Zahvaljujem se svojem mentoru izv. prof. dr. sc. Zoranu Kalafatiću na savjetima i pomoći tijekom pisanja rada.

Sadržaj

1. Uvod

Računalni vid je grana računarske znanosti čiji je cilj omogućiti računalu identifikaciju i razumijevanje objekata na slikama i video snimkama. Osim identifikacije objekata, često javljajući problemi kojima se bavi računalni vid su klasifikacija, prebrojavanje i praćenje objekata. Kako bi se efikasno i olakšano rješavali takvi problemi, razvijen je model pod nazivom YOLOv8 koji se temelji na konvolucijskim neuronskim mrežama, a one se koriste upravo za raspoznavanje objekata i uzoraka. YOLOv8 je novija inačica modela YOLO kojeg je prvi puta predstavio Joseph Redmon 2016. godine kao novi pristup detekciji objekata. Novo predstavljeni model je bio znatno brži i učinkovitiji od svojih prethodnika i imao je bolje svojstvo generalizacije.

Kako bi model radio dobre predikcije, potrebno je trenirati ga na velikom i raznovrsnom skupu podataka. Skup podataka se najčešće dijeli u tri skupine, a to su podatci za učenje, koji se koriste za iterativno podešavanje težina u neuronskoj mreži, podatci za provjeru na kojima se provjerava generalizacijsko svojstvo neuronskih mreža i podatci za testiranje na kojima se provjerava cjelokupna točnost i kvaliteta modela. YOLOv8 nudi nekoliko unaprijed treniranih modela čiji odabir ovisi o raspoloživom vremenu za treniranje i kvaliteti sklopovlja s obzirom na to da je treniranje modela iznimno zahtjevan proces. Osim unaprijed treniranih modela, postoji opcija trenirati vlastiti model iz nule sa željenim klasama.

U okviru ovoga rada najprije su definirani pojmovi kao što su umjetna inteligencija, duboko učenje, neuroni, neuronske mreže itd. Zatim je dan pregled korištenih tehnologija i opisan je sustav koji koristi YOLOv8 model. Model se koristi za detekciju, praćenje i prebrojavanje automobila na slikama i video snimkama prometnica te je napravljena jednostavna web aplikacija za olakšano korištenje sustava.

2. Umjetna inteligencija

Umjetna inteligencija je tehnologija koja omogućuje računalu rješavanje problema za koje je potrebna ljudska inteligencija kao što su razumijevanje govora, vid i analiza scene, prepoznavanje uzoraka, klasifikacija objekata u pripadne skupine itd. Umjetna inteligencija objedinjuje mnogo različitih znanstvenih disciplina poput računarske znanosti, statistike i analize podataka, inženjerstva programske potpore i sklopovlja, lingvistiku, neuroznanost, filozofiju i psihologiju. [1] Automatizacija raznih poslova, smanjenje faktora ljudske greške, brzina, točnost i efikasnost samo su neke od prednosti korištenja umjetne inteligencije. Područje kojim se bavi umjetna inteligencija je široko i obuhvaća grane poput strojnog učenja, robotike, računalnog vida, ekspertnih sustava, obradu prirodnog jezika itd.

2. 1. Strojno učenje

Strojno učenje je grana umjetne inteligencije čiji je glavni cilj razvoj programa i algoritama na temelju skupa podataka bez eksplicitnog navođenja svih instrukcija koje bi program trebao izvršavati. Na raspolaganju se nalazi model koji je definiran parametrima potrebnim za zadaću koju izvršava, a učenjem modela upravo se ti parametri optimiziraju. Cilj učenja modela je razvijanje mogućnosti da predviđa svojstva na neviđenim, novim podatcima, dakle cilj strojnog učenja je izgraditi model koji dobro generalizira. [2]

Strojno učenje možemo podijeliti u tri vrste, a to su nadzirano, nenadzirano i podržano učenje. Nadzirano učenje definirano je treniranjem modela na označenom skupu podataka, dakle s definiranim ulazom i izlazom, a zatim se koristi za predikciju (klasifikacija u slučaju diskretnih vrijednosti ili regresija u slučaju kontinuiranih vrijednosti). Nenadzirano učenje koristi se kada imamo zadani skup podataka bez zadanih oznaka. Cilj takve metode je otkriti sličnosti i razlike, odnosno pravilnosti u podatcima i napraviti adekvatno grupiranje podataka. Posljednja metoda jest podržano učenje i ono se temelji na učenju kroz iskustvo; svaka dobra akcija je nagrađena, a loša akcija penalizirana.

2.2. Duboko učenje

Duboko učenje je grana strojnog učenja koja se temelji na dubokim neuronskim mrežama. Duboke neuronske mreže su neuronske mreže s puno skrivenih slojeva, odnosno pokazalo se da je efikasnije proširiti mrežu na način da se koristi više slojeva s manje neurona nego

manje slojeva s više neurona. Praktični primjeri korištenja dubokog učenja su autonomna vozila koja prepoznaju prometne znakove, okolinu, pješake i sl., obrambeni sustavi koji koriste duboko učenje za označavanje područja interesa na slikama satelita, medicinske slike za olakšanu ili automatsku dijagnostiku i sustavi za sigurnost koji detektiraju ako je čovjek ili neki drugi objekt u opasnome području. [3] Za provođenje dubokog učenja, potrebne su velike količine označenih podatka (npr. slike s vozilima i informacija o pripadnim koordinatama pojedinog vozila na predanoj slici) i visokokvalitetne grafičke procesorske jedinice, budući da je obrada i rad sa slikama računalno vrlo zahtjevan proces i zahtjeva mnogo resursa.

2.3. Računalni vid

Računalni vid je grana računarske znanosti koja je fokusirana na omogućivanju računala da identificira i razumije objekte i ljude na slikama i video snimkama, odnosno cilj računalnog vida je replicirati na računalu način i razumijevanje onoga što ljudi vide. [4] Postoji mnogo načina na koji se koristi računalni vid od kojih su najčešći i najvažniji:

- Segmentacija slike – razdvajanje slike na particije interesantne za problem kojemu pristupamo.
- Detekcija objekata – identificiranje specifičnih objekata na slikama ili video snimkama kao što su npr. vozila i pri identifikaciji se objekt adekvatno uokviri.
- Raspoznavanje simbola – identifikacija brojeva, slova i ostalih simbola na slici te pretvorba u tekst kojeg razumiju računala.
- Prepoznavanje lica – napredna vrsta detekcije objekata koja se danas često koristi kao opcija pri biometrijskoj sigurnosti i cilj je ne samo prepoznati lice, nego i identificirati osobu koja se nalazi na slici.
- Detekcija rubova – tehnika koja se koristi za poboljšanje kvalitete slike ili identificiranja specifičnih objekata na slici.
- Detekcija uzoraka – proces prepoznavanja oblika, boja i ostalih vizualnih pojava koje se ponavljaju na slici.
- Klasifikacija objekata na slici – pri identifikaciji objekata na slici, svrstavaju se u pripadne klase, npr. pri detekciji vozila, treba odlučiti radi li se o automobilu, autobusu, kamionu i sl. [5]

3. Umjetne neuronske mreže

Poznato nam je da se mozak sastoji od velikog broja bioloških neurona koji rade paralelno i glavna im je uloga primitak, obrada i prijenos informacije. Oponašanjem neuronske mreže kakva se nalazi u mozgu, smišljena je umjetna neuronska mreža koja omogućava rješavanje problema koji su presloženi da bi se riješili na klasični, algoritamski način. Umjetna neuronska mreža je skup međusobno povezanih elemenata koji imaju ulogu matematičkog modela čija se funkcionalnost temelji na biološkom neuronu i služe za prijenos te obradu podataka [6].

3.1. Neuroni

Neuron je osnovna građevna jedinica umjetnih neuronskih mreža i svaki neuron, odnosno svaki čvor predstavlja matematički model. Prikaz strukture neurona nalazi se na slici Sl. 3.1. Ulazni parametri neuronske mreže pomnože se s pripadnim težinama i opcionalno zbroje s vektorom pristranosti, a zatim kumulativno sumiraju:

$$y = \sum_{i=0}^{n} w_i x_i + b = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n + \ \ b \tag{3.1.}$$

Zatim se primjenjuje prijenosna funkcija čija je uloga odlučiti koliko je važna informacija koju nosi trenutni neuron za izračun daljnjih predikcija. Postoje tri glavne vrste prijenosnih funkcija u umjetnim neuronskim mrežama, a to su:

- Funkcija skoka – definira se prag koji odlučuje o aktivaciji neurona.

$$\hat{y} = g(y) = \begin{cases} 0, & y < 0 \\ 1, & y \geq 0 \end{cases} \tag{3.2.}$$

- Linearne funkcije (funkcije identiteta) – funkcija ne radi nikakve transformacije, već samo prenosi ono što je dobiveno.

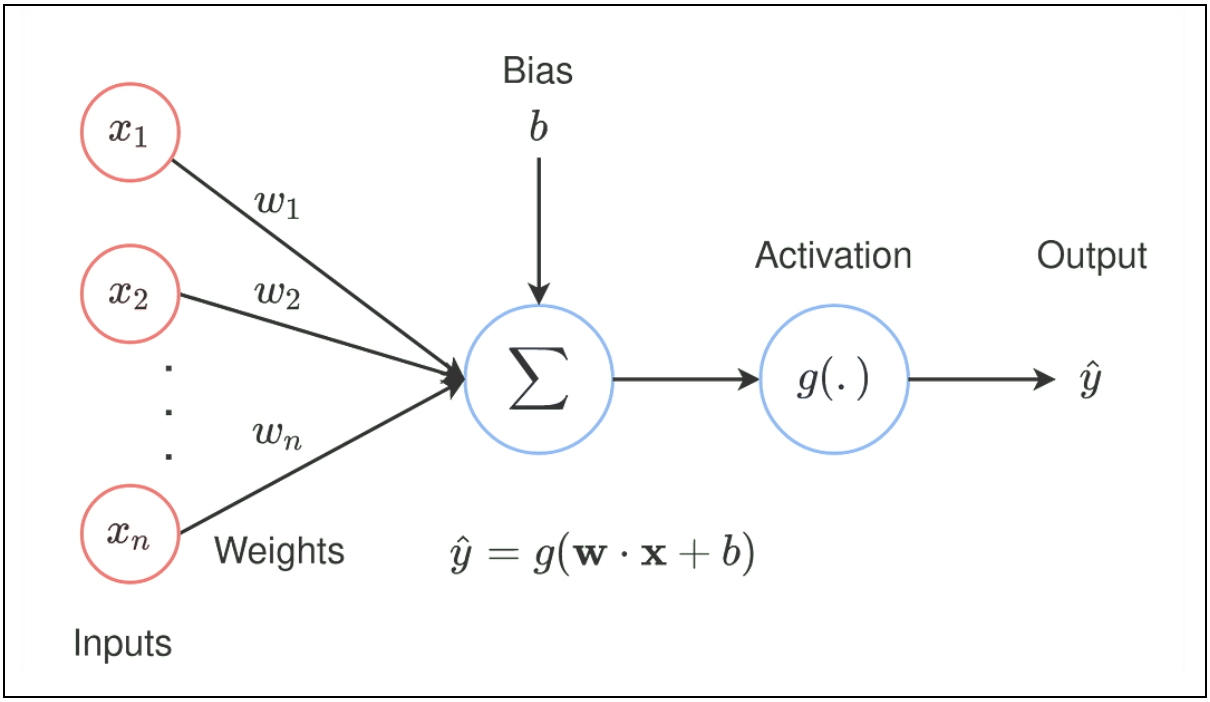
$$\hat{y} = g(y) = y \tag{3.3.}$$

- Nelinearne funkcije – dodavanjem nelinearnog sloja u neuronsku mrežu, omogućen je razvoj bolje i kompleksnije neuronske mreže s povećanom ekspresivnosti. Kao primjer nelinearnih prijenosnih funkcija često se primjenjuje logistička sigmoida:

$$\hat{y} = g(y) = \frac{1}{1+e^{-y}} \tag{3.4}$$

Međutim, danas se sve više koristi ReLU (eng. *Rectified linear unit*) nelinearna funkcija koja omogućava treniranje dubljih mreža i računalno je učinkovitija za izračunati:

$$\hat{y} = g(y) = \max(0, y) \tag{3.5}$$

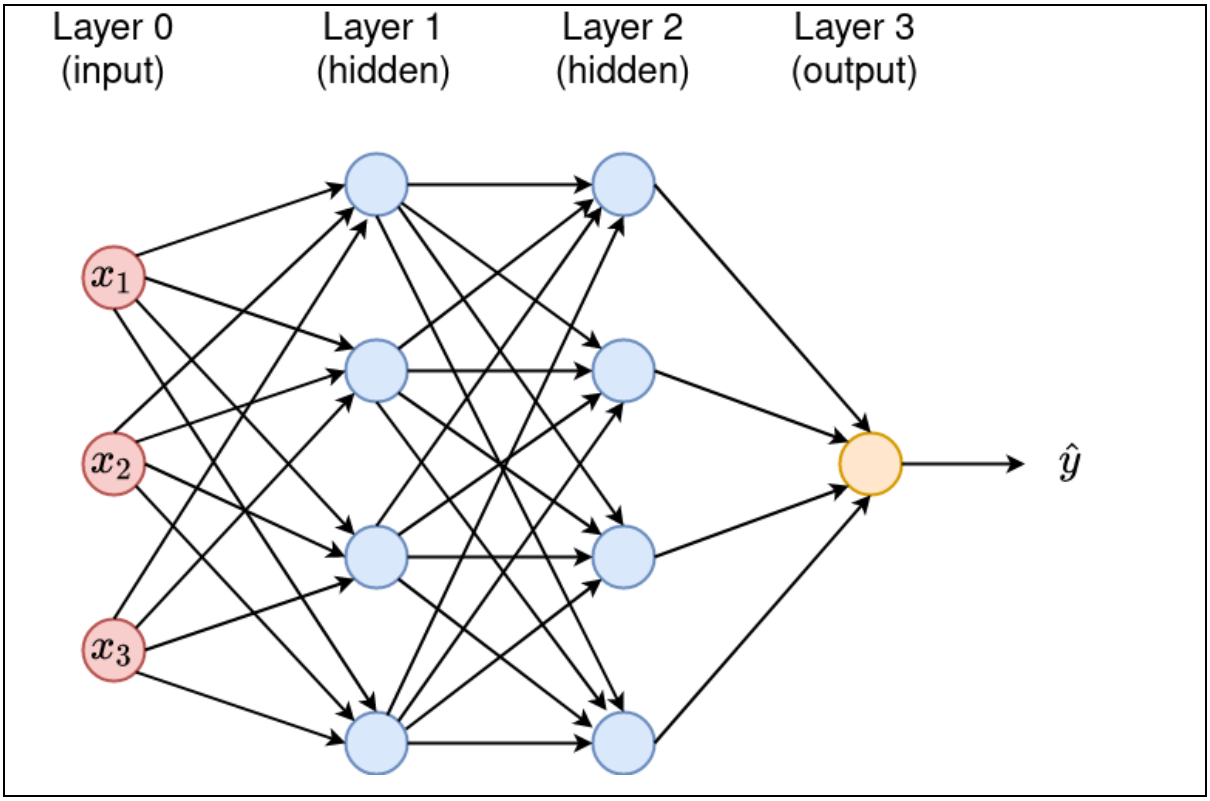


Sl. 3.1: Neuron u umjetnoj neuronskoj mreži [7]

3.2. Arhitektura neuronske mreže

Neuronska mreža sastoji se od niza slojeva međusobno povezanih neurona kao što je to prikazano na slici Sl. 3.2. Prvi sloj je uvijek ulazni sloj i on ima onoliko čvorova koliki je broj ulaznih parametara. U tom sloju ne izvršavaju se nikakve matematičke operacije, nego se samo prenose informacije u sljedeći sloj. Zatim slijedi jedan ili više skrivenih slojeva, a taj broj ovisi o kompleksnosti problema koji se rješava. U skrivenom sloju izvršavaju se razne matematičke operacije koje su definirane u neuronima i prenose se iz sloja u sloj sve do posljednjeg, izlaznog sloja. Informacije na izlazu neuronske mreže predstavljaju ciljnu vrijednost, odnosno rezultat obrade zadanih podataka koji se može koristiti za donošenje

odluke (u slučaju da se radi o klasifikaciji, dobili bi na izlazu o kojoj se klasi najvjerojatnije radi sudeći po zadanom skupu podataka) ili se prosljeđuje dalje, u iduću neuronsku mrežu.



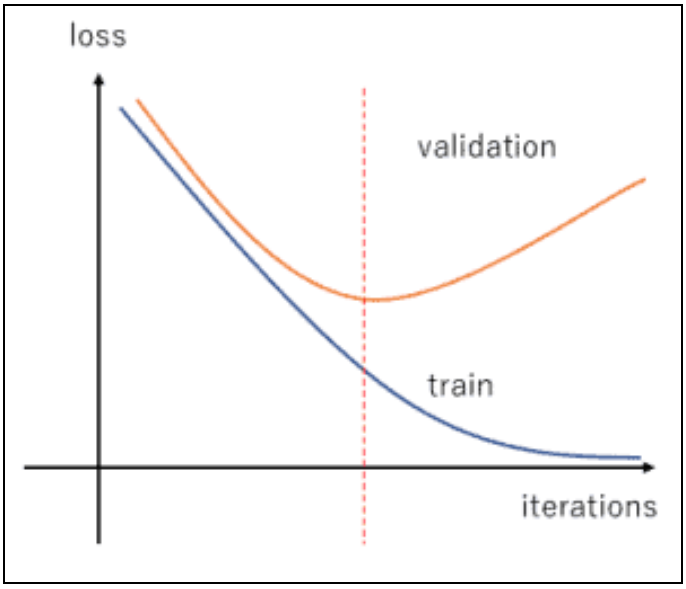
Sl. 3.2: Arhitektura neuronske mreže [7]

3.3. Učenje umjetne neuronske mreže

Za učenje umjetne neuronske mreže, najprije treba pripremiti veliki skup podataka koji je točno označen kako bi dobili model koji može raditi kvalitetne predikcije budući da se ti podatci koriste se kao ulazni parametri u neuronskoj mreži. Prikupljeni skup podataka se najčešće dijeli u tri skupa podataka:

- Skup podataka za učenje – obično čini oko 70% priređenih podataka i koristi se za iterativno treniranje modela. Poželjno svojstvo skupa podataka za učenje je reprezentativnost budući da je cilj strojnog učenja napraviti model koji što bolje generalizira.
- Skup podataka za provjeru – obično čini oko 15% priređenih podataka i s njime se provjerava koliko je dobro generalizacijsko svojstvo mreže. Na slici Sl. 3.3 vidimo

ovisnost pogreške o broju iteracija, odnosno epoha na skupu podataka za učenje i provjeru, dakle kako se povećava broj iteracija, tako se povećava i točnost na skupu podataka za učenje, model savršeno radi na takvom skupu. Međutim, u takvom slučaju javlja se problem kojeg nazivamo pretreniranost, što bi značilo da mreža gubi svojstvo generalizacije, stoga je cilj naći optimum (kojeg na grafu predstavlja crvena isprekidana linija) koji predstavlja kompromis između točnosti modela i svojstva generalizacije.



Sl. 3.3: Prikaz ovisnosti pogreške o broju epoha učenja [8]

- Skup podataka za testiranje – obično čini oko 15% priređenih podataka i koristi se za krajnju provjeru kvalitete mreže nakon što smo dovoljno istrenirali model i provjerili da radi dobro na skupu podataka za provjeru.

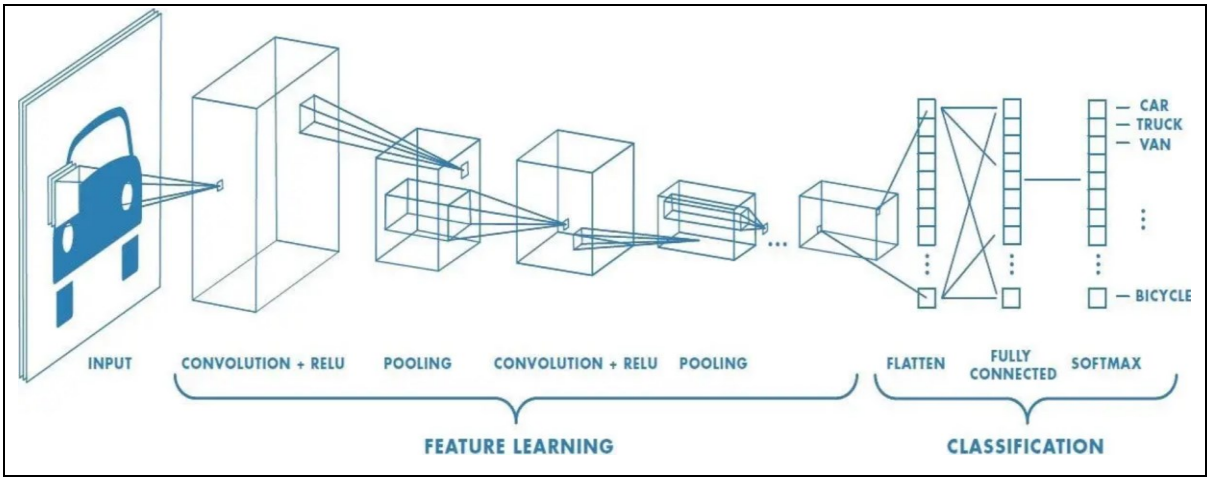
Učenje modela znači prilagođivanje težina i koeficijenta pristranosti pojedinih neurona u neuronskoj mreži kroz zadani broj epoha. Epoha predstavlja jedan obilazak cijelog skupa podataka za učenje i prilagodbu parametara neuronske mreže u ovisnosti o obiđenim podatcima kroz tu iteraciju. Kako bi neuronska mreža adekvatno prilagodila težine, potrebno je definirati funkciju pogreške, a jedna od takvih najčešće korištenih funkcija je srednje kvadratno odstupanje.

$$err = \frac{1}{N}\sum_i^N (y_i - \hat{y}_i)^2 = \frac{1}{N}\sum_i^N (y_i - NN(x_i))^2 \tag{3.6}$$

Funkcija izračunava prosjek kvadrata razlike između unaprijed poznate izlazne vrijednosti i izlazne vrijednosti koje izračuna neuronska mreža za zadani ulaz. Nakon definiranja funkcije pogreške, često se koristi postupak optimiranja težina neuronske mreže koji se temelji na izračunu gradijenta, odnosno iz parcijalnih derivacija moguće je saznati kako će se funkcija pogreške mijenjati ako mijenjamo ili smanjujemo težinu pojedinog neurona. Postupak učenja neuronskih mreža koji se temelji na prethodno opisanom postupku prilagodba težina naziva se postupak propagacije pogreške unatrag. [6] Neke od prednosti korištenja takvog postupka su činjenica da neuronska mreža ne treba imati nikakvo prethodno znanje za izvršavanje algoritma, nego se sve težine inicijalno postave na slučajne vrijednosti i podešavaju učenjem, efikasnost izvođenja algoritma i velika skalabilnost, odnosno postupak propagacije pogreške unatrag radi izvrsno na velikim količinama podataka.

3.4. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže su vrsta umjetnih neuronskih mreža koje se koriste za prepoznavanje uzoraka i objekata na slikama, dakle često se koriste u području računalnog vida. Kao i klasične umjetne neuronske mreže, konvolucijske neuronske mreže također imaju ulazni i izlazni sloj, međutim, kod konvolucijskih mreža skriveni sloj je nešto kompleksniji. U skrivenom sloju javljaju se konvolucijski sloj, aktivacijski sloj, sloj sažimanja (eng. *pooling*), sloj za izravnavanje i potpuno povezani sloj kao što vidimo na slici Sl. 3.4.



Sl. 3.4: Arhitektura konvolucijske neuronske mreže [9]

Prvi sloj u konvolucijskim neuronskim mrežama je konvolucijski sloj, a njegova svrha je iz zadanog skupa podataka izvršiti ekstrakciju značajki kao što su rubovi i boje na slici uporabom raznih filtera. Konvolucija je operacija kojom iz dva izvora informacija, što su u ovome slučaju izvorna slika i filter, nastaje nova informacija koja može biti u različitim oblicima kao što su nova slika, vektor brojčanih vrijednosti ili samo jedan piksel. Konvolucija se tipično koristi za zamućenje ili izoštravanje slike, izoštravanje rubova i sl. međutim, u kontekstu dubokog učenja, koristi se za smanjenje dimenzionalnosti slike i ekstrakciju relevantnih značajki. [23]

Konvolucijski sloj često dolazi u kombinaciji s aktivacijskim slojem koji ima ključnu ulogu u prijenosu i obradi podataka te uvodi element nelinearnosti u mrežu. Najčešće se kao prijenosna funkcija u aktivacijskom sloju uzima prethodno definirana (3.5.) ReLU funkcija.

Zatim slijedi sloj sažimanja (eng. *pooling*) čija je glavna uloga smanjiti dimenzije dobivenih podataka kako bi se izračun u računalu ubrzao i olakšao te kako bi se izbjegao problem prenaučenosti. Postoje dvije vrste sažimanja koje se najčešće koriste a to su sažimanje maksimalnom vrijednošću (eng. *max pooling*) i sažimanje prosječnom vrijednošću (eng. *average pooling*). [9] Maksimalno sažimanje odabire najveću vrijednost na dijelu slike koji se trenutno obrađuje i odlično je za uklanjanje šuma. S druge strane, prosječno sažimanje odabire prosječnu vrijednost na dijelu slike koji se trenutno obrađuje. U kontekstu klasifikacije i identifikacije objekata, sažimanje prosječnom vrijednošću ima nešto lošija svojstva u odnosu na sažimanje maksimalnom vrijednošću.

Sloj za izravnavanje služi kako bi se podatci pretvorili u jednodimenzionalni vektor, što je pogodan oblik za daljnju obradu podataka u potpuno povezanom sloju.

Potpuno povezani sloj s primitkom prethodno obrađenih podataka izvršava klasifikaciju u slučaju da se radi o diskretnim izlaznim vrijednostima, odnosno regresiju ako se radi o kontinuiranim vrijednostima. [11]

4. Pregled korištenih tehnologija

U ovome poglavlju nalazi se prikaz tehnologija korištenih za implementaciju sustava za detekciju vozila u video snimkama prometnica.

4.1. YOLO model za detekciju objekata

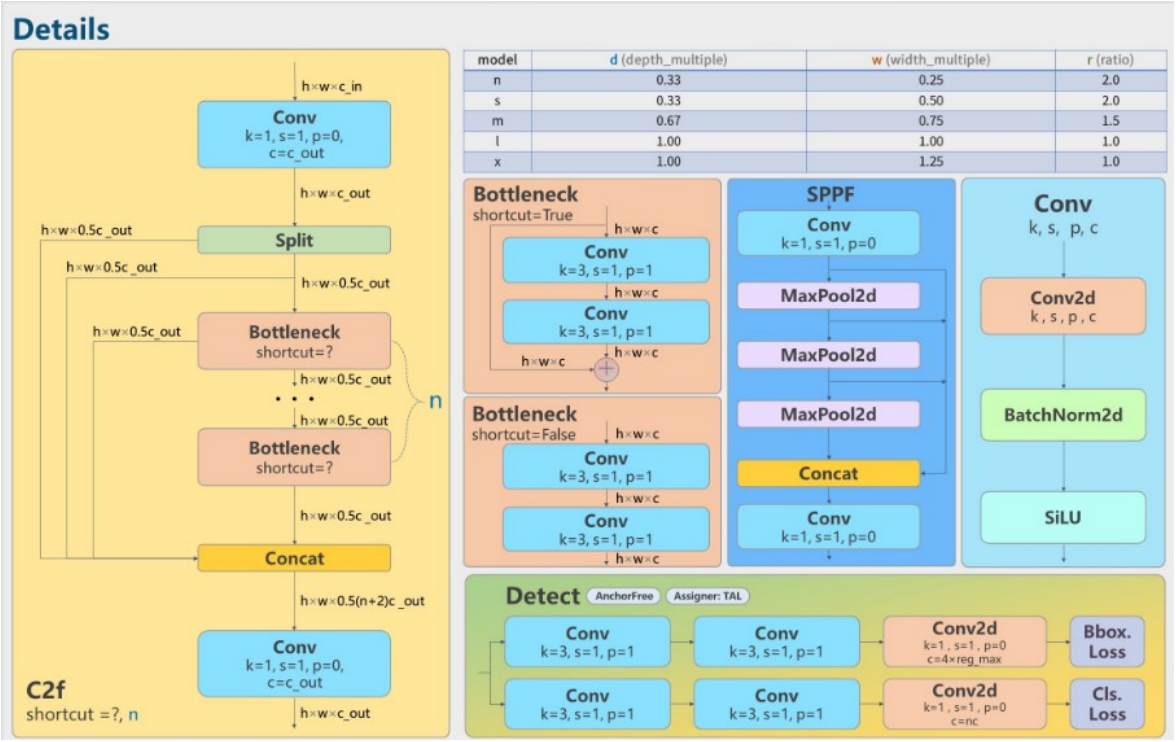
YOLO (eng. *You only look once*) je jedan od najpopularnijih visokorazvijenih modela za detekciju objekata, a predstavio ga je Joseph Redmon 2016. godine [22]. Glavna uloga YOLO modela je identifikacija i klasifikacija objekata jednim prolaskom kroz konvolucijsku neuronsku mrežu. Činjenica da YOLO model problem detekcije rješava kao regresiju, a ne klasifikaciju, čini ga mnogo bržim i efikasnijim od svojih prethodnika kao što je R-CNN (eng. *Region-based Convolutional Neural Network*). [12] R-CNN je također tip arhitekture koji se koristi u dubokom učenju za detekciju objekata, a radi na način da se slika postepeno dijeli na regije koje bi mogle sadržavati objekte i pritom se ekstrahiraju relevantne značajke. [13] Osim brzine i efikasnosti, prednosti korištenja YOLO modela su visoka točnost u identifikaciji, bolja generalizacija u odnosu na prijašnje modele i činjenica da je model otvorenog koda, što znači da ga svatko može slobodno koristiti i distribuirati.

U međuvremenu, YOLO modeli su znatno napredovali od 2016. kada su prvi puta predstavljeni, odnosno iz inicijalnog YOLO modela, razvili su se i YOLOv2, YOLOv3, pa sve do YOLOv10 koji je izašao u svibnju 2024. godine. Međutim, odabrani model koji će biti relevantan za izgradnju sustava za detekciju vozila je YOLOv8.

4. 1. 1. YOLOv8 model

YOLOv8 model izašao je u siječnju 2023. godine kao inačica YOLO modela kojeg je razvila tvrtka Ultralytics, a koristi se za detekciju, klasifikaciju i praćenje objekata te segmentaciju slike. Prednosti korištenja YOLOv8 modela su činjenica da ima veću točnost predikcija pri identifikaciji objekata i napravljen je kao paket u programskom jeziku Python, što programerima olakšava korištenje. Osim toga, oko YOLOv8 modela razvila se velika zajednica, što znači da se može naći kvalitetna podrška pri korištenju tog modela. [14] Neke od inovacija koje čine YOLOv8 boljim odabirom od svojih prethodnika su uvođenje mehanizma za dodatno opažanje relevantnih dijelova slike, poboljšana točnost detekcije za male objekte i smanjena vremenska i prostorna kompleksnost programa uz zadržavanje točnosti.

Za treniranje i evaluaciju YOLOv8 modela koristi se COCO (eng. *Common objects in context*) skup podataka u kojemu se nalazi preko dvjesto tisuća slika iz realističnih situacija i definirano je osamdeset različitih klasa kao što su osoba, automobil, mačka, laptop itd. COCO skup podataka je vrlo raznovrstan i često se koristi kao mjera kvalitete performansa modela u području računalnog vida.



Sl. 4.1: Moduli u YOLOv8 arhitekturi [28]

U arhitekturi modela YOLOv8 nalazi se nekoliko modula koji su prikazani na slici Sl. 4.1. Konvolucijski modul (Conv) sastoji se od sloja koji izvršava dvodimenzionalnu konvoluciju, sloja koji normalizira ono što se dovede na ulaz centriranjem i skaliranjem kako bi proces bio brži i stabilniji, te na posljetku SiLU aktivacijskog sloja koji na izlazu daje umnožak ulaznih vrijednosti i logističke sigmoide ulaznih vrijednosti. Modul usko grlo (eng. *bottleneck*) sastoji se serije konvolucijskih modula i koristi se kao ključni dio C2f modula koji je zadužen za ekstrakciju značajki na visokoj razini i unapređenju brzine modela. SPPF modul (eng. *Spatial Pyramid Pooling Fusion*) osim konvolucijskih slojeva, sadrži i tri sloja sažimanja i njegova glavna uloga je uhvatiti i uočiti relevantne značajke na različitim skalama, što dodatno poboljšava kvalitetu detektora. Posljednji modul koji se koristi je modul za detekciju koji na izlazu daje vrijednosti pogreške detektiranog okvira u odnosu na referentni okvir i pogrešku klasifikacije. Arhitektura modela YOLOv8 može se podijeliti na

tri glavne komponente, a to su kralježnica / oslonac (eng. *backbone*), vrat (eng. *neck*) i glava (eng. *head*). [27] Kralježnica se sastoji od serije izmjenično poredanih konvolucijskih i C2f modula te završava s SPPF modulom. Glavni cilj kralježnice je ekstrakcija važnih značajki sa slike. Vrat kombinira prethodno ekstrahirane značajke iz kralježnice i prenosi ih do glave u kojoj se izvršavaju detekcija i klasifikacija budući da se u glavi nalaze moduli za detekciju.

4. 2. Programski jezik Python

Sama implementacija sustava za detekciju vozila na video snimkama prometnica napravljena je u Python-u, koji je najpopularniji programski jezik za rad u području strojnog učenja, pa tako i računalnog vida. Python je objektno orijentirani programski jezik visoke razine koji se ne prevodi, nego za izvođenje koristi interpreter. Biblioteke u Python-u korištene za izvedbu sustava su:

- ultralytics – Biblioteka koja omogućava korištenje YOLOv8 modela, dakle moguće je koristiti unaprijed trenirani model, ali postoji i opcija treniranja modela od početka s vlastito odabranim skupom podataka. Osim treniranja, moguće je raditi i predikcije s ostvarenim modelom na slikama i video snimkama, a postoje još i neke zanimljive opcije poput brojanja objekta interesa ili procjena brzine vozila u prometu.
- cv2 – Biblioteka OpenCV je jedna od najpopularnijih biblioteka za područje računalnog vida, budući da nudi razne opcije za rad sa slikama poput prikaza slike, binarizaciju, razne manipulacije poput promjene dimenzija, rotiranje, izrezivanje, filtriranje itd.
- flask - Radni okvir za razvoj programske potpore za web koji se nudi kao biblioteka u Pythonu. Koristi se za izradu dinamičnih, odnosno interaktivnih web aplikacija što ga danas čini popularnim odabirom. Flask nudi programsku potporu za upravljanjem korisnikovih zahtjeva, prikazivanje HTML sadržaja i interakciju s bazom podataka te je jednostavan za naučiti, koristiti i održavati. [10]

Za razvoj sustava i web aplikacije korišten je PyCharm. PyCharm je jedno od najpopularnijih integriranih razvojnih okruženja (eng. IDE – *Integrated development environment*) za programski jezik Python. Razvila ga je tvrtka JetBrains i široko se koristi u područjima kao što su razvoj programske potpore za Web i znanost o podatcima.

4.3. Google Colab

Prije svega, treba napomenuti da je Jupyter bilježnica aplikacija koja nudi mogućnosti pisanja i pokretanja Python koda u blokovima koji se mogu izvršavati zasebno, a služi i za pisanje dokumentacije. Google Colab je internetski servis koji upravo nudi usluge Jupyter bilježnice bez potrebe za bilo kakvim instalacijama na računalo, što ga čini praktičnim za korištenje. Razlog koji čini Google Colab jednim od najpopularnijih servisa za korištenje u području strojnog učenja je besplatan pristup grafičkim procesorskim jedinicama (kao što je Nvidia Tesla T4, kojeg nudi besplatna verzija Google Colab-a), što je vrlo važno za efikasno treniranje na velikom skupu podataka. Osim toga, Google Colab ima već unaprijed instalirane popularne biblioteke za područje računalnog vida (kao što su TensorFlow ili PyTorch) i integriran je s Google Diskom što omogućava olakšani prijenos relevantnih datoteka.

4.4. Kaggle

Kaggle je popularna internetska platforma s javno dostupnim skupovima podatka relevantnih za područja kao što su znanost o podatcima i strojno učenje. Osim toga, Kaggle organizira brojna natjecanja u kojima je cilj napraviti što bolji model za zadanu problematiku.

4.5 Alati za razvoj programske potpore za web

Za razvoj web aplikacije na poslužiteljskoj strani korišten je Python Flask, međutim, na klijentskoj strani korišteni su HTML, CSS i Javascript. HTML (eng. *HyperText Markup Language*) je označni jezik koji se koristi za izradu kostura web stranice i često dolazi u paru s CSS-om (eng. *Cascading Style Sheets*). CSS dodaje stil web stranici, odnosno koristi se kako bi vizualno uljepšali stranicu, dodali animacije, prilagodili izgled web stranice za različite uređaje itd. Ako želimo web stranicu učiniti dinamičnom, u tom slučaju koristimo Javascript. To je skriptni jezik koji dodaje element interaktivnosti, a često se koriste i Javascript radni okviri na klijentskoj i poslužiteljskoj strani.

5. Implementacija, eksperimenti i rezultati

Sustav za detekciju vozila na video snimkama prometnica može se implementirati korištenjem unaprijed treniranog YOLOv8 modela budući da je treniran na COCO skupu podataka u kojemu se nalaze klase relevantne za takav sustav kao što su automobil, motor, autobus i kamion. Osim toga, sustav je moguće ostvariti treniranjem YOLOv8 modela od početka na odabranom skupu podataka kroz određeni broj epoha. Glavni zadatci koje bi sustav trebao zadovoljavati uključuju identifikaciju i praćenje vozila na video snimci te prebrojavanje ulaznih i izlaznih vozila. U ovome poglavlju nalazi se prikaz i usporedba rezultata predikcija korištenjem i unaprijed treniranog modela, ali i modela treniranom na odabranom skupu podataka.

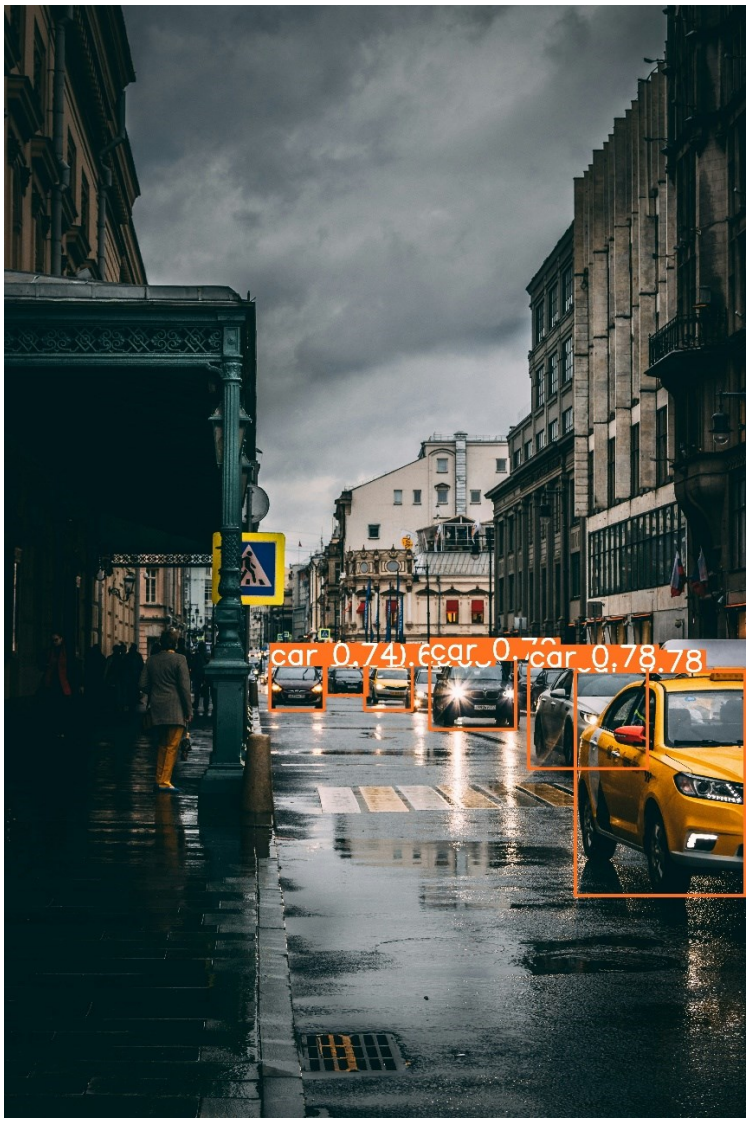
5.1. Unaprijed trenirani YOLOv8 model

Kako bi imali referencu za usporedbu vlastito implementiranog modela za detekciju vozila, koristimo unaprijed trenirani YOLOv8 model koji se može vrlo jednostavno implementirati ako imamo instaliranu Python biblioteku *ultralytics* ili ako koristimo servis poput Google Colab-a. Nakon uvoza modela, možemo ga koristiti za predikcije na video snimaka i slikama te možemo zadavati dodatne parametre kao što su klase objekata koje nas zanimaju ili prag pouzdanosti ispod koje model ne bi detektirao objekte. Za ilustraciju prethodno opisane funkcionalnosti, pogledajmo isječak koda sa slike Sl. 5.1. Kao argumente za poziv funkcije za predikciju postavljamo put do željene datoteke nad kojom želimo da se izvrši detekcija, prag pouzdanosti, opcija želimo li spremiti rezultat, što u ovome slučaju želimo, i identifikacijski broj klasa koje nas interesiraju (automobil – 2, motor – 3, autobus – 5, kamion – 7). Kao rezultat dobivamo sliku Sl. 5.2 na kojoj vidimo da je detekcija automobila na slici uspješna i vozila su točno uokvirena. [20]

```python
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
model.predict(source='../data/proba.jpg',
              conf=0.5,
              save=True,
              classes=[2, 3, 5, 7])
```

Sl. 5.1: Demonstracija predikcije vozila korištenjem YOLO modela



Sl. 5.2: Primjer detekcije automobila

5.2. YOLOv8 model treniran na odabranom skupu podataka

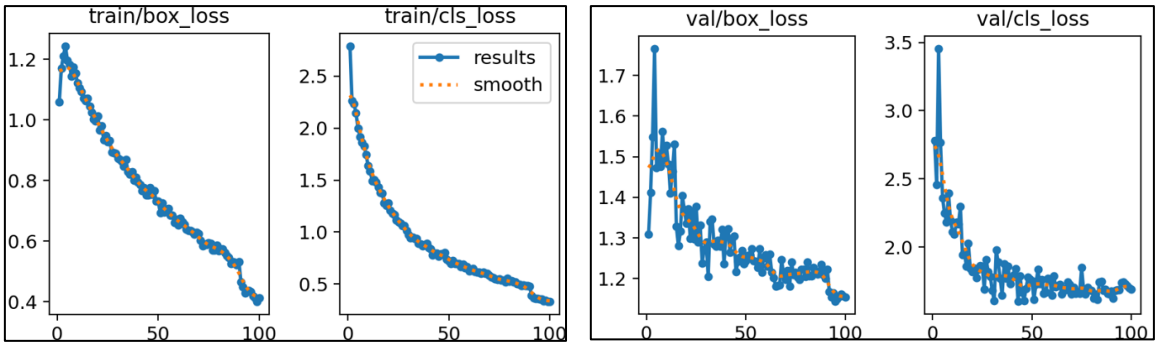
Kako bi omogućili YOLOv8 modelu prepoznavanje željenih objekata, najprije treba pronaći kvalitetan skup podataka koji sadrži reprezentativni uzorak objekata koje želimo detektirati. Međutim, za treniranje YOLOv8 modela nisu dovoljne samo slike, nego su potrebne i oznake u točno specificiranom formatu: (klasa objekta, x koordinata centra, y koordinata centra, širina, visina). Ako skup podataka nije označen, treba ga ručno anotirati korištenjem nekih od internetskih servisa poput CVAT [16] ili Roboflow [17].

Za treniranje modela u okviru ovoga rada koristimo unaprijed označeni skup podataka koji se može se pronaći na internetskoj stranici Kaggle: [18]. Skup podataka sastoji se od 1254 slike s pripadnim oznakama u obliku tekstualnih datoteka i podijeljen je u tri manja skupa podataka, odnosno u skupove za treniranje, validaciju i testiranje. Klase koje su podržane tim skupom podataka su vozilo hitne pomoći, autobus, automobil, motor i kamion.

5.2.1. Pregled rezultata treniranja modela

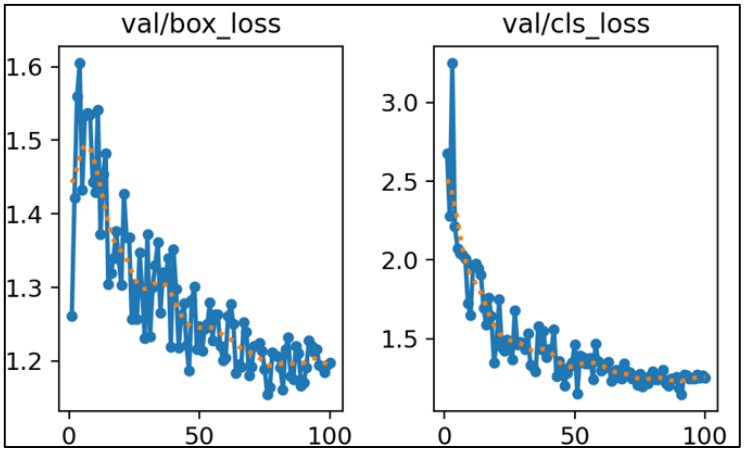
Treniranje modela vrši se na platformi Google Colab korištenjem prethodno opisanog skupa podataka kroz 100 epoha. Kao rezultat treniranja dobijemo razne automatski generirane metrike za provjeru kvalitete modela i možemo postepeno pratiti razvoj modela kroz svaku epohu u procesu učenja.

Za početak, pogledajmo nekoliko metrika za napredak učenja po epohama. Na slici Sl. 5.3 nalazi se nekoliko grafova koji predočavaju kako napreduje proces treniranja na skupu za učenje (lijeva dva drafa) i na skupu za provjeru (desna dva grafa). Budući da grafovi prikazuju ovisnost različitih gubitaka o epohama, a grafovi su padajući, pretpostavljamo da učenje ide u dobrome smjeru. Na slikama su prikazane po dvije metrike: *box_loss* koji predstavlja grešku poklapanja pravokutnika odziva dobivenog detektorom s referentnim oznakama i *cls_loss* koji predstavlja grešku prilikom klasifikacije objekata. Kako niti jedan graf nije krenuo stagnirati tijekom proteklih 100 epoha, pretpostavljamo da model još nije došao do stanja prenaučenosti, odnosno povećavanjem broja epoha, model bi postao još kvalitetniji.



Sl. 5.3: Pogreške na skupu za učenje i na skupu za provjeru

Osim na evaluaciji modela na skupu podataka za učenje i skupu podataka za provjeru, zanima nas kako model radi na skupu podataka za testiranje. Na slici Sl. 5.4 nalazi se graf koji prikazuje kako se mijenjaju prethodno definirani *box_loss* i *cls_loss*, dakle budući da su i ovi grafovi padajući, zaključujemo da model radi dobro i na skupu za testiranje.



Sl. 5.4: Pogreška na skupu za testiranje

Kao iduće dvije metrike za evaluaciju modela uzimamo preciznost i odziv. Kako bi definirali preciznost i odziv, trebamo definirati sljedeće pojmove: [19]

- TP (eng. *true positives*) – broj uzoraka na kojima se nalaze objekti koji su ispravno klasificirani. Kao primjer uzimamo sliku na kojoj se nalazi automobil i model je stvarno ispravno klasificirao automobil na slici.
- TN (eng. *true negatives*) – broj uzoraka na kojima nema objekata koje bi model trebao identificirati i model tada ne prepoznaje nikakve objekte. Kao primjer uzimamo sliku na kojoj se nalazi ulica bez vozila i model ne identificira niti jedan objekt.

- FP (eng. *false positives*) – broj uzoraka na kojima nema objekata koje bi model trebao identificirati, ali model identificira i neispravno klasificira objekte sa slike. Kao primjer uzimamo sliku na kojoj se nalazi mačka, a model ju klasificira kao autobus.
- FN (eng. *false negatives*) – broj uzoraka na kojima se nalaze objekti koje bi model trebao identificirati, ali ne prepoznaje nikakve objekte, stoga možemo reći da je neispravna klasifikacija. Kao primjer uzimamo sliku na kojoj se nalaze dva kamiona i model ne prepoznaje niti jedno vozilo na slici.

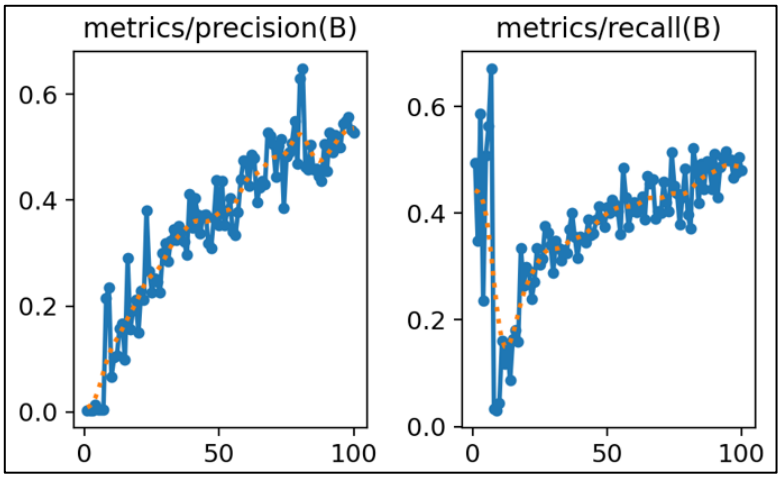
Preciznost (eng. *precision*) modela predstavlja postotak stvarnih pozitivno klasificiranih primjera u skupu u kojemu se nalaze svi pozitivno klasificirani primjeri i računa se formulom (5.1.) :

$$P = \frac{TP}{TP+FP} \tag{5.1.}$$

Odziv (eng. *recall, true positive rate*) modela predstavlja postotak stvarnih pozitivno klasificiranih primjera u skupu u kojemu se nalaze svi pozitivni primjeri i računa se formulom (5.2.):

$$R = TPR = \frac{TP}{TP+FN} \tag{5.2.}$$

Kako bi model radio dobro, treba pronaći balans između preciznosti i odziva modela. Na slici Sl. 5.5 nalaze se dva grafa koja predstavljaju preciznost i odziv modela te možemo uočiti kako oba dvije metrike rastu s brojem epoha, što je dobar znak prilikom treniranja modela.

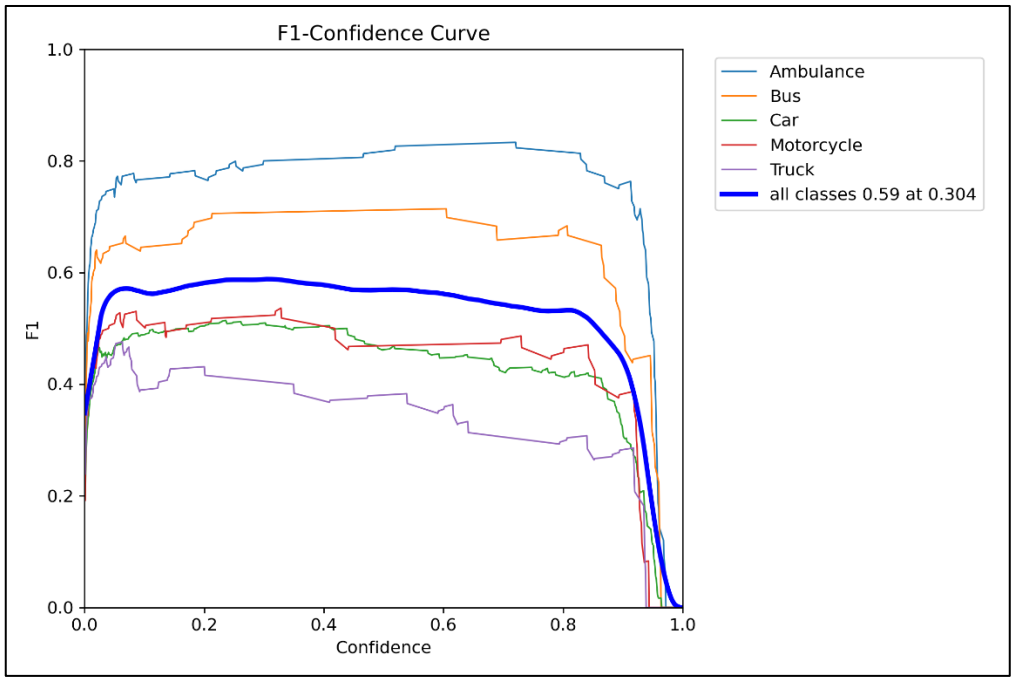


Sl. 5.5: Prikaz preciznosti i odziva modela

Nakon što smo definirali što su to preciznost i odziv, možemo definirati F1 vrijednost. F1 vrijednost je metrika koja uzima u obzir odziv i preciznost te nam govori o njihovom balansu. Računa se kao harmonijska sredina između odziva i preciznosti što je prikazano formulom (5.3.):

$$F1 = 2 * \frac{P*R}{P+R} \tag{5.3.}$$

F1 krivulja predstavlja F1 vrijednost (y-os) na različitim pragovima pouzdanosti (x-os) i iz grafa možemo saznati ponašanje za svaku klasu za model kojeg treniramo. Cilj kojem trebamo težiti prilikom treniranja modela je imati sve F1 vrijednosti što bliže 1 jer bi to značilo da model radi savršene predikcije, ali to je naravno u praksi teško izvedivo. Na slici Sl. 5.6 nalazi se F1 krivulja za trenirani model i možemo zaključiti kako je F1 krivulja najbolja za klasu vozila hitne pomoći, a najgora za klasu kamiona. Najbolja F1 krivulja gledana za cijeli model je na pragu pouzdanosti 0.304 i tada F1 vrijednost iznosi 0.59, što nije baš najbolji rezultat budući da je prag pouzdanosti poprilično malen.



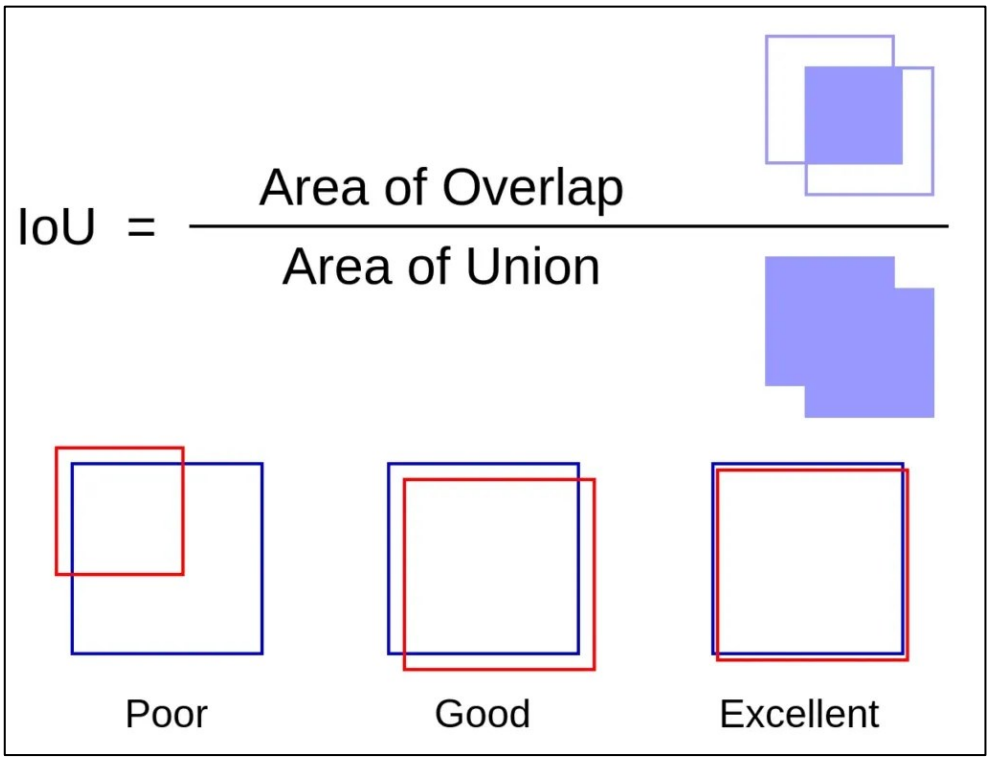
Sl. 5.6: F1 krivulja

Sljedeća metrika koja je interesantna za evaluaciju kvalitete modela za detekciju objekata je srednja prosječna preciznost – mAP (eng. *mean average precision*). Srednja prosječna

preciznost računa se kao prosjek svih prosječnih preciznosti za svaku klasu (AP) i prikazana je formulom (5.4.):

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{5.4.}$$

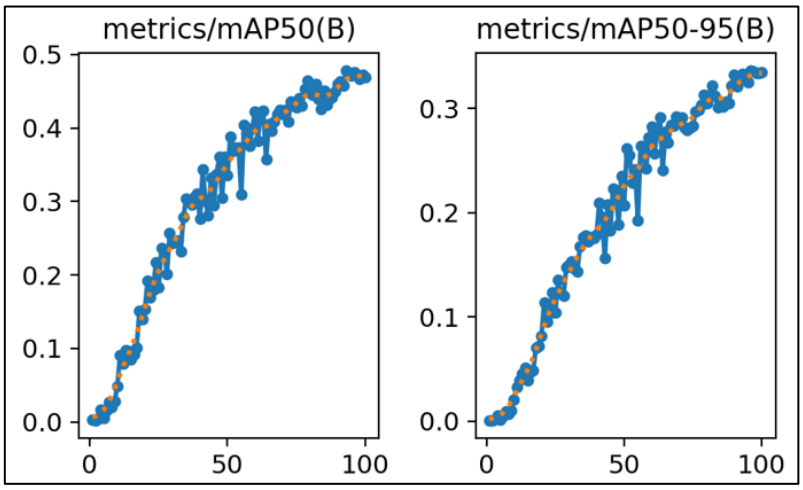
Zatim definirajmo što je to presjek preko unije - IoU (eng. intersection over union). IoU je vrijednost koja se računa kao omjer između površine u kojoj se presijecaju pravokutnik odziva dobivenog detektorom i referentni pravokutnik te površine koja predstavlja uniju pravokutnika odziva dobivenog detektorom i referentnog pravokutnika. Kako bi tu definiciju lakše shvatili, pogledajmo sliku Sl. 5.7 na kojoj je prethodna definicija lijepo vizualizirana.



Sl. 5.7: Presjek preko unije [24]

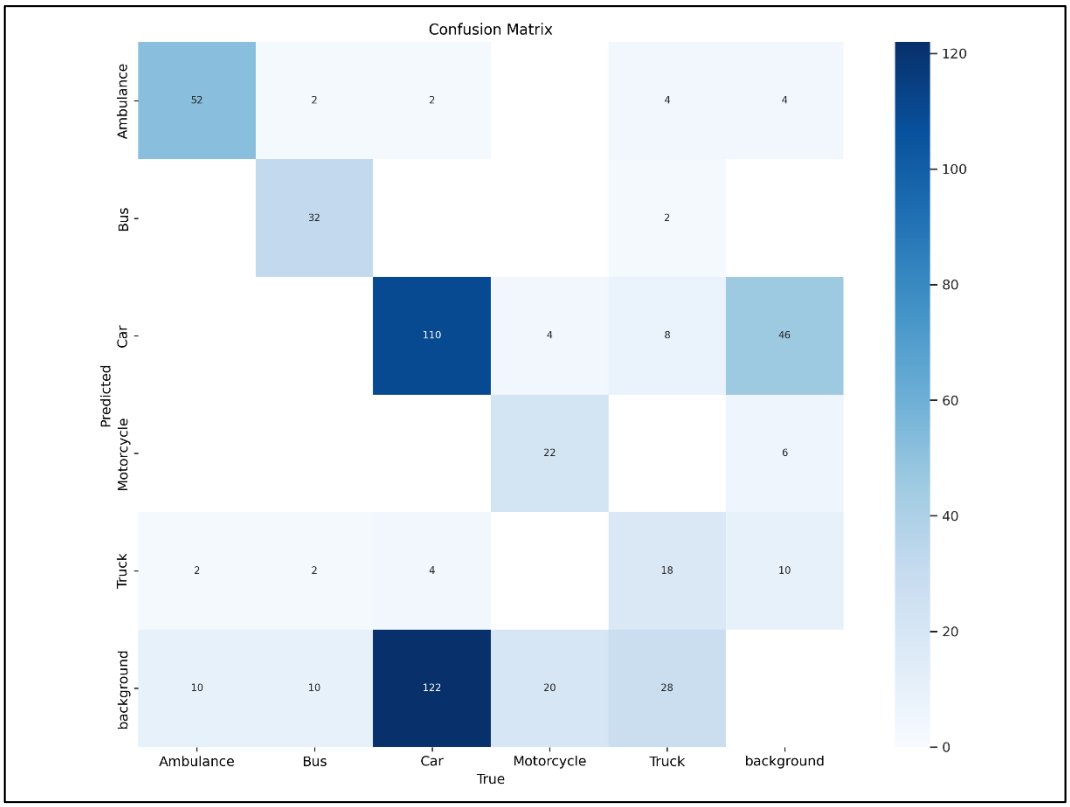
Sada kada znamo što su to srednja prosječna preciznost (mAP) i presjek preko unije (IoU), možemo definirati što su to mAP50 i mAP50-95. mAP50 je srednja prosječna vrijednost koja se računa s IoU vrijednosti jednakoj 0.5 i to je metrika za provjeru točnosti modela na „laganim“ detekcijama. mAP50-95 je srednja prosječna vrijednosti koja se računa s IoU vrijednosti između 0.5 i 0.95 te nam daje procjenu točnosti modela na raznovrsnim uzorcima, uključujući i „teže“ detekcije. Na slici Sl. 5.8 nalaze se grafovi koji prikazuju

prethodno opisane metrike i budući da su grafovi rastući, možemo zaključiti da učenje modela dobro napreduje, odnosno povećava se preciznost modela. [25]



Sl. 5.8: Metrike mAP50 i mAP50-95

Jedna od najpopularnijih metoda za evaluaciju modela u području strojnog učenja naziva se matrica zabune (eng. *confusion matrix*). Matrica zabune koristi se za vizualizaciju kvalitete predikcija pojedinih klasa, odnosno redovi u matrici zabune predstavljaju stvarnu klasu koja je predstavljena slikom, a stupci predstavljaju klasu koju je model identificirao. S obzirom na tu tvrdnju, pod pretpostavkom da se sve klase poredaju istim redoslijedom po redovima i stupcima, možemo uočiti kako bi idealna matrica zabune bila dijagonalna matrica, odnosno sve vrijednosti bi se nalazile na dijagonali. Matrica zabune za model koji prepoznaje vozila nalazi se na slici Sl. 5.9. Možemo uočiti kako se pojedine klase poprilično dobro prepoznaju, ali problem se javlja u tome što se puno primjeraka nalaze u posljednjem retku kojeg čine lažne detekcije (FP) i u posljednjem stupcu kojeg čine propuštene detekcije (FN). S obzirom na to da puno primjeraka nije identificirano ili je lažno identificirano, a po prethodnim metrikama smo zaključili da se ne radi o prenaučenosti, možemo zaključiti kako bi model imao višu uspješnost kada bi se skup podataka za učenje proširio kvalitetno označenim slikama ili kada bi se povećao broj epoha učenja modela. Osim toga, najveći broj slika koje se nalaze u skupu podataka predstavljaju automobile, što znači da skup podataka nije najbolje balansiran, odnosno skup podataka bi trebalo proširiti sa slikama ostalih klasa.



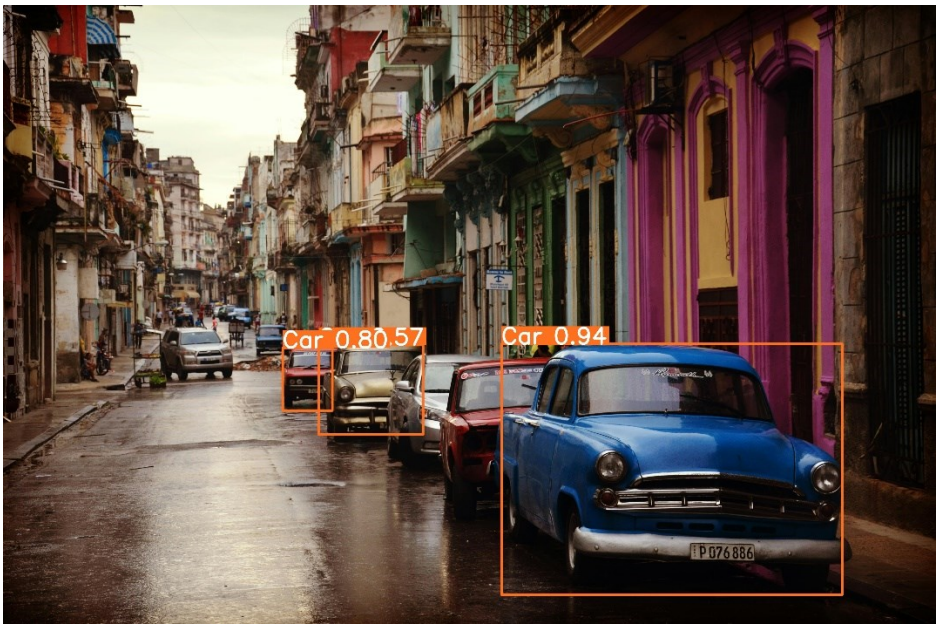
Sl. 5.9: Matrica zabune

5.3. Usporedba i evaluacija unaprijed treniranog i vlastitog modela

Za početak, pogledajmo kako bi dva prethodno opisana modela identificirala objekte na istoj slici. Kao rezultat predikcije korištenjem unaprijed treniranog modela dobivamo sliku Sl. 5.10 i vidimo da model radi izvrsno, kao što je očekivano. [15]

Sl. 5.10: Identifikacija automobila korištenjem unaprijed treniranog modela

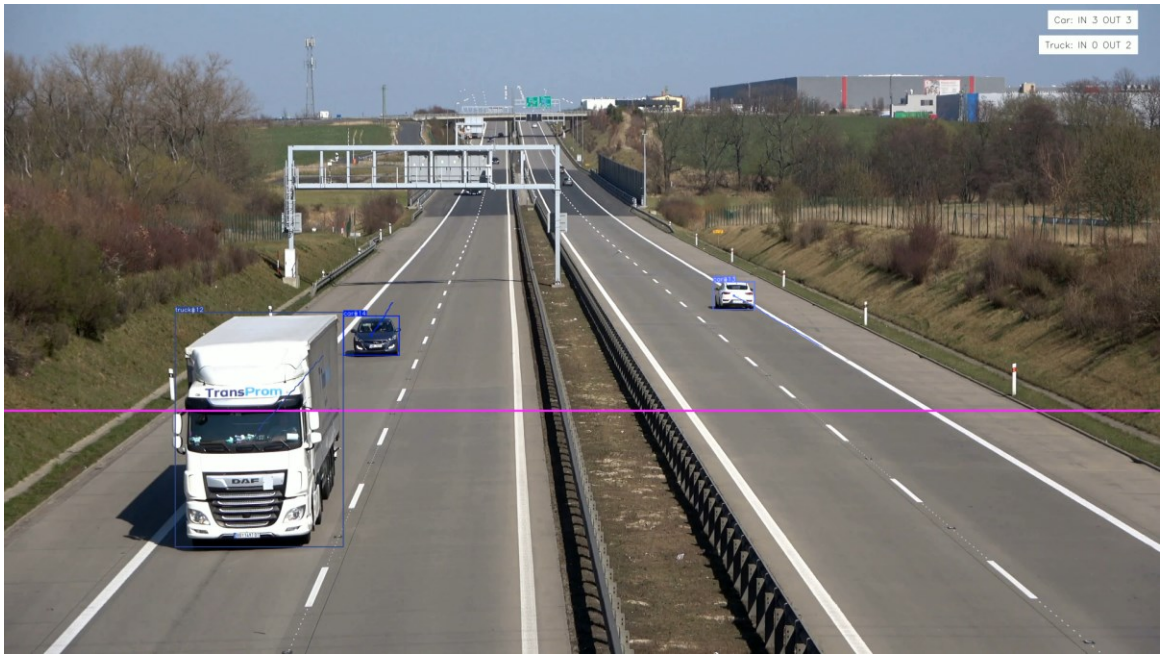
S druge strane, korištenjem modela treniranog na vlastito odabranom skupu podataka dobivamo sliku Sl. 5.11 i vidimo da model ne radi baš najbolje, odnosno prepoznaje samo nekoliko automobila na slici.



Sl. 5.11: Identifikacija automobila korištenjem vlastito treniranog modela

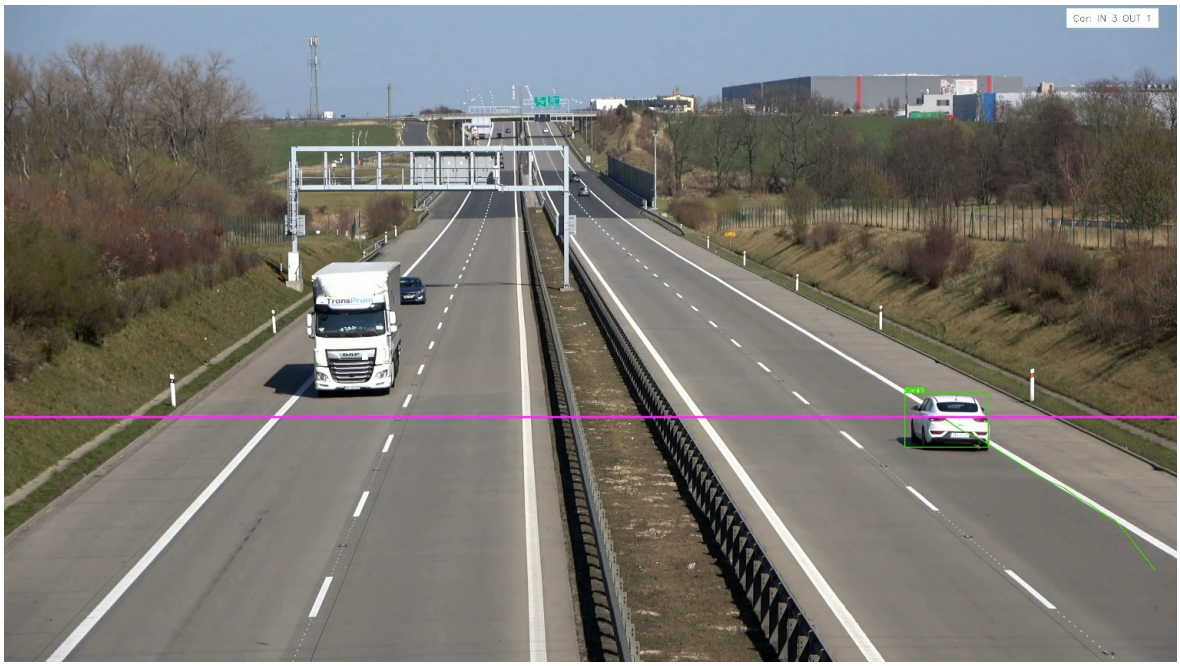
Zatim pogledajmo kako radi brojanje vozila na video snimci. Na slici Sl. 5.12 nalazi se jedan isječak iz videosnimke [21] na kojemu se vozila na slici dijele u dvije kategorije: vozila koja ulaze, odnosno idu „prema nama“, i vozila koja izlaze, odnosno udaljavaju se iz naše

perspektive. Pred kraj snimke prebrojani su 3 ulazna i 3 izlazna automobila te 2 izlazna kamiona. Granica koja prati prolaz vozila nalazi se nešto ispod sredine slike kako bi se povećala vjerojatnost za točnim prebrojavanjem budući da će model lakše detektirati veće, odnosno bliže objekte.



Sl. 5.12: Brojanje vozila na video snimci korištenjem unaprijed treniranog modela

Rezultati na vlastito treniranom modelu nalaze se na slici Sl. 5.13 i su izbrojana su samo četiri vozila, odnosno tri ulazna i jedan izlazni automobil, dakle drugi model ima znatno slabiju uspješnost prilikom brojanja vozila u odnosu na prvi model. Problem koji uočavamo na ovom primjeru je što model nikako nije prepoznao kamione, po čemu dodatno zaključujemo da skup podataka nije najbolje balansiran.

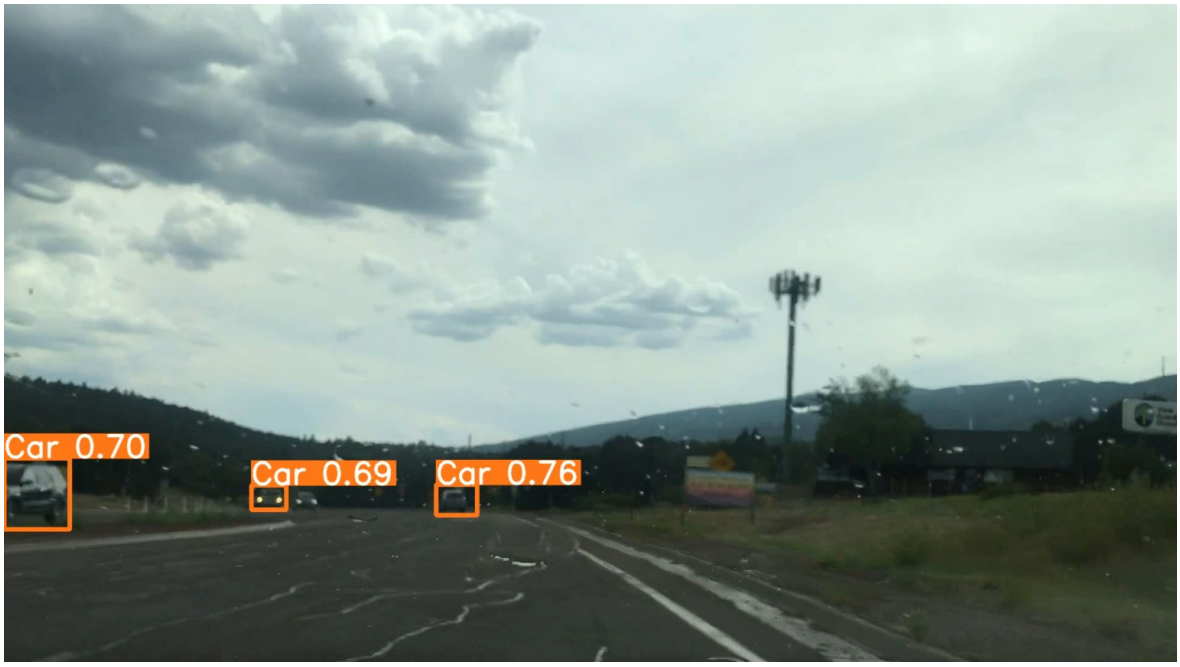


Sl. 5.13: Brojanje vozila na video snimci korištenjem vlastito treniranog modela

Zatim pogledajmo kako rade modeli na video snimci u kojoj su nešto lošiji uvjeti, odnosno budući da je kiša, smanjena je vidljivost. Na slici Sl. 5.14 nalazi se isječak iz video snimke [26] i vidimo kako unaprijed trenirani model radi dobro i u lošijim uvjetima.

Sl. 5.14: Identifikacija vozila pri smanjenoj vidljivosti korištenjem unaprijed treniranog modela

Na slici Sl. 5.15 nalazi se isječak iz iste video snimke, no koristimo vlastito trenirani model i vidimo da se i ovdje poprilično dobro prepoznaju automobili na cesti, što je možda neočekivani rezultat s obzirom na rezultate treniranja.

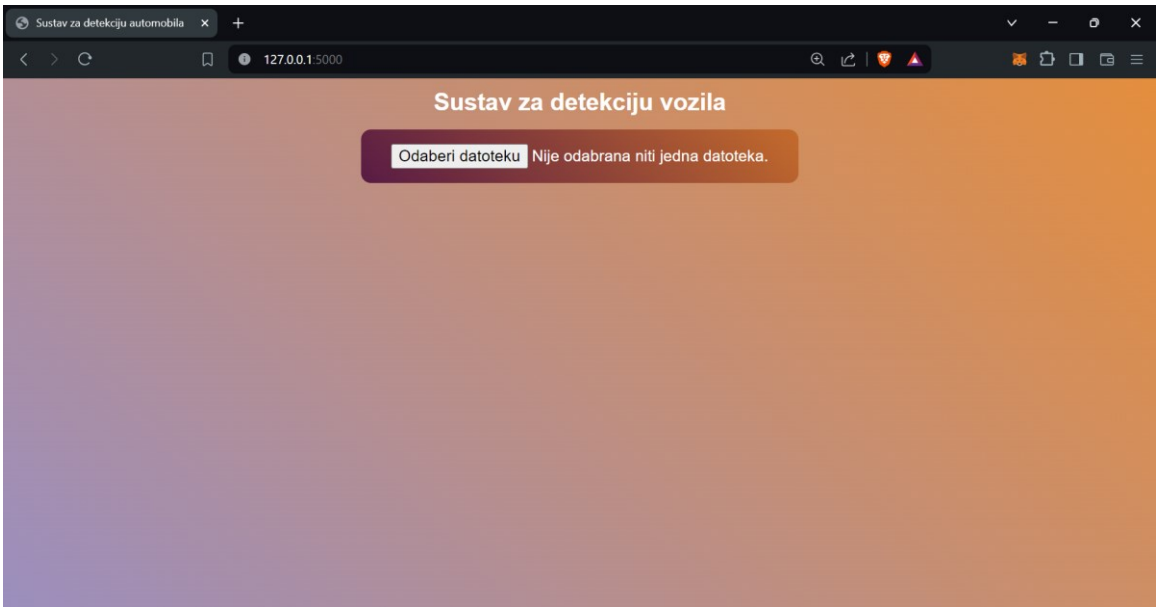


Sl. 5.15: Identifikacija vozila pri smanjenoj vidljivosti korištenjem vlastito treniranog modela

Uzimajući u obzir statistike i eksperimente, možemo zaključiti kako unaprijed trenirani model ima znatno višu i konzistentniju uspješnost od vlastito treniranog modela pri identifikaciji i klasifikaciji objekata. Međutim, to je očekivano, budući da bi za kvalitetniji model trebali puno više podataka za učenje, koji bi trebali biti dobro balansirani te bi trebali utvrditi idealan broj epoha za učenje (npr. 150 ili 200).

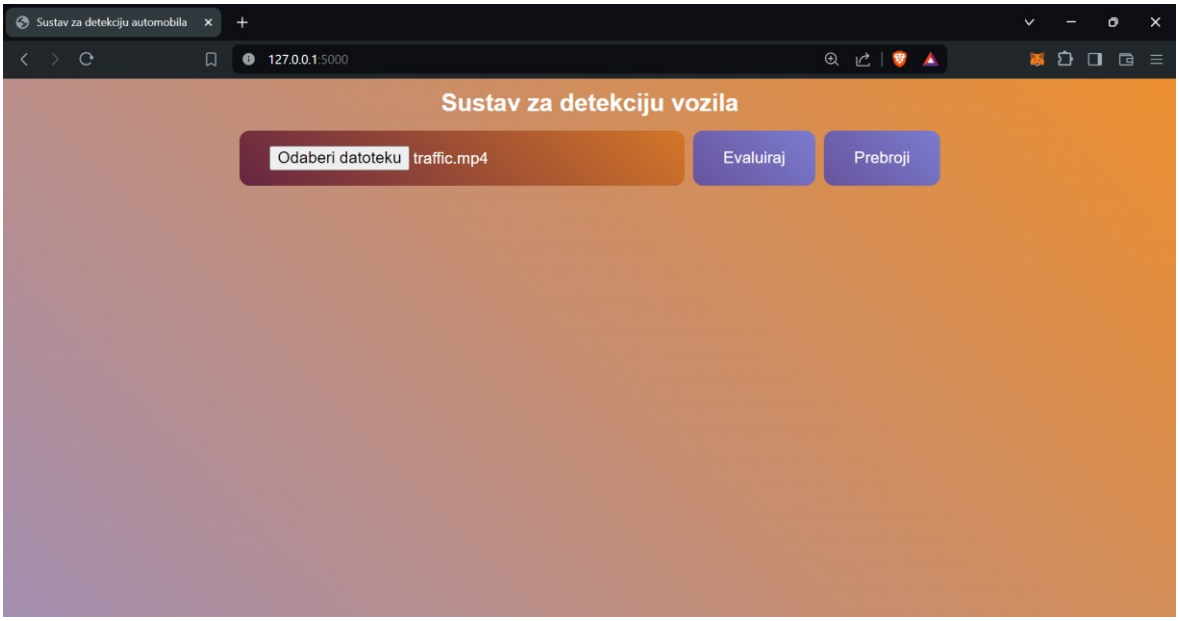
5.4. Web aplikacija za sustav za detekciju vozila

Za olakšani pristup funkcionalnostima sustava za detekciju vozila, napravljena je jednostavna web aplikacija koja koristi Python Flask na poslužiteljskoj strani, a HTML, CSS i Javascript na klijentskoj strani. Na slici Sl. 5.16 nalazi se prikaz početne stranice web aplikacije. Aplikacija nam nudi opciju odabira željene datoteke za obradu koja može biti slika (u formatu png, jpg ili jpeg) i video snimka (u formatu mp4, mov, avi ili webm).



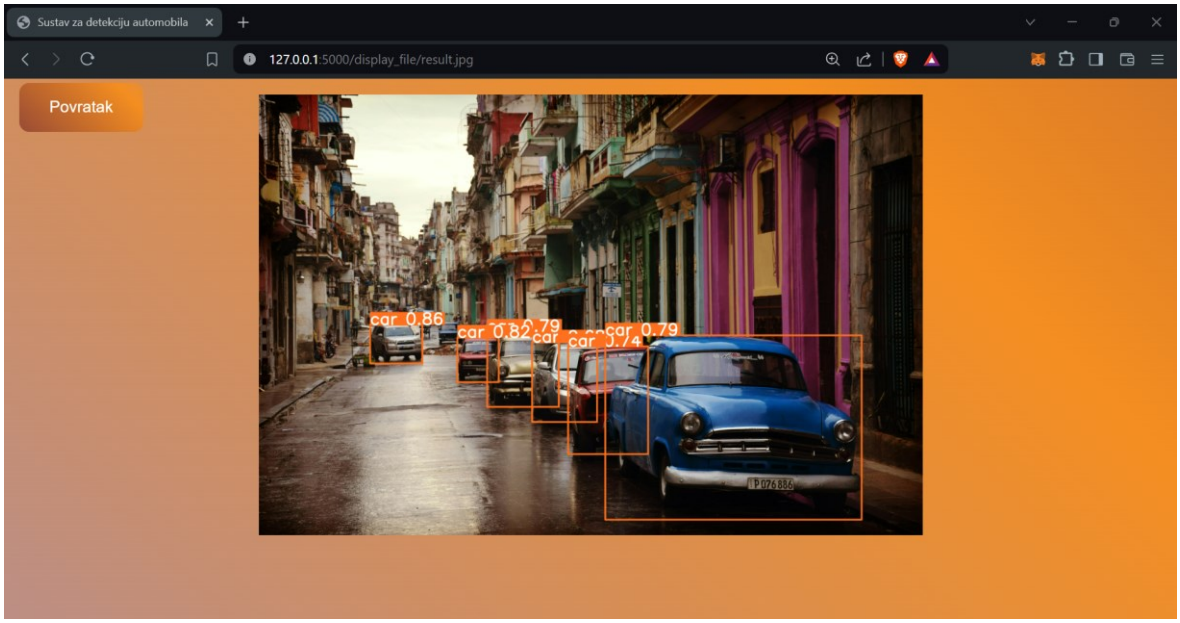
Sl. 5.16: Početna stranica web aplikacije

Nakon što smo učitali odabranu datoteku, pojavljuju nam se opcije kao što je prikazano na slici Sl. 5.17. Ako smo predali sliku, pojavljuje se opcija „Evaluiraj“ koja će identificirati i označiti vozila na slici, a ako smo predali video snimku, osim opcije „Evaluiraj“, pojavljuje se i opcija „Prebroji“ koja broji vozila na video snimci kao što je pokazano u prethodnom poglavlju.



Sl. 5.17: Web stranica kada učitamo video snimku

Na slici Sl. 5.18 nalazi se primjer kada smo predali sliku i pritisnuli „Evaluiraj“, dakle prikazana je slika nakon što ju je model obradio. U slučaju da smo učitali video snimku, obrađena video snimka ne bi bila prikazana u web pregledniku nego bi korisnici imali opciju preuzeti video lokalno na svoje računalo.



Sl. 5.18: Web stranica nakon obrade učitane slike

6. Zaključak

Nagli razvoj umjetne inteligencije u 21. stoljeću nudi mnogo novih i interesantnih koncepata za rješavanje raznih problema za koje je potrebna ljudska inteligencija i kao takva uključuje i područje računalnog vida. Neki od interesantnih problema kojima se bavi računalni vid predstavljeni su i u ovome radu, a to su identifikacija i praćenje objekata, klasifikacija i prebrojavanje. U okviru rada definirani su glavni pojmovi vezani uz umjetnu inteligenciju, strojno učenje i neuronske mreže. Cilj rada je bio razviti sustav za detekciju vozila i to je ostvareno koristeći YOLOv8 model.

U okviru rada treniran je vlastiti model na odabranom skupu podataka i dan je pregled rezultata treniranja pomoću metrika koje se često koriste u području računalnog vida kao što su pogreške prilikom učenja, preciznost, odziv, matrica zabune itd. Za referencu je prikazan i unaprijed trenirani YOLOv8 model koji se koristi za usporedbu kvalitete modela na stvarnim primjerima. Napravljena je jednostavna web aplikacija za lakši i intuitivan pristup sustavu.

Nastavno na ovaj rad, najprije bi se mogao koristiti kvalitetniji i mnogo veći skup podataka od onoga koji se koristi u ovome radu, što bi značilo veću točnost modela. Osim toga, mogle bi se dodati i nove funkcionalnosti kao što su procjena brzine vozila na video snimci ili identifikacija i čitanje registracijskih pločica kako bi se razvio potpuniji sustav za detekciju vozila.

Literatura

[1] Google Cloud, What is Aritificial Intelligence? Poveznica: https://cloud.google.com/learn/what-is-artificial-intelligence; 31. svibnja 2024.

[2] Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder, Uvod u umjetnu inteligenciju: Strojno učenje, prezentacija, 2024.; 31. svibnja 2024.

[3] AWS, What is Deep Learning? Poveznica: https://aws.amazon.com/what-is/deep-learning/; 31. svibnja 2024.

[4] Azure, What is Computer Vision? Poveznica: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision#object-classification; 1. lipnja 2024.

[5] SAS, Computer Vision – What it is and why it matters. Poveznica: https://www.sas.com/en_us/insights/analytics/computer-vision.html; 1. lipnja 2024.

[6] Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder, Uvod u umjetnu inteligenciju: Umjetne neuronske mreže, prezentacija, 2024.; 1. lipnja 2024.

[7] Kiprono Elijah Koech, The Basics of Neural Networks. (2022, svibanj). Poveznica: https://towardsdatascience.com/the-basics-of-neural-networks-neural-network-series-part-1-4419e343b2b; 1. lipnja 2024.

[8] Baeldung, What Is a Learning Curve in Machine Learning? Poveznica: https://www.baeldung.com/cs/learning-curve-ml; 2. lipnja 2024.

[9] Sumit Saha, A Guide to Convolutional Neural Networks. (2018, December). Poveznica: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/; 2. lipnja 2024.

[10] BairesDev, What is Flask? Poveznica: https://www.bairesdev.com/blog/what-is-flask/; 10. lipnja 2024.

[11] GeeksForGeeks, Introduction to Convolution Neural Network. (2024, ožujak) Poveznica: https://www.geeksforgeeks.org/introduction-convolution-neural-network/; 2. lipnja 2024.

[12] Joseph Nelson. What is YOLO? The Ultimate Guide. (2021, lipanj). Poveznica: https://blog.roboflow.com/guide-to-yolo-models/; 3. lipnja 2024.

[13] Petru Potrimba. What is R-CNN? (2023, rujan). Poveznica: https://blog.roboflow.com/what-is-r-cnn/; 3. lipnja 2024.

[14] Jacob Solawetz, Francesco. What is YOLOv8? The Ultimate Guide. (2023, siječanj). Poveznica: https://blog.roboflow.com/whats-new-in-yolov8/; 3. lipnja 2024.

[15] Slika: Pexels - Cars Parked Near Buildings during Daytime. Poveznica: https://www.pexels.com/photo/cars-parked-near-buildings-during-daytime-92866/; 5. lipnja 2024.

[16] CVAT – Open Dana Annotation Platform. Poveznica: https://www.cvat.ai/; 4. lipnja 2024.

[17] Roboflow. Poveznica: https://roboflow.com/, 4. lipnja 2024.

[18] Alkan Erturan, VehicleDetection-YOLOv8 – kaggle. Poveznica: https://www.kaggle.com/datasets/alkanerturan/vehicledetection?select=VehiclesDetectionDataset; 12. svibnja 2024.

[19] EvidentlyAI, Accuracy vs precision vs recall; Poveznica: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall; 12. lipnja 2024.

[20] Slika: Unsplash – City with street cars during daytime. Poveznica: https://unsplash.com/photos/city-street-with-cars-during-daytime-kzsMak9p9L0; 6. lipnja 2024.

[21] Video : Roboflow – Vehicle Counting. Poveznica: https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/how-to-track-and-count-vehicles-with-yolov8.ipynb ; 6. lipnja 2024.

[22] Joseph Redmon, You Only Look Once: Unified, Real-Time Object Detection. (2016). Poveznica: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html; 12. lipnja 2024.

[23] databricks, What is a Convolutional Layer? Poveznica: https://www.databricks.com/glossary/convolutional-layer; 12. lipnja 2024.

[24] Nikhil Tomar, What is intersection over union? (2023, veljača) Poveznica: https://idiotdeveloper.com/what-is-intersection-over-union-iou/; 12. lipnja 2024.

[25] G. Jocher, M. R. Munawar, A. Vina, Ultralytics – YOLO Performance metrics. (2023, studeni) Poveznica: https://docs.ultralytics.com/guides/yolo-performance-metrics/#interpretation-of-results; 12. lipnja 2024.

[26] Video: Raindrops falling on the windshield of a car. Poveznica: https://www.pexels.com/video/raindrops-falling-on-the-windshield-of-a-car-4672774/; 13. lipnja 2024.

[27] Jane Torres - YOLOv8 Architecture: A Deep Dive into its Architecture. (2024, siječanj) Poveznica: https://yolov8.org/yolov8-architecture/; 14. lipnja 2024.

[28] Brief summary of YOLOv8 model structure (2023, siječanj); Poveznica: https://github.com/ultralytics/ultralytics/issues/189; 14. lipnja 2024.

Sustav za detekciju vozila u video snimkama prometnica

Sažetak

U okviru rada napravljen je sustav za detekciju vozila koji se temelji na često korištenim konceptima iz područja računalnog vida kao što su identifikacija, klasifikacija i prebrojavanje objekata. To je ostvareno korištenjem YOLOv8 modela čiji se rad temelji na konvolucijskoj neuronskoj mreži. Osim unaprijed treniranog modela, korišten je i model učen na javno dostupnom skupu podataka. U radu se nalazi pregled rezultata treniranja modela i usporedba kvalitete pojedinih modela testiranih na uobičajenim primjerima s prometnica. Sustav je ostvaren kao jednostavna web aplikacija koja nudi korisnicima odabir vlastite slike ili video snimke koju će model koristiti za detekciju ili brojanje vozila.

Ključne riječi: strojno učenje, računalni vid, neuronske mreže, YOLO model, detekcija, klasifikacija

System for detecting vehicles in traffic video sequences

Abstract

In this paper, a vehicle detection system has been developed based on commonly used concepts in the field of computer vision, such as object identification, classification, and object counting. The system utilizes YOLOv8 model, which is based on a convolutional neural network. In addition to the pre-trained model, a new model has been trained on a public dataset. The paper includes an overview of the model training results and a comparison of each model by testing their quality on common traffic images. The system is implemented as a simple web application that allows the users to select their own image or video that the model will use for vehicle detection or counting.

Keywords: machine learning, computer vision, neural networks, YOLO model, detection, classification