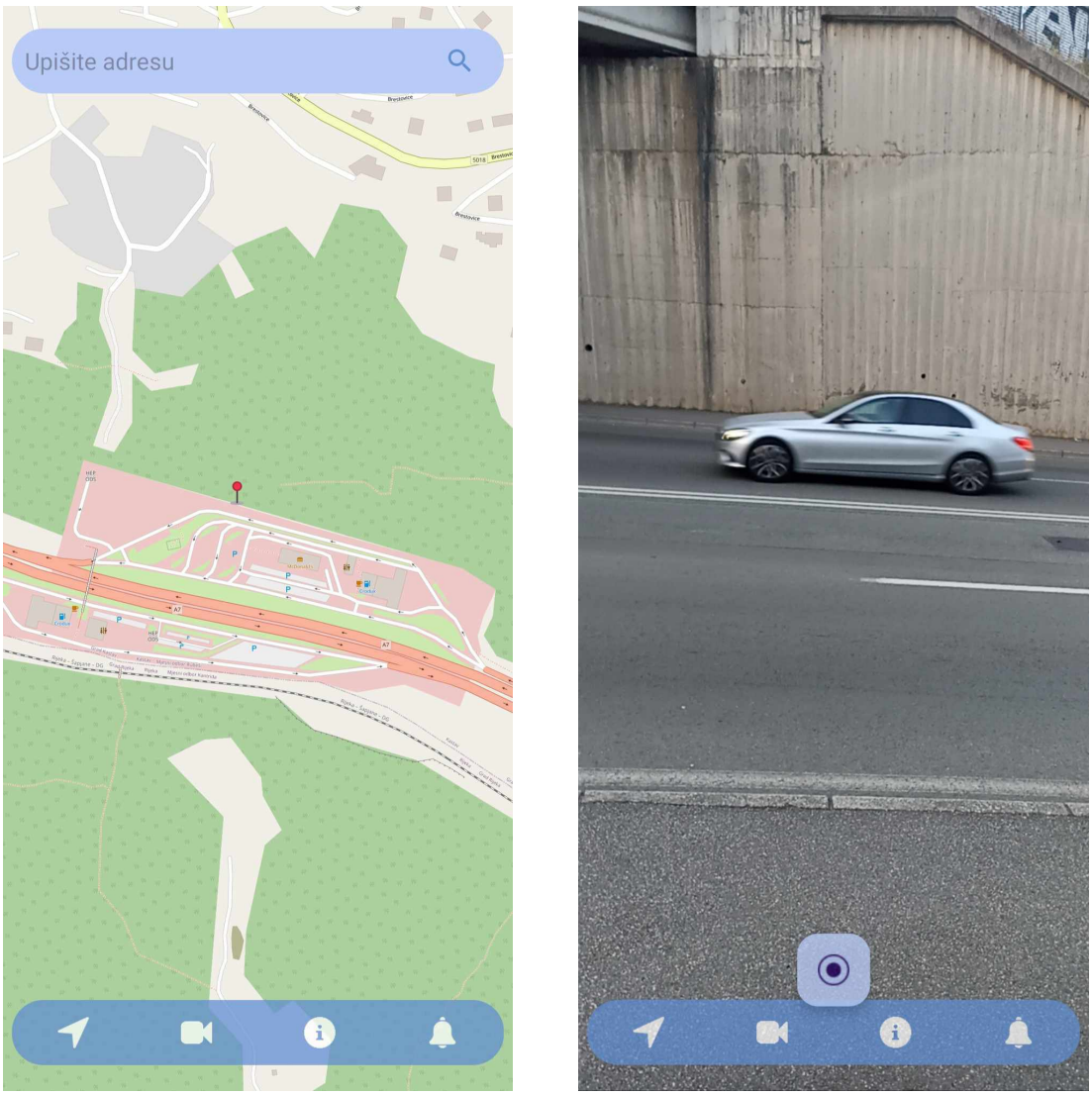
Korisniku su dostupni podaci o korištenom statističkom testu za usporedbu skupina podataka, njegovim rezultatima te interpretacija rezultata statističkog testa. Ukoliko je post-HOC analiza odrađena, dostupni su i njeni rezultati te interpretacije. Primjer dostupnih podataka nakon odrađenih testova prikazan je slikom 5.5.

S obzirom na rezultate, korisnik sam može odabrati rutu kretanja nakon što subjektivno procjeni koja ruta mu je bolja u odnosu na duljinu puta i podacima o statističkoj analizi detektiranih vozila na toj ruti. Omogućeno je i dodavanje prilagođenih podsjetnika u obliku obavijesti kojima se upozorava na početak kretanja prema određenoj lokaciji. Podsjetnike stvara korisnik samostalno, birajući pritom kojim danima u koje vrijeme i s kojom namjerom podsjetnik postoji, a svi podaci se spremaju u SharedPreferences objekt kako bi se podsjetnik mogao koristiti dok

Slika 5.5 Snimka zaslona s prikazanom usporedbom podataka s odabranih prometnica pomoću statističkih testova, uključujući i post-HOC analizu podataka.

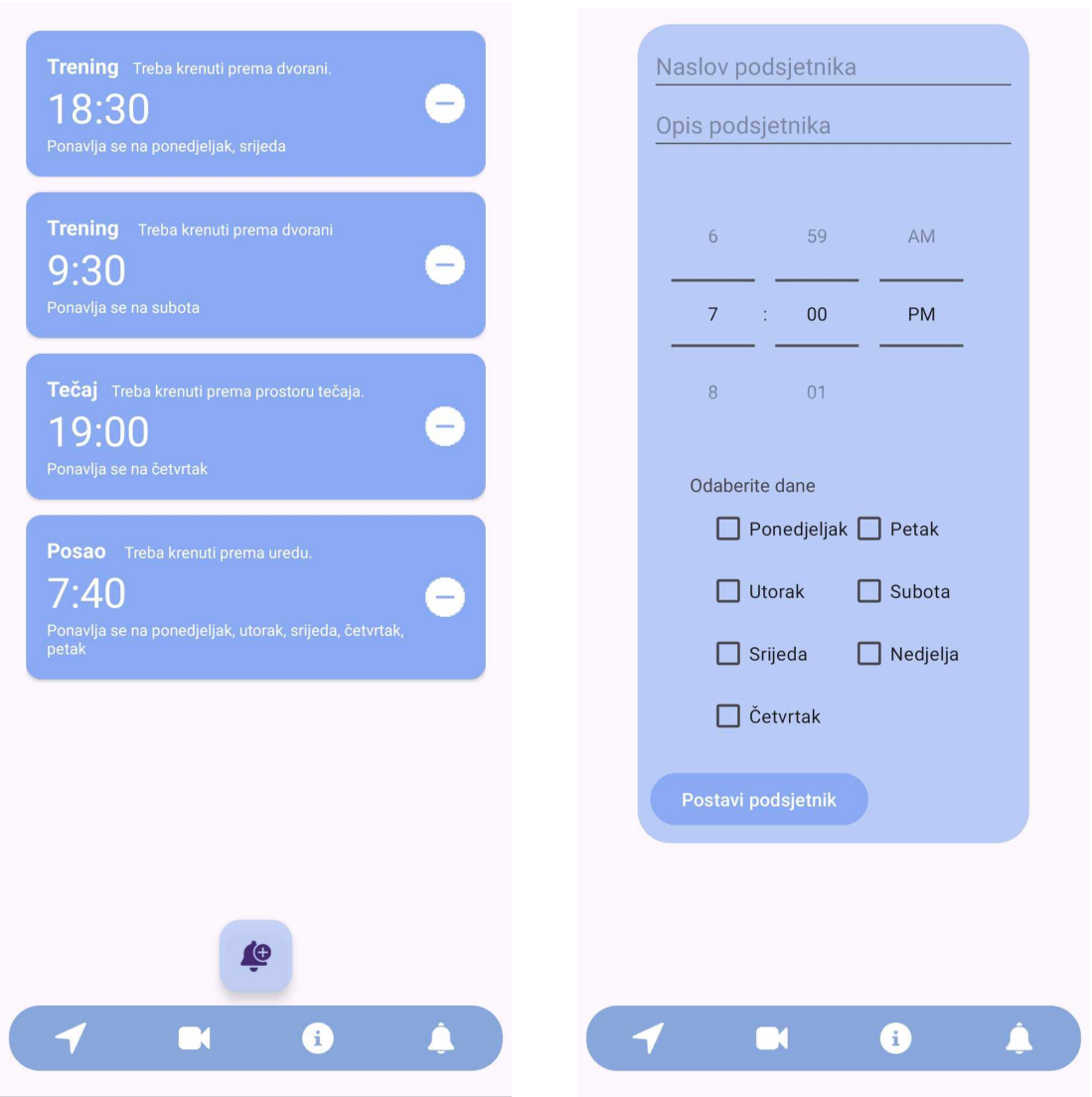
ga korisnik samostalno ne izbriše. Klijentska aplikacija također ima funkcionalnost snimanja video zapisa prometa. Snimanje video zapisa ima funkciju povećanja baze podataka, odnosno te video snimke su obrađene računalnim vidom i statističkom analizom te su rezultati zapisani u bazu za daljnje korištenje. Kako bi se pristupilo kameri uređaja, koristi se JetPack knjižica CameraX. CameraX omogućuje jednostavnu integraciju korištenja kamere uređaja za fotografiranje i snimanje video zapisa. Za snimanje video zapisa potrebno je dati dopuštenje za snimanje video i audio zapisa te je za slanje potrebno imati pristup internetu i trenutnoj lokaciji uređaja. Snimanje video zapisa ograničeno je na petnaest sekundi i limitirano kvalitetom kako bi svi

video zapisi imali jednak broj vremenskih okvira (zasebnih snimaka) u svakom video zapisu, nakon čega se video zapis automatski šalje na pohranu na oblaku s informacijama o lokaciji snimanja, vremenu snimanja, danu u tjednu te datumu snimanja. Te snimke se potom obrađuju s poslužiteljske strane čime se proširuje baza podataka. Sama aplikacija sastoji se od četiri aktivnosti u kojima su raspoređene različite funkcionalnosti, tri fragmenta te trinaest pomoćnih klasa koje definiraju jedinstveno ponašanje unutar aplikacije i pridonose potrebne informacije za definiranje drugih ponašanja unutar aplikacije. Na sljedećim slikama prikazani su najbitniji dijelovi korisničkog sučelja aplikacije.



(a) (b)

Slika 5.6 Korisničko sučelje aplikacije: glavna aktivnost (a) i aktivnost snimanja video zapisa (b)



(a) (b)

Slika 5.7 Korisničko sučelje aplikacije: popis postojećih podsjetnika (a) i stvaranje novog podsjetnika (b)

5.3 Integracija komponenti

Integracija komponenti odrađena je pomoću računarstva u oblaku preko Googleove platforme Firebase. Firebaseove usluge korištene unutar ovoga projekta su Firebase Storage za pohranu video snimaka i *csv* datoteka s podacima obrade računalnim vidom te Realtime Database, baza podataka ažurirana u stvarnom vremenu gdje se čuvaju podaci desktiptivne statističke analize za lakši i brži pristup. Integracija Firebase platforme napravljena je i na klijentskoj i na poslužiteljskoj strani projekta. Na poslužiteljskoj strani korišten je komplet za razvoj softvera (eng. *software development kit*, skraćeno SDK) koji omogućuje interakciju aplikacija napisanih u programskom jeziku Python s Firebase uslugama na oblaku. Za upotrebu Firebase usluga u poslužiteljskom dijelu potrebno je imati instaliran Firebase Admin SDK modul koji mora biti uvezen u program koji ga koristi. Kako bi se projekt u Python-u povezao s točnim projektom na Firebase platformi, postoji JSON datoteka koja služi kao ključ za povezivanje korisnika Firebase platforme te konkretnog projekta samog korisnika unutar Firebase platforme sa željenim Python programom. Integracija Firebase platforme i njenih usluga u Python kod prikazana je slikom 5.8.

```
#Firebase setup
import firebase_admin
from firebase_admin import credentials
cred = credentials.Certificate("serviceAccountKey.json")
#initialization of relevant Firebase services (storage and database)
firebase_admin.initialize_app(cred, {
    'databaseURL': '[redacted]',
    'storageBucket': '[redacted]
})
from firebase_admin import storage
from firebase_admin import db
```

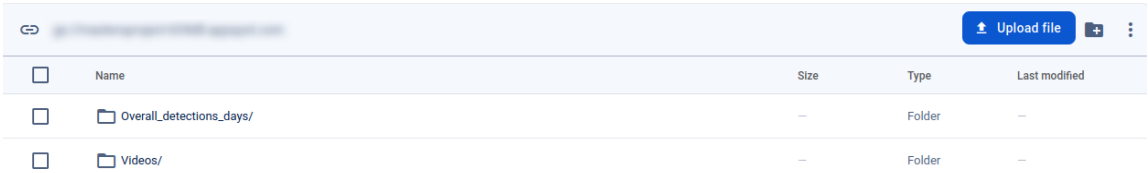
Slika 5.8 Kod za povezivanje projekta i usluga Firebase platforme s projektom u Python programskom jeziku.

Nakon povezivanja projekta s platformom na oblaku, slanje rezultata obrade računalnim vidom odvija se u nekoliko jednostavnih koraka:

1. definicija podataka koji se žele poslati na oblak,

2. definicija putanje za pohranu podataka ili definicija čvorova za upis u bazu podataka,
3. slanje podataka na definirano mjesto.

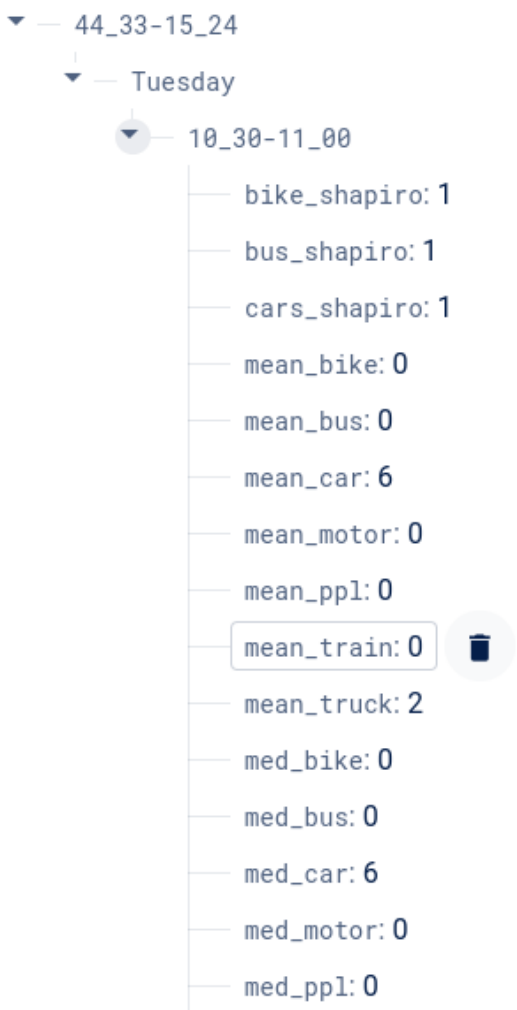
Osim slanja rezultata obrade računalnim vidom, povezivanje s Firebase platformom služi i za preuzimanje korisničkih videa iz pohrane na oblaku kako bi se nad njima obavile funkcionalnosti poslužiteljske strane. Pohrana u Firebase Storage funkcionira na sličan način kao pohrana lokalno na računalo. Za svaki projekt koji koristi ovu uslugu postoji zaseban korijenski direktorij nazvan *bucket* (hrv. kanta) unutar kojega se nalaze sve datoteke i direktoriji stvoreni u Firebase Storage pohrani za određeni projekt. Unutar korijenskog direktorija stvaraju se ostale mape ili prenose datoteke kao i na lokalnoj pohrani. Korijenski direktorij je nemoguće izbrisati s obzirom da bi se u tom slučaju onemogućilo korištenje ove usluge. Prikaz korijenskog direktorija Firebase Storage usluge projekta nalazi se na slici 5.9.



Slika 5.9 Korijenski direktorij Firebase Storage usluge projekta.

Za razliku od Firebase Storage usluge, arhitektura Firebase Realtime Database baze je u formatu JSON stabla s čvorovima te su podaci spremljeni kao par ključ - podatak. Primjer zapisa podataka u Realtime Database bazi podataka prikazan je na slici 5.10.

Integracija Firebase platforme s Android aplikacijom olakšana je unutar Android Studio razvojnog okruženja te se automatizirano odradi unutar samog okruženja. U *build.gradle* datoteku aplikacije potrebno je implementirati svaku korištenu knjižicu te svaku uslugu Firebase platforme koju koristimo u aplikaciji što se može automatizirano odraditi kroz alat Firebase unutar Android Studija. U dokumentaciji Firebase platforme može se pronaći primjere kodova te upute za upotrebu različitih usluga same platforme. Kao i u poslužiteljskoj strani, i u Android aplikaciji koriste se usluge Firebase Storage i Firebase Realtime database, pri čemu se na Firebase



Slika 5.10 Struktura Firebase Realtime Database baze podataka.

Storage pohranu šalju podaci (snimljeni video zapisi spremni za obradu) te se preuzimaju CSV datoteke s prethodnim rezultatima obrade određenih prometnica, a s Firebase Realtime Database se samo preuzimaju podaci o deskriptivnoj statistici određenoj unutar polužiteljskog dijela. Za slanje i preuzimanje podataka potrebno je imati pristup internetskoj vezi, a imajući na umu da je baza podataka u stvarnom vremenu, preuzimanje najnovijih podataka vrši se u stvarnom vremenu korisnikovog zahtjeva.

Poglavlje 6

Zaključak

Cilj rada bio je iskoristiti tehnologiju detekcije objekata računalnim vidom kako bi se dobili podaci za statističku analizu prometa. Korišten je prethodno trenirani model YOLO arhitekture za obradu video snimaka, dok su one dobivene iz korisničke aplikacije za Android operativni sustav. Statistička analiza provedena je u vidu deskriptivne statistike unutar poslužiteljskog dijela pomoću Python programskog jezika uz podršku NumPy knjižice, dok je inferencijalna statistika inicirana od strane korisnika unutar Android aplikacije te je odrađena pomoću Apache Commons Math knjižice za Android. Deskriptivna statistika pokrila je osnovne karakteristike skupova podataka poput srednje prosječne vrijednosti, medijana i standardne devijacije podataka, odnosno mjera centralne tendencije podataka. Inferencijalna statistika odnosi se na usporedbe podataka o prometnicama te ovisi o broju uspoređenih skupina i njihovim obilježjima, a testovi koji se koriste su upareni t-test (odnosno Wilcoxon Signed Rank test kao neparametarska alternativa) u slučaju usporedbe dvije skupine podataka te ANOVA test (odnosno Friedman test kao neparametarska alternativa) u slučaju više od dvije skupine podataka. Android aplikacija koristi dobivene podatke obrade računalnim vidom i deskriptivnom statistikom kako bi korisniku prikazala statistički vjerojatno stanje prometa na prometnicama koje ga zanimaju u trenutku upita, a grafički prikaz statističke analize izveden je grafovima pomoću MPAndroidChart knjižice. Unutar same Android aplikacije korisniku je dostupna digitalna karta koja je implementirana pomoću besplatne i otvorene baze geolokacijskih podataka OpenStreetMap te su na njoj prikazane različite lokacije i rute kretanja na

koje se statistička analiza odnosi. Poslužiteljski dio povezan je s Android aplikacijom pomoću Firebase platforme za računarstvo u oblaku. Usluge Firebase platforme korištene za provedbu projekta su Firebase Storage pohrana u oblaku te Firebase Realtime Database baza podataka u stvarnom vremenu, a korištene su za razmjenu podataka i datoteka između Android aplikacije i poslužiteljskog dijela projekta.

Bibliografija

[1] Marshall, J., Jonker, L. "An introduction to inferential statistics: A review and practical guide" 2009 University of Cumbria

[2] Terven, J., Córdova-Esparza, D.-M., Romero-González, J.-A. "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS" 2023 Mach. Learn. Knowl. Extr.

[3] Wikipedija - Content-based image retrieval, s interneta: https://en.wikipedia.org/wiki/Content-based image retrieval (pristupljeno 1. kolovoza 2024.)

[4] Statcounter Global Stats - Mobile Operating System Market Share Worldwide, s interneta: https://gs.statcounter.com/os-market-share/mobile/worldwide (pristupljeno 5. lipnja 2024.)

[5] Android Developers - The activity lifecycle, s interneta: https://developer.android.com/guide/components/activities/activity-lifecycle (pristupljeno 6. lipnja 2024.)

[6] Pankaj, C., Vaibhav, Y., Anil, G. "Firebase - Overview and Usage" International Research Journal of Modernization in Engineering Technology and Science, 2021

[7] Firebase, s interneta: https://firebase.google.com/ (pristupljeno 12. lipnja 2024.)

[8] OpenStreetMap - Wikipedija, s interneta: https://en.wikipedia.org/wiki/OpenStreetMap (pristupljeno 12. lipnja 2024.)

[9] Wu, J., "Complexity and accuracy analysis of common artificial neural networks on pedestrian detection" MATEC Web of Conferences, 232, 01003, 2018

[10] Android Developers - Save simple data with SharedPreferences, s interneta: https://developer.android.com/training/data-storage/shared-preferences (pristupljeno 9. kolovoza 2024.)

[11] Firebase - Add the Firebase Admin SDK to your server, s interneta: https://firebase.google.com/docs/admin/setup (pristupljeno 13. kolovoza 2024.)

[12] Gašparović, B., Mauša, G., Rukavina, J., Lerga, J., "Evaluating Yolov5, Yolov6, Yolov7, and Yolov8 in underwater environment: Is there real improvement?" 8th International Conference on Smart and Sustainable Technologies (SpliTech), 2023

[13] Gašparović, B., Lerga, J., Mauša, G., Ivašić-Kos, M., "Deep learning approach for objects detection in underwater pipeline images" Applied artificial intelligence, 2022

[14] Field, A., "Discovering statistics using SPSS" SAGE Publications Ltd, 2009

[15] Beins, B., McCarthy, M., "Research Methods and Statistics" Pearson Education Inc., 2012

[16] Ortataş, F., Mahir, K. "Performance Evaluation of YOLOv5, YOLOv7, and YOLOv8 Models in Traffic Sign Detection." 2023 8th International Conference on Computer Science and Engineering

[17] Liu, Q., Yang, L., Da Lin "Revolutionizing Target Detection in Intelligent Traffic Systems: YOLOv8-SnakeVision." Electronics 12.24(2023): 4970

[18] Ćorović, A., Ilić, V., Đurić, S., Marijan, M., Pavković, B., "The Real-Time Detection of Traffic Participants Using YOLO Algorithm", 2018 26th Telecommunications Forum (TELFOR)

[19] Zimmerman, D., Zumbo, B. "Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks." The Journal of Experimental Education 62.1 (1993): 75-86.

[20] Core Features - MPAndroidChart, s interneta: https://weeklycoding.com/mpandroidchart/core-features/ (pristupljeno 29. kolovoza 2024.)

[21] Medium - What is DeepSORT and how to implement YOLOv7 Object Tracking using DeepSORT, s interneta: https://medium.com/@m.moinfaisal/what-is-object-tracking-and-why-deepsort-and-how-to-implement-yolov7-object-tracking-using-deepsort-f0c952f89b06 (pristupljeno 30. kolovoza 2024.)

Pojmovnik

NLP eng. Natural Language Processing - obrada prirodnog jezika

YOLO eng. You Only Look Once - pogledaš samo jednom

ANOVA eng. Analysis of Variance - analiza varijanci

COCO eng. Common Objects in Context - uobičajeni objekti u kontekstu

csv eng. Comma-separated values - zarezom odvojene vrijednosti

xml eng. Extensible Markup Language - proširivi označni jezik

AWS eng. Amazon Web Services - Amazon Web usluge

JSON eng. JavaScript Object Notation - JavaScript objektna notacija

OSM eng. OpenStreetMap - otvorena karta ulica

pt eng. Place-Text - mjesto teksta

HTTP eng. Hypertext Transfer Protocol - protokol prijenosa hiperteksta

API eng. application programming interface - programsko sučelje aplikacije

SDK eng. software development kit - komplet za razvoj softvera

DCNN eng. Deep Convolutional Neural Network - duboka konvolucijska neuronska mreža

Deep SORT eng. Simple Online and Realtime Tracking with Deep Association Metric - jednostavno praćenje na mreži i u stvarnom vremenu s metrikom duboke povezanosti

Sažetak

Ovaj rad istražuje primjenu računalnog vida za automatizaciju analize prometa. Korišten je prethodno treniran YOLO model za detekciju objekata. Statistička analiza odrađena je nad rezultatima detekcije objekata računalnim vidom u dva dijela: deskriptivna statistika koristeći Python programski jezik i NumPy knjižicu te inferencijalna statistika unutar Android aplikacije pomoću Apache Commons Math knjižice. Deskriptivna statistika uključuje izračun mjera centralne tendencije, dok se inferencijalna statistika fokusira na usporedbu prometa na različitim lokacijama pomoću statističkih testova. Unutar Android aplikacije grafički su prikazani rezultati statističke analize kako bi se prikazalo statistički vjerojatno trenutno stanje prometa. Aplikacija također uključuje digitalnu kartu s OpenStreetMap-om i koristi Firebase platformu za pohranu podataka i komunikaciju između Android aplikacije i poslužitelja, te ima mogućnost direktnog slanja video snimki u bazu podataka kako bi ih poslužiteljska strana obradila.

Ključne riječi — **yolo, računalni vid, statistička analiza, statistika, vizualizacija, neuronske mreže, strojno učenje, Python, detekcija objekata, praćenje objekata, analiza prometa, Android, aplikacija, OpenStreetMap, računarstvo u oblaku**

Abstract

This project explores the application of computer vision for automating traffic analysis. A pre-trained YOLO model was used for object detection. The statistical analysis was conducted on the results of object detection in two parts: descriptive statistics using the Python programming language and the NumPy library, and inferential statistics within the Android application using the Apache Commons Math library. Descriptive statistics include the calculation of measures of central tendency, while inferential statistics focus on comparing traffic at different locations using statistical tests. Within the Android application, the results of the statistical analysis are graphically displayed to show the statistically probable current traffic conditions. The application also includes a digital map using OpenStreetMap and utilizes the Firebase platform for data storage and communication between the Android application and the server. It also has the capability to directly upload video recordings to the database for processing on the server side.

Keywords **— yolo, computer vision, statistical analysis, statistics, visualization, neural networks, machine learning, Python, object detection, object tracking, traffic analysis, Android, application, OpenStreetMap, cloud engineering**

Dodatak A

Izvorni kod rada

Izvorni kod rada može se pronaći na javnom repozitoriju *Github* platforme na poveznici: https://github.com/ivanatus/masters_project.